 Palaniappan Meiyappan, Alagappan Ramu, Vinoth Selvaraju & Angad Gadre

CSE 535 | December 2, 2013

Expert Q&A System using Apache Solr

PROJECT 3 | TEAM OrangeHUMMER



Table of Contents

[1. Introduction 2](#_Toc373581528)

[2. System Overview 3](#_Toc373581529)

[3. Configuration Details 7](#_Toc373581530)

[4. Solr Statistics 8](#_Toc373581531)

[5. Search Interface (UI) 9](#_Toc373581532)

[6. Future work 13](#_Toc373581533)

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 design and implementation of the Q&A search system we are implementing as a 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 list 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.

1. System Overview

We are building a Q&A search system which would give the user the facility to pose question on personalities, places and films documented on the Wikipedia (English) website. As the first step in developing such a system, we will be providing the user 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.

* 1. Closed domain Questions – based on Persons, Places and Films

Closed-domain question answering deals with questions under a specific domain (in our case, the domains mentioned above). 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 include 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. The snippet from the Solr schema will be described in section 3.

* 1. Interactive QA

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 two strategies to mitigate this inconvenience.

* Spell Check

Here we suggest ‘similar searched question’ lists. We have made use of Solr spell check

* Auto-suggest/auto-fill
* More Like this

We have also implemented this feature using a ‘More like this’ similar searches. 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.

* 1. 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.

* 1. 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.

**Check this** For other cases, the presentation of the answer required the use of fusion techniques that combine the partial answers from multiple documents.

* 1. 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 seconds irrespective of the question complexity and ambiguity. **Need to check what is the answering speed.**

* **Check if we are implementing this. Would be cool to show as an add-on. For example, for a question – ‘Where was Rajnikanth born?’ we could, in addition to displaying the pertinent answer, display films (infoboxes should mention Rajnikanth’s presence within them) in which he has acted, or/and display trivia of his birthplace (content present in the infobox where Rajnikanth was born).**

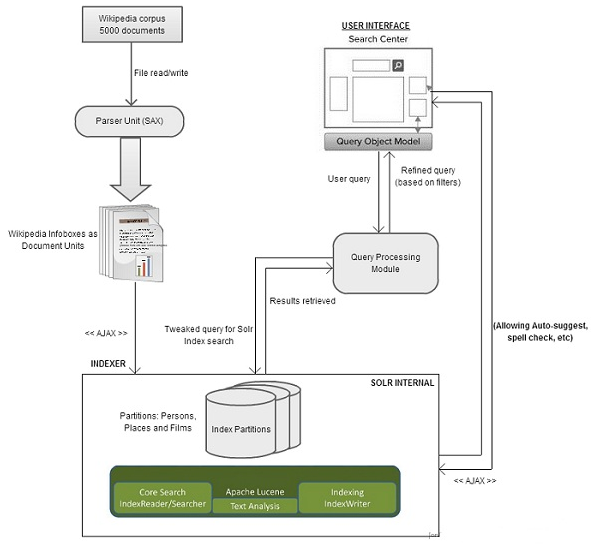
Concretely, our system will contain components illustrated in the diagram below.

Figure 1

Each module present in the above diagram has been described in the following section. 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.

# Configuration Details

We have tweaked and 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. Schema fields used for the ‘persons’ category are as follows. Most of the field types are self-explanatory.

|  |  |  |
| --- | --- | --- |
| **Person** | **Films** | **Places** |
| 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 analyzer.

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

Having made use of the above configuration within Solr, we proceed to describing our learnings and processes 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.

1. Parsing Wikipedia pages

* 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.
* 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.
* 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.
* 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.
* 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.
* 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.

1. Solr Schema

* For the purpose of indexing our defined 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. We reason our use of a single index on the basis of our need of providing real-time search results. Also, it would allow us the flexibility to integrate additional features such as providing trivia in addition to the answer, displaying all information about the subject in question etc.
* Some of the fields we have chosen to employ in the Solr schema, based on the frequency of and importance of their content, are as follows. See section 3 for detailed schema design.
  + Person – Name, birthdate, birthplace, occupation, spouse, website
  + Place (tag: Settlement) – Sub-division\_type, sub-division\_name, population, population density, postal\_code
  + Film – Name, producer, director, music, writer, language, country

1. Query processing module

* The query processing (QP) module for our system will function for two purposes. 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). The second function would be to filter out selection options (that is ‘When’, ‘Where’ would be supported for persons and film category, not place) based on the selected category.
* For our baseline system, where we would be specifying 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 Rajnikanth born?’ the QP would reduce the search terms in the query to ‘Rajnikanth birthplace’. While on the other hand for the query ‘**When** was Rajnikanth born?’, the QP would process it as ‘Rajnikanth birthdate’.
* We have implemented ‘similar questions’ feature. Here, we will provide the user similar searched questions based on the subject’s features. Say, a person is searched for who is an actor, we will provide ‘similar questions’ for similar questions on other actors.

1. User Interface

* 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.
* For the purpose of implementing the auto suggest feature, we have made use of Twitter Typehead.js

Additional subsystems (may be implemented)

* The user may use this system to search for navigational and exploratory purposes to round up answers in a particular domain. Here, we have provided the user with ‘similar questions’ to the one asked by developing a system where the search is implemented for similar persons – based on occupation, places – based on population or films – based on director.

We foresee the use of the Solr features described below.

* **NOT REQUIRED** Spell Check

This Solr feature will be used in the special case when users do not choose to use an auto-suggested term (for any of the question predicates). In this setting, we will run a spell check on the entered term (in addition to querying the user-entered term) and provide a ‘Did you mean this?’ list. If there exists no indexed content to the question asked by the user, it would be better to return a ‘Sorry, OrangeHummer Q&A has not been trained with this question!’ message, as the corpus in use limits the answering ability of the system.

* Changing the similarity model and analyzers

After processing the query, we can change the similarity model and look to optimize which model might work best. Also, it may be useful to apply different analyzers to retrieve more relevant results.

* Generating unique field

Use this feature to generate a unique id for each document we index. This would prove useful in cases of duplicate documents.

* **SEARCH FOR THIS, DON’T REMEMBER** Primary key field optimization

Since, most of the fields in our data will have differing primary keys, a search on one such primary field is often not as fast as compared to databases. We could use this Solr feature to increase speed of response.

* Plural to singular form without stemming

This Solr feature would be most useful for our Q&A system, where we would not want to lose information regarding the subject of the question (person, place, film name) posed by the user. Plurals of names of places, persons and films would also be required.

* Auto-suggest/auto-fill/auto-complete feature

In order to implement this feature, we tweaked the solrconfig.xml file to implement suggestions based on the input string. This input string is by default considered for a spell check routine. **Mention which amongst these was implemented – indexbasedspellchecker, filebasedspellchecker, directsolrspellchecker**

**Check if true -The user would be suggested the names, places or films which have been indexed in Solr by implementing this feature, thereby avoiding incorrect results. However, if the user does not opt to choose this suggestion, the results set if empty, prompts the QP module to search with the spell checked term.**

# Solr Statistics

Need to add screenshots of the Solr admin and other running stats.

Index size.

Stats on performance

Stats on caching – hits, queries

# Search Interface (UI)

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: 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).

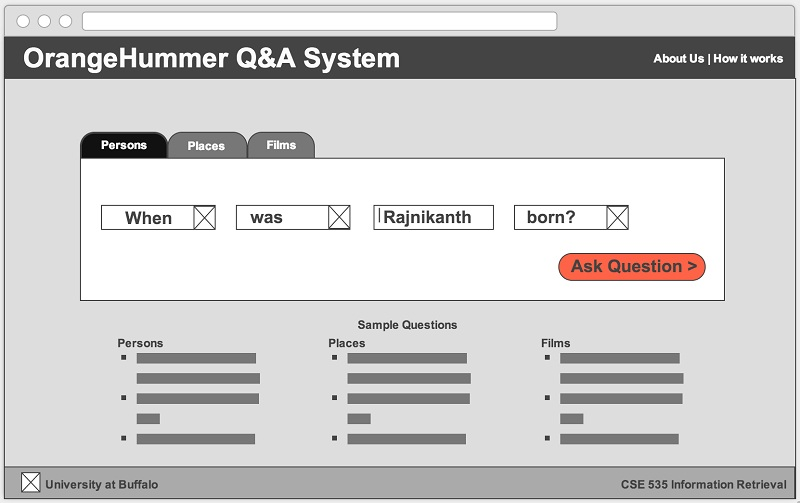
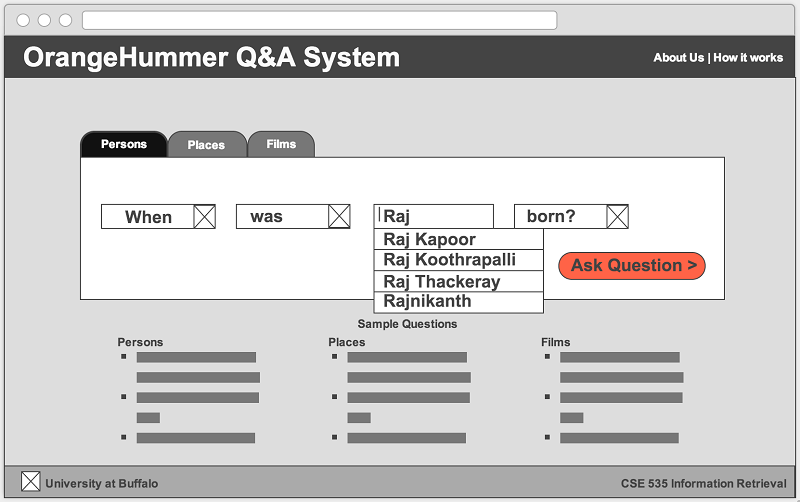


Illustration 2: The user will be provided with drop-down lists of auto-suggest terms in relation to the subject, object and verb of the question to be answered. There will be a fixed number of question and information predicates (that is, Why, where, how, when, who and so on for the question. ‘Born’, ‘birthdate’, ‘located’, and so on for the information)



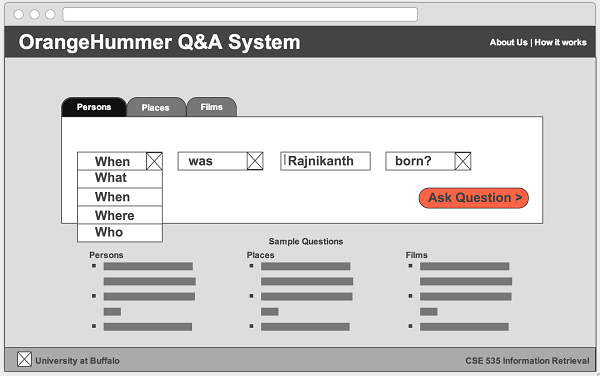


Illustration 3: Upon putting forth a question, a result page will be displayed with the answer to the requested question in addition to other features we might present. These may include – Photograph of person, poster of film, or location of the place on a map.

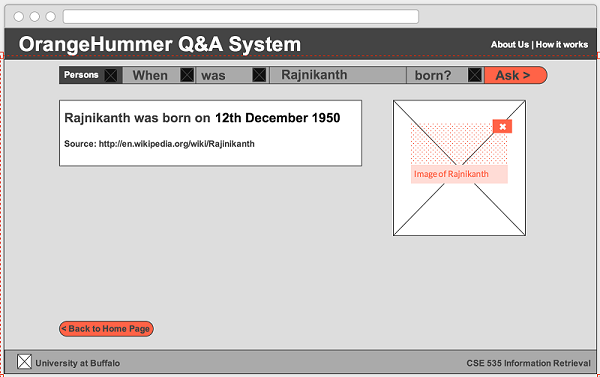


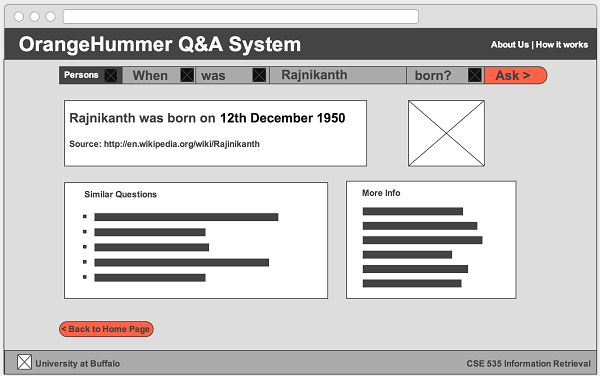
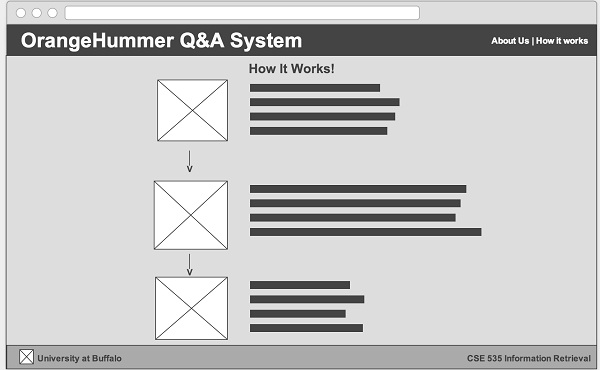
Illustration 4: Upon getting an answer to the question initially posted, the user may want to change or refine the question for further questioning. The results page will provide capabilities of providing ‘similar questions’ and additional information about the subject of the question. Also, question drop-lists will be provided on this page as well to allow for convenient querying.

Illustration 5: In addition to providing an easy to use question and answer interface, we will also provide instructions to the user in the ‘How it works’ section to instruct the user on how to use our system, what queries will the system support and how to obtain the most from the system.



# 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 using Solrconfig.xml
* 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.

* Location information

For questions relating to places, such as ‘What is the capital of New York?’, we intend to use the geospatial feature of Solr in conjunction with the Google Map API to locate the place on the map. Additionally, we could retrieve the user’s location and show the distance he/she is from the location in question.

* 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 and the team will be working cohesively. However, some of the tasks have been shared amongst all the team members. Work Distribution for this project is based on prior experience and interest of the teammates among the various modules of the Q&A system.

**Consensus on special mentions**

Member contributions

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

**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>