2013

Question Answering Search System Using Apache Solr

PROJECT REPORT

AUTHOR: HELLOWORLD

Table of Contents

1. INTRODUCTION:

This is a Question Answering [QA] search system which provides an answer to any question asked by the user. This question is referred further in the document as User Query and is the main input given to the system.
We have already indexed more than 1.6 million Wikipedia documents so that a huge variety of questions can be answered. User query is processed against these indexed documents and most relevant information is displayed on the screen.
User interface is kept simple and minimalistic so that the user is not overwhelmed by intricate design. As it is a system of general public UI is made user friendly.

We support following three topics of queries:

- People
- Places
- Organizations/Clubs/Teams/Event Matches

All other topics are regarded as one category i.e. miscellaneous.

2. ASSUMPTIONS AND CONSTRAINTS:

- Questions asked by the user should be complete sentences and grammatically correct.
  We are concerned only with English Wikipedia pages.

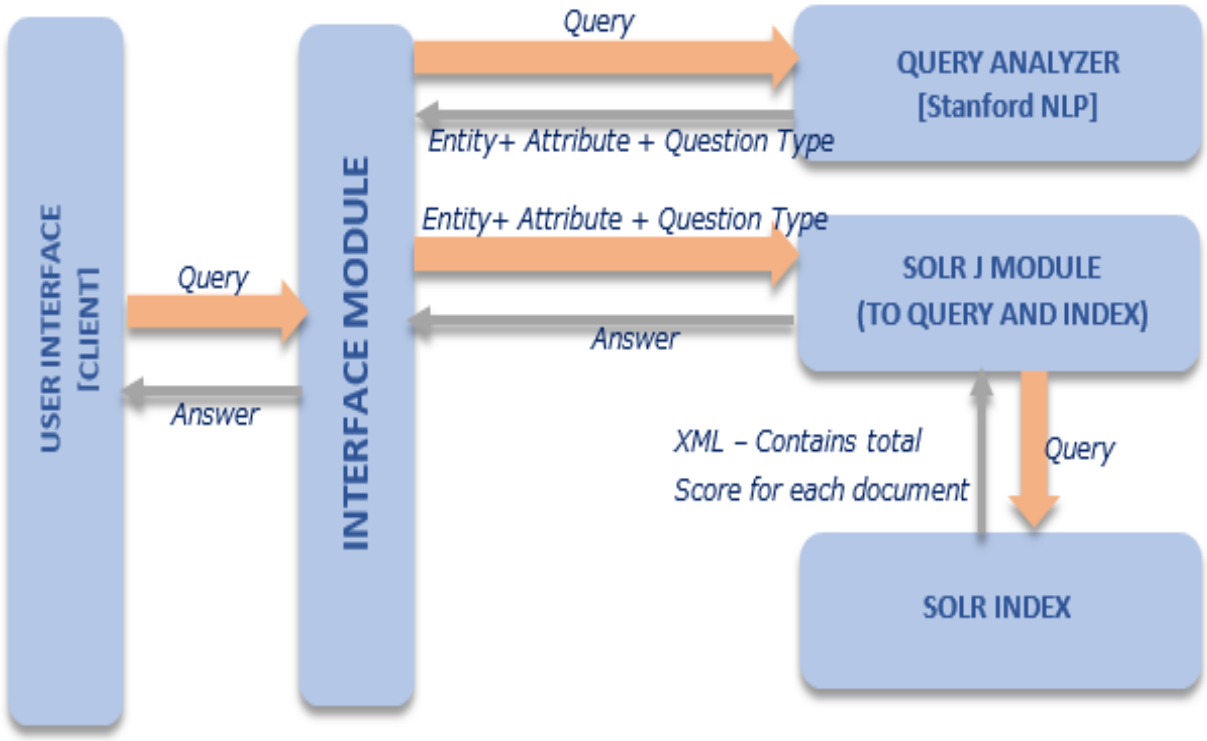
- This is the first attempt at making a Question Answering system, so we will be addressing only simple queries. For example “Who is Albert Einstein?” is a valid and simple query.

- Complicated queries like “When was Sachin Tendulkar’s wife born?” is a complex query as Sachin Tendulkar is mentioned as the main entity but the person whose information is actually needed is his spouse. Such queries do not give any answer in the current system.

  A work around for such a query can be – user asks “Who is Sachin Tendulkar’s wife” and then then asks for her birth date.

- If there is no specific answer in Wikipedia for the question asked, then the system returns a short summary form Wikipedia, stating—this might be a possible answer.

3. SYSTEM ARCHTECTURE:



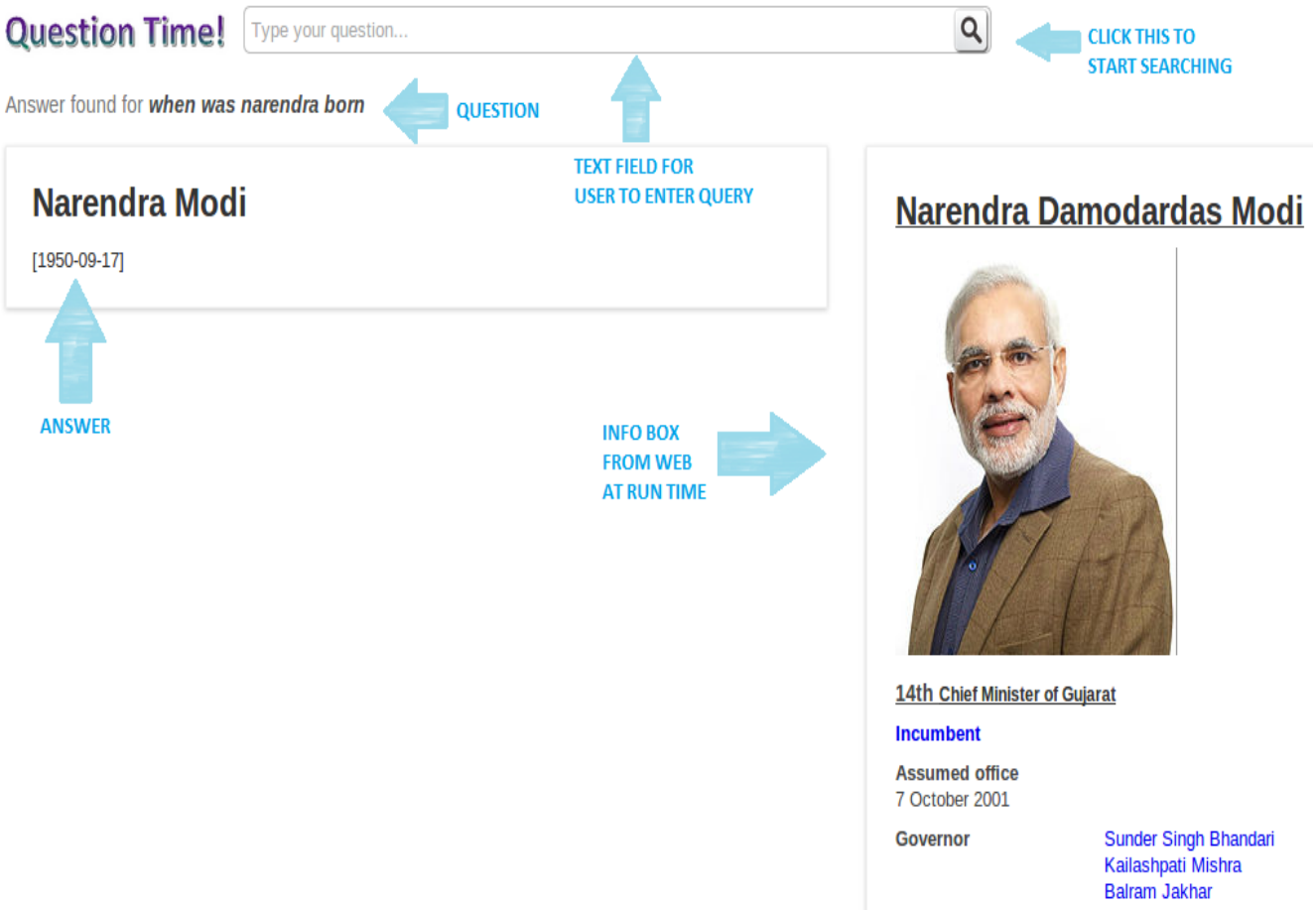
This is the basic structure of our system. Each of these modules and their functionalities are explained below one by one.

3.1 User Interface:

- User inter is a simple web page constructed using JSP and HTML5. Look and feel is kept very simple and minimalistic. Main intent is not to overwhelm users by an intricate and complex interface. The UI has a text field where user can enter their question and press Enter or click the search button.
- The Only way User Interface can interact through the rest of the system is through Interface Module. UI sends user query to Interface module from where it is forwarded to appropriate module for processing.
- UI also receives the answer [after query processing by the rest of the system] from interface module and displays it on the screen.

- On the right hand side we have the Wikipedia Info box extracted from web at run time for main entity queried by the user.
- Here are a few sample screen shots that will give a better idea of how results are shown on UI.

1) User Query: When was Narendra born
   System's Answer: Birth date is displayed on screen, along with the Wiki Info box for any extra information. Info box is also clickable. If the user clicks on any of the links, system will take them to the corresponding page.



2) User Query: Who was doctoral advisor of Albert einstein
System's Answer: Alfred_Kleiner and the Info box of Albert Einstein.

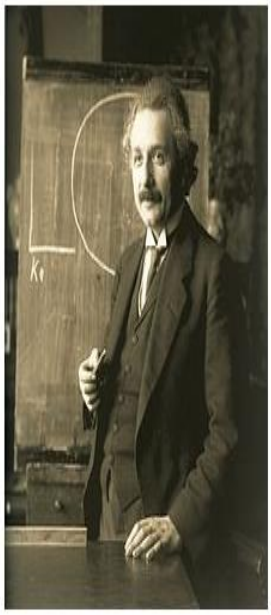
Question Time! Type your question...

Answer found for *who was doctoral advisor of albert einstein*

Albert Einstein

WIKI:Alfred_Kleiner

Albert Einstein



Albert Einstein in 1921

Born	14 March 1879 Ulm, Kingdom of Württemberg, German Empire
Died	18 April 1955 (aged 76) Princeton, New Jersey, U.S.

3.2 Interface Module:

This is the interaction layer between user interface module and all the other system modules where query processing takes place.
It controls the flow of user query to appropriate components and redirects the final answer from SOLR J module to UI to display.
Interface Module is constructed using JSP.

3.3 Query Analyzer:

Main purpose of this module is to extract three parameter for any incoming user query: **Question Type, Topic and Attribute.** Based on these three parameters we pass **Wiki_Handle** [title of the Wikipedia page we are looking for] and **Attribute** [value to be looked in the given Wikipedia page] are then passed on to SOLR J module for extracting the most relevant answer.

Here is a step by step explanation of how query analyzer module does extracts the three required parameters.

a) Extracting Parts of Speech :
Input query from the user is passed through Stanford POS tagger which divides the query into parts of speech like nouns, adjectives, verbs etc. all these parts of speech are tagged on to each word in the query.

b) Extracting Topic:
Topic has to a NNP [Proper noun, singular] or NN [Noun, singular or mass]. If NNP is not present in the query then only we look for NN. If none of them are identified in the query then Topic Parameter is null.
A topic can be Sachin Tendulkar, Delhi, University of Buffalo etc.

c) Extracting Question Type:
A question type is extracted from words with POS tagging like WDT [Wh determiner], WP [Wh pronoun], WPS [Possessive wh pronoun], WRB [Wh adverb].
Question type can be what, who, where, when, who and how.

d) Extracting Attribute:
Attributes can the extracted from words with tagging like FW [foreign word], IN [Preposition or subordinating conjunction], JJ [Adjective], JJR [Adjective, comparative], JJS [Adjective, Superlative], VB [verb base form] and other verb forms, etc.
Attributes are typically properties like born, height, population, climate etc.

e) Processing the Three parameters:
First step is to remove any stop words. Question type does not have any stop words so we process only Attribute and Topic in this step. Further, if there are any repetitive words in any of the three parameters then they are also removed. These two steps are carried out in the mentioned order.

f) Disambiguation and attribute mapping:

There are a lot of words mean the same thing depending upon the context they are used in. For example born, birth mean the same thing, big when used as How big is a city it means we are looking for the population of a city but when we use it as how big was a person , we actually want to know the height of the person.
So, in order to eradicate all such ambiguities we use following steps.

First thing is using Stanford NER [Named Entity Recognizer] Parser - Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. It is used for parameter "Topic" in analyzing what type of topic we have in hand. There can be four types of topics in our case – person, place, organization and everything else is placed into miscellaneous category. We use these topic types further in a map manually created as explained below.

Second thing is we have created a map manually – Now, taking the example used above for "big" , when used for topic type "person" means attribute is height and when used for topic type "place" means attribute is population. So a map is created with various probabilities of how an attribute can be used with different type of topics and what is the actual context of the attribute.

Here is a part of the map file used in the application.

```
#Question;Entity;word1,word2...;Field
#Person , Dates
when;person;birth,born,genesis,hail,created,create;WikiDateOfBirth
when;person;death,died,dead,kill,assassinate,killed,assassinated,slay,slayed,execute,executed;WikiDateOfDeath
#Person , location
where;person;birth,born,genesis,hail,created,create;WikiPlaceOfBirth
where;person;death,died,dead,kill,assassinate,killed,assassinated,slay,slayed,execute,executed;WikiPlaceOfDeath
#Person , relation
who;person;wife,spouse,mate,partner,bride,husband,groom,other half,boyfriend,girlfriend;WikiSpouce
who;person;kids,child,children,baby,offspring;WikiChildren
who;person;father,mother,guardian,wellspring,creater;WikiParents
#Person , self
what;person;occupation,work,chosen work, line of work,walk of life,job,do,did for;WikiOccupation
where;person;office,workplace,appointed,posted,appointment;WikiOffice
what;person;net worth,worth,assets,total assets;WikiNetWorth
how;person;net worth,worth,assets,total assets;WikiNetWorth
how;person;death,died,dead,kill,assassinate,killed,assassinated,slay,slayed,execute,executed;WikiDeathCause
what;person;nationality,national,country,contry;WikiNationality
how;person;education,educated,study,studied,education,educate,trained;WikiEducation
where;person;education,educated,study,studied,education,educate,trained;WikiEducation
```

Based on the above file, if n-gram similarity is greater than 0.95 then only a result from this file are returned and a flag is set to tell Solr J module that an exact match for attribute value is found. Otherwise the attribute field is as it is and flag is not set.

3.4 SOLR Index:

We have used online DBPedia repository as our source document. It is an online data source of extracted structured data from Wikipedia and is available in NT format. Converted NT files downloaded from DBPedia to Text files by removing the DBPedia markups.

Indexes are explained in detail below:

1) **Info Box index:** Contains all the infobox details along with Wiki Handle as the ID.

2) **Short Abstract Index:** Contains all short abstract indexed with following fields"
   Wiki Handle and short abstract from RDF data source.
   In this case Wiki handle is a unique key. There will be only one entry per wiki handle in the index.

3) **Long wiki abstract index:** Contains all the long wiki abstract indexed with following fields:
   Wiki handle, info box, long abstract, short abstract static score, long wiki abstract static score, external link static score.

Similar to short abstract index, there will be only one entry per wiki handle in the index.

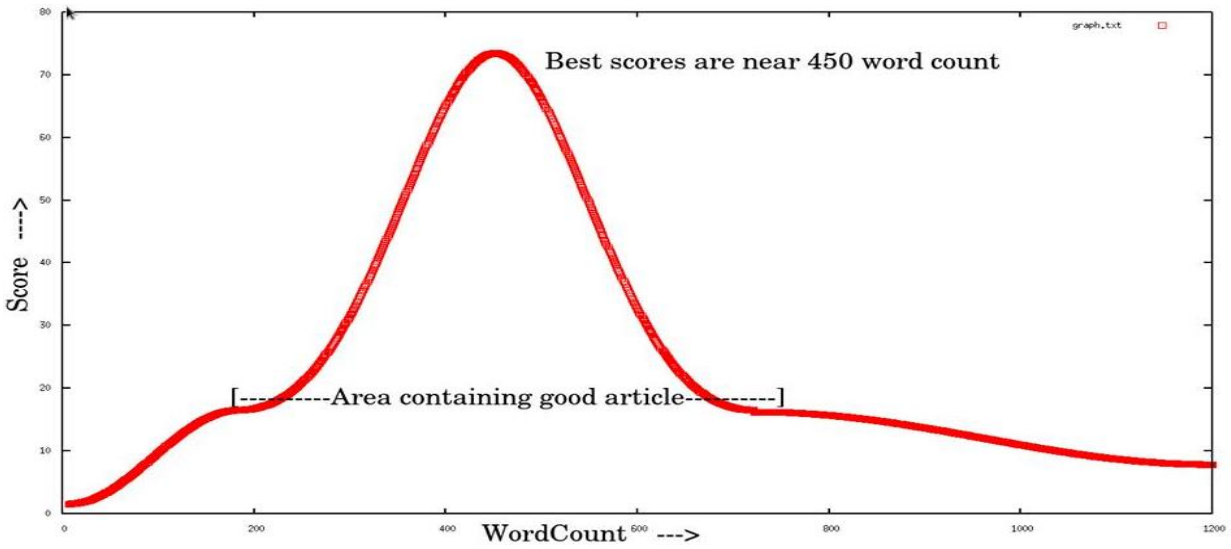
How to calculate static scores:

i) **Long wiki Abstract static score** is calculated using following three features:
- Normalized word count of long wiki abstract.
- Normalized count of external links.
- Normalized word count of short abstract.

For calculating normalized score using word count of long wiki abstract, we first did a simple word count and then normalized the score is calculated using following function:

```
if(WordCount>180 AND WordCount<720)
{
    if($words < 450)
    {
            tBoost = (540 - (450-WordCount))/270;
            Score = score * tBoost;
            Sint = ((Score - 45)/30)*PI/2;
            Wt = sin(Sint)*0.95;
            Score = 45+30*Wt;
    }
    else
    {
            tBoost = (540 - (WordCount-450))/270;
            Score = (75-Score) * tBoost;
            $sint = ((Score - 45)/30)*PI/2;
            $wt = sin(Sint)*0.95;
            Score = 45+30*Wt;
    }
}
```

Here is graphical representation of normalized word count of a document to the corresponding score it gets. It can be seen that documents with word count of around 450 get the highest score.



ii) Short abstract static score and External link static score are is calculated using Logarithmic functions. External link static score is calculated by taking the Logarithm of number of links in the Wikipedia page. If this score comes out to be greater than 220 [again heuristically decided. It is a better idea to learn it with machine learning], then penalty is subtracted from static score. Penalty is calculated using following formula.

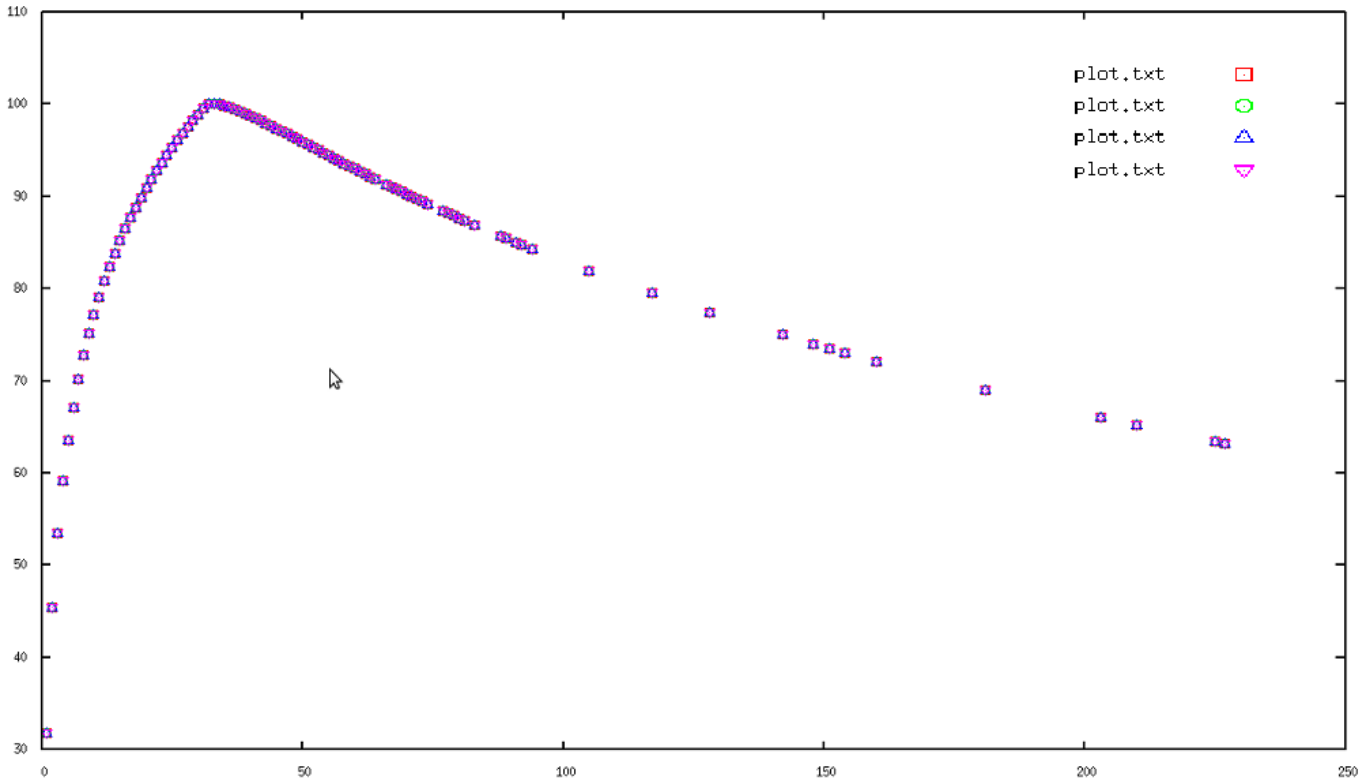
Penalty = 0.8* Math.pow((val-220), 1.2)

For short abstract static score we use number of words in the abstract to calculate the logarithmic value and if it is greater than 220 we subtract penalty from the score as explained above.

Here is a graph showing the relation between external link static score and number of links [x-axis] / short abstract static score and number of words in short abstract [x-axis].

It is clear that optimum number of external links or number of words in short abstract is between 30 and 40 to get maximum static score.

All the static scores range between 5 and 100.



3.5 SOLR J Module

Wiki_Handle received from Query analyzer is first searched on Long wiki abstract index. If a match is found then based on the static scores and Query scores the most relevant document is returned.

Now if Query analyzer was able to find an exact attribute it sets a flag and sends it to Solr J module. If this flag is set then we simply search for given attribute from infobox index for the given wiki_handle and return it. Otherwise in case where the flag was not set, an attribute with maximum similarity [calculated using n-gram similarity] with the input attribute parameter is returned as final result.

Query score is the tf-idf SOLR score and is used to extract only top 20 documents. These documents are then sorted as per their static score calculated by the formula given below:

Static score = 0.6 * long static score + 0.25 * short abstract static score + 0.1 * external link score

Where numerical values are the weights assigned to each of these parameters. These weights are currently heuristically decided, but it is a better idea to learn these weights through machine learning.

The top most document is extracted and returned in form of a java object with all required field values in it.
If no results are found then short description for corresponding documents are returned from Short abstract index.

4. SPECIAL FEATURES:

- More than 1.6 million documents have been indexed and the system is considerably fast for such a vast index.
- Static score is used to improve the quality of the results
- This QA system supports all the five types of queries but priority given to People, Places and organization/club/team in that order. Other types of queries are taken into one category, so any type of query is supported by this QA system.
- User Interface also shows the Wikipedia Info box for the entity being queried. This info box is clickable will take the user to Wikipedia page of the entity shown. It is easier for user to get more information if they need.
- As this is a QA system, the response time and accuracy of retrieved results is very important. So main emphasis is on improving speed of the system.
- User interface look and feel is kept very simple and minimalistic. Main concern is not to overwhelm users by an intricate and complex interface.
- System has robust exception handling.

5. CONFIGURATION DETAILS AND TWEEKS:

Most of the schema fields are of TYPE: text_general.

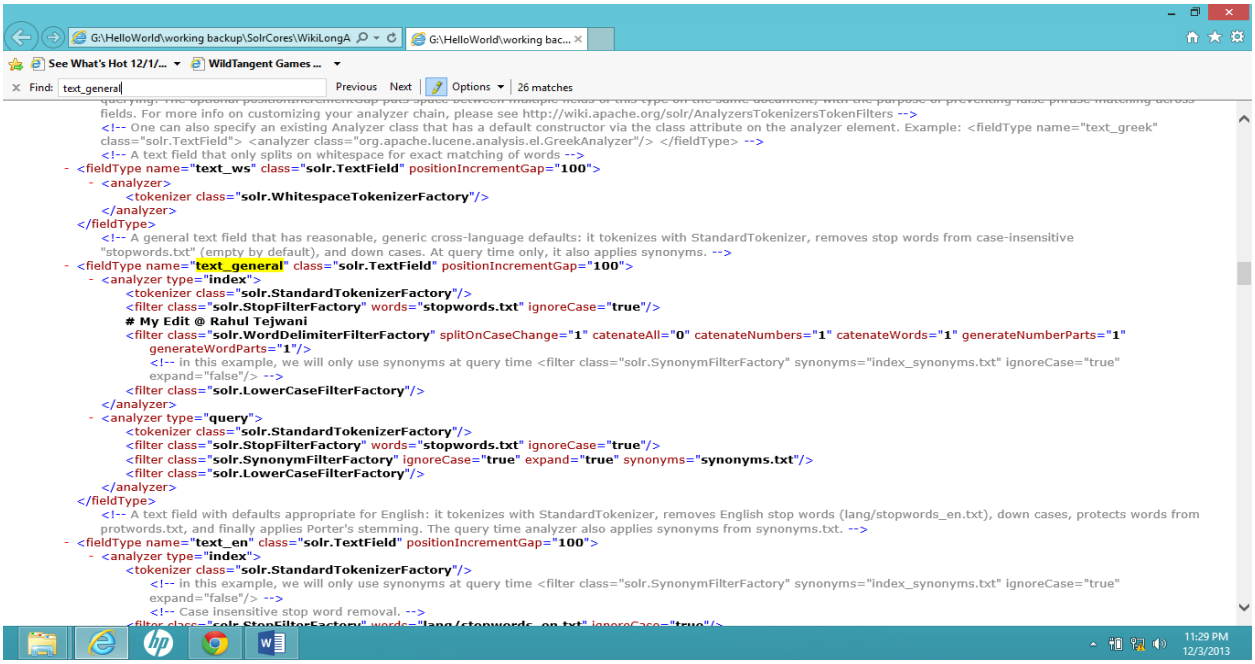
There are over 50 schema fields in WikiInfobox core. Some of which are as follows:

```
<field name="WikiName" multiValued="false" required="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiTitle" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiIndustry" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiFounder" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiFounded" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiEmployees" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiWebsite" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiShortDescription" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiTitle" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiKnownFor" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiProducts" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiRevenue" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiHeadquarters" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiType" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiBirthName" multiValued="true" stored="true" indexed="true" type="text_general" omitNorms="true"/>
<field name="WikiPlaceOfBirth" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiPlaceOfDeath" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiDateOfDeath" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiDateOfBirth" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiSpouce" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiChildren" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiParents" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiOccupation" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiOffice" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiNetWorth" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiDeathCause" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiNationality" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiEducation" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiSalary" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiHeight" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiWeight" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiAge" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiYearsActive" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiCapital" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiLanguage" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiCurrency" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiArea" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiPopulation" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiCurrency" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiArea" multiValued="true" stored="true" indexed="true" type="text_general"/>
<field name="WikiPopulation" multiValued="true" stored="true" indexed="true" type="text_general"/>
```

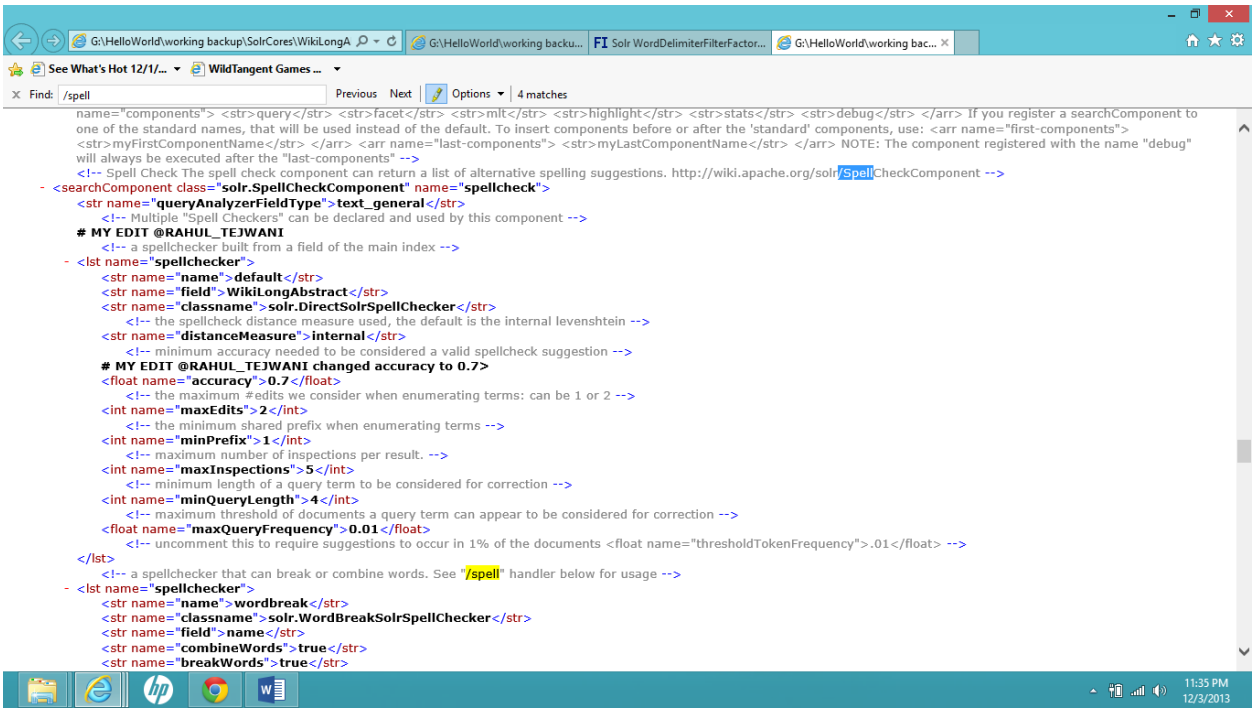
Filters and analyzer:

Added WordDelimiterFilter to distinguish between queries so that it doesn't split on letter-number transitions. It will provide results for queries like "wi fi" for "WI-FI".

We also configured Spell Check. Please find below the screenshots of solrconfig.xml and schema.xml files:



Solrconfig.xml



6. SOLR STATISTICS:

Infobox Index: 1.58 GB

Statistics

Last Modified:	about 19 hours ago
Num Docs:	2983227
Max Doc:	2983227
Deleted Docs:	0
Version:	5262
Segment Count:	1
Optimized:	✔
Current:	✔

Replication (Master)

	Version	Gen	Size
Master (Searching)	1385975722259	1826	1.58 GB
Master (Replicable)	1385975722259	1826	-

Short Abstract Index: 1.38 GB

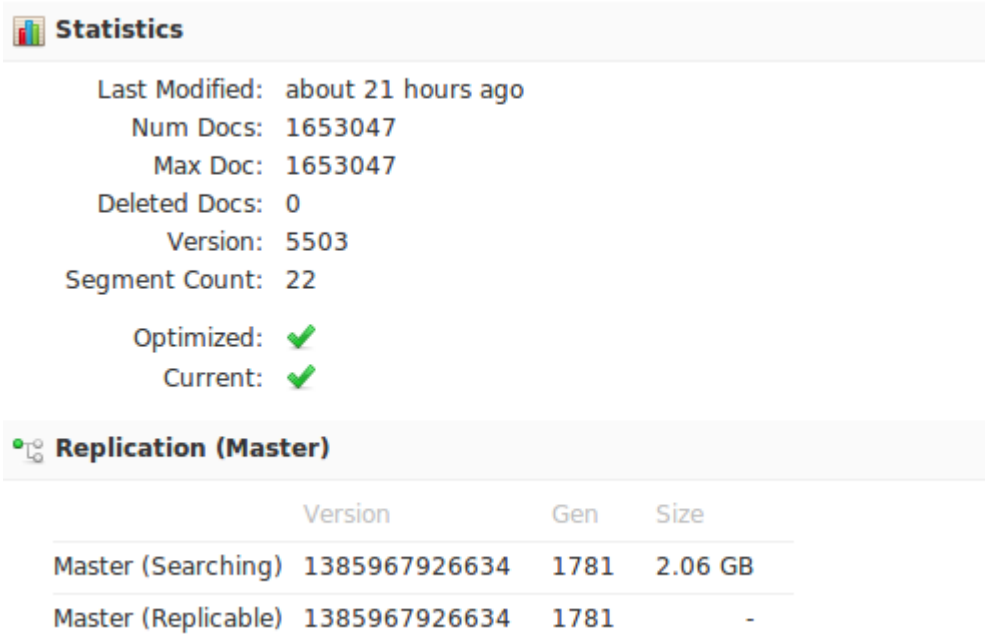
Statistics

Last Modified:	about 2 hours ago
Num Docs:	3980510
Max Doc:	3980510
Deleted Docs:	0
Version:	5178
Segment Count:	1
Optimized:	✔
Current:	✔

Replication (Master)

	Version	Gen	Size
Master (Searching)	1386036080599	1901	1.38 GB
Master (Replicable)	1386036080599	1901	-

Long Wiki Abstract Index: 2.06GB



Statistics

Last Modified:	about 21 hours ago
Num Docs:	1653047
Max Doc:	1653047
Deleted Docs:	0
Version:	5503
Segment Count:	22
Optimized:	✔
Current:	✔

Replication (Master)

	Version	Gen	Size
Master (Searching)	1385967926634	1781	2.06 GB
Master (Replicable)	1385967926634	1781	-

7. FUTURE WORK:

- Learning weights for each of the static score through machine learning techniques.
- Using Wikipedia Reference category as another parameter for calculating a page's static score.
- Finding results for complicated queries like "When was Sachin Tendulkar's wife born?" This query is complicated because the question asked is about a second person who is not mentioned in the query. One possible way of addressing these queries is through using two level topic detection.
- Implementing Answer analyzer module. As if now it only displays the raw answer.

8. MEMBER CONTRIBUTION:

Rahul Tejwani: SOLR, GUI

Sneha Banerjee: SOLR, GUI

Puneet Singh: Question Analyzer, Data Cleaning.

Prateeksha Mudgal: Question Analyzer, Data Cleaning.