Autocomplete using Solr with RankingAlgorithm

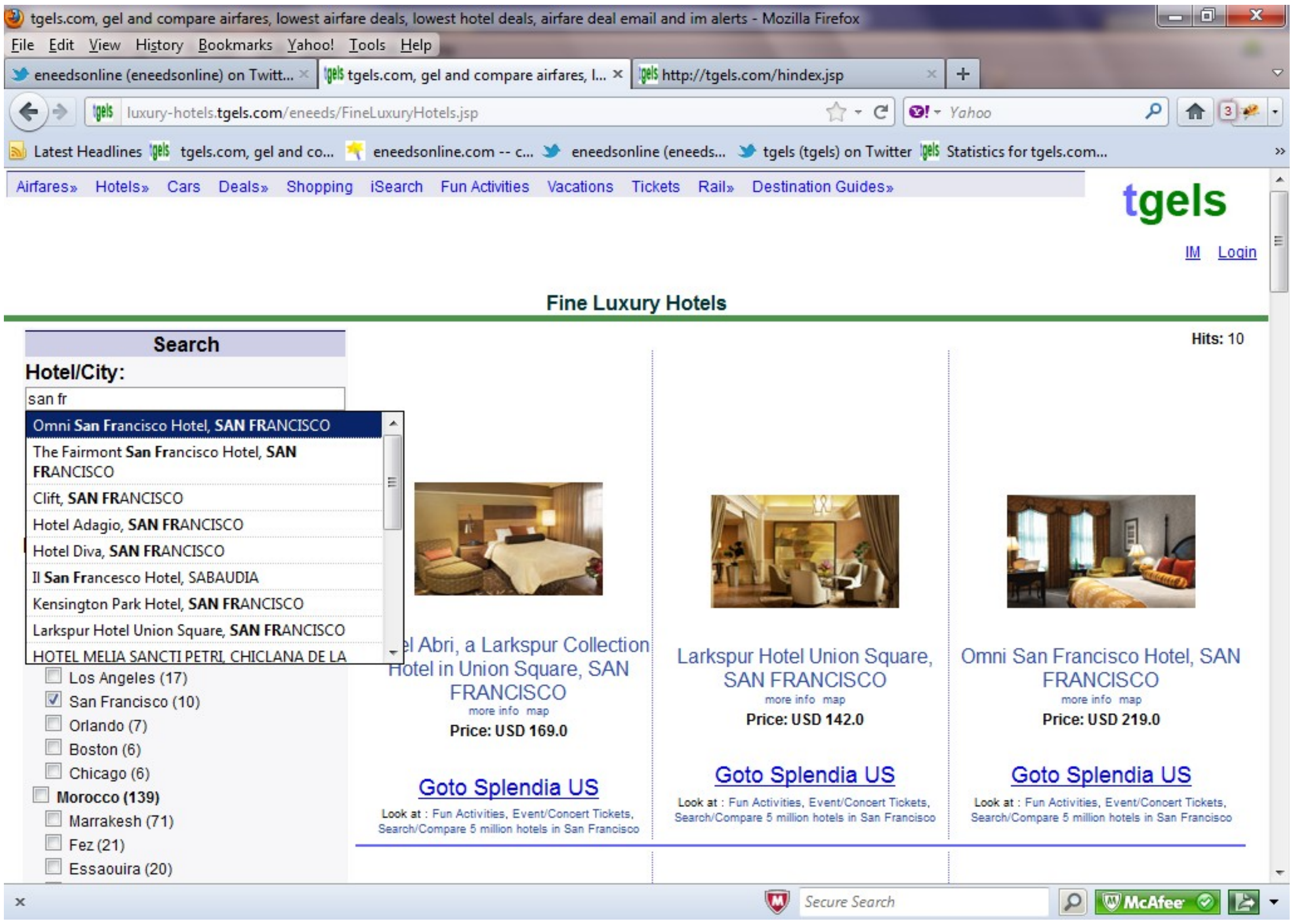
Nagendra Nagarajayya
solr-ra.tgels.org

1. Introduction

This article discusses an autocomplete implementation using Solr and RankingAlgorithm (solr-ra) that works similar to google/yahoo/bing autocomplete ie. the autocomplete matches the prefix of terms irrespective of its position in a title, name of a product/search terms, etc. The match could be setup so that it matches in reverse, etc. The names can include images, links, etc, including suggestions to enhance the input mechanism. The autocomplete mechanism can be used in browsers on the desktop, mobile phones, and desktop and mobile apps.

2. How does it work ?

An input field is marked as autocomplete enabled and any characters typed are sent across to the server side using Ajax. On the server side, a script next makes a Solr with RankingAlgorithm request with the typed characters and the returned results are sent back to client, which highlights the matches and shows the returned results in a popup window. So as the user starts typing characters, the popup window gets updated with new matches, allowing the user to easily selected the needed item. Autocomplete is very useful to guide the user to interact and select from a range of choices, while also allowing the user to enter new input if needed. See screen snapshot from an autocomplete demo implementation at http://http://luxury-hotels.tgels.com/eneeds/FineLuxuryHotels.jsp


tgels.com, gel and compare airfares, lowest airfare deals, lowest hotel deals, airfare deal email and im alerts - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
eneedsonline (eneedsonline) on Twitt... tgels.com, gel and compare airfares, l... http://tgels.com/hindex.jsp
luxury-hotels.tgels.com/eneeds/FineLuxuryHotels.jsp
Latest Headlines tgels.com, gel and co... eneedsonline.com -- c... eneedsonline (eneeds... tgels (tgels) on Twitter Statistics for tgels.com...
Airfares» Hotels» Cars Deals» Shopping iSearch Fun Activities Vacations Tickets Rail» Destination Guides»
tgels
IM Login
Secure Search
McAfee

Fine Luxury Hotels

Search

Hits: 10

Hotel/City:

san fr

Omni San Francisco Hotel, SAN FRANCISCO
The Fairmont San Francisco Hotel, SAN FRANCISCO
Clift, SAN FRANCISCO
Hotel Adagio, SAN FRANCISCO
Hotel Diva, SAN FRANCISCO
Il San Francesco Hotel, SABAUDIA
Kensington Park Hotel, SAN FRANCISCO
Larkspur Hotel Union Square, SAN FRANCISCO
HOTEL MELIA SANCTI PETRI, CHICLANA DE LA

- [ ] Los Angeles (17)
- [x] San Francisco (10)
- [ ] Orlando (7)
- [ ] Boston (6)
- [ ] Chicago (6)
- [ ] Morocco (139)
- [ ] Marrakesh (71)
- [ ] Fez (21)
- [ ] Essaouira (20)

el Abri, a Larkspur Collection Hotel in Union Square, SAN FRANCISCO
more info map
Price: USD 169.0
Goto Splendia US
Look at : Fun Activities, Event/Concert Tickets, Search/Compare 5 million hotels in San Francisco

Larkspur Hotel Union Square, SAN FRANCISCO
more info map
Price: USD 142.0
Goto Splendia US
Look at : Fun Activities, Event/Concert Tickets, Search/Compare 5 million hotels in San Francisco

Omni San Francisco Hotel, SAN FRANCISCO
more info map
Price: USD 219.0
Goto Splendia US
Look at : Fun Activities, Event/Concert Tickets, Search/Compare 5 million hotels in San Francisco

3. Implementation

The autocomplete implementation comprises of two sides a client side and server side. On the server side, it again comprises of two sides the web-tier and the search tier. The client side is the browser or mobile app which passes on the request as the user types characters onto the server side by making http requests. On the server a script is invoked that process and retrieves the input characters and it in turn makes a query request to the search tier running Solr with RankingAlgorithm. The search tier matches the input characters in its index and returns back the matching results. The results are sent back to the client side.

On the client side where javascript support is available,JQuery is used to process the input characters, make the ajax call to the server side, wait for a response and process the returned results, popup a window showing the returned results, highlighting the matching letters. For mobile apps, if javascript support is not available, the app can make a http or tcp request to the server side.

On the server side for the web tier, jsp/servlet, php, c/c++, python, ruby or perl can be used to process the input request from JQuery client. The web tier can run on a servlet container or Apache web server. The search tier is comprised of a Solr with RankingAlgorithm server.

3.1 Server Side

The server side is comprised of the web tier and the search tier. The request from the client browser is received first by the web tier which in turn makes a request to the search tier and sends back the returned response back to the client browser.

3.1.1 Web Tier

The web tier receives a http request from the client browser, retrieves the q parameter and makes a http call to the search tier, and returns the response in json format back to the client.

Sample jsp code:

```
search = request.getParameter("q");
search = URLEncoder.encode("\""+search+"\"");
String url = "http://searchtier/solr/wikipedia/select/?
q=ac_field:"+search+"&rows=15&wt=json&fl=a_name";
HttpMethod method = new GetMethod(url);
method.setFollowRedirects(true);
method.setStrictMode(false);
String ret = null;

client.executeMethod(method);
ret = method.getResponseBodyAsString();

out.print(ret);
```

3.1.2 Search tier

The search tier waits for a query request, and on a request searches through its index and returns the matching documents in json format as response.

The search tier runs Apache Solr with RankingAlgorithm server in a jetty/tomcat/glassfish web container. Solr with RankingAlgorithm is enhancing the Solr server with the RankingAlgorithm which provides multiple algorithms so as to rank documents more accurately, and also is very high performance. Solr with RankingAlgorithm can be installed from here, http://solr-ra.tgels.org (it is free).

The autocmplete search index needs a edgetoken field and the the below steps discuss adding a new schema fieldtype and field needed for autocomplete. If you download Solr with RankingAlgorithm, it comes with two demo schemas for wikipedia and mbartists. You can start with these schemas or copy the wikipedia schema to create a new one. You should find the schemas under multicore/wikipedia and multicore/mbartists.

3.1.2.1 Steps

1. If you already have a schema add the below type to the schema or create a new schema with the field shown below. If you need images, other information that you want shown then add those fields:

```
<field name="ac_field" type="edgetoken" />
```

Note:
a. ac_field index should be the text that will be matched on the server side when the user types text into an input field that is marked as an autocomplete field

2. Create the index

For eg:

```
curl "http://localhost:9993/solr/wikipedia1m/update?
overwrite=true&stream.file=/eneeds/fs/solr-ra-sites/apache-solr-ra-
3.4.0_2/work/1m.xml&commit=true&optimize=true"
```

A snippet of 1m.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<add xmlns:common="http://exslt.org/common">
<doc>
<field name="id">1</field>
<field name="name">AccessibleComputing</field>
</doc>
<doc>
...
```

3.1.2.2 edgeToken fieldtype : How does it work ?

<edgetoken> fieldtype splits the input text (ac_field input) into edge tokens. For eg:

“San Francisco City” will be broken into:

```
s
sa
san

f
fr
fra
fran
franc
franci
```

```
francis
francisc
francisco

c
ci
cit
city
```

The tokens start from the 1st character as `minGramSize` attribute is set to 1. If you set this to 2, then the edge tokenisation will start from the second character. For eg:

"San Francisco City" will now be broken as:

```
sa
san …

fr
fra …
```

3.3 Client side

Client side uses a jquery plugin to enable autocomplete. The input field is marked as autocomplete enabled by giving it an id which is next included in a jquery script as below. This allows jquery to recognize that the input field is autocomplete enabled. When a user types a few characters (configurable), the autocomplete plugin sends across the request to the server side script waits for a json response, evaluates the response and pops up the results in a popup window below the text field as in Fig 1.

```
<INPUT id=ac1 TYPE="TEXT" class=nobold12  size=45 NAME="from" maxlength="90"
VALUE='' >
```

Add the below jquery includes and script to your html file in the head section:

```
<script src="jtgels/jquery.1.6.8/jquery.js"></script>
<script type='text/javascript' src='jtgels/javascripts/jquery.autocomplete_solr-
ra.js'></script>

<script>
$("#ac1").autocomplete("/SearchAC.jsp\?a=1",
{
      minChars:1,
      width:350,
      scroll:true,
      scrollHeight:200
      });
</script>
```

Note:
1. ac1 the input field id should match the jquery key id, "#ac1".
2. *jquery.autocomplete_solr-ra.js* is an open source script. You can use other autocomplete

plugins available on jquery web site.

4. Performance

The performance of the autocomplete was tested against different indexes, a 3000 document Hotel names index with just 1 field, hotel name being the autocomplete field, a 390K document MBArtists index (artist name is the autocomplete field), a 1 million doc Wikipedia titles index (title of the wikipedia doc is the autocomplete field), and a 10.5 milllion Wikipedia titles index (complete index) (title of the wikipedia doc is the autocomplete field).

The load was generated using Grinder 3.x on a Linux 2 core system. The search system was also a Linux 2 core system with 6GB of memory. Solr 3.4 with RankingAlgorithm 1.3 ran inside a Jetty container started with JDK 1.6 and made use of the concurrent collector. The two systems were connected across a 100Mb network. A small Jython script was written that randomly selects a name from a list of 10 partial names. This was then transformed to a Solr with RankingAlgorithm query request and sent across the network, and waiting for the response. Once a search request was processed and returned, the next request was immediately initiated without any think time, etc. For the Hotel tests, the name was selected from a list of 10 partial hotel names, for the MBArtists test, the name was 10 partial artist names and for the Wikipedia test, the name was 10 partial Wikipedia titles. The first 50 samples were not collected so as to reach steady state, and at steady state 20,000 requests were executed and the test stopped. The standard I/O output of the Solr with RankingAlgorithm server was sent to a log file and the last 20,000 tests Qtime extracted from the log and the mean and std. deviation calculated as shown in the below table. The grinder mean time or tps is not shown since the tsearch times are are so small the overhead of the grinder network stack is much higher for the 3k Hotel index test and resulted in Grinder showing erroneous numbers.

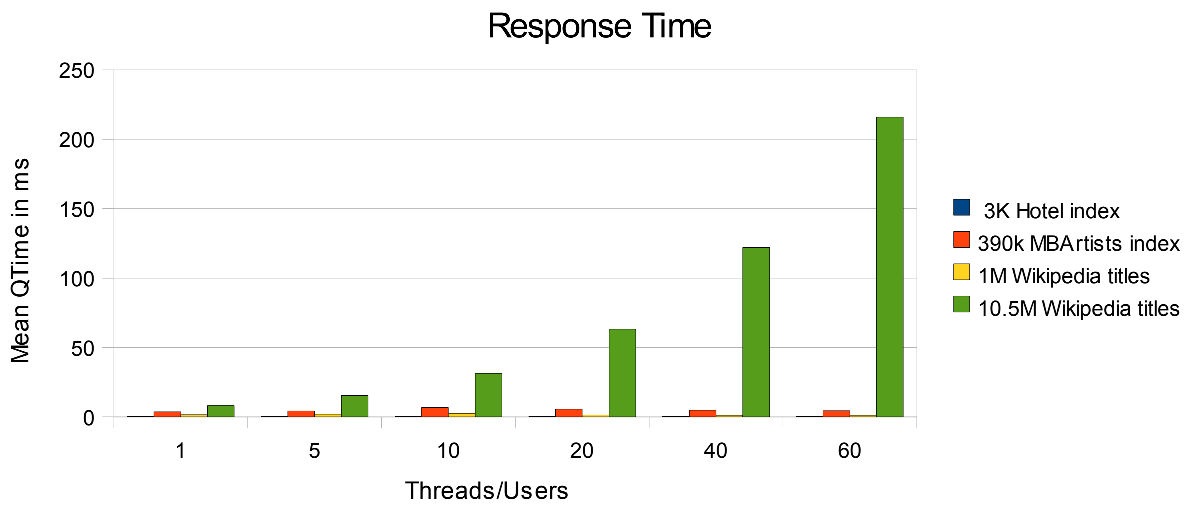
Note:
The Query Cache on the Solr with RankingAlgorithm server had been disabled so as to eliminate performance benefits seen with a cached request.

Threads or Users	3K Hotel index			390k MBArtists index			1M Wikipedia titles			10.5M Wikipedia titles		
	Mean Qtime ms	Std. Dev	Fail	Mean Qtime ms	Std. Dev	Fail	Mean Qtime ms	Std. Dev	Fail	Mean Qtime ms	Std. Dev	Fail
1	0.25	0.66	0	3.66	1.14	0	1.52	1.63	0	8.18	9.4	0
5	0.34	0.66	0	4.19	5.68	0	2	3.7	0	15.4	25.4	0
10	0.34	0.71	0	6.8	12	0	2.31	5.16	0	31.12	55.03	0

20	0.35	0.7	0	5.6	16	0	1.38	3.64	0	63.2	109	0
40	0.29	0.52	0	4.85	15	0	1.13	2.13	0	121.9	206.7	0
60	0.29	0.57	0	4.5	10	0	1.09	1.16	0	216	455	80

Table 1

Note: Mean and standard deviation calculated from a 20k sample



Graph 1 (less is better)

Graph 1 shows the Qtimes in ms for the different tests. As seen the best performance is with the 1 million Wikipedia index. The times are less than a couple of ms with very low deviation. The 10 million Wikipedia test shows increased latency with increase in load.

5. Conclusion

Autocomplete operation needs to be very fast as it is an interaction with the user and latency should be as small as possible maybe < 5 or 10 ms with low deviation. Solr with RankingAglorithm autocomplete is very fast up to 1million docs with query times of <3 ms and very low deviation. More than 1 million docs still has reasonable performance but with increased latency and deviation as more docs are in the index.

6. Download

autocomplete with Solr and RankingAlgorithm (Free)

(Need to add “Uses Solr with RankingAlgorithm” visibily with autocomplete, See license for more info)

autocomplete with Solr and RankingAlgorithm ($0.99)
(No Need to include anything, but you need to buy one for each usage. See license for more info)

1 yr support $29.99
[10 support requests]

1 yr unlimited support $299.99