International Journal of Applied Science and Mathematics
Volume 4, Issue 2, ISSN (Online): 2394-2894

Query Based Cluster Searching using DBPEDIA – Knowledge Base

Ms. Kavitha A[(1)], Mr. Naveen Kumar G[(2)], Mr. Nijanthan V G[(3)], Mr. Satheesh Kumar R[(4)] and Mr. Santhosh Kumar [(5)]
Assistant Professor (1), Final year (2), (3), (4), (5)
Department of Information Technology (1), (2), (3), (4), (5)
Bannari Amman Institute of Technology (1), (2), (3), (4),(5)
Sathyamangalam. (1), (2), (3), (4), (5)
*Corresponding author email id: kavithaa @bitsathy.ac.in

Date of publication (dd/mm/yyyy): 11/03/2017

***Abstract* – The main objective of this project is to get the answer to the questions by the use of search engine by automatically, not manual whether questions are in word, pdf etc. We would find the answer key for a question one by one which is in the document and the user may check the answer by the use of search engine. By searching the answer a user will get only the URL and HTTP links on the screen. This process can take more time and unusual loss of manpower. To rectify this problem and to manage the time, the Cluster Searching application might be used. In this application, we would upload the question paper and click find button which is in application. It will check the answer key for a questions in a file automatically with time efficient and store the answer key automatically to your google drive and sent response to gmail. The application is mainly used to avoid showing irrelevant answers of the questions over the web pages and showing the straight forward answers to the questions and also used for time management for users.**

***Keywords* – Question, Serch Engine, Cluster Serching Application, DBPEDIA.**

I. INTRODUCTION

The answers of a question there are lots of search engine available. But the problem with the search engine is that instead of giving a straight forward answer they usually gives the links/URL to the webpages which might have the answers. Instead of providing links Cluster Searching will provide the straight forward answer to user's question.

Tasks intended by the project:

- Takes question as an input from the user.
- Analyses the sentiment behinds the question.
- Look over the information available in the knowledge base related to question.
- Compute the answer of the question from the knowledge base.
- Present the answer to the user if it is available.

Straight forward answer will be very useful and time efficient and it is really helpful for the users who are using small screen devices, since in those devices it is very hard to find answers in webpage with lots of irrelevant content.

This section talks about the need of a Query answering Application. It talks of how Query Answering can be used to get the answer of the question.

II. EXISTING SYSTEM

Finding the answer for a questions is done by manual. Questions will be check one by one in the document. Takes question as an input from the user. Analyses the sentence behinds the question. Look over the information available in the knowledge base related to question. Compute the answer of the question from the knowledge base present the answer to the user if it is available.

III. PROPOSED SYSTEM

Makes the process friendly and reduces the manual work. Every process is made on online. Just upload the file or a document through an application and then click search button manually to get answer links. In that, there is a option for number of questions to be search. This is used to give the limit to search the number of queries. Currently the Cluster Searching is taking text as an input and returns text as an output.

This could be enhanced by integrating voice library, so that it can able to answer by listening to the question. We can use any map reduce technologies in this system, so that the load can be distributed on multiple nodes. Furthermore the system is not able to answer the question of logical reasoning, like why something happened.

- We can enhance its functionality so that it could be able to answer those questions also.

IV. NEED OF QUERY ANSWERING APPLICATION

To get an answer of a question from the Internet, we have one search engine available. But the problem with most of the search engine is that they provide you the link of the web page where you can find the links with answers instead of exact answer. So, for this we need a new kind of system, which will give you the answers for such kind of question.

V. PROBLEM ADDRESSED

For some kind of question user needs the direct answers, instead of the links, which leads him to some webpage, which might have the answer somewhere in the body, or user need to compute the answer from the

information in the body, But this is not what user wants. Instead, the user wants a very interactive Query answering Application. So, sometime straight forward answer to user question is better than finding the answer in the some of the web site links, returned from search engine. It is generally the case that the information needed by the user is not well captured by the Query answering Application. So the system's questioner must reformulate a question and put a dialog "The system is not able to find the answer of this question, Try rephrasing the question or Type another question". Also, direct answer is even more useful if user is using small screen devices like cell- phone, tablet because it is hard to Fig out what you need on small screen. There needs to be some system which can understand your question, sentiments behind it and the according to this, it will look over the answer in knowledge base and give you the direct answer for the question.

V.1. *Theory of Query Answering Application*

This system will reads the input from the user, scan it, and parse it. After that the question is fed to Natural language processing software, which will tokenize the question in part of speech tag, like nouns, pronouns, verb, adjective, Wh-Determiner etc. By using the part of speech tags the system will try to understand the sentiments behind the question, and will try to come up with subject and predicate. These subject and predicate would be used as input in knowledge base open source software.

Once the system finds the object using the subject and predicate from the knowledge base it will use the Wh-Determiner to verify the answers using the class property obtained from the knowledge base. Like if the Wh-Determiner is 'Where' then the class label property must be place or any other class label from place hierarchy. If the Wh-Determiner is 'Who' then the class label property must be person or any other class label from person hierarchy. After verifying the class label with Wh-Determiner the system will show up the answer computed during the whole process. And if the answer is unavailable, the system will show a message the system is not able to find the answer.

V.2. *DBPEDIA-Knowledge Base*

DBPEDIA gathers its information from Wikipedia and store it in its mapping which is called as mapping wiki. And we can retrieve this information in any of the given specified format. Flow Chart of knowledge update of DBPEDIA is given as follows:

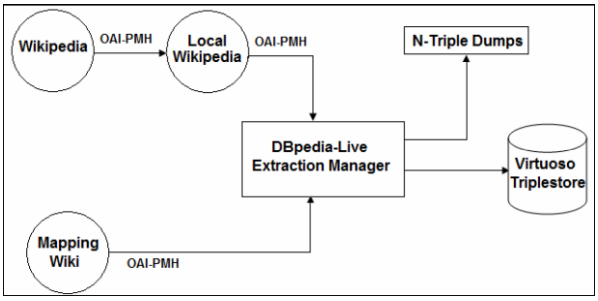


Fig. 1. Flow Chart of knowledge update of DBpedia

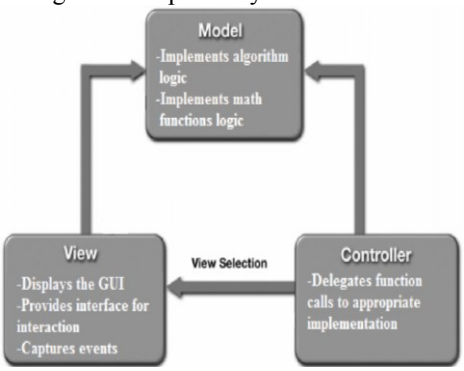
If we have more than one predicate, the system will try to look answer for either one of them. The sequence of looking the predicate in data structure is according to the pos tag of the predicate, some pos tag's predicate has higher priority than others. While looking for the answer in triplet the system will store the information in data structure for each and every predicate. If the answer is not found in the information provided from response of DBPedia.

VI. System Architecture

This section describes the detailed explanation of how Cluster Searching is implemented. The workflow and design pattern implemented in the system is also discussed in this section.

VI.1 *Technologies and Architecture Used*

The project is developed in Java platform and for User Interface JSP with JQuery and AJAX is used extensively. The project is designed using MVC (Model View Controller) architecture. The view generates the GUI (User Interface) which will interact with the user. Controller is used to manage the View and the Model, on the basis of user's input the controller will decide that which model should be called. Controller basically manages the workflow of the whole system. The model constituted the core algorithm and the logic of the system. It includes the RESTful calls to various APIs and transforming their output to system usable format.



MVC Architecture

VI.2 *Framework Design*

The whole system is designed in JAVA with JSP, JQuery and AJAX on front-end. Several RESTful web services are called at different phases of the answer evaluation. Apache Tomcat server is used as server to interact between client (browser) and backend. Backend is implemented in Java Servlet. To deal with various standardized format like XML, triplets, JSON various libraries are used like SAX parser, DOM parser, JSON simple etc.

VII. RESULTS

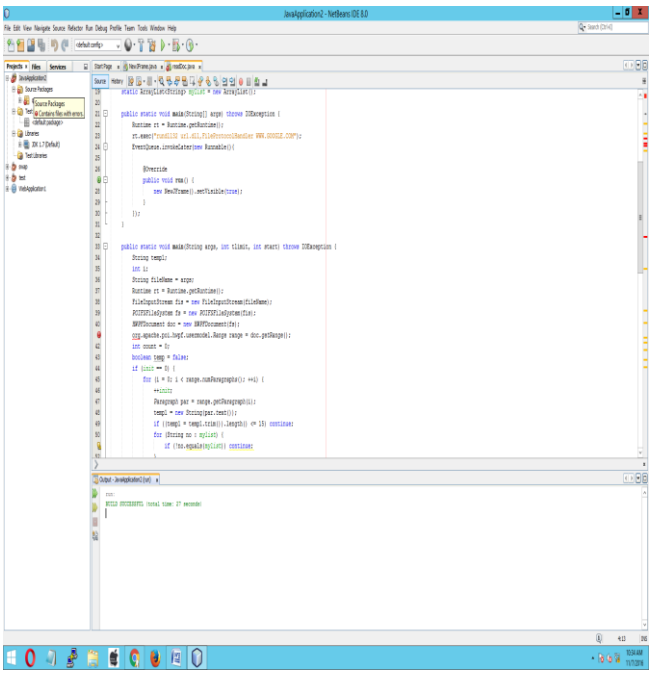


Fig. 2. NetBeans Source code page

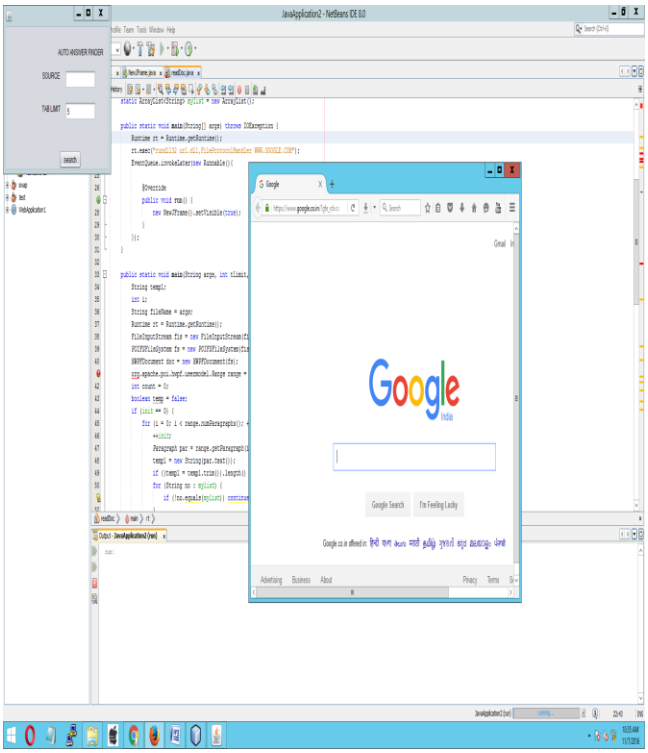


Fig. 3. Display of terminal page

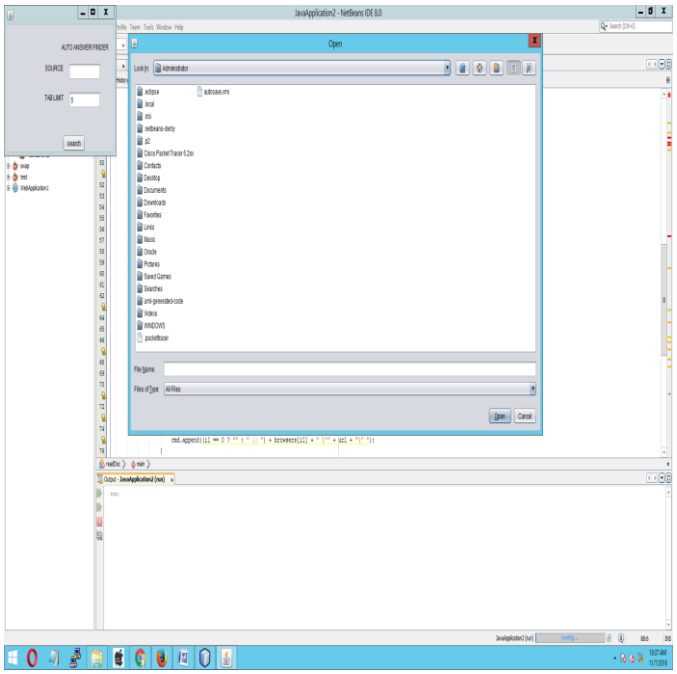


Fig. 4. Select the document for search

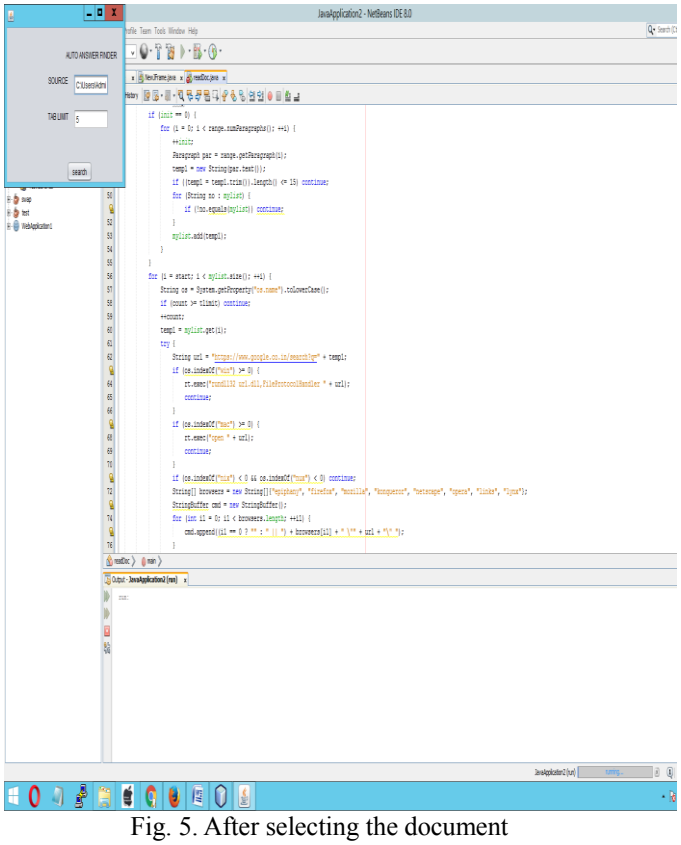


Fig. 5. After selecting the document

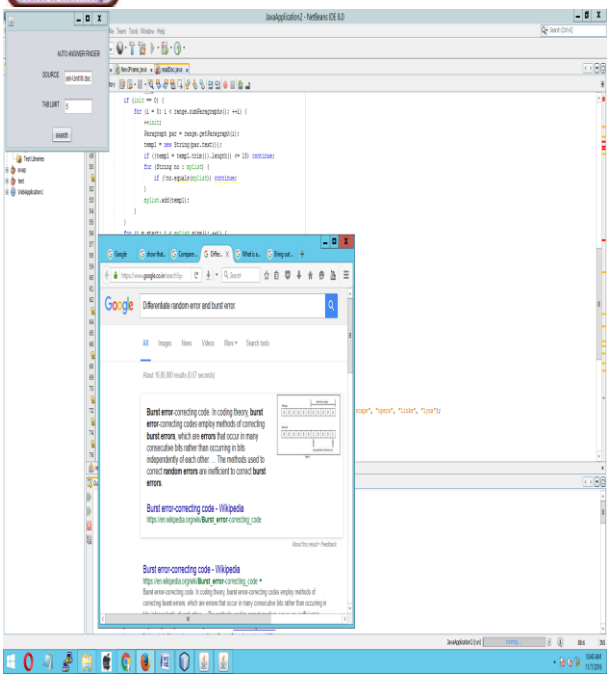


Fig. 6. Finding Answer using Default Search Engine

VII. Conclusion

In this project there was an implantation of Cluster Searching and to implement this system using open source software. This software is need to getting the straight forward answer to some particular question instead of getting the links/URLs which might have the desired information. This system will be helpful for the user which means that time efficiency. The processing is done on server side, so light weight client can also use this system very efficiently. The system's ability to answer variety of question by using various open source software is achieved in an efficient manner. The implementation of Cluster Searching is done using Model View Controller architecture which can be reused in further enhancements and expanding the knowledge base.

References

[1] https://www.ibm.com/developerworks/mydeveloperworks/InsideSyst
[2] http://opennlp.apache.org/documentation/1.5.2-incubating/manual/ opennlp.html
[3] http://nlp.stanford.edu/software/lex-parser.shtml
[4] http://www.monlp.com/2011/11/08/part-of-speech-tags/
[5] http://mappings.dbpedia.org/server/ontology/classes/
[6] http://mappings.dbpedia.org/server/ontology/classes/Place
[7] http://wiki.dbpedia.org/DBpediaLive
[8] http://dbpedia.org/lookup
[9] http://dbpedia.org/page/Chicago