Summer 2012

Question Answering System using Open Source Software

Gaurav Gangwal
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Gangwal, Gaurav, "Question Answering System using Open Source Software" (2012). *Master's Projects*. 258.

DOI: <https://doi.org/10.31979/etd.natg-n6nw>
https://scholarworks.sjsu.edu/etd_projects/258

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Question Answering System using Open Source Software

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of Requirements for the Degree

Master of Science

by

Gaurav Gangwal (007481253)

Spring 2012

Abstract

To get 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 Question Answering System will provide the straightforward answer to user's question.

Tasks intended by the project :

- Takes question as an input from the user.
- Analyses the sentiment behinds the question.
- Lookover 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.

Strightforward 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.

Table of Contents

1. Introduction	6
1.1 Need of Question Answering system	6
1.2 Problem Addressed	6
2. Theory	7
3. Literature Survey and Workflow of the system	8
3.1 Natural Language processing.....	9
3.1.1 Open NLP	9
3.1.2 Stanford Parser: A statistical parser	13
3.2 Look up API – knowledge base.....	14
3.2.1 Input and Output for the Lookup API	15
3.2.2 Class Label used by Lookup API	21
3.3 DBPedia – Knowledge base.....	32
3.2.4 Freebase – Knowledge Base.....	41
4 System Architecture	43
4.1 Technologies and Architecture used:	43
4.2 Framework Design:	44
4.3 Workflow of the System’s User Interface	45
5 Conclusion.....	45
6 The Future of Question Answering System.....	46
7 References.....	47
[1] IBM Watson Jr. – How to build your https://www.ibm.com/developerworks/mydeveloperworks/blogs/InsideSystemStorage/entry/ibm_watson_how_to_build_your_own_watson_jr_in_your_baselement7?lang=en	47
[2] Apache OpenNLP http://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html	47

List of Figures

Figure 1 Flow Chart of knowledge update of DBpedia.....	32
Figure 2 : Output of DBPedia in browser.....	34
Figure 3 : MVC Architecture.....	44

Acknowledgement

Through this acknowledgement, I am expressing my gratitude to all the people who have helped me and are associated with this project and have made this a worthwhile experience.

The project is made under the guidance of **Dr. Chris Tseng**. Firstly, I thank **Professor Dr. Chris Tseng and Computer Science Department**, who gave me this opportunity to learn new things and implement them in my Master's Project on **"Question Answering System using Open Source Software"**. His continuous guidance with practical approach and valuable suggestions regarding the project helped me in visualize the project design. Without his guidance and efforts it wouldn't have been possible to come up with this project so effectively and efficiently. His kind cooperation helped me complete my project work.

I also sincerely express my thanks to my parents and friends who have shared their opinions as well as experience with me. Their support was with me throughout. I have worked hard and tried my best efforts and to complete this project and make this report an informative one.

1. Introduction

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

1.1 Need of Question Answering system

To get an answer of a question from the Internet, we have lots of 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. For example, if the user wants to search for birthplace of Barack Obama, or the user wants to know the number of employees in IBM, then in that case, the accurate answer would be more useful rather than web page links, which might have desired information. So, for this we need a new kind of system, which will give you the answers for such kind of question.

1.2 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 question answering system. So, sometime straightforward 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 question answering system. At times the question processing part may fail to understand the question properly or other times the information needed for generating the answer is not received very easily. 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 figure 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.

2. Theory

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. The system will fed subject as an input in knowledge base and retrieve the information in output. The open source knowledge base software used in this system will produce the output in standardized format (XML, triplet and JSON). On the basis of output, the system will parse the information about the noun, like some

knowledge base will provide information about category, class hierarchy structure of the noun that whether it is a *person*, *place*, *organization* etc. and some other knowledge base will provide complete information about the subject either in triplet format or JSON format. The system will match the predicate in the output and also store it in data structure. If the system will be able to find out a match in the output then it is the answer otherwise it will use synonym API to get the synonyms of the given predicate and look over the stored data structure for all the synonym of the given predicate.

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.

3. Literature Survey and Workflow of the system

For developing the open source Question Answering System, we need different components at different phases of computing the answer. These components will be used to understand the meaning, semantics behind the question, information about

subject and predicate, understanding the relationship between subject and predicate, class of subject, and detailed knowledge about the subject. For this we used various kind of open source software system.

3.1 Natural Language processing

Natural Language processing is a field of Artificial Intelligence and linguistic, and it is primarily concerned on interaction between computer and human language.

Functionality of NLP encompasses broad range of topic like, token detection, part of speech analysis of sentence, semantic analysis, parsing etc. There is couple of open source natural language processing software for tokenization and semantic analysis.

Before implementing the NLP software we did literature survey on couple of Open source NLP software. Some of them are OpenNLP, Stanford parser etc.

3.1.1 Open NLP

Open NLP is a toolkit available in variety of programming languages which supports most of the Natural Language Processing tasks like tokenization, pos tagging, chunking, name recognition, sentence segmentation. These tasks are generally required to produce services that require more advanced text processing.

Sentence Detection

Open NLP sentence detector, detects the sentences in the given text. It basically determines the sentences on the basis of white spaces between the punctuation

marks. In this first and last case are exceptional. It uses the training data to detect the sentence boundary.

Tokenization

Open NLP tokenize is a tool which tokenize and de-tokenize the input. By tokenization, it means segmenting the input sequence into words/tokens. The tokens are given as input for further processing such as parsing and text mining. It is used in the form of text segmentation as well as lexical analysis. There are different types of tokenizer tools and they work on different criteria, like, whitespace tokenizer tokenizes the sentence/input sequence on the basis of whitespaces, simple tokenizer, tokenizes the sentence/input sequence on the basis of same class of characters.

Name Finder

Open NLP name finder will find the name and number in the given input sequence. For doing this it requires a model using which it can identify the name. This model is specific to language, for each language OpenNLP have different model.

Part of speech tagging

POS tagger is used to tokenize the given input sequence with their corresponding word type. This word type is decided on the basis of token itself and on the basis of context of the token in the input character sequence. To perform the POS tagging we have a set of POS tags and a model which is used to determine the tags for each token. Again OpenNLP has different model for different languages. This tool is most

useful in Question answering system since using these tool, we can figure out the subject and predicate in the question. And if there could be multiple set of subject predicate then, which set of subject object, is the primary one.

The list of tags used in OpenNLP Part of speech tagging is as follows [4]:

POS Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	there is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	big
JJR	adjective, comparative	bigger
JJS	adjective, superlative	biggest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	door
NNS	noun plural	doors

NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	both the boys
POS	possessive ending	friend's
PRP	personal pronoun	I, he, it
PRP	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give up
TO	to	to go, to him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VCN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP\$	possessive wh-pronoun	whose

WRB

wh-abverb

where, when

Chunker

OpenNLP chunker tool is to chunk the input sequence on the basis of POS tags and make groups like noun group, verb group etc., but it specifies the group only not the structure and the relationship among tokens in the group.

3.1.2 Stanford Parser: A statistical parser

Stanford parser is a JAVA based open source Natural Language Processing tool which performs the linguistic analysis of the given input character sequence to show output in various forms like, tokenization, POS tagging, sentence determination. In Stanford parser we can input the part of speech tag and force the parser to use those tags.

Stanford parser also provides parse tree with type dependencies, but the issue with the Stanford parser is that its memory usage for using Stanford parser is very expensive.

3.1.3 Natural Language Processing tool in Question Answering system

Natural language processing tools are required in question answering system at the very first initial phase to find out subject- predicate set. Only after getting to know about the subject we can proceed further and look over knowledge base to find out the answer. Also, OpenNLP is used to figure out the type of question using Wh-

Determiner tag; this is used to compute the answer after we have gathered the whole knowledge about the system. This is also used to validate the answer with the class label of the answer.

Among the above tools, we used Part of Speech tagging tool of OpenNLP in our system to tokenize the input character sequence and after that the system will try to figure out the subject and predicate in the given input character sequence. If there are more than one subject-predicate set then the system will decide that which set is primary set and which one is not. On the basis of this ranking the system will look over the knowledge base for finding the answer of the question.

3.2 Look up API – knowledge base

After getting the subject from the natural language processing tool, the subject is fed to lookup API which is a RESTful service to lookup DBPedia URI by related words. It also provides the class label and category for each subject if available. If there is no class label then, the noun we are considering is not a subject; instead it is a predicate.

If there are multiple result for the same keyword then Lookup API will provide the results according to the ranking, and this ranking is based on number of in-links to the Wikipedia page of that subject. To understand this properly let's consider an example, if we got the subject as Java, then the question might be regarding Java island , located in Indonesia, Java programming language or Java coffee and all results have different DBPedia URI, different class label and different description. So

now the question in this type of scenario which URI label should be considered or actually considered first. So, in DBpedia we can make the output results sorted on the basis of their ranking on Wikipedia which is calculated using various factors and major factor among those is on basis of number of in-links to that particular entry with some particular meaning on Wikipedia page.

3.2.1 Input and Output for the Lookup API

Lookup API is a RESTful service, which provides the facility to lookup URI's which have information about the subject we entered.

Input to LookupAPI

Lookup API is a RESTful service and it takes the input as a URL with the input like type of search, keyword string, Query class, maximum number of results etc.

There are 2 types of search to get the output from Lookup API ^[8].

1. Keyword Search
2. Prefix Search (i.e. auto complete)

Keyword Search: In keyword search the input is type of search (i.e.

KeywordSearch), Keyword (subject), Keyword class (optional like person, place etc.)

This API will look over the knowledge base for all the entries of that particular

keyword and if class is mentioned then it will provide result of that particular class

only. Multiple words keyword is also allowed as an input for this API. If the keyword is of multiple words then the keyword to be fed to the system with space separated word.

Example: URL to get all the places related to

Chicago <http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?QueryClass=place&QueryString=Chicago>

Prefix Search (i.e. auto complete):

In prefix search the input is type of search (i.e. PrefixSearch), Keyword Prefix

(partial subject), Keyword class (optional like person, place etc.), MaxHits (Max results required from API)

This API will work just like auto complete box in which if you write partial word like 'Chica' look over the knowledge base for all the entries of that keyword that starts

with 'Chica' and in this case the top results will be related to Chicago and if class is mentioned then it will provide result of that particular class only. MaxHits will be some numeric and it denotes the maximum number of results expected from API.

Defaults max hits is 5. In prefix search also, multiple words keyword is also allowed as an input for this API. If the keyword is of multiple words, then only the last word could be taken as partial rest all words should be complete and the keyword to be fed to the system with space separated word.

Example: URL to get all the places starting from Chica

<http://lookup.dbpedia.org/api/search.asmx/PrefixSearch?QueryClass=place&MaxHits=10&QueryString=Chica>

Output from Lookup API

Lookup API server will return the result in XML format, whether we make call to server to search from Prefix Search or Keyword search. If the max hit is not specifies the default result will have 5 entries related to the keyword/subject the system searched for.

Each result entry will have a *Label, URI, Description, Classes, Categories, Templates, Redirects and Refcount*.

Label: The label of the result, which could be the subject name itself, or full form of subject (if the subject is Acronym) or any other label related to the subject given as input.

<Label>Chicago**</Label>**

URI: URI would be the link of DBPedia API which has the more information about the subject. DBpedia is explained later in this report. For each different result entry, the URI would point to different DBPedia page.

<URI><http://dbpedia.org/resource/Chicago>**</URI>**

Description: In each result set there a description tag for each result tag which describe briefly about the result. It's a short paragraph describing the information about the label.

<Description>

Chicago is the largest city in the state of Illinois. With over 2.8 million residents, it is the most populous city in the Midwestern United States and the third most populous city in the country. Its metropolitan area, commonly named "Chicagoland," is the 26th most populous in the world, home to an estimated 9.7 million people spread across the U.S. states of Illinois, Wisconsin, and Indiana. Chicago is the county seat of Cook County.

</Description>

Classes: Classes is a nested tag in the output. This tag can have one or more class tag. Each class tag specifies about the class of the result entry like whether it is a place, person, organization, university etc.

Class.Label: This is the label of the class ontology the result entry belongs to.

Class.URI: This URI denotes the URI for the DBpedia page for this particular class ontology.

```
<Classes>
  <Class>
    <Label>city</Label>
    <URI>http://dbpedia.org/ontology/City</URI>
  </Class>
  <Class>
    <Label>place</Label>
    <URI>http://dbpedia.org/ontology/Place</URI>
  </Class>
```

```

<Class>
  <Label>settlement</Label>
  <URI>http://dbpedia.org/ontology/Settlement</URI>
</Class>
<Class>
  <Label>populated place</Label>
  <URI>http://dbpedia.org/ontology/PopulatedPlace</URI>
</Class>
<Class>
  <Label>owl#Thing</Label>
  <URI>http://www.w3.org/2002/07/owl#Thing</URI>
</Class>
</Classes>

```

Categories: Categories tag is also a nested output tag. This tag can have multiple category tag in it. Category tag specifies that the result entry fall under what categories.

```

<Categories>
  <Category>
    <Label>Populated places on the Great Lakes</Label>
    <URI>
      http://dbpedia.org/resource/Category:Populated_places_on_the_Great_Lakes
    </URI>
  </Category>
  <Category>
    <Label>Chicago, Illinois</Label>
    <URI>
      http://dbpedia.org/resource/Category:Chicago,_Illinois
    </URI>
  </Category>
  <Category>
    <Label>Port settlements in the United States</Label>
    <URI>
      http://dbpedia.org/resource/Category:Port_settlements_in_the_United_States
    </URI>
  </Category>
</Categories>

```

```

</Category>
<Category>
  <Label>Chicago metropolitan area</Label>
  <URI>
    http://dbpedia.org/resource/Category:Chicago_metropolitan_
    area
  </URI>
</Category>
<Category>
  <Label >County seats in Illinois</Label>
  <URI>
    http://dbpedia.org/resource/Category:County_seats_in_Illinoi
  </URI>
</Category>
<Category>
  <Label>Populated places in DuPage County, Illinois</Label>
  <URI>
    http://dbpedia.org/resource/Category:Populated_places_in_D
    uPage_County,_Illinois
  </URI>
</Category>
<Category>
  <Label>Irish-American culture</Label>
  <URI>
    http://dbpedia.org/resource/Category:Irish-American_culture
  </URI>
</Category>
<Category>
  <Label>Communities on U.S. Route 66</Label>
  <URI>
    http://dbpedia.org/resource/Category:Communities_on_U.S._
    Route_66
  </URI>
</Category>
<Category>
  <Label>Polish American history</Label>
  <URI>
    http://dbpedia.org/resource/Category:Polish_American_histor
    y
  </URI>
</Category>
<Category>
  <Label>Populated places in Cook County, Illinois</Label>

```

```

        <URI>
            http://dbpedia.org/resource/Category:Populated_places_in_Cook_County,_Illinois
        </URI>
    </Category>
    <Category>
        <Label>Cities in Illinois</Label>
        <URI>
            http://dbpedia.org/resource/Category:Cities_in_Illinois
        </URI>
    </Category>
    <Category>
        <Label>
            United States places with Orthodox Jewish communities
        </Label>
        <URI>
            http://dbpedia.org/resource/Category:United_States_places_with_Orthodox_Jewish_communities
        </URI>
    </Category>
    <Category>
        <Label>Populated places established in 1833</Label>
        <URI>
            http://dbpedia.org/resource/Category:Populated_places_established_in_1833
        </URI>
    </Category>
</Categories>

```

RefCount: RefCount denotes the number of links in which the current page is mentioned i.e. its denotes the count of number in inlinks referring to current topic's web page.

```
<Refcount>19646</Refcount>
```

3.2.2 Class Label used by Lookup API

Lookup API follows the convention provided by schema.org for class labels. Each label has its own property list. Its not mandatory, but most of the properties have

some value for that particular subject. This class label could be used to verify the answer. On the basis of Wh-Determiner in the question, we can look over the class label for the answer and verify that it is same as expected. Primarily the results are divided into 7 classes as follows:

- [CreativeWork](#)
- [Event](#)
- [Intangible](#)
- [Organization](#)
- [Person](#)
- [Place](#)
- [Products](#)

In my system I used this convention of classes. There is one more detailed hierarchy of the class list with properties, which is as follows ^[5]:

- [Thing](#)
 - [Activity](#) ([edit](#))
 - [Game](#) ([edit](#))
 - [Sport](#) ([edit](#))
 - [Agent](#) ([edit](#))
 - [Organisation](#) ([edit](#))
 - [Band](#) ([edit](#))
 - [Broadcaster](#) ([edit](#))

- [BroadcastNetwork](#) ([edit](#))
- [RadioStation](#) ([edit](#))
- [TelevisionStation](#) ([edit](#))
- [Company](#) ([edit](#))
 - [Airline](#) ([edit](#))
 - [LawFirm](#) ([edit](#))
 - [RecordLabel](#) ([edit](#))
- [EducationalInstitution](#) ([edit](#))
 - [College](#) ([edit](#))
 - [Library](#) ([edit](#))
 - [School](#) ([edit](#))
 - [University](#) ([edit](#))
- [GeopoliticalOrganisation](#) ([edit](#))
- [GovernmentAgency](#) ([edit](#))
- [Legislature](#) ([edit](#))
- [MilitaryUnit](#) ([edit](#))
- [Non-ProfitOrganisation](#) ([edit](#))
- [PoliticalParty](#) ([edit](#))
- [SportsLeague](#) ([edit](#))
 - [AmericanFootballLeague](#) ([edit](#))
 - [AustralianFootballLeague](#) ([edit](#))
 - [AutoRacingLeague](#) ([edit](#))
 - [BaseballLeague](#) ([edit](#))
 - [BasketballLeague](#) ([edit](#))
 - [BowlingLeague](#) ([edit](#))
 - [BoxingLeague](#) ([edit](#))
 - [CanadianFootballLeague](#) ([edit](#))
 - [CricketLeague](#) ([edit](#))
 - [CurlingLeague](#) ([edit](#))
 - [CyclingLeague](#) ([edit](#))
 - [FieldHockeyLeague](#) ([edit](#))
 - [GolfLeague](#) ([edit](#))
 - [HandballLeague](#) ([edit](#))
 - [IceHockeyLeague](#) ([edit](#))
 - [InlineHockeyLeague](#) ([edit](#))
 - [LacrosseLeague](#) ([edit](#))
 - [MixedMartialArtsLeague](#) ([edit](#))
 - [MotorcycleRacingLeague](#) ([edit](#))

- [PaintballLeague](#) ([edit](#))
- [PoloLeague](#) ([edit](#))
- [RadioControlledRacingLeague](#) ([edit](#))
- [RugbyLeague](#) ([edit](#))
- [SoccerLeague](#) ([edit](#))
 - [SoccerLeagueSeason](#) ([edit](#))
- [SoftballLeague](#) ([edit](#))
- [SpeedwayLeague](#) ([edit](#))
- [TennisLeague](#) ([edit](#))
- [VideogamesLeague](#) ([edit](#))
- [VolleyballLeague](#) ([edit](#))
- [SportsTeam](#) ([edit](#))
 - [AmericanFootballTeam](#) ([edit](#))
 - [BasketballTeam](#) ([edit](#))
 - [CanadianFootballTeam](#) ([edit](#))
 - [HockeyTeam](#) ([edit](#))
 - [SoccerClub](#) ([edit](#))
 - [NationalSoccerClub](#) ([edit](#))
 - [SpeedwayTeam](#) ([edit](#))
- [SportsTeamSeason](#) ([edit](#))
 - [SoccerClubSeason](#) ([edit](#))
- [TradeUnion](#) ([edit](#))
- [Person](#) ([edit](#))
 - [Ambassador](#) ([edit](#))
 - [Architect](#) ([edit](#))
 - [Artist](#) ([edit](#))
 - [Actor](#) ([edit](#))
 - [AdultActor](#) ([edit](#))
 - [VoiceActor](#) ([edit](#))
 - [Comedian](#) ([edit](#))
 - [ComicsCreator](#) ([edit](#))
 - [MusicalArtist](#) ([edit](#))
 - [Writer](#) ([edit](#))
 - [Astronaut](#) ([edit](#))
 - [Athlete](#) ([edit](#))
 - [AustralianRulesFootballPlayer](#) ([edit](#))
 - [BadmintonPlayer](#) ([edit](#))
 - [BaseballPlayer](#) ([edit](#))

- [BasketballPlayer](#) ([edit](#))
- [Boxer](#) ([edit](#))
- [Cricketer](#) ([edit](#))
- [Cyclist](#) ([edit](#))
- [FigureSkater](#) ([edit](#))
- [FormulaOneRacer](#) ([edit](#))
- [GaelicGamesPlayer](#) ([edit](#))
- [GolfPlayer](#) ([edit](#))
- [GridironFootballPlayer](#) ([edit](#))
 - [AmericanFootballPlayer](#) ([edit](#))
 - [CanadianFootballPlayer](#) ([edit](#))
- [IceHockeyPlayer](#) ([edit](#))
- [MartialArtist](#) ([edit](#))
- [NascarDriver](#) ([edit](#))
- [NationalCollegiateAthleticAssociationAthlete](#) ([edit](#))
- [RugbyPlayer](#) ([edit](#))
- [SnookerPlayer](#) ([edit](#))
 - [SnookerChamp](#) ([edit](#))
- [SoccerPlayer](#) ([edit](#))
- [Swimmer](#) ([edit](#))
- [TeamMember](#) ([edit](#))
- [TennisPlayer](#) ([edit](#))
- [VolleyballPlayer](#) ([edit](#))
- [Wrestler](#) ([edit](#))
- [BritishRoyalty](#) ([edit](#))
- [BullFighter](#) ([edit](#))
- [Celebrity](#) ([edit](#))
- [ChessPlayer](#) ([edit](#))
- [Cleric](#) ([edit](#))
 - [Cardinal](#) ([edit](#))
 - [ChristianBishop](#) ([edit](#))
 - [Pope](#) ([edit](#))
 - [Priest](#) ([edit](#))
 - [Saint](#) ([edit](#))
- [CollegeCoach](#) ([edit](#))
- [Criminal](#) ([edit](#))
- [FictionalCharacter](#) ([edit](#))

- [ComicsCharacter](#) ([edit](#))
 - [Journalist](#) ([edit](#))
 - [Judge](#) ([edit](#))
 - [MilitaryPerson](#) ([edit](#))
 - [Model](#) ([edit](#))
 - [Monarch](#) ([edit](#))
 - [OfficeHolder](#) ([edit](#))
 - [OrganisationMember](#) ([edit](#))
 - [SportsTeamMember](#) ([edit](#))
 - [Philosopher](#) ([edit](#))
 - [PlayboyPlaymate](#) ([edit](#))
 - [PokerPlayer](#) ([edit](#))
 - [Politician](#) ([edit](#))
 - [Chancellor](#) ([edit](#))
 - [Congressman](#) ([edit](#))
 - [Deputy](#) ([edit](#))
 - [Governor](#) ([edit](#))
 - [Lieutenant](#) ([edit](#))
 - [Mayor](#) ([edit](#))
 - [MemberOfParliament](#) ([edit](#))
 - [President](#) ([edit](#))
 - [PrimeMinister](#) ([edit](#))
 - [Senator](#) ([edit](#))
 - [VicePresident](#) ([edit](#))
 - [VicePrimeMinister](#) ([edit](#))
 - [Referee](#) ([edit](#))
 - [Royalty](#) ([edit](#))
 - [PolishKing](#) ([edit](#))
 - [Scientist](#) ([edit](#))
 - [SoccerManager](#) ([edit](#))
- [AnatomicalStructure](#) ([edit](#))
 - [Artery](#) ([edit](#))
 - [Bone](#) ([edit](#))
 - [Brain](#) ([edit](#))
 - [Embryology](#) ([edit](#))
 - [Lymph](#) ([edit](#))
 - [Muscle](#) ([edit](#))
 - [Nerve](#) ([edit](#))

- [Vein](#) ([edit](#))
- [Asteroid](#) ([edit](#))
- [Award](#) ([edit](#))
- [Biomolecule](#) ([edit](#))
 - [Gene](#) ([edit](#))
 - [HumanGene](#) ([edit](#))
 - [MouseGene](#) ([edit](#))
 - [Protein](#) ([edit](#))
- [CelestialBody](#) ([edit](#))
- [ChemicalSubstance](#) ([edit](#))
 - [ChemicalCompound](#) ([edit](#))
 - [ChemicalElement](#) ([edit](#))
- [Colour](#) ([edit](#))
- [Constellation](#) ([edit](#))
- [Currency](#) ([edit](#))
- [Database](#) ([edit](#))
 - [BiologicalDatabase](#) ([edit](#))
- [Device](#) ([edit](#))
 - [AutomobileEngine](#) ([edit](#))
 - [Weapon](#) ([edit](#))
- [Disease](#) ([edit](#))
- [Drug](#) ([edit](#))
- [EthnicGroup](#) ([edit](#))
- [Event](#) ([edit](#))
 - [Convention](#) ([edit](#))
 - [Election](#) ([edit](#))
 - [FilmFestival](#) ([edit](#))
 - [MilitaryConflict](#) ([edit](#))
 - [MusicFestival](#) ([edit](#))
 - [SpaceMission](#) ([edit](#))
 - [SportsEvent](#) ([edit](#))
 - [FootballMatch](#) ([edit](#))
 - [GrandPrix](#) ([edit](#))
 - [MixedMartialArtsEvent](#) ([edit](#))
 - [Olympics](#) ([edit](#))
 - [Race](#) ([edit](#))
 - [SoccerTournament](#) ([edit](#))
 - [WomensTennisAssociationTournament](#) ([edit](#))

- [WrestlingEvent](#) ([edit](#))
 - [Year](#) ([edit](#))
 - [YearInSpaceflight](#) ([edit](#))
- [Flag](#) ([edit](#))
- [Food](#) ([edit](#))
 - [Beverage](#) ([edit](#))
- [Galaxy](#) ([edit](#))
- [GeneLocation](#) ([edit](#))
 - [HumanGeneLocation](#) ([edit](#))
 - [MouseGeneLocation](#) ([edit](#))
- [GovernmentType](#) ([edit](#))
- [Holiday](#) ([edit](#))
- [Ideology](#) ([edit](#))
- [Language](#) ([edit](#))
- [LegalCase](#) ([edit](#))
 - [SupremeCourtOfTheUnitedStatesCase](#) ([edit](#))
- [MeanOfTransportation](#) ([edit](#))
 - [Aircraft](#) ([edit](#))
 - [Automobile](#) ([edit](#))
 - [Locomotive](#) ([edit](#))
 - [Rocket](#) ([edit](#))
 - [Ship](#) ([edit](#))
 - [SpaceShuttle](#) ([edit](#))
 - [SpaceStation](#) ([edit](#))
 - [Spacecraft](#) ([edit](#))
- [MusicGenre](#) ([edit](#))
- [Name](#) ([edit](#))
 - [GivenName](#) ([edit](#))
 - [Surname](#) ([edit](#))
- [OlympicResult](#) ([edit](#))
- [PersonFunction](#) ([edit](#))
- [Place](#) ([edit](#))
 - [ArchitecturalStructure](#) ([edit](#))
 - [Building](#) ([edit](#))
 - [Arena](#) ([edit](#))
 - [Church](#) ([edit](#))
 - [HistoricBuilding](#) ([edit](#))
 - [Hospital](#) ([edit](#))

- [Hotel](#) ([edit](#))
- [Lighthouse](#) ([edit](#))
- [Museum](#) ([edit](#))
- [Restaurant](#) ([edit](#))
- [ShoppingMall](#) ([edit](#))
- [Skyscraper](#) ([edit](#))
- [Stadium](#) ([edit](#))
- [Theatre](#) ([edit](#))
- [Infrastructure](#) ([edit](#))
 - [Airport](#) ([edit](#))
 - [LaunchPad](#) ([edit](#))
 - [PowerStation](#) ([edit](#))
 - [RouteOfTransportation](#) ([edit](#))
 - [Bridge](#) ([edit](#))
 - [PublicTransitSystem](#) ([edit](#))
 - [RailwayLine](#) ([edit](#))
 - [Road](#) ([edit](#))
 - [RoadJunction](#) ([edit](#))
 - [Tunnel](#) ([edit](#))
 - [RailwayTunnel](#) ([edit](#))
 - [RoadTunnel](#) ([edit](#))
 - [WaterwayTunnel](#) ([edit](#))
 - [Station](#) ([edit](#))
- [Park](#) ([edit](#))
- [HistoricPlace](#) ([edit](#))
- [Monument](#) ([edit](#))
- [MountainPass](#) ([edit](#))
- [NaturalPlace](#) ([edit](#))
 - [BodyOfWater](#) ([edit](#))
 - [Lake](#) ([edit](#))
 - [Stream](#) ([edit](#))
 - [Canal](#) ([edit](#))
 - [River](#) ([edit](#))
 - [Cave](#) ([edit](#))
 - [LunarCrater](#) ([edit](#))
 - [Mountain](#) ([edit](#))
 - [MountainRange](#) ([edit](#))
 - [Valley](#) ([edit](#))

- [PopulatedPlace](#) ([edit](#))
 - [AdministrativeRegion](#) ([edit](#))
 - [Atoll](#) ([edit](#))
 - [Continent](#) ([edit](#))
 - [Country](#) ([edit](#))
 - [Island](#) ([edit](#))
 - [Settlement](#) ([edit](#))
 - [City](#) ([edit](#))
 - [Town](#) ([edit](#))
 - [Village](#) ([edit](#))
- [ProtectedArea](#) ([edit](#))
- [SiteOfSpecialScientificInterest](#) ([edit](#))
- [SkiArea](#) ([edit](#))
- [WineRegion](#) ([edit](#))
- [WorldHeritageSite](#) ([edit](#))
- [Planet](#) ([edit](#))
- [ProgrammingLanguage](#) ([edit](#))
- [Project](#) ([edit](#))
 - [ResearchProject](#) ([edit](#))
- [Sales](#) ([edit](#))
- [SnookerWorldRanking](#) ([edit](#))
- [Species](#) ([edit](#))
 - [Archaea](#) ([edit](#))
 - [Bacteria](#) ([edit](#))
 - [Eukaryote](#) ([edit](#))
 - [Animal](#) ([edit](#))
 - [Amphibian](#) ([edit](#))
 - [Arachnid](#) ([edit](#))
 - [Bird](#) ([edit](#))
 - [Crustacean](#) ([edit](#))
 - [Fish](#) ([edit](#))
 - [Insect](#) ([edit](#))
 - [Mammal](#) ([edit](#))
 - [Mollusca](#) ([edit](#))
 - [Reptile](#) ([edit](#))
 - [Fungus](#) ([edit](#))
 - [Plant](#) ([edit](#))
 - [ClubMoss](#) ([edit](#))

- [Conifer](#) ([edit](#))
 - [Cycad](#) ([edit](#))
 - [Fern](#) ([edit](#))
 - [FloweringPlant](#) ([edit](#))
 - [Grape](#) ([edit](#))
 - [Ginkgo](#) ([edit](#))
 - [Gnetophytes](#) ([edit](#))
 - [GreenAlga](#) ([edit](#))
 - [Moss](#) ([edit](#))
- [Tax](#) ([edit](#))
- [TopicalConcept](#) ([edit](#))
- [Unknown](#) ([edit](#))
- [Work](#) ([edit](#))
 - [Film](#) ([edit](#))
 - [Musical](#) ([edit](#))
 - [MusicalWork](#) ([edit](#))
 - [Album](#) ([edit](#))
 - [Single](#) ([edit](#))
 - [Song](#) ([edit](#))
 - [EurovisionSongContestEntry](#) ([edit](#))
 - [Painting](#) ([edit](#))
 - [Sculpture](#) ([edit](#))
 - [Software](#) ([edit](#))
 - [VideoGame](#) ([edit](#))
 - [TelevisionEpisode](#) ([edit](#))
 - [TelevisionSeason](#) ([edit](#))
 - [TelevisionShow](#) ([edit](#))
 - [Website](#) ([edit](#))
 - [WrittenWork](#) ([edit](#))
 - [Book](#) ([edit](#))
 - [ComicBook](#) ([edit](#))
 - [PeriodicalLiterature](#) ([edit](#))
 - [AcademicJournal](#) ([edit](#))
 - [Magazine](#) ([edit](#))
 - [Newspaper](#) ([edit](#))
 - [Play](#) ([edit](#))

3.3 DBPedia – Knowledge base

DBPedia API provides structured information from Wikipedia. It gives information about particular subject various standardized format. In Question Answering system, once the system gets the URI from Lookup API it will send a RESTful request to the server and expects the output in triplet format. The information of this triplet is used to search for the answer of the question, and side by side stored in already existing data structure for later use.

DBPedia^[9] 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 ^[7]:

Figure 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 API then there might be probability that the predicate is used in knowledge base with some different name. So, the system will call for synonyms of the predicate and look for the answer in the stored data structure and if we are using data structure (like HashMap) which looks up the answer in constant time, then in that case this look up will not be expensive. The system then lookup all the synonyms provided by the synonym API and the sequence for lookup of the synonym in stored information in data structure will be in same sequence as the original predicate. DBPedia provides information in various language but for now Question Answering system supports information in English only, so information in other languages are eliminated while storing it in data structure. Once the system finds the answer from the response, the system will verify it by checking the class of the answer. This matching will be done on the basis of question type, which could be determined using Wh-Determiner.

If the answer is not found in the DBpedia knowledge base, then there is a tag of "sameAs" in DBPedia which will provide the URI of some other knowledge base for the same subject like, URI for freebase, Opencyc etc. The system will use the freebase URI if present otherwise create the URI for freebase using RESTful call and look over the answer in the knowledge base. If the answer is not found in that freebase response also, then the system will print in the output that "The system is not able to find the answer of this question, Try rephrasing the question or Type another question".

The output of the DBpedia in browser will look like as follows:

About: Chicago
 An Entity of Type : [City](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)

Chicago is the largest city in the US state of Illinois. With nearly 2.7 million residents, it is the most populous city in the Midwestern United States and the third most populous in the USA, after New York City and Los Angeles.

Property	Value
dbpedia-owl:PopulatedPlace/areaMetro	28163.530711793665
dbpedia-owl:PopulatedPlace/areaTotal	606.1
dbpedia-owl:PopulatedPlace/areaUrban	5498.026760621261
dbpedia-owl:PopulatedPlace/populationDensity	4447.4
dbpedia-owl:abstract	Chicago is the largest city in the US state of Illinois. With nearly 2.7 million residents, it is the most populous city in the Midwestern United States and the third most populous in the USA, after New York City and Los Angeles. Its metropolitan area, commonly named "Chicagoland," is the 27th most populous urban agglomeration in the world, the largest in the Great Lakes Megalopolis, and the third largest in the United States, home to an estimated 9.5 million people spread across the US states of Illinois, Wisconsin, and Indiana. Chicago is the county seat of Cook County, the second most populous county in the United States, after Los Angeles County, California. Chicago was founded in 1833, near a portage between the Great Lakes and the Mississippi River watershed. Today, the city retains its status as a major hub for industry, telecommunications and infrastructure, with O'Hare International Airport being the second busiest airport in the world in terms of traffic movements. In 2008, the city hosted 45.6 million domestic and overseas visitors. As of 2010, Chicago's metropolitan area has the 4th largest Gross Domestic Product (GDP) amongst world metropolitan areas. The city is a center for business and finance and is listed as one of the world's top ten Global Financial Centers. The World Cities Study Group at Loughborough University rated Chicago as an "alpha world city". In a 2010 survey collaboration between Foreign Policy and A. T. Kearney ranking cities, Chicago ranked 6th, just after Paris and Hong Kong. The ranking assesses five dimensions: value of capital markets, diversity of human capital, international information resources, international cultural resources, and political influence. Chicago has been ranked by Forbes as the world's 5th most economically powerful city. Chicago is a stronghold of the Democratic Party and has been home to many influential politicians, including the current President of the United States, Barack Obama. The city's notoriety expressed in popular culture is found in novels, plays, movies, songs, various types of journals (for example, sports, entertainment, business, trade, and academic), and the news media. Chicago has numerous nicknames, which reflect the impressions and opinions about historical and contemporary Chicago. The best known include: "Chi-town," "Windy City," "Second City," and the "City of Big Shoulders." Chicago has also been called "the most American of big cities". Chicago (Vorlage:Audio) ist eine Stadt am Südwestufer des Michigansees im Bundesstaat Illinois in den Vereinigten Staaten von Amerika. Mit einer Einwohnerzahl von 2.695.598 (2010) ist sie die drittgrößte Stadt der USA. In der Agglomeration leben 8,7 Millionen, in der Metropolregion Chicago 9,7 Millionen Menschen (2007). Chicago ist seit der Mitte des 19. Jahrhunderts eine wichtige Handelsstadt in den Vereinigten Staaten. Die Lage an der Mündung des Illinois Waterways, der die Großen Seen mit dem Mississippi River und dem Golf von Mexiko verbindet, sowie als Knotenpunkt wichtiger Eisenbahnstrecken, auf denen die West- mit der Ostküste verbunden werden, haben diese Funktion begünstigt. Chicago ist Sitz der Chicago Mercantile Exchange, der größten Warenterminbörse der Vereinigten Staaten, und der Chicago Board of Trade, der größten Rohstoff-, Futures- und Optionsbörse der USA.

Figure 2 : Output of DBpedia in browser

Properties of a Class:

The DBpedia follows the property ontologies provided by schema.org. Each class has a set of property, and the response of the DBpedia have values of these properties. List of properties of place class is as follows^[6]:

Properties on Place:

Name	Label	Domain	Range
abbreviation (edit)	abbreviation	Thing	<i>string</i>
abstract (edit)	has abstract	Thing	<i>string</i>
access (edit)	access	Thing	<i>string</i>
accessDate (edit)	access date	Thing	<i>Year</i>
activeYearsEndDate (edit)	active years end date	Thing	<i>date</i>
activeYearsEndYear (edit)	active years end year	Thing	<i>Year</i>
activeYearsStartDate (edit)	active years start date	Thing	<i>date</i>
activeYearsStartYear (edit)	active years start year	Thing	<i>Year</i>
album (edit)	album	Thing	Album
alias (edit)	alias	Thing	<i>string</i>
alongside (edit)	alongside	Thing	Person

animal (edit)	animal	Thing	Animal
annualTemperature (edit)	annual temperature	Place	<i>Temperature</i>
apoapsis (edit)	apoapsis	Thing	<i>Length</i>
appointer (edit)	appointer	Thing	Person
area (edit)	area	Thing	<i>Area</i>
areaLand (edit)	area land	Place	<i>Area</i>
areaMetro (edit)	area metro	Thing	<i>Area</i>
areaTotal (edit)	area total	Place	<i>Area</i>
areaWater (edit)	area water	Place	<i>Area</i>
ascent (edit)	ascent	Thing	<i>string</i>
associate (edit)	associate	Thing	Person
associateStar (edit)	associateStar	Thing	Constellation
associatedBand (edit)	associated band	Thing	Band
associatedMusicalArtist (edit)	associated musical artist	Thing	MusicalArtist
atcPrefix (edit)	ATC prefix	Thing	<i>string</i>
atcSuffix (edit)	ATC suffix	Thing	<i>string</i>
atcSupplemental (edit)	ATC supplemental	Thing	<i>string</i>
authority (edit)	authority	Thing	Thing
averageSpeed (edit)	average speed	Thing	<i>Speed</i>
background (edit)	background	Thing	<i>string</i>
battle (edit)	battle	Thing	MilitaryConflict
biome (edit)	biome	Thing	Thing
boilingPoint (edit)	boiling point	Thing	<i>Temperature</i>
budget (edit)	budget	Thing	<i>Currency</i>
builder (edit)	builder	Thing	Thing
callSign (edit)	call sign	Thing	<i>string</i>
casSupplemental (edit)	CAS supplemental	Thing	<i>string</i>
category (edit)	category	Thing	Thing
chairLabel (edit)	chair label	Thing	Thing
chairman (edit)	chairman	Thing	Person
chancellor (edit)	chancellor	Thing	Person
city (edit)	city	Thing	City

climate (edit)	climate	Place	Thing
closingDate (edit)	closing date	Thing	<i>date</i>
closeYear (edit)	closing year	Thing	<i>Year</i>
coach (edit)	coach	Thing	Person
code (edit)	code	Thing	<i>string</i>
college (edit)	college	Thing	College
colour (edit)	colour	Thing	Colour
colourHexCode (edit)	colour hex code	Thing	<i>string</i>
colourName (edit)	colour name	Thing	<i>string</i>
commander (edit)	commander	Thing	Person
committee (edit)	committee	Thing	<i>string</i>
commonName (edit)	common name	Thing	<i>string</i>
company (edit)	company	Thing	Organisation
composer (edit)	composer	Thing	Person
connotation (edit)	connotation	Thing	Thing
cosparId (edit)	COSPAR id	Thing	<i>integer</i>
country (edit)	country	Thing	Country
county (edit)	county	Thing	PopulatedPlace
creationYear (edit)	year of creation	Thing	<i>Year</i>
criteria (edit)	criteria	Thing	<i>string</i>
currency (edit)	currency	Thing	Currency
currencyCode (edit)	currency code	Place	<i>string</i>
currentMember (edit)	current member	Thing	Person
date (edit)	date	Thing	<i>date</i>
dateOfAbandonment (edit)	date of abandonment	Thing	<i>date</i>
daylightSavingTimeZone (edit)	daylight saving time zone	Place	Thing
dc:language (edit)	language	Thing	<i>string</i>
dc:rights (edit)	rights	Thing	Thing
dc:subject (edit)	subject	Thing	Thing
dct:subject (edit)	subject	Thing	Thing
deliveryDate (edit)	delivery date	Thing	<i>date</i>
demonym (edit)	demonym	Thing	<i>string</i>

density (edit)	density	Thing	<i>Density</i>
depth (edit)	depth	Place	<i>Length</i>
deputy (edit)	deputy	Thing	Person
designer (edit)	designer	Thing	Person
destructionDate (edit)	destruction date	Thing	<i>date</i>
developer (edit)	developer	Thing	Company
diameter (edit)	diameter	Thing	<i>Length</i>
director (edit)	director	Thing	Person
discharge (edit)	discharge	Thing	<i>FlowRate</i>
dischargeAverage (edit)	discharge average	Thing	<i>FlowRate</i>
discoverer (edit)	discoverer	Thing	Person
distance (edit)	distance	Thing	<i>Length</i>
distributor (edit)	distributor	Thing	Organisation
district (edit)	district	Thing	PopulatedPlace
division (edit)	division	Thing	Thing
draft (edit)	draft	Thing	<i>string</i>
draftYear (edit)	draft year	Thing	<i>Year</i>
drugbank (edit)	DrugBank	Thing	<i>string</i>
editor (edit)	editor	Thing	Agent
electionMajority (edit)	election majority	Thing	<i>nonNegativeInteger</i>
elementAbove (edit)	element above	Thing	ChemicalSubstance
elevation (edit)	elevation	Place	<i>Length</i>
endangeredSince (edit)	endangered since	Place	<i>date</i>
engineer (edit)	engineer	Thing	Person
era (edit)	era	Thing	Thing
ethnicGroupsInYear (edit)	ethnic groups in year	Thing	<i>Year</i>
event (edit)	event	Thing	Event
executiveProducer (edit)	executive producer	Thing	Person
failedLaunches (edit)	failed launches	Thing	<i>nonNegativeInteger</i>
fileSize (edit)	size	Thing	<i>InformationUnit</i>
firstLeader (edit)	firstLeader	Thing	Person
firstPopularVote (edit)	firstPopularVote	Thing	Person
floodingDate (edit)	flooding date	Thing	<i>date</i>

foaf:based_near (edit)	based near	Thing	Thing
foaf:depiction (edit)	depiction	Thing	Thing
foaf:familyName (edit)	family name	Thing	<i>string</i>
foaf:givenName (edit)	given name	Thing	<i>string</i>
foaf:homepage (edit)	homepage	Thing	Thing
foaf:image (edit)	image	Thing	Thing
foaf:logo (edit)	logo	Thing	Thing
foaf:name (edit)	name	Thing	<i>string</i>
foaf:nick (edit)	nickname	Thing	<i>string</i>
foaf:page (edit)	A page or document about this thing.	Thing	foaf:Document
foaf:primaryTopic (edit)	primary topic	Thing	Thing
foaf:surname (edit)	surname	Thing	<i>string</i>
foaf:thumbnail (edit)	thumbnail	Thing	Thing
format (edit)	format	Thing	Thing
formerName (edit)	former name	Thing	<i>string</i>
foundedBy (edit)	founded by	Thing	Agent
foundingDate (edit)	founding date	Thing	<i>date</i>
foundinYear (edit)	founding year	Thing	<i>Year</i>
frequency (edit)	frequency	Thing	<i>Frequency</i>
fuel (edit)	fuel	Thing	Thing
gender (edit)	gender	Thing	Thing
genre (edit)	genre	Thing	Thing
geologicPeriod (edit)	geologic period	Place	<i>string</i>
geology (edit)	geology	Thing	<i>string</i>
governingBody (edit)	governing body	Place	Organisation
governmentType (edit)	government type	Thing	GovernmentType
governor (edit)	governor	Thing	Person
governorGeneral (edit)	governor general	Thing	Person
gridReference (edit)	grid reference	Place	<i>string</i>
hasSurfaceForm (edit)	undefined	Thing	<i>string</i>
hasVariant (edit)	variant or variation	Thing	Thing
height (edit)	height	Thing	<i>Length</i>

highestMountain (edit)	highest mountain	Thing	Mountain
highestPlace (edit)	highest place	Thing	PopulatedPlace
highestPosition (edit)	highest position	Thing	geo:SpatialThing
highschool (edit)	highschool	Thing	School
homeArena (edit)	home arena	Thing	Arena
homeStadium (edit)	home stadium	Thing	Stadium
imageSize (edit)	image size (px)	Thing	<i>integer</i>
incumbent (edit)	incumbent	Thing	Person
intercommunality (edit)	intercommunality	Thing	PopulatedPlace
isPartOf (edit)	is part of	Thing	Thing
isPrimaryTopicOf (edit)	is primary topic of	Thing	foaf:Document
iso31661Code (edit)	ISO 3166-1 code	Place	<i>string</i>
jurisdiction (edit)	jurisdiction	Thing	Thing
keyPerson (edit)	key person	Thing	Person
language (edit)	language	Thing	Language
languageRegulator (edit)	language regulator or academy	Thing	Language
leader (edit)	leader	Thing	Person
league (edit)	league	Thing	SportsLeague
length (edit)	length	Thing	<i>Length</i>
licensee (edit)	licensee	Thing	<i>string</i>
lieutenant (edit)	lieutenant	Thing	Person
localAuthority (edit)	local authority	Thing	Thing
locatedInArea (edit)	located in area	Place	Place
location (edit)	location	Thing	Place
locationCountry (edit)	location country	Thing	Country
logo (edit)	logo	Thing	<i>string</i>
longName (edit)	longName	Thing	<i>string</i>
lowest (edit)	lowest	Place	<i>string</i>
lowestMountain (edit)	lowest mountain	Thing	Mountain
lowestPlace (edit)	lowest place	Thing	PopulatedPlace
lowestPosition (edit)	lowest position	Thing	geo:SpatialThing
mainInterest (edit)	main interest	Thing	Thing

majorityFloorLeader (edit)	majority floor leader	Thing	<i>integer</i>
majorityLeader (edit)	majority leader	Thing	<i>integer</i>
management (edit)	management	Thing	Thing
managementElevation (edit)	management elevation	Thing	<i>Length</i>
managementMountain (edit)	management mountain	Thing	Mountain
managementPlace (edit)	management place	Thing	PopulatedPlace
managementPosition (edit)	management position	Thing	geo:SpatialThing
manufacturer (edit)	manufacturer	Thing	Organisation
map (edit)	map	Place	Thing
mapDescription (edit)	map description	Place	<i>string</i>
mascot (edit)	mascot	Thing	<i>string</i>
mass (edit)	mass	Thing	<i>Mass</i>
maximumDepth (edit)	maximum depth	Place	<i>Length</i>
maximumDischarge (edit)	maximum discharge	Thing	<i>FlowRate</i>
timeInSpace (edit)	time in space	Thing	<i>Time</i>
timeZone (edit)	time zone	Place	Thing
title (edit)	title	Thing	<i>string</i>
topSpeed (edit)	top speed	Thing	<i>Speed</i>
totalLaunches (edit)	total launches	Thing	<i>nonNegativeInteger</i>
tradeMark (edit)	TradeMark	Thing	Thing
type (edit)	type	Thing	Thing
unloCode (edit)	UN/LOCODE	Place	<i>string</i>
updated (edit)	updated	Thing	<i>date</i>
utcOffset (edit)	UTC offset	Place	<i>string</i>
variantOf (edit)	variant or variation	Thing	Thing
vehicle (edit)	vehicle	Thing	Automobile
vehicleCode (edit)	vehicle code	Place	<i>string</i>
viceChancellor (edit)	vice chancellor	Thing	Person
vicePresident (edit)	vice president	Thing	Person
vicePrimeMinister (edit)	vice prime minister	Thing	Person
volume (edit)	volume	Thing	<i>Volume</i>
webcast (edit)	webcast	Thing	Thing

weight (edit)	weight	Thing	<i>Mass</i>
width (edit)	width	Thing	<i>Length</i>
wikiPageDisambiguates (edit)	Wikipedia disambiguates	Thing	Thing
wikiPageExternalLink (edit)	Link from a Wikipedia to an external page	Thing	Thing
wikiPageID (edit)	Wikipedia page ID	Thing	<i>integer</i>
wikiPageRedirects (edit)	Wikipedia redirect	Thing	Thing
wikiPageRevisionID (edit)	Wikipedia revision ID	Thing	<i>integer</i>
wikiPageWikiLink (edit)	Link from a Wikipedia to another Wikipedia	Thing	Thing
yearOfConstruction (edit)	year of construction	Place	<i>Year</i>

3.2.4 Freebase – Knowledge Base

Freebase is a large collection of online structured knowledge base gathered from many sources ^[10]. In Question answering system it is the secondary source of knowledge base. If the system is not able to find answer in DBpedia it will look over the answer in Freebase. Freebase is available to both commercial and non-commercial users. For non-commercial user, there is a limit on query fetching in single day. The output from the freebase API is in JSON format. Once the system gets the output from the API it will parse and scan the output and put it in same data structure which was created while accessing DBpedia information. If there is some tag already exists in data structure, it will not insert the value again and simply skip that predicate.

Again same, if the predicate is not found in the information provided by the Freebase API response, then the system will look for synonym API and find the match for all the synonym in the existing data structure.

If the answer is not found for any of the synonym also, then the system will return the message that “The system is not able to find the answer of this question, Try rephrasing the question or Type another question”.

4 System Architecture

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

4.1 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.

Figure 3 : MVC Architecture

4.2 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.

4.3 Workflow of the System's User Interface

The home page has a text box in which user enters the Question and clicks on Process Button. The process button will call the OpenNLP service and then process the OpenNLP output to the tabular format with accurate labels.

5 Conclusion

In this project there was an implantation of Question answering system and to implement this system, couple of open source softwares/systems is used. The system caters the need of getting the answer to some particular question instead of getting the links/URLs which might have the desired information. This system will be helpful in mobile devices like cell phones, tablets etc. where the screen size is so small that to find answer on a particular page is really difficult and also all 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 a achieved in an efficient manner.

The implementation of Question Answering system is done using Model View Controller architecture which can be reused in further enhancements and expanding the knowledge base.

6 The Future of Question Answering System

Currently the Question answering system 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 the by listening to the question.

And currently this system is using only 3 open source software, though the information in those open source knowledge base is very vast and contains data on variety of topics, but at times the system is not able to answer all type of question, so for that we can incorporate some more knowledge base system in it. But by doing that there might be chances for the system to take more time to compute the answer. 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 question of logical reasoning, like why something happened. We can enhance its functionality so that it could be able to answer those questions also.

7 References

[1] IBM Watson Jr. – How to build your

https://www.ibm.com/developerworks/mydeveloperworks/blogs/InsideSystemStorage/entry/ibm_watson_how_to_build_your_own_watson_jr_in_your_basement7?lang=en

[2] Apache OpenNLP <http://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html>

[3] Stanford Parser <http://nlp.stanford.edu/software/lex-parser.shtml>

[4] Part of Speech Tags <http://www.monlp.com/2011/11/08/part-of-speech-tags/>

[5] List of DBPedia Ontology

<http://mappings.dbpedia.org/server/ontology/classes/>

[6] List of properties for Place class

<http://mappings.dbpedia.org/server/ontology/classes/Place>

[7] DBpedia Documentation <http://wiki.dbpedia.org/DBpediaLive>

[8] Lookup API documentation <http://dbpedia.org/lookup>

[9] DBpedia browser result for Chicago <http://dbpedia.org/page/Chicago>

[10] Freebase documentation.

http://wiki.freebase.com/wiki/What_is_Freebase%3F