Expert Q&A System Using Apache Solr

PROJECT 3 | TEAM ORANGEHUMMER

Angad Gadre, Alagappan Ramu, Palaniappan Meiyappan & Vinoth Selvaraju

CSE 535 | December 2, 2013

Table of Contents

1. Introduction

Question & Answer (Q&A) systems have become a significant part of on-going research in the area of Information retrieval. NIST has been holding Q&A tracks for TREC document collections leading to pushing of the envelope in developing efficient Q&A systems.

This report describes the implementation details of the Q&A search system we have developed as part of CSE 535 course (Fall 2013) on Information retrieval. The project report is divided into 4 sections.

The first section provides you with a high level description of the entire system. It lists what our system components are along with a system diagram. The system is based on processing and indexing a corpus of XML documents. We will describe how our document dump was processed and indexed using Solr. Then we will describe our query processing module, in relation to the Solr framework. We will discuss briefly how the Solr features have been implemented for our project.

The second section will delve into the configuration details and schema description we have used within Solr. Also, we will discuss features we have implanted to make our system more efficient and robust. These features, although may not be unique, make our system different. We will briefly describe their utility in the face of developing the Q&A system.

The third section contains details of the Solr statistics collected from the present production system. This section will give you an idea of how the system is utilizing Solr resources.

Section 4 contains our user interface (UI) design and sample use illustrations. The last section will briefly discuss further avenues of work and member contributions to the project.

2. System Overview

We have built a Q&A system which would give the user the facility to pose questions on personalities, places and films documented on the Wikipedia (English) website. As the first step in developing such a system, we provide the user with options in terms of lists (later illustrated in UI screenshots, refer section 3) to select the subject, object and verb of the question he/she needs answered. The user would be able to specify his area of search among the three categories and the type of information he/she needs from that topic.

We have developed a Q&A system with the properties discussed below, addressing some of the challenges and issues that contemporary Q&A systems face.

Note, the properties here are the features of the OrangeHummer Q&A system. The Solr features implemented have been discussed in section 3.

2.1. CLOSED DOMAIN QUESTIONS

Closed-domain question answering deals with questions under a specific domain (in our case, the domains - Persons, Places and Films). Alternatively, closed-domain might refer to a situation where only a limited type of questions are accepted, such as questions asking for descriptive rather than procedural information. Our Q&A project model includes both these aspects of closed domain questions.

For this purpose, we have divided the index into three logical partitions giving appropriate field types in the Solr schema.

2.2. INTERACTIVE Q&A

It is often the case that the information need is not well captured by a QA system, as the question processing part may fail to classify properly the question or the information needed for extracting and generating the answer is not easily retrieved. In such cases, we have employed the below discussed strategies to mitigate this inconvenience.

- Spell Check

  In order to implement this feature, we tweaked the solrconfig.xml file to implement suggestions based on the input string. When a query results in no documents being retrieved, we use the entered term for a similarity based retrieval. Note that, we do not pass the term to retrieve the most popular terms (within the same field, for example – name); that has been implemented for the auto-suggest feature. It is pertinent to mention that we used the 'indexbasedspellchecker'.

- Auto-suggest/auto-fill

  This is a feature which has become ubiquitous in search systems over the web. The advantage of this provision is that even if the user does not have complete information about the subject he/she wants to query, he/she still succeeds on the task of getting

the required. We provide the user with multiple options based on what is typed. The suggested options are based on the popularity of the search term starting with the typed character. The 'suggest' handler implement this by retrieving items indexed in the relevant schema field. Thus, allowing the user convenient way to question our system.

- More Like this

We have implemented this feature using 'Morelikethishandler' to allow us to search within schema fields parameterized as 'termVectors' = "true". When a user queries the system with a question, say about a person who is an actor, we would use the same question predicate for similar persons (i.e. actors) and allow the user to further query. This is useful when the user is using our system for exploratory purposes.

2.3. FACETED SEARCH

Faceting is a useful feature for users who have incomplete knowledge of the subject they want to query about. The user is allowed to select different characteristics of the subject. In our system, this would allow the user to drill down on the person's occupation, a place's state territory or the film's director. Concretely, this has been implemented using Solr's 'filterQuery' parameterized by the facets indexed.

2.4. QUESTION CLASSES

In our system, question classes try to classify the question type in terms of what specific part of document we are trying to access. For example: What is Rajnikanth's birthdate? Where was Rajnikanth born? The difference between the two questions lies in the question class and the information need from the same document. Since, we are looking at getting exact answers to questions, not a list of possible answers, identifying the question class is important. We have done this by providing limited list of field predicates.

2.5. ANSWER FORMULATION

The result of our QA system would be presented in a way as natural as possible. In some cases, simple extraction is sufficient. For example, when the question classification indicates that the answer type is a name (of a person), a place (city, country etc.) or a film (known film name) the extraction of a single datum is sufficient. The answers to questions have been presented in a readable form by identifying the question type and mapping the answer sentence.

2.6. REAL TIME QUESTION ANSWERING

Since, the Q&A system represents one where the users would expect answers to questions they would want answered for the purposes of further questioning, our

system will present answers to questions in the order of milli-seconds irrespective of the question complexity and ambiguity.

2.7. TRIVIA

Our Q&A system additionally provides trivia to the user in addition to the answer. Providing trivia provides an aspect of enriching the experience while using the system. For exploratory users, it becomes a helpful to tool to understand more about the answer to the question the user poses. For this purpose, when the category queried upon is movies, the system provides movie posters

Concretely, our system will contain components illustrated in the diagram below.

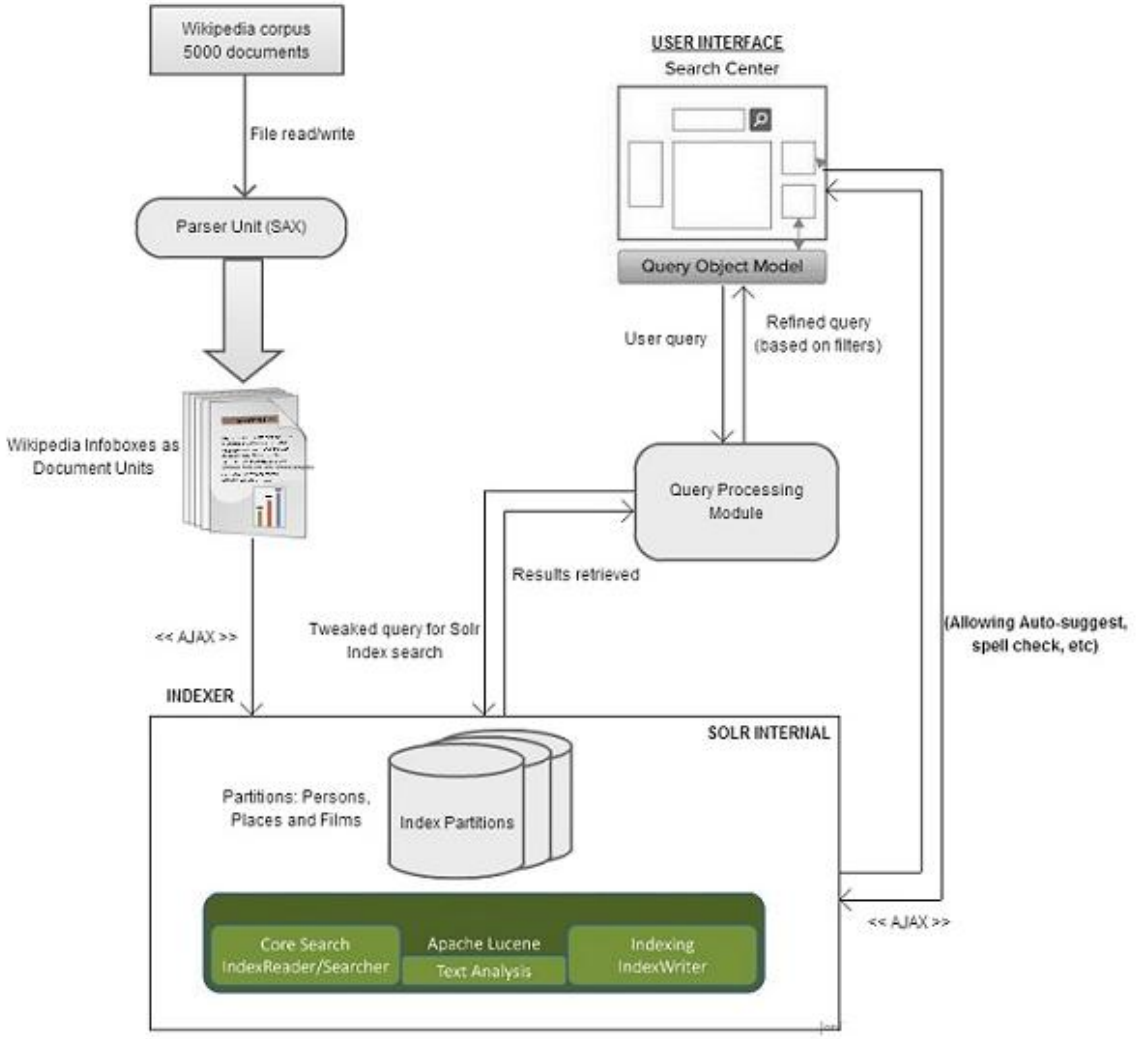

Figure 1: System Diagram

Each module present in the above diagram has been described in section 3. We have given a walk-through of all the components in our system. Below also contains the challenges faced during development of these modules and the ways in which we dealt with them.

3. Configuration Details

We have modified Solr to configure our system. For most parts we have used the schema.xml and solrconfig.xml files to configure our settings. Refer to the project code for more details. For the purpose of indexing documents, we have used a single index housed on the single core. This index will be type-differentiated into three, to house our three categories of interest using Solr's inherent implementation.

Schema fields used for the 'persons' category are as follows. Most of the field types are self-explanatory.

Person	**Movie**	**Place**
type	releaseddate	settlementtype
name	director	nativename
birthname	producer	nickname
education	writer	officialname
birthplace	screenplay	leader
deathplace	story	state
occupation	basedon	county
awards	narrator	latd
religion	starring	latm
parents	music	lats
deathdate	cinematography	latns
partner	editor	longd
spouse	studio	longm
yearsactive	distributor	longs
website	runtime	longew
residence	country	coordinates
knownfor	language	founder
nationality	budget	area
birthdate	gross	population
children		populationdensity
denomination		timezone
networth		postalcode
othernames		establisheddate
family		motto
salary		
alias		
employer		
ethnicity		
siblings		
citizenship		

For the fields listed above we have used the same Lucene analyzers for index and query. The fields have been treated with the following analyzers/filters.

- StandardTokenizerFactory
- StopFilterFactory
- SynonymFilterFactory – On the query side only
- LowerCaseFilterFactory

To implement features discussed in section 2 within our system, we have configured the 'solrconfig.xml' for the following handlers:

- 'select' handler (spell check)
- 'suggest' handler (auto-suggest)
- 'mergeFactor' = 2 (To increase search speed)

Having made use of the above configuration within Solr, we proceed to describe our learnings while we developed the various modules of the Q&A system. In order to understand the flow of the development, please refer to figure 1. The following section describes each process.

PARSER MODULE

1) We have used the Wikipedia (English) documents hosted by Wikimedia foundation to create our repository. In particular, we will be indexing 5000 documents containing content from our domains of interest, namely Persons, Places and Films. Within these documents, we will extract content from Infoboxes using SAX parser supported by the java language. Therefore, we have defined an Infobox as our unit of document for indexing within Solr.
2) These infoboxes, although a permanent structure in all Wikipedia pages, pose a not so straight forward way to extract content. Some of the challenges faced in parsing content from the infoboxes are the following.
   a) The infoboxes use a pipe (|) operator to separate different tags within. However, the content before and after these separators in not consistent. We formatted each infobox prior to parsing for consistency.
   b) We have used an external API for HTML markup removal, however, we had to tweak it so that it would identify some of the fields (such as birth date), which it was ignoring.
3) Also, we had to standardize the date format as 'yyyy-mm-dd' for different variations, 'spouse', URL of person websites, empty boxes, and occupation required special processing.
4) While parsing, we are also taking care of converting plural forms to singular without stemming. In order to do this we created a mapping file for infobox fields needed to be added to the schema, and applied the mapping at the time of creating 'infobox_person_livingpeople.xml' (to index on Solr)

5) In addition to the infoboxes, we also parsed out the first paragraph of the body of each Wikipedia page. This will be tagged using the XML tag 'summary'. At the time of answer presentation we will use this summary to provide additional information about the subject in question as a static summary.

QUERY PROCESSING MODULE

The QP module performs the function of the brain in our system. It has been developed to implement various features of the Q&A system.

1) Query formation

First, it would assimilate the query posted by the user and convert it into coherent form to search on the Solr index. This would involve converting the question predicate (Where, why, when, etc) and the information field (birthdate, birthplace, etc). For our baseline system, where we have specified a list of questions the user could select using a drop-down list, we created a mapping of question and its semantic content (for searching). Such as for the question – '**Where** was Bill Gates born?' the QP reduces the search terms in the query to 'Bill Gates birthplace'. While on the other hand for the query '**When** was Bill Gates born?', the QP would process it as 'Bill Gates birthdate'.

- 'QueryMapper.java' class performs this task.

2) Similar questions

We have implemented 'similar questions' feature. Here, we will provide the user similar searched questions based on the subject's features. For instance, if a person is searched for, who is an actor, we will provide a list of 'similar questions' on other 'persons'. This list of persons would be based on the availability of answers to the original question for these persons.

- 'QueryMltVerExp.java' class performs this task. Essentially, this class identifies similar persons by expanding on the token vertically in the same field; hence named as multiple vertical expansion of the search term.

3) Faceting

Faceting is a useful feature for the user when he/she has incomplete idea about the question. We have implemented faceting by identifying related fields in the category which might help the user develop a more concrete question. We have used faceting on 'occupation' for persons, 'directors' on films and 'state' on places. So, for instance if the user is trying to find out the birthdate of a person who is also an actor, and does not remember his/her name, he/she could drill down on the occupation facet and narrow down the names which will be suggested while searching using the interface.

- This task has been handled by the 'QueryFacet.java' class.

USER INTERFACE MODULE

1) We have used the dJango web framework to develop the front end system for the user interface. Thereby, using Python to program the web server application. The client side scripting has been done using Javascript/jQuery. We have borrowed the Twitter bootstrap for designing the functionalities and interface.

2) Question formulation
To filter out selection options (that is 'When', 'Where' would be supported for persons and film category, not place) based on the selected category, javascript is used with the 'filter' option.

3) Auto-suggest
For the purpose of implementing the auto suggest feature, we have made use of twitter API - Twitter 'typehead.js'

4) Location information
The UI module has also used the google geocoding API and google maps API to display map based location descriptions of the answer, in case of the query type selected as 'Places' by the user. This has been implemented by sending javascript requests to the API through the standard interface.

5) Movie information
On similar lines, we have used the OMDB API to retrieve the movie poster of the searched movie. The UI module handles API calls to OMDB (Open Movie Database) in javascript and displays the poster with the answer.

4. Solr Statistics

Solr provides a comprehensive view of the statistics based on the index which has been uploaded. Below we have displayed screenshots of Solr admin, index size, performance and hits/queries.

Figure 2: Solr Admin Dashboard

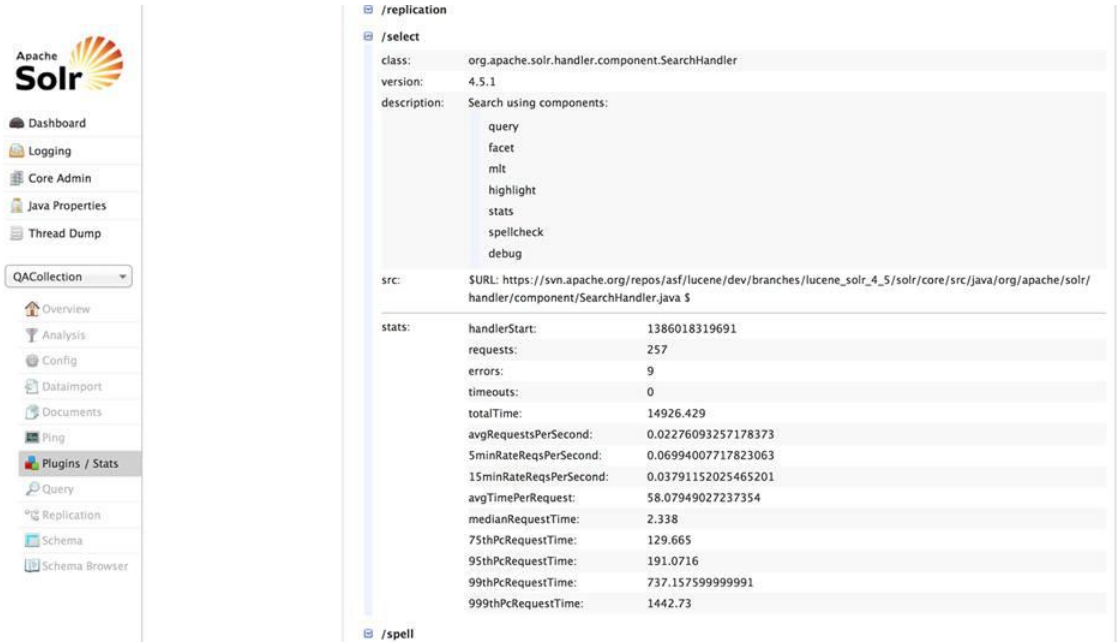

Figure 3: Select handler stats

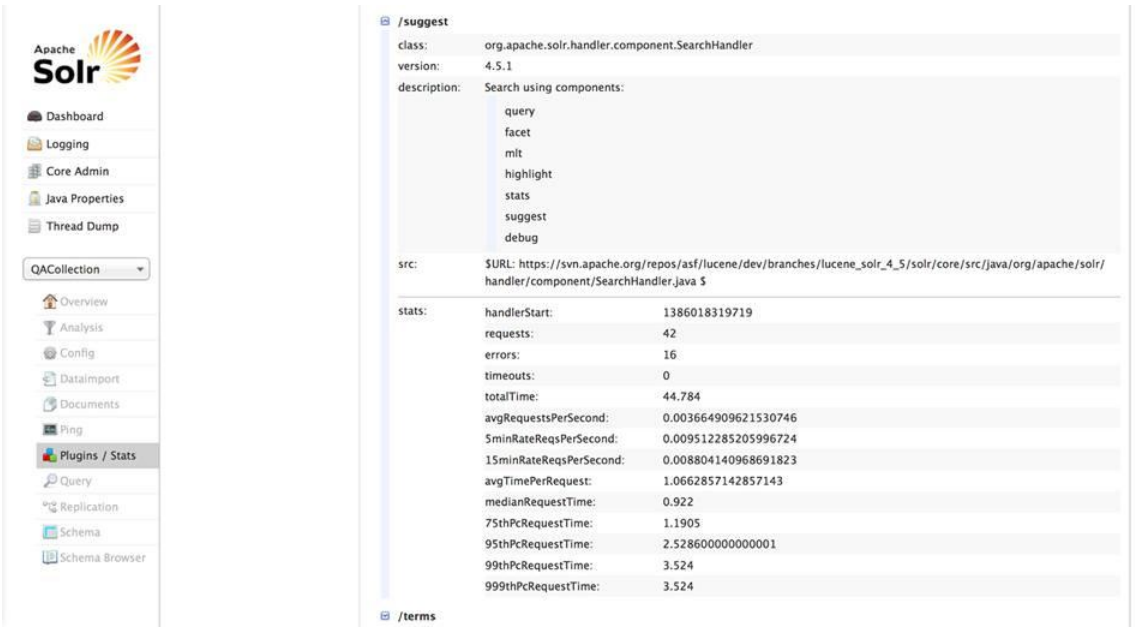

Figure 4: Suggest handler stats

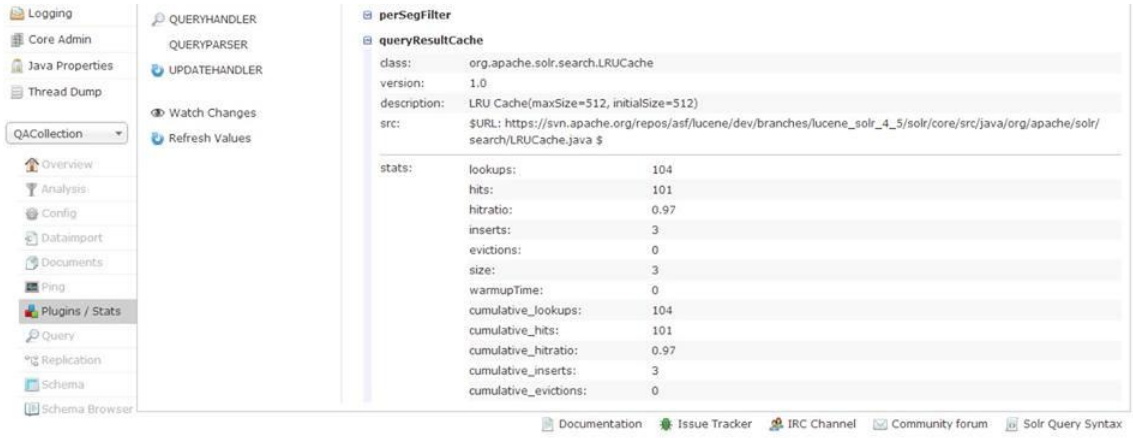

Figure 5: Displaying hits and other cache stats

5. Search Interface (UI)

The Q&A system has been hosted on the django webserver.

Below are some of the user interface screenshots from the OrangeHummer website. Each user screen during the process of question and answer has been illustrated.

Illustration 1: HOMEPAGE

The Q&A system's opening page. Here, the user has the option of selecting in which domain he/she wants questions answered (refer section 1 for more detail regarding closed domain search).

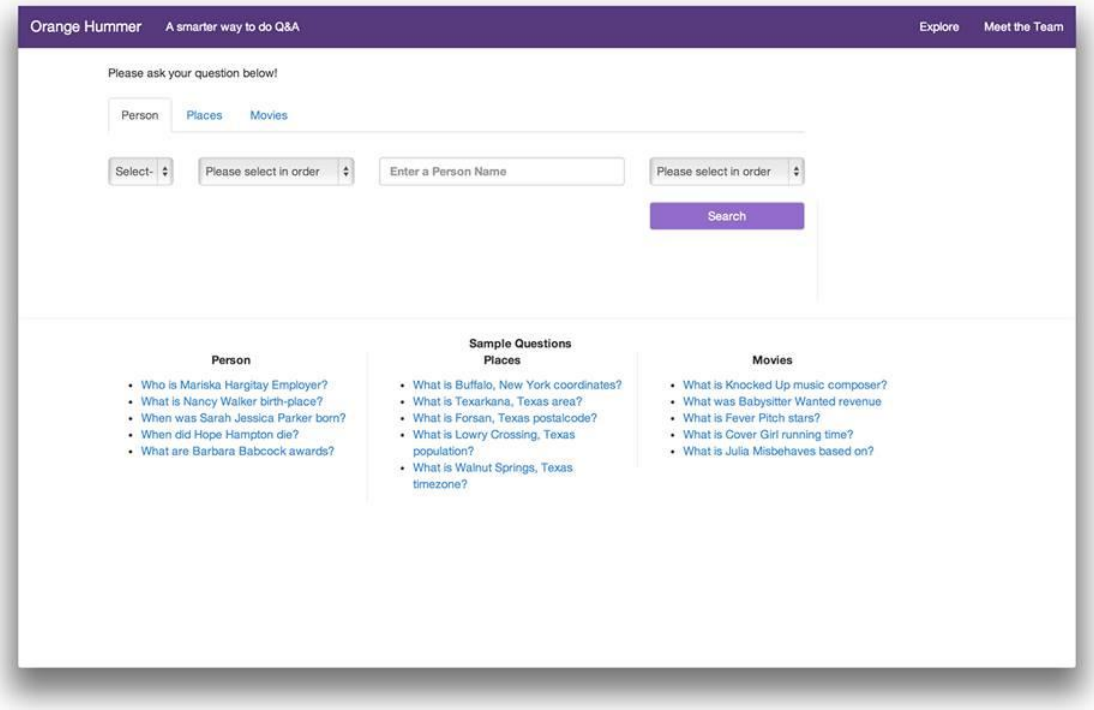

Figure 6: Q&A Opening webpage

Illustration 2: FEATURING AUTO-SUGGEST

The user is provided with drop-down lists of auto-suggest terms in relation to the subject, object and verb of the question to be answered. There will be a fixed number of question and information predicates (that is, Why, where, how, when, who and so on for the question. 'Born', 'birthdate', 'located', and so on for the information)

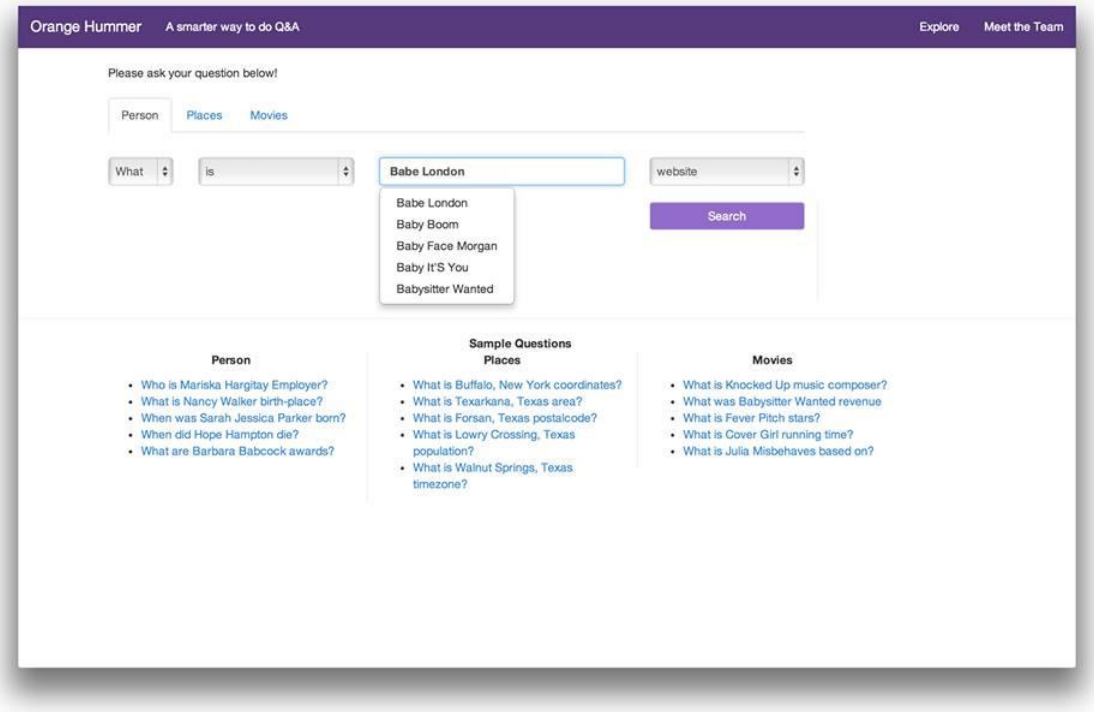

Figure 7: Displaying Auto-suggest feature

Illustration 3: RESULTS

Upon putting forth a question, a result page is displayed with the answer to the requested question in addition to other trivia. Below we have shown screenshots of sample queries on all three categories

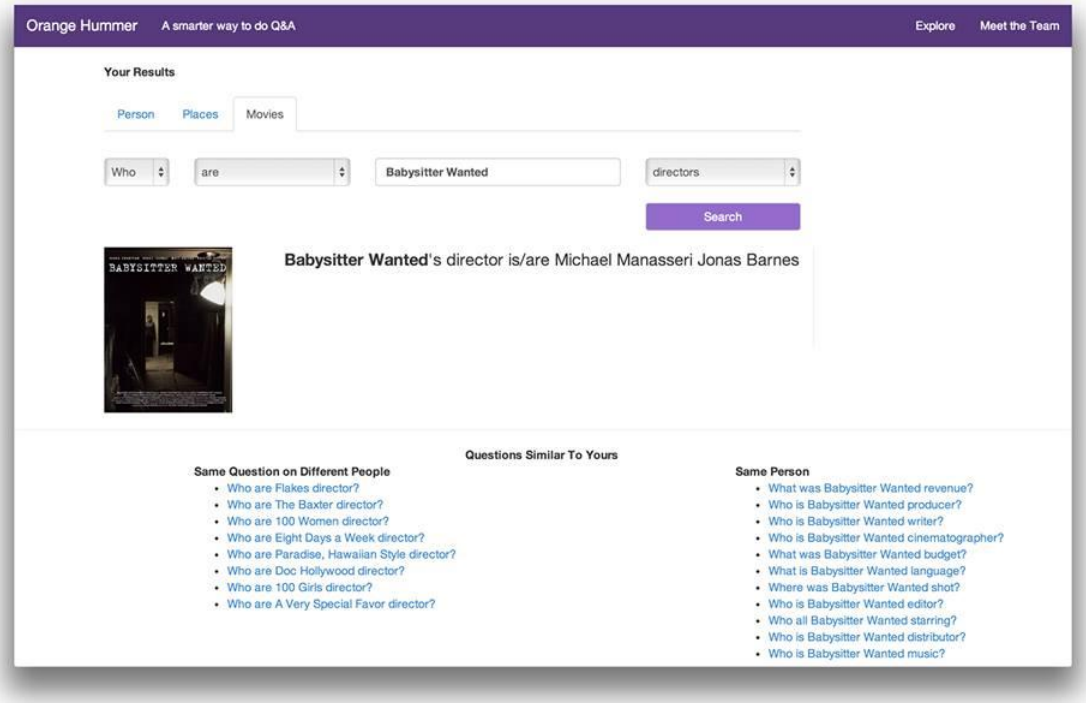

Figure 8: Sample Films Answer page

Figure 9: Sample Persons Answers page

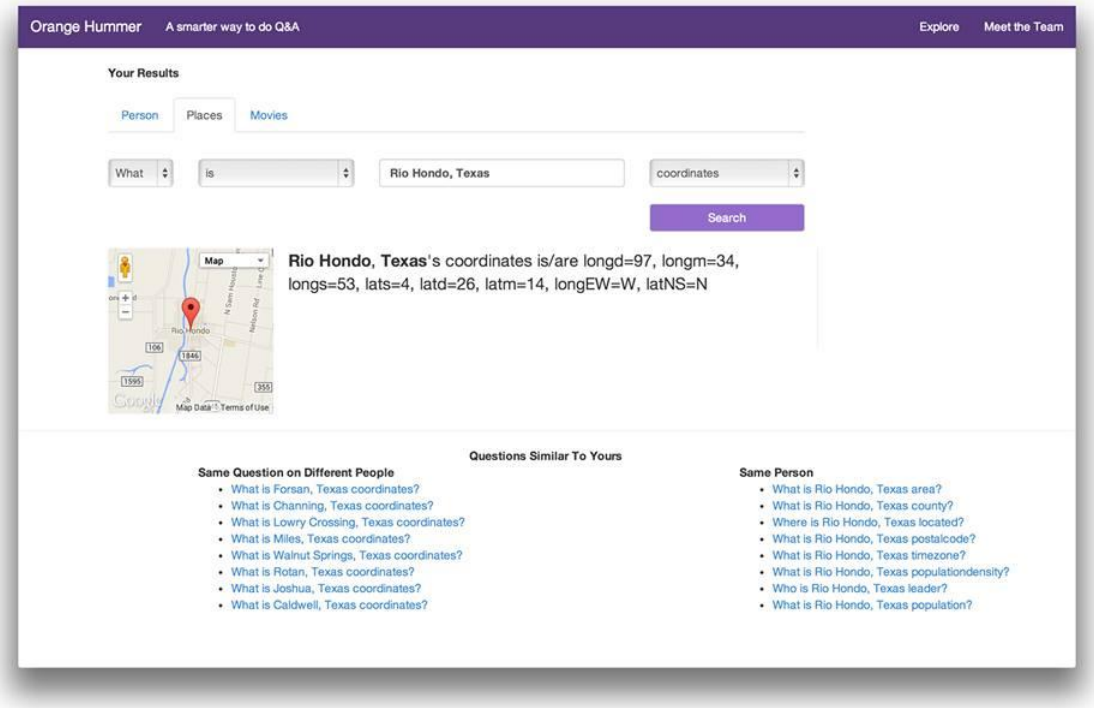

Figure 10: Sample Places Answer page

Illustration 4: SIMILAR QUESTIONS

Upon getting an answer to the question initially posted, the user may want to change or refine the question for further questioning. The results page provides capabilities of providing 'similar questions' and additional information about the subject of the question. The user will be allowed to conveniently select more questions on the same lines as he had earlier asked. Below you can observe the highlighted area where the similar questions will be displayed on the website.

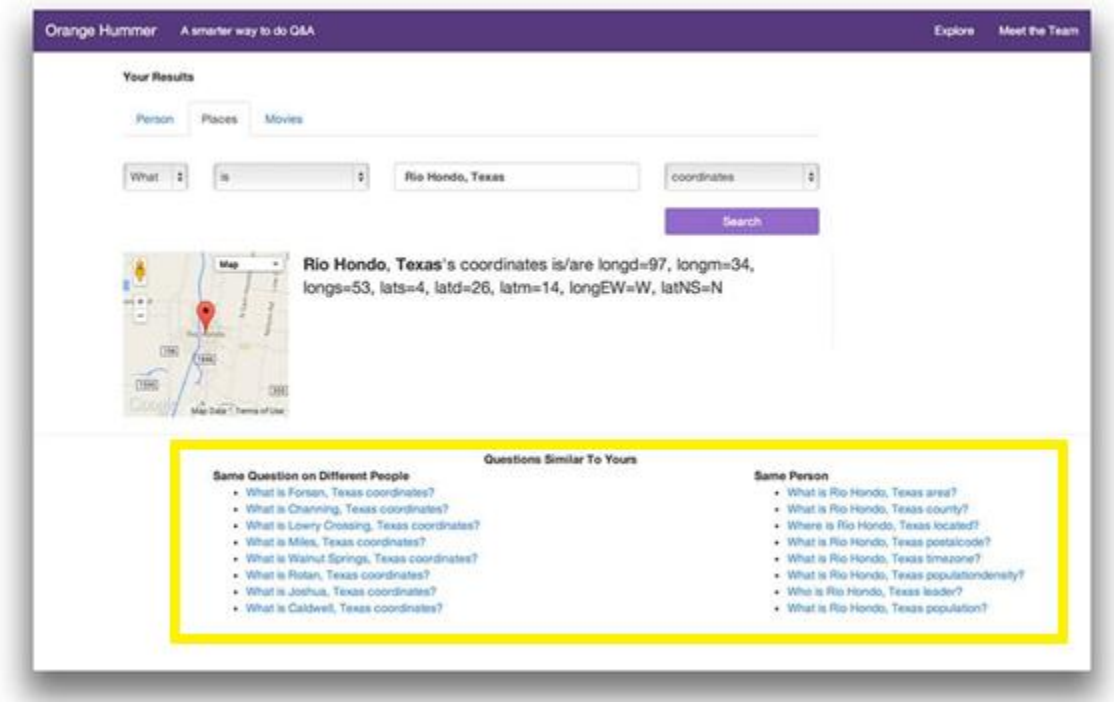

Figure 11: Featuring 'similar questions'

Illustration 5: EXPLORE

In addition to providing an easy to use question and answer interface, we will also provide faceting. This will be useful when the user has incomplete information on the topic he wants to query about. Here, each category can be drilled into as shown below. After selecting a topic, the user will be navigated back to the Q&A page.

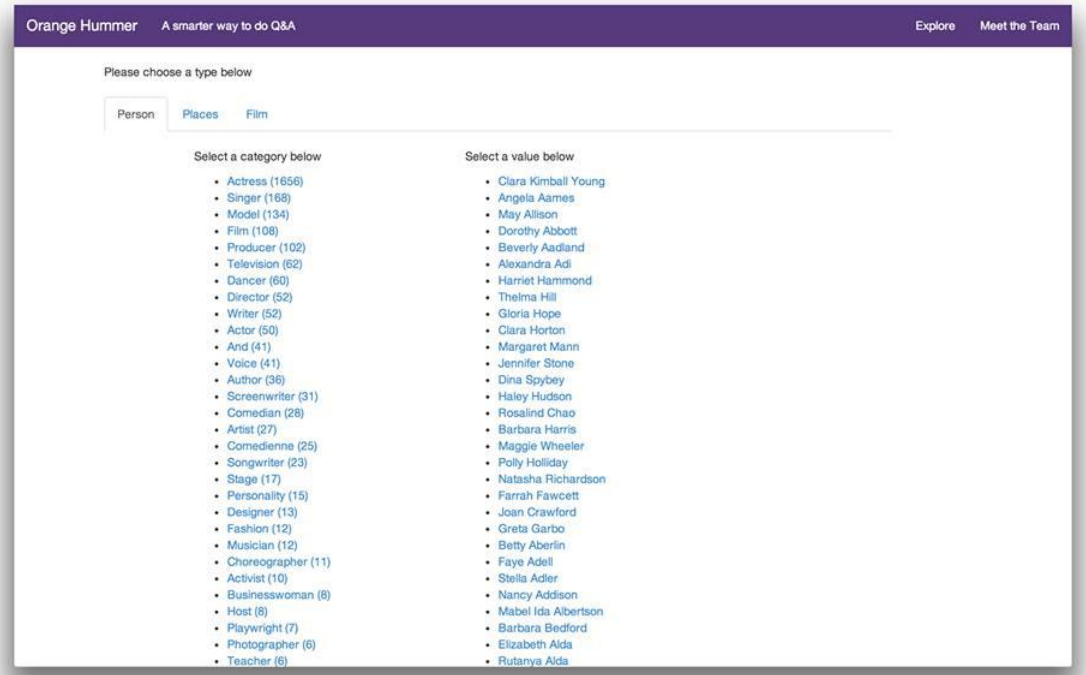

Figure 12: Faceting – Letting the user explore

6. Conclusions & Future work

This project leads us to identify numerous ways in which we can expand this system. Firstly, it would be more user friendly to allow for free text searches. In order to allow free text searching, we need to extract the noun-part (NP), which in most cases would be the name of the person, film or place. We could use OpenNLP to identify this (although tagging names and nouns with the help of OpenNLP is prone with errors). Additionally, we identify the following broad avenues in which we could consider future work.

- Providing further information

The Q&A system provides an interesting way to provide trivia and more information based on the search results. For a result retrieved as a single datum, by running this term on the corresponding index (if the datum is place, film or person, for our use-case) and providing the user with trivia about the answer.

To build on the thought of providing straight forward trivia about the answer, we could also provide the user with static summaries about the topic. This could be done by indexing the first paragraph of the Wikipedia pages parsed during extraction of infoboxes. These static summaries could be presented at the time of answer presentation.

- Improving Solr performance

We have used a static database to create an index and all the indexing has been done a-priori. Therefore, here we could optimize on parameters such as 'mergeFactor' and cache hits. By finding the optimum number of segments for merging, we could speed up our searching.

- Using Google API

The Google web search API provides the option to search a keyword in multiple areas such web search, News search, Blog search, Image search. We could use this API to provide answers to questions which do not have indexed documents. Furthermore, we could use this API to provide trivia about the subject in question.

- Provide movie trivia

When the question involves Films as the subject, the user could be provided with enhanced information about the film in question by getting information from The Movie Database API. In addition, we could suggest similar movies with the help of the genre field provided by this API.

- Synonym based expansion

We could expand the query by keeping the synonyms of words we think could be used to define the verb in the user question and search the index for those too. This could be

implemented by choosing the synonym filter for the query side. This would be pertinent only when we go on to implement a free text search.

The table below aptly describes our work distribution. The work has been distributed with mutual agreement. Work Distribution for this project is based on prior experience and interest of the teammates among the various modules of the Q&A system. However, some of the tasks have been shared amongst all the team members.

Member contributions

Name/Work Description →	Query processing	Parsing XML Dump	Solr Configuration	UI Design Implementation	Project Report Documentation
Palaniappan Meiyappan	✓		✓		
Alagappan Ramu			✓	✓	
Vinoth Selvaraju		✓	✓		
Angad Gadre			✓		✓

7. References

- http://en.wikipedia.org/wiki/Question_answering
- https://cwiki.apache.org/confluence/display/solr/A+Step+Closer
- https://cwiki.apache.org/confluence/display/solr/Configuring+solrconfig.xml
- http://www.solrtutorial.com/solr-in-5-minutes.html
- http://stackoverflow.com/
- http://wiki.apache.org/solr/SpellCheckComponent
- http://mappings.dbpedia.org/server/ontology/classes/Film
- http://wiki.apache.org/solr/SolrPerformanceFactors
- http://java.dzone.com/news/solr-optimization-%E2%80%93-document
- http://www.solrtutorial.com/schema-xml.html
- http://wiki.apache.org/solr/MultipleIndexes