Personalized search of news articles

Team: Oof1

Masters in Science

University at Buffalo

CSE 535 Information Retrieval (Fall 2014)

Harish Varadarajan

Kaushik Raj Palanichamy

Sankaravadivel Dhandapani

Sathish Kumar Deivasigamani

Table of Contents

ABSTRACT:

The purpose of this project is to build a personalized search engine for three different news articles using Apache Solr. The aim is to apply different concepts learned in the class and implement it.

SEARCH KEYWORDS:

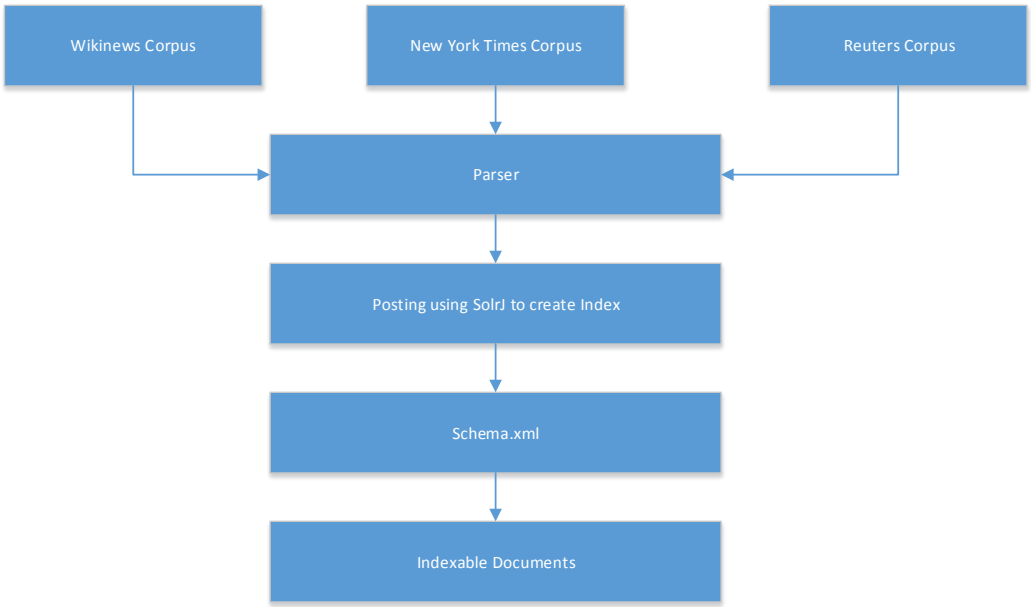
Apache Solr, Apache Lucene, NewYork Times Search, Wiki News Search, Reuters search

INTRODUCTION:

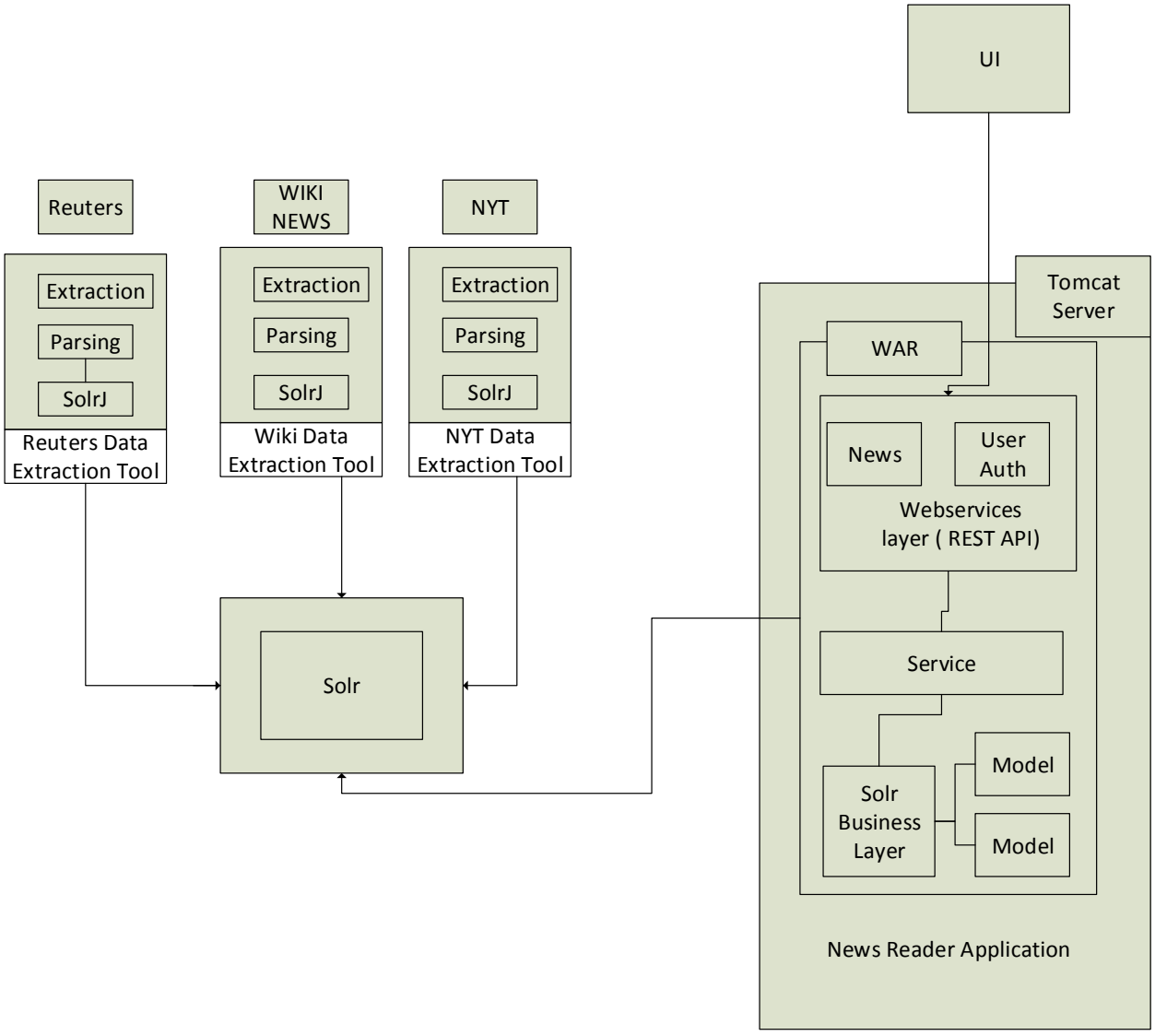
The aim of this project is to give personalized search results for the users based on the user preference and his/her previous searches. We have indexed three different corpus and each corpus has around 10000 articles. We allow user to choose their own preference and also learn from their searches and show results based on their interests and preferences. We also track anonymous user activity and when they login/sign up we map this user activity to the user created. We provide both Facebook OAuth authentication and form authentication for the user.

INDEXING MODULE:

The main goal of the indexing module is to parse the news corpus and convert it into a common JSON format and post it to the Solr to build index. All the news articles (New York Times, wiki news and Reuters) are converted into a JSON format [1].



APPLICATION ARCHITECHTURE:



SOLR CONFIGURATION DETAILS:

```
<field name="news_id" type="string" indexed="true" stored="true" required="true" multiValued="false" />
<field name="user_id" type="string" indexed="true" stored="true" required="true" multiValued="true" />
<field name="user_count" type="string" indexed="false" stored="true" multiValued="true" />
<field name="url" type="text_general" indexed="true" stored="true"/>
<field name="authors" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="place" type="string" indexed="true" stored="true" />
<field name="lat" type="float" indexed="true" stored="true" />
<field name="lon" type="float" indexed="true" stored="true" />
<field name="title" type="text_en_splitting" indexed="true" stored="true"/>
<field name="published_date" type="string" indexed="true" stored="true" />
<field name="category" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="sub_category" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="tags_descriptors" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="tags_locations" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="tags_people" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="tags_orgs" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="content" type="text_en_splitting" indexed="true" stored="true" />
<field name="publisher" type="text_en" indexed="true" stored="true" />
<field name="count" type="int" indexed="true" stored="true" />
<field name="subject" type="text_general" indexed="true" stored="true"/>
```

IMPORTANT FIELDS:

***news_id* -** This is the unique identifier of the document.

***user_id* -** This indicates whether the user is clicked or not. It's used in the preprocessing of queries.

***user_count* –** if a user clicks the document, this fields is updated. If the user id is already existing, the value will be incremented or the user will be added with value 1. It is used in the post processing of queries

***url* -** The URL of the news article.

***authors* –** Authors of the news articles. Represented in array

***place* -** place of the news article

***lat* –** Latitude of the news article

***lon* -** longitude of the news article

***title* -** title of the news article

***published_date* -** published_date of the news article. It's represented in YYYY-MM-DD HH:mm:ss format

***category* -** category of the news article**.** This is the highest level classification like politics, technology, sports etc.

***sub_category* –** sub_category of the news article**.** This is the next level category in which we metntion whether it belongs to football or Asia or Republican etc.,

***tags_descriptors* -** tags_descriptors of the news article

***tags_locations* -** tags_locations of the news article

***tags_people* -** tags_people of the news article

***tags_orgs* -** tags_org of the news article

***content* -** content of the news article

***publisher* -** publisher of the news article

***count* -** count is the cumulative number of times all the users have clicked on this particular news article.

Solr Configuration:

Auto Suggest:

```
<searchComponent name="custom_suggest" class="solr.SpellCheckComponent">
  <lst name="spellchecker">
    <str name="name">suggester</str>
    <str name="classname">org.apache.solr.spelling.suggest.Suggester</str>
    <str name="lookupImpl">org.apache.solr.spelling.suggest.tst.TSTLookupFactory</str>
    <str name="field">autoCorrect</str>
    <str name="buildOnCommit">true</str>
    <float name="threshold">0.0</float>
  </lst>
</searchComponent>
<requestHandler class="org.apache.solr.handler.component.SearchHandler" name="/suggest">
  <lst name="defaults">
    <str name="df">text</str>
    <str name="spellcheck">true</str>
    <str name="spellcheck.dictionary">suggester</str>
    <str name="spellcheck.collate">true</str>
    <str name="spellcheck.onlyMorePopular">true</str>
    <str name="spellcheck.extendedResults">true</str>
    <str name="spellcheck.count">5</str>
  </lst>
  <arr name="last-components">
    <str>custom_suggest</str>
  </arr>
</requestHandler>
```

SpellCheck1:

```
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">
   <str name="queryAnalyzerFieldType">text_general</str>
   <lst name="spellchecker">
     <str name="name">default</str>
     <str name="field">autoCorrect</str>
     <str name="classname">solr.DirectSolrSpellChecker</str>
     <!-- the spellcheck distance measure used, the default is the internal levenshtein -->
     <str name="distanceMeasure">internal</str>
     <!-- minimum accuracy needed to be considered a valid spellcheck suggestion -->
     <float name="accuracy">0.6</float>
     <!-- the maximum #edits we consider when enumerating terms: can be 1 or 2 -->
     <int name="maxEdits">2</int>
     <!-- the minimum shared prefix when enumerating terms -->
     <int name="minPrefix">1</int>
     <!-- maximum number of inspections per result. -->
     <int name="maxInspections">5</int>
     <!-- minimum length of a query term to be considered for correction -->
     <int name="minQueryLength">4</int>
     <!-- maximum threshold of documents a query term can appear to be considered for correction -->
     <float name="maxQueryFrequency">0.1</float>
     <!-- uncomment this to require suggestions to occur in 1% of the documents-->
       <float name="thresholdTokenFrequency">.0001</float>
   </lst>
   <!-- a spellchecker that can break or combine words.  See "/spell" handler below for usage -->
   <lst name="spellchecker">
     <str name="name">wordbreak</str>
     <str name="classname">solr.WordBreakSolrSpellChecker</str>
     <str name="field">autoCorrect</str>
     <str name="combineWords">true</str>
     <str name="breakWords">true</str>
     <int name="maxChanges">10</int>
   </lst>
</searchComponent>
```

SpellCheck2:

```
<requestHandler name="/spellcheck" class="solr.SearchHandler" >
   <lst name="defaults">
      <str name="df">autoCorrect</str>
      <str name="spellcheck">true</str>
      <str name="spellcheck.collate">true</str>
      <str name="spellcheck.dictionary">default</str>
      <str name="spellcheck.dictionary">wordbreak</str>
      <str name="spellcheck.count">5</str>
      <str name="spellcheck.maxCollations">2</str>
      <str name="spellcheck.maxCollationTries">5</str>
      <str name="spellcheck.alternativeTermCount">5</str>
      <str name="spellcheck.extendedResults">false</str>
   </lst>
   <arr name="last-components">
     <str>spellcheck</str>
   </arr>
</requestHandler>
```

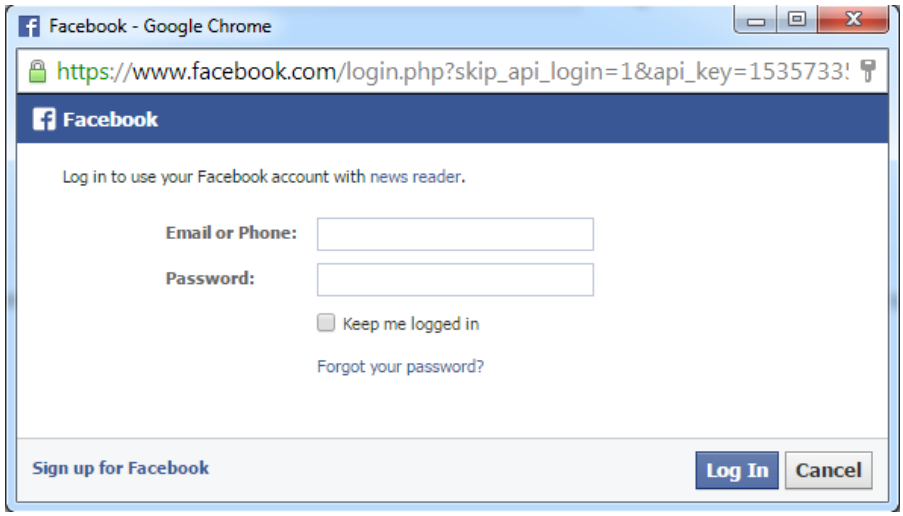
Search Handler:

```
<requestHandler name="/query" class="solr.SearchHandler">
     <lst name="defaults">
       <str name="echoParams">explicit</str>
       <str name="wt">json</str>
       <str name="indent">true</str>

       <str name="defType">edismax</str>
       <str name="qf"> place^7.0 title^10.0 content^3.0 authors^7.0 publisher^7.0 category^0.5</str>
       <str name="df">text</str>
       <str name="mm">100%</str>
       <str name="q.alt">*:*</str>
       <str name="rows">10</str>
       <str name="fl">*,score</str>

       <str name="hl">on</str>
       <str name="hl.fl">content authors title publisher</str>
       <str name="hl.snippets">3</str>
       <str name="hl.fragsize">200</str>
     </lst>
     <arr name="last-components">
        <str>clustering</str>
     </arr>
  </requestHandler>
```

LOGIN:

Facebook Login:

A Facebook app has been created for this project and used for Facebook authentication.

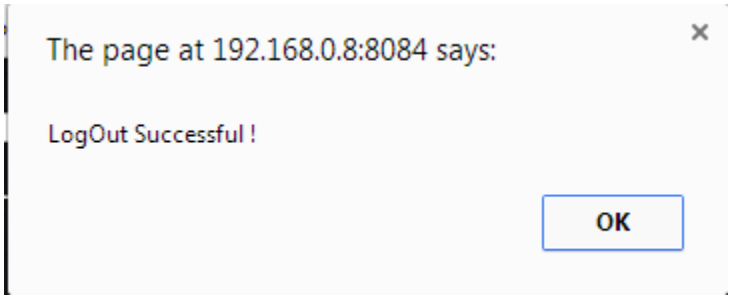


Normal Login:

Other than Facebook login the user can also login with native form authentication mechanism. Common functionalities like Username creation, authentication is done in java server in- memory. No persistent storage of user details is done. Since we are not using databases, if the server is restarted we would lose all the user information.

“User already exists” and “Invalid login credentials” are the error messages shown for duplicate user names and wrong passwords respectively.

Screen shot of native login screen



Browser Cookie is Cleared during logout

Anonymous Tracking:

To track the user the user need not be logged into the system. The user can be tracked anonymously. For example, if the user, without logging in starts searching items, we will generate a tracker id (a 36 character guid) and store it in her cookie. All her searches will be tracked against the guid and search history will be stored in the database. Whenever, the user signs in or signs up, we then link the guid with a username and from then onwards, the search history will be stored against the user.

When the user logs out of the system, the cookie is cleared. If a new user starts searching anonymously, a new guid is created to track the user's search history.

Personalization:

Personalization of search is achieved by preprocessing and post processing queries. Whenever a user clicks a news article link shown in the search page, the id of the news article is captured and stored in the document itself. If user "sam" clicks a news article for the first time, then

1. "**count**" field in the Solr xml will be incremented as it represents the global counter.
2. **user_id** field will have the user name "sam" added to its array of userids. This is used in preprocessing.
3. **user_count** field will have the user name "sam" with value 1 added to its array of user counts. This is used in the post processing of results.

At any given point of time, user_id arry length and user_count array length will be same.

Preprocessing of queries:

Queries are preprocessed with "AND userid" and if the query returns n results and n less than 10 (10 is the number of results displayed in the page), we fire another query with "AND NOT userid" and append the top 10-n results to the n result set obtained in the first query.

If the both result set does not add up to 10, then it highly likely that the user has misspelled the query. So spellcheck handler is called to get suggestions and in the UI, the suggestion with most search results will be shown as "Did you mean <suggestion>" in hyperlink.

Post – Processing of queries:

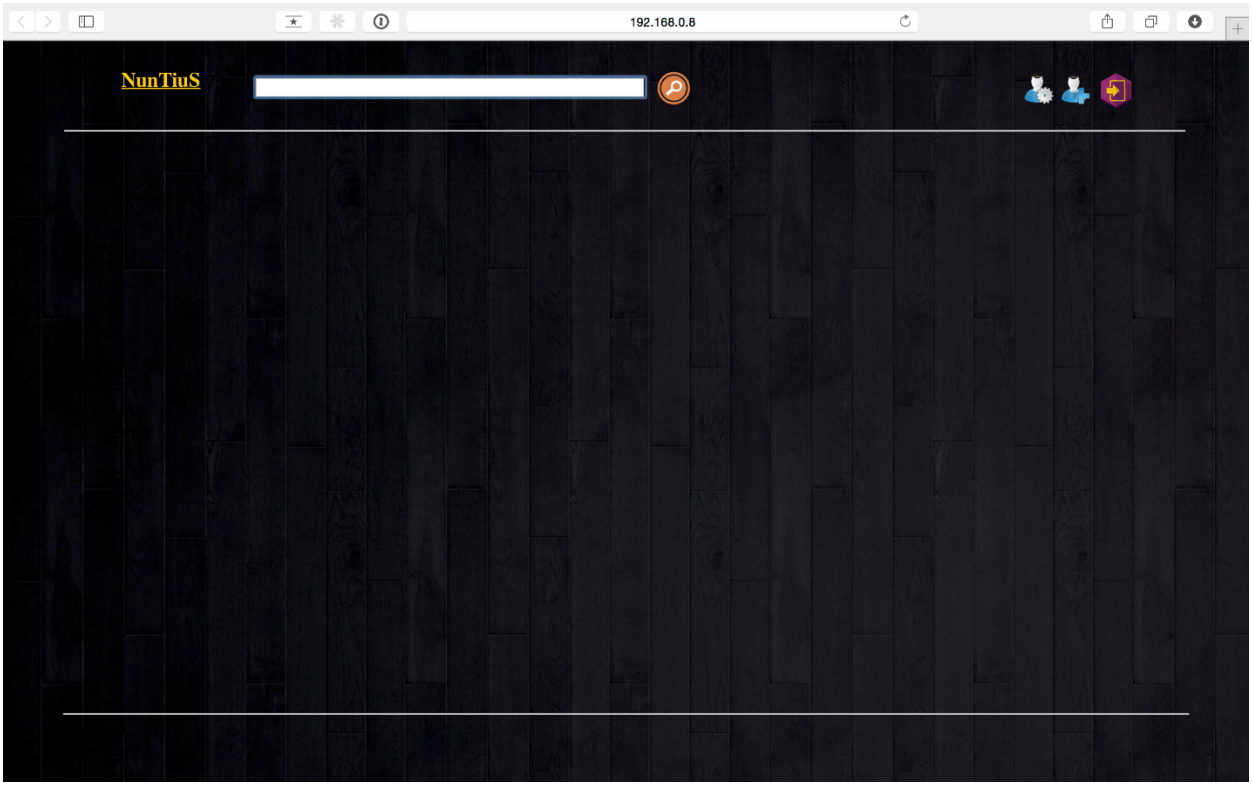
In the post processing of queries we make use of both global counter and the local user counter. Higher weightage is given to the local user hits as it closely represents the user preference thereby user personalization.

$$Finalscore = (userhits \times 0.25) + (globalhits \times 0.1) + Solrscore$$

The results are sorted in the descending order of the final score.

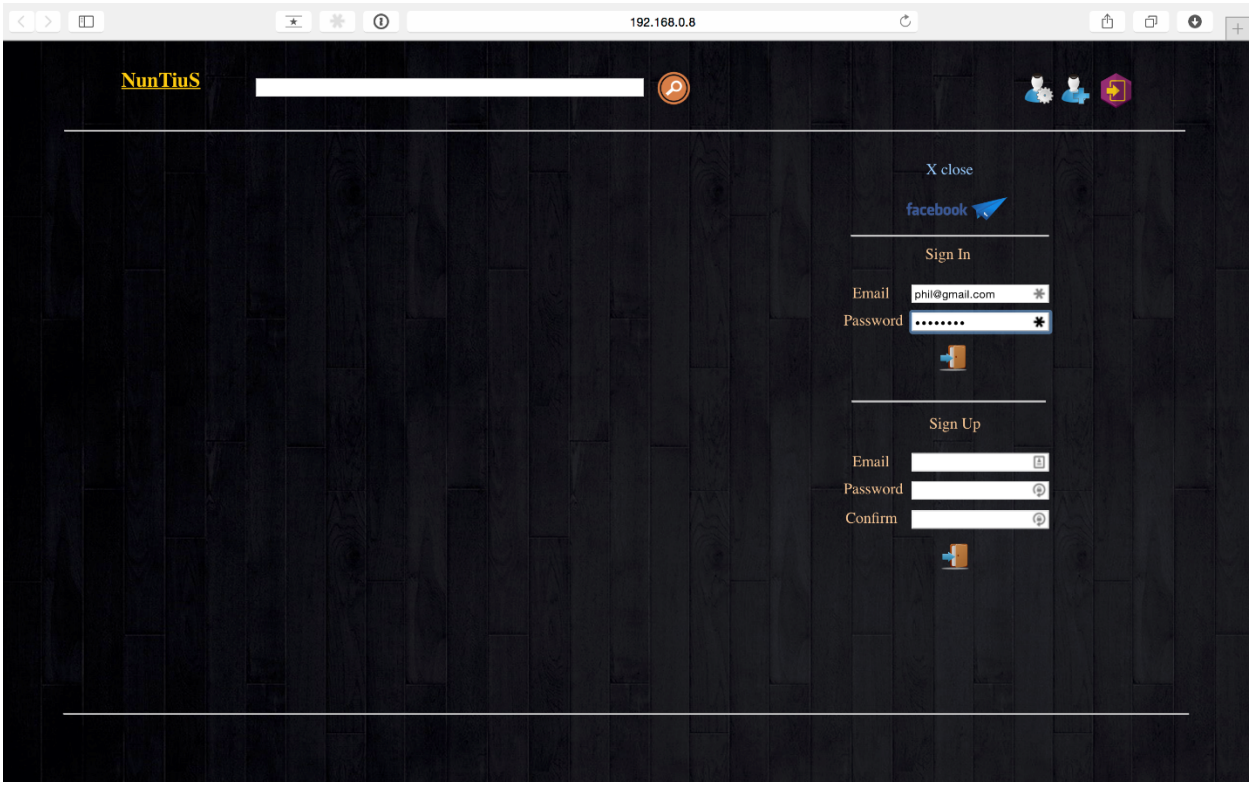
Home Screen

During page load the IP is sent to the server, a user object is created with the IP and it sends a user primary key as a response, which is stored in the browser cookie. Hence even though the user hasn't logged in, personalization is in process corresponding to his IP.



Login:

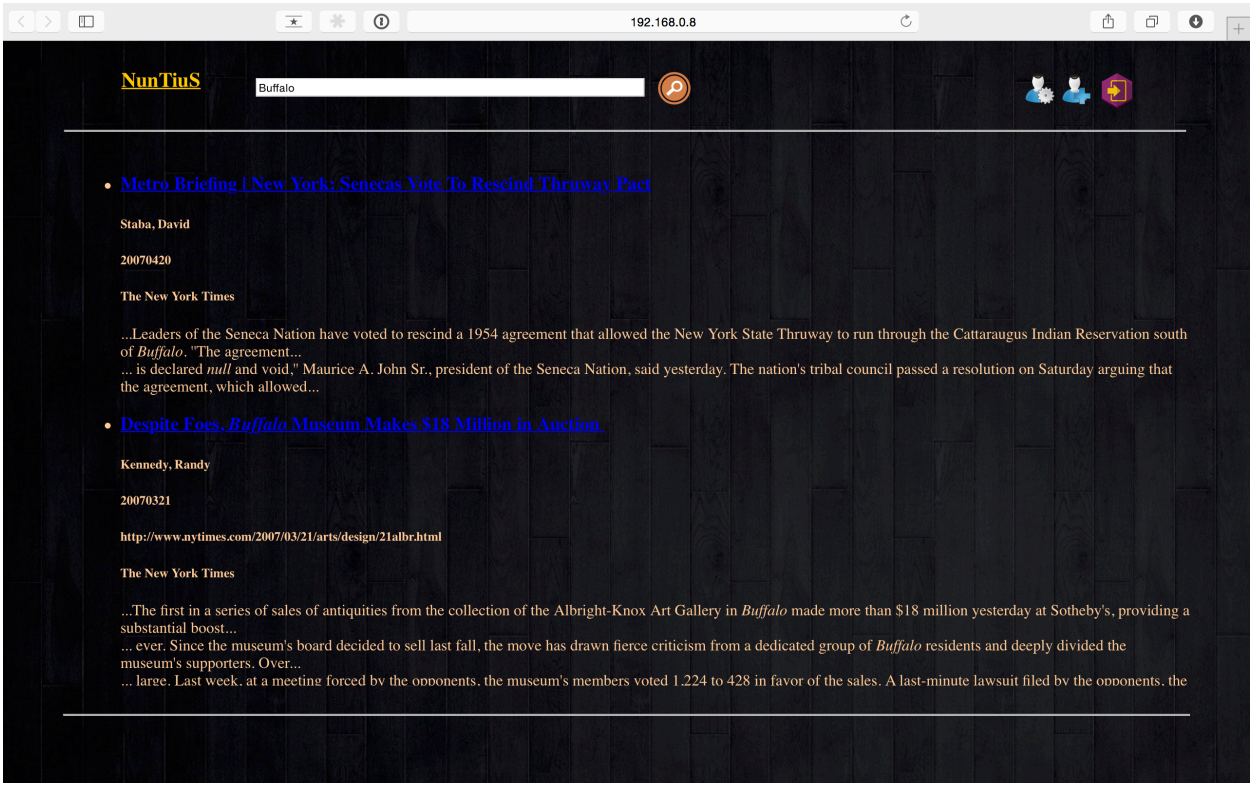
Now when the user sign up is successful, the signed up user id is sent to the server along with the IP. The user object corresponding to this IP is now updated with the user id and password. And all his previous personalization mapped to the user object still remains.



Search results:

The search query is sent as an AJAX request to the server and the personalized results returned from the server is displayed in the UI in the default order (descending order of the final score).

The first result by default is **“Metro Briefing | New York: Senecas Vote To Rescinf Thruway Pact”**

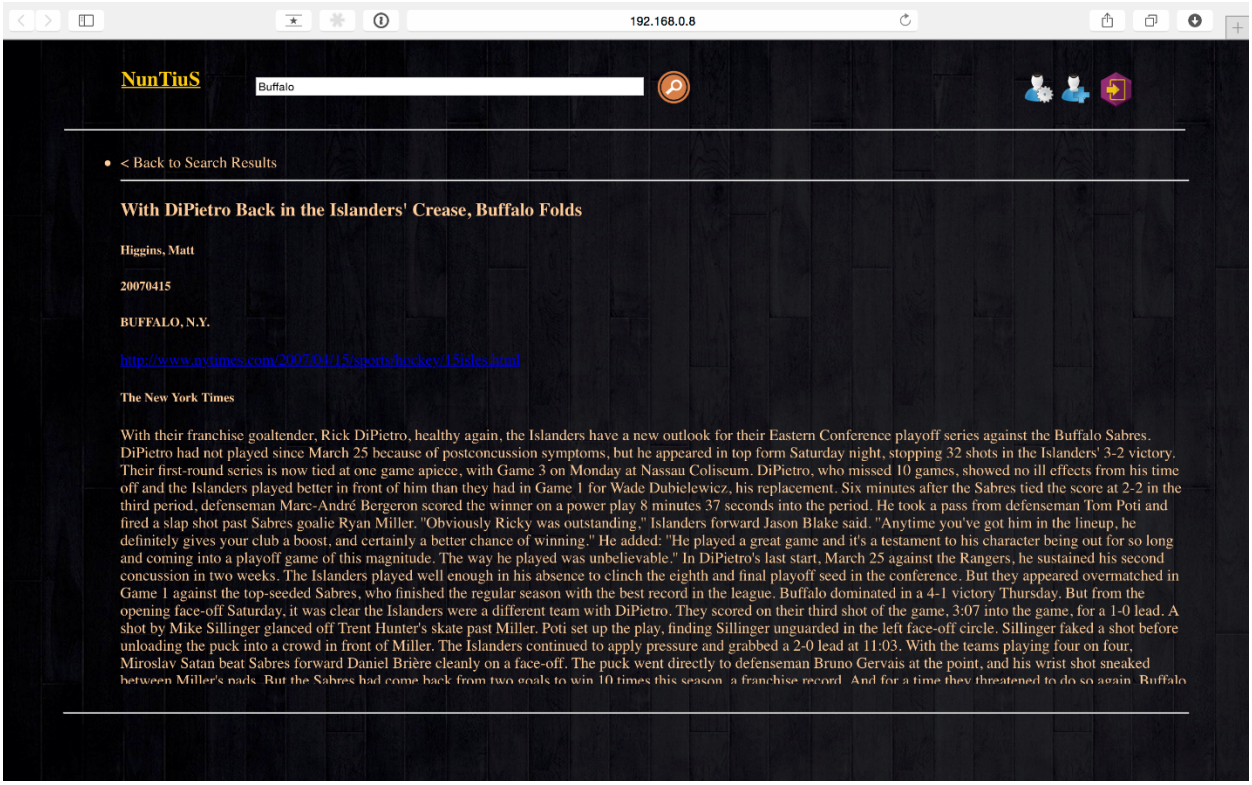


Complete news article displayed on click:

Once the user clicks on a particular news Title, the entire news content for that particular news is displayed to the user.

In addition to the above, once the user clicks on particular news, the news id is sent to the user and his click on that particular news item is stored for relevancy feedback score.

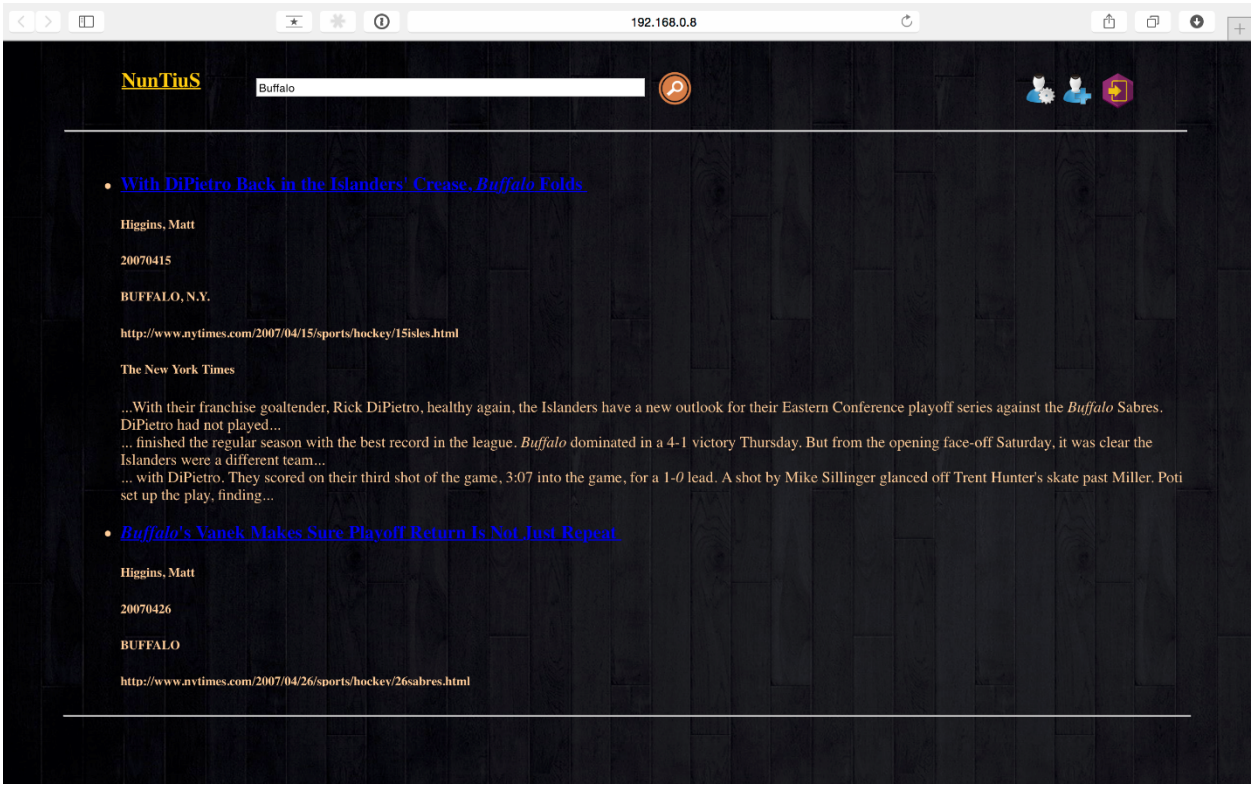
User Clicked on the news title, **“With DiPietro Back in the Islanders’ Crease, Buffalo Folds”**



Relevancy Feedback:

Hence, after few clicks on a particular news item, the next time the user searches for the same query that particular news item which the user had already clicked is displayed as the first result.

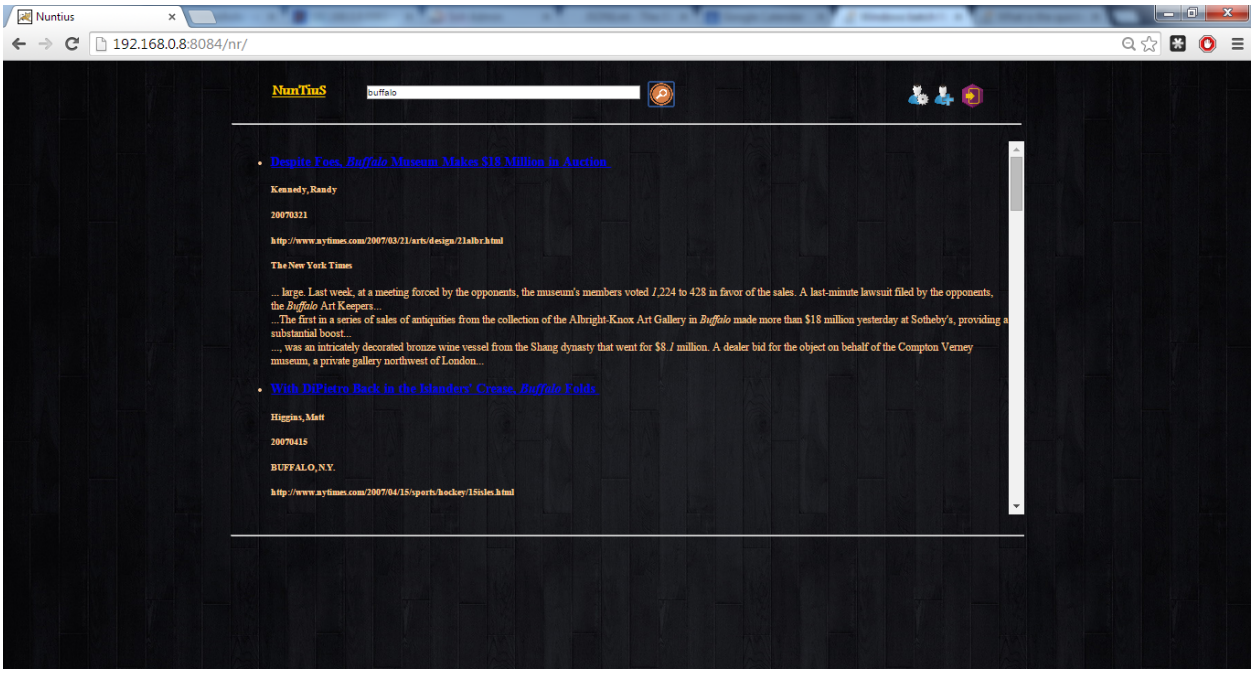
The first result after personalization is, **“With DiPietro Back in the Islanders’ Crease, Buffalo Folds”**



Search Results – User 2:

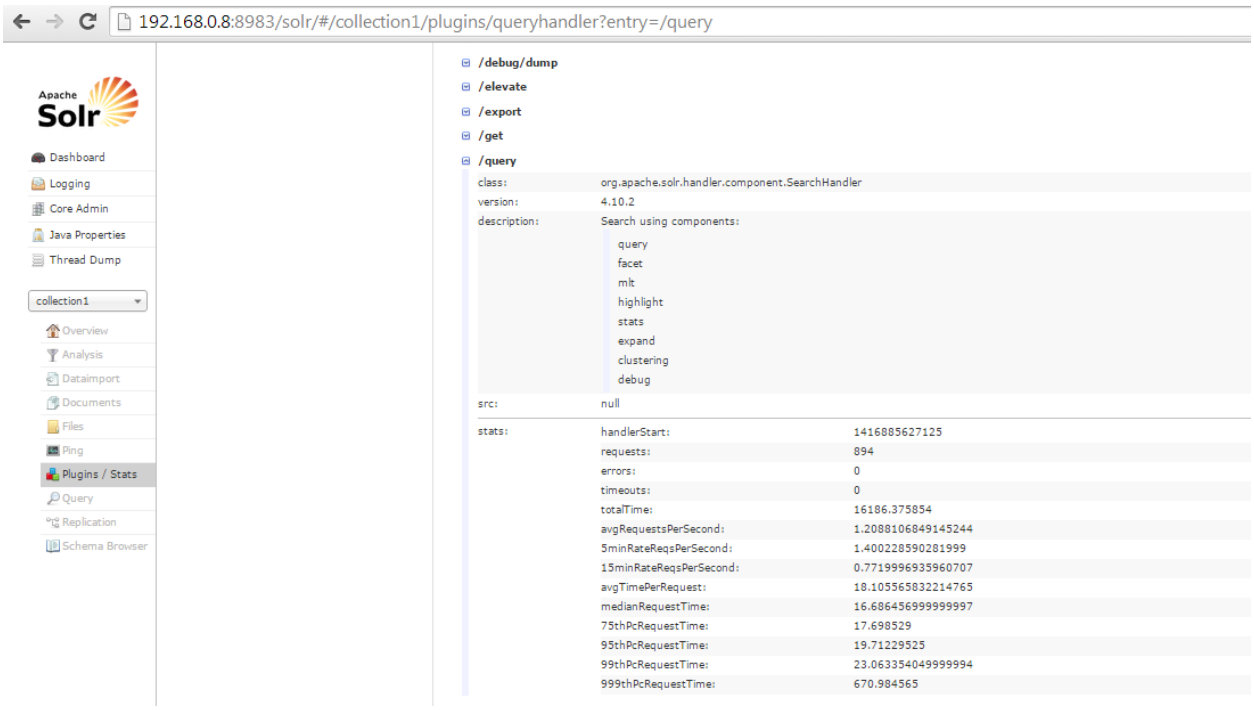
Whereas another user searching for the same query in a different machine has the result ordered in a default order (in case he is searching for the first time) or as per his own personalization (if he had already searched the portal).

Here results are displayed as per his personalization and the first result is, **“Despite Foes, Buffalo Museum Makes $18 Million in Auction”**

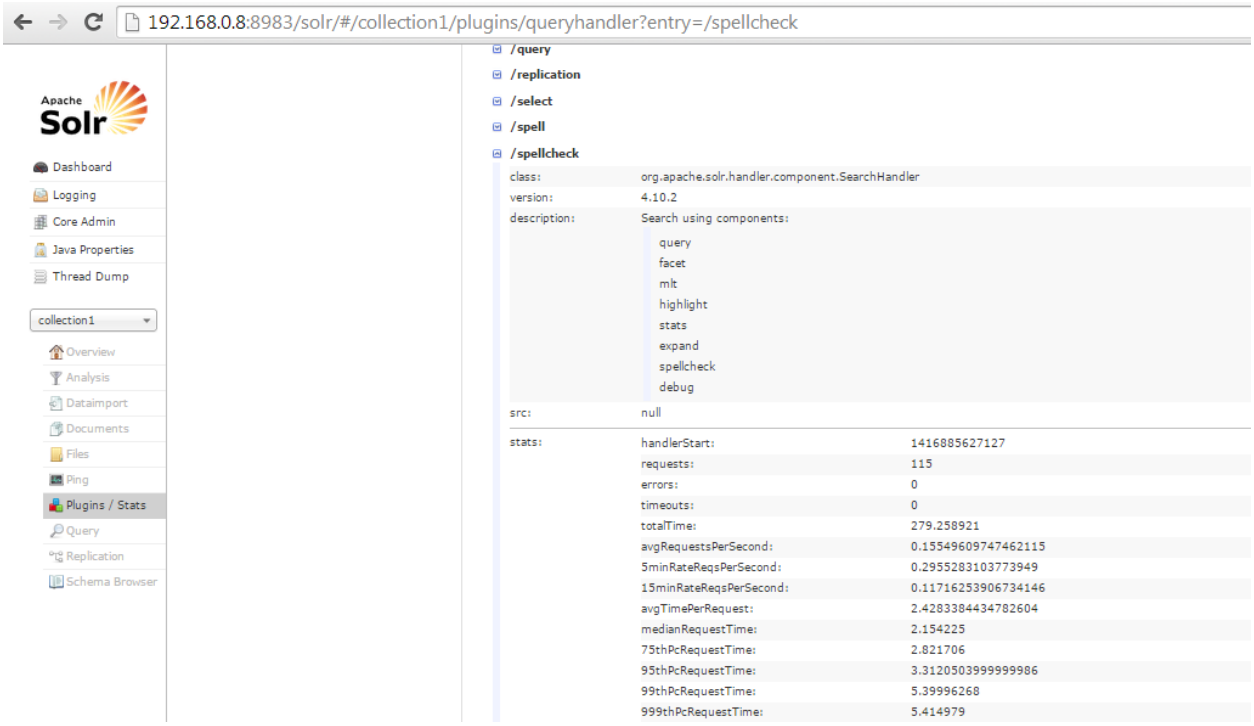


Handlers:

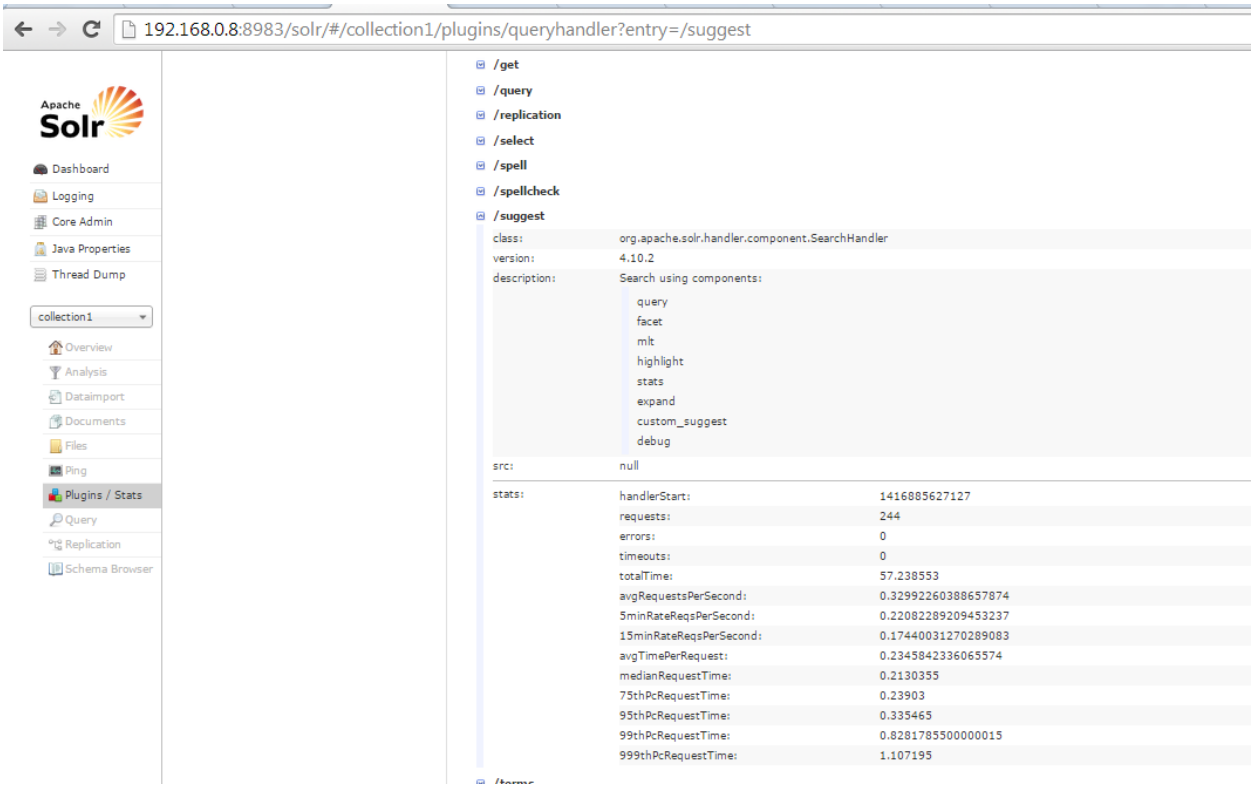
Query Handler:



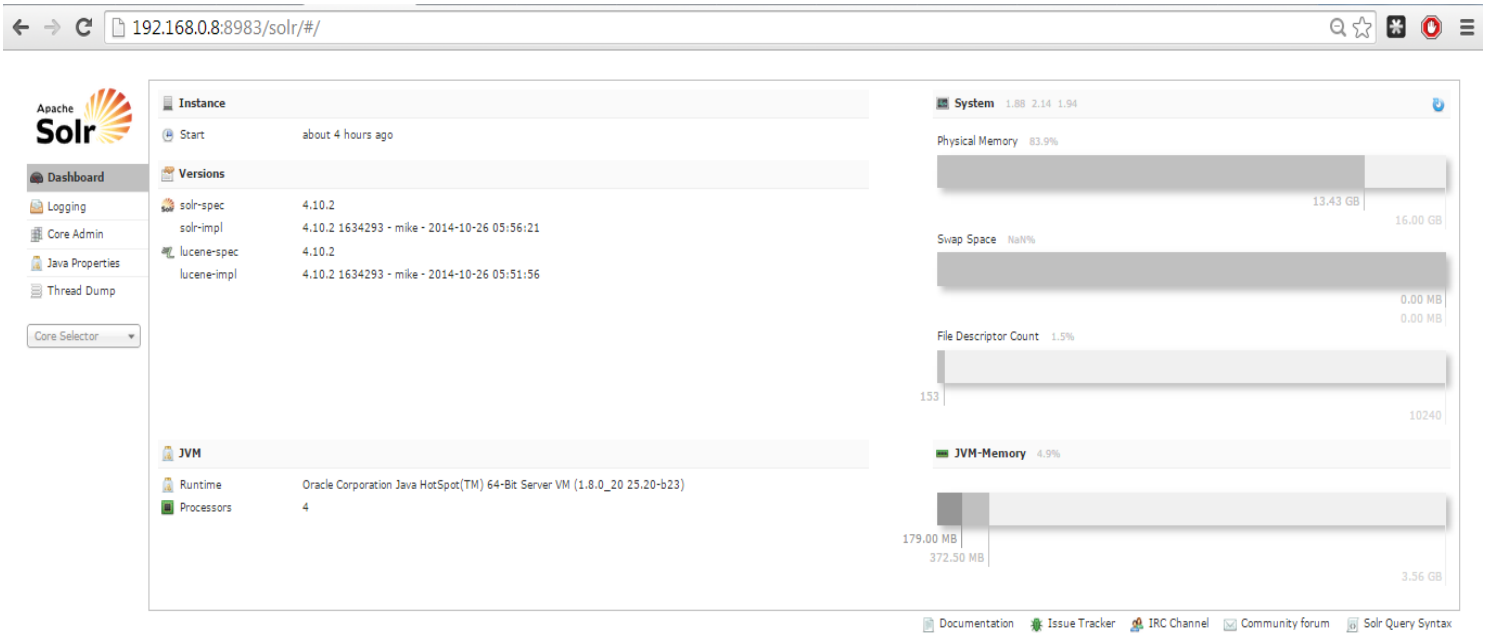
SpellCheck:


192.168.0.8:8983/solr/#/collection1/plugins/queryhandler?entry=/spellcheck
Apache Solr
Dashboard
Logging
Core Admin
Java Properties
Thread Dump
collection1
Overview
Analysis
Dataimport
Documents
Files
Ping
Plugins / Stats
Query
Replication
Schema Browser
/query
/replication
/select
/spell
/spellcheck
class: org.apache.solr.handler.component.SearchHandler
version: 4.10.2
description: Search using components:
query
facet
mlt
highlight
stats
expand
spellcheck
debug
src: null
stats:
handlerStart: 1416885627127
requests: 115
errors: 0
timeouts: 0
totalTime: 279.258921
avgRequestsPerSecond: 0.1554960974746211 5
minRateReqsPerSecond: 0.2955283103773949
15minRateReqsPerSecond: 0.11716253906734146
avgTimePerRequest: 2.4283384434782604
medianRequestTime: 2.154225
75thPcRequestTime: 2.821706
95thPcRequestTime: 3.3120503999999986
99thPcRequestTime: 5.39996268
999thPcRequestTime: 5.414979

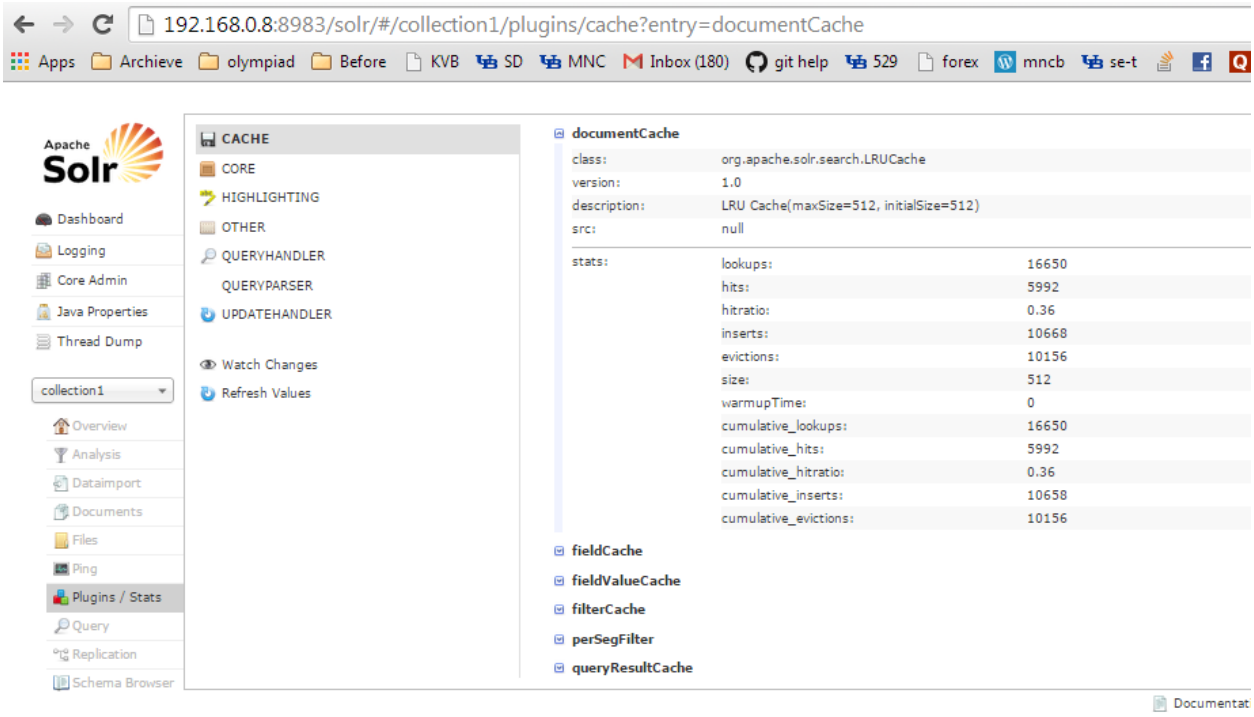
Suggest:


192.168.0.8:8983/solr/#/collection1/plugins/queryhandler?entry=/suggest
Apache Solr
Dashboard
Logging
Core Admin
Java Properties
Thread Dump
collection1
Overview
Analysis
Dataimport
Documents
Files
Ping
Plugins / Stats
Query
Replication
Schema Browser
/get
/query
/replication
/select
/spell
/spellcheck
/suggest
class: org.apache.solr.handler.component.SearchHandler
version: 4.10.2
description: Search using components:
query
facet
mlt
highlight
stats
expand
custom_suggest
debug
src: null
stats:
handlerStart: 1416885627127
requests: 244
errors: 0
timeouts: 0
totalTime: 57.238553
avgRequestsPerSecond: 0.32992260388657874
5minRateReqsPerSecond: 0.22082289209453237
15minRateReqsPerSecond: 0.17440031270289083
avgTimePerRequest: 0.2345842336065574
medianRequestTime: 0.2130355
75thPcRequestTime: 0.23903
95thPcRequestTime: 0.335465
99thPcRequestTime: 0.8281785500000015
999thPcRequestTime: 1.107195

Dashboard:


192.168.0.8:8983/solr/#/
Apache Solr
Dashboard
Logging
Core Admin
Java Properties
Thread Dump
Core Selector
Instance
Start
about 4 hours ago
Versions
solr-spec
4.10.2
solr-impl
4.10.2 1634293 - mike - 2014-10-26 05:56:21
lucene-spec
4.10.2
lucene-impl
4.10.2 1634293 - mike - 2014-10-26 05:51:56
JVM
Runtime
Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8.0_20 25.20-b23)
Processors
4
System 1.88 2.14 1.94
Physical Memory 83.9%
13.43 GB
16.00 GB
Swap Space NaN%
0.00 MB
0.00 MB
File Descriptor Count 1.5%
153
10240
JVM-Memory 4.9%
179.00 MB
372.50 MB
3.56 GB
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

Document Cache:



Work Distribution:

1	Architecture design	Harish, Kaushik, Sankar, Sathish
2	Web service layer	Sankar
3	Service layer	Sankar
4	SolrJ integration layer (Preprocessing)	Kaushik, Sathish
5	SolrJ integration layer (Post processing)	Kaushik, Sathish
6	Utility Development	Sathish
7	User Authentication	Sathish
8	Models Creation	Sathish
9	NYT Data Analysis	Sankar
10	NYT Data Extraction Utility	Sankar
11	Wiki News Data Analysis	Kaushik
12	Wiki news data extraction utility	Kaushik
13	Reuters News Data Analysis	Harish
14	Reuters news data extraction utility	Harish
15	Solr Schema	Harish, Sankar
16	Solr Configuration	Harish, Sankar
17	Facebook OAuth	Harish
18	Application build setup	Harish, Kaushik
19	User Interface + AJAX	Kaushik, Sathish

References

http://terranceasnyder.com/2014/05/user-based-personalization-engine-with-solr/

http://java.dzone.com/news/how-write-custom-solr

http://www.slideshare.net/LucidImagination/bialecki-andrzej-clickthroughrelevancerankinginsolrlucidworksenterprise-8419715

http://www.supermind.org/blog/1059/separating-relevance-signals-from-document-content-in-solr-or-lucene

http://www.solrtutorial.com/custom-solr-functionquery.html