

Question Answering System for Yioop

A Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

By

Niravkumar Patel

Dec 2015

© 2015

Niravkumar Patel

ALL RIGHTS RESERVED

The Designated Thesis Committee Approves the Thesis Titled

Question Answering System for Yioop

by

Niravkumar Patel

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2015

Dr. Chris Pollett Department of Computer Science

Dr. Thomas Austin Department of Computer Science

Dr. Robert Chun Department of Computer Science

ABSTRACT

QUESTION ANSWERING SYSTEM FOR YIOOP

by Niravkumar Patel

Yioop is an open source search engine developed and managed by Dr. Christopher Pollett. Currently, Yioop returns the search results of the query in the form of list of URLs, just like other search engines (Google, Bing, DuckDuckGo, etc.)

This paper created a new module for Yioop. This new module, known as the Question-Answering (QA) System, takes the search queries in the form of natural language questions and returns results in the form of a short answer that is appropriate to the question asked. This feature is achieved by implementing various functionalities of Natural Language Processing (NLP). By using NLP, the new Question-Answering (QA) System attempts to extract the necessary information from the query provided by the user and provides an appropriate answer from the available data.

ACKNOWLEDGEMENT

I would like to express my gratitude to everyone who contributed directly or indirectly towards the completion of this project.

First and foremost, I would like to thank my project advisor, Dr. Christopher Pollett, for his unending patience and constant guidance towards this project, without which the success of this project would not have been achievable.

I would also like to extend my thanks to my committee members, Dr. Thomas Austin and Dr. Robert Chun, for their suggestions and time.

Contents

INTRODUCTION	1
BACKGROUND	4
2.1 Components of Yioop	4
2.1.1 Queue Server	5
2.1.2 Fetcher	5
2.1.3 Summarizer.....	5
2.1.4 Mini Indexer	6
QUESTION ANSWERING (QA) SYSTEM OVERVIEW	14
3.1 Representation of a Question Answering (QA) System:	14
3.2 Information Extraction approaches and it challenges	16
3.2.1 The Bag of Words Approach.....	16
3.2.2 The Triplet Extraction Approach.....	16
DESIGN AND IMPLEMENTATION OF THE QUESTION ANSWERING SYSTEM 20	
4.1 Role of the Question Answering (QA) System during crawl time:	20
4.1.1 Major components of the Question Answering (QA) System.....	22
4.2 Role of Question Answering (QA) System during Query Time:	37
EXPERIMENTS	39
5.1 Experiments on a standalone Question Answering (QA) system	39
5.2 Experiments on an integrated Question Answering (QA) system	45
5.2.1 Experiments during crawl time.....	46
5.2.2 Experiments during query time	48
5.3 Observations after Integration.....	49
CONCLUSION.....	52
REFERENCES	53

List of Figures

Fig. 1 Web interface to test Summarizer and Mini Index.....	9
Fig. 2 Generated summary by Summarizer	10
Fig. 3 Example of mini Index on the Summary.....	11
Fig. 4 Overview of information flow.....	12
Fig. 5 Generated Parsed Tree.....	28
Fig. 6 Parse tree generated from the complex sentence.....	29
Fig. 7 Triplet Extraction from the Tree.....	34
Fig. 8 Generated Tree from the statement	41
Fig. 9 Generated Tree from the statement	42
Fig. 10 Generated Tree from the statement	43
Fig. 11 Generated Tree from the statement	44
Fig. 12 Generated Question Triplet at crawl time.....	47
Fig. 13 Generated Question Triplet from the crawling.....	48
Fig. 14 Results before integration.....	49
Fig. 15 Results after integration.....	50
Fig. 16 Page processing time before/after the integration	51

List of Tables

Table 4.1 Part of Speech Tags.....23

CHAPTER 1

INTRODUCTION

The World Wide Web (WWW) is an abundant source of information that is growing every second. However, there are not many techniques that focus on data retrieval based upon a question provided by the user. This led to the development of the Question Answering (QA) System. The Question Answering (QA) System is a discipline under the Computer Science field of Information Retrieval (IR) and Natural Language Processing (NLP). The Question Answering (QA) System works by building intelligent systems that generate answers so as to imitate the answers given by a real human being [1].

Since the World Wide Web contains a huge amount of information which is growing exponentially every day, this information can be used in answering various questions. Several methods are followed to enter the information into the World Wide Web, for example, social media websites (Facebook, Twitter, etc.), information websites (Wikipedia), and many others. However, very few methods try to provide an answer to complex question asked by the user. A lot of research and resources have been employed in order to make this a better experience for the everyday user, with Voice Assisted Systems (VAS) such as “Siri” & “Cortana”. Such systems are constantly evolving in order to attain better accuracy and a better user experience. The Question Answering (QA) System is not a modern concept in the Computer Science domain; rather, it has been the topic of discussion for several decades. A few examples of such Question Answering

(QA) Systems include IBM's "Watson" [2] and "START" by Boris Katz and associates of the InfoLab Group at the MIT Computer Science and Artificial Intelligence Laboratory [3]. However, the best accuracy obtained by any such Question Answering (QA) System is not more than 70%.

This project aims to add a Question Answering (QA) System to the open source search engine, Yioop, developed by Dr. Christopher Pollett. Like any other search engine, Yioop also has a crawler that runs through the URLs on the web, extracts the information, processes it, and stores it in the Yioop index for later retrieval. The extraction phase of Yioop involves several components that are employed before storing the data to the Yioop index. One such important component is the Summarizer. This component runs as a part of the crawling process and is mainly responsible for limiting the amount of information, as well as providing the data with much-needed conciseness. The Summarizer produces a short summary of the data fetched by the crawler, which might otherwise be too long. The output of the Summarizer, known as the Summaries, obtained from several documents is used to generate the mini inverted index.

After building the mini inverted index on the summaries, the proposed Question Answering (QA) System comes into the picture. The Question Answering (QA) System iterates over all the phrases in the mini inverted index and stores the information in the form of [SUBJECT-PREDICATE-OBJECT] triplets [4]. These triplets are then stored in Yioop by appending them to the same mini inverted index. This is used when the user inputs a query in the form of a question. The Question Answering (QA) System generates all possible combinations of triplets, based on the types of question that a user can ask.

This saves processing time while extracting the query search results, leading to faster information retrieval.

The Question Answering (QA) System includes a query processing mechanism, which handles the questions entered by the users. The Question Answering (QA) System processes the user input, checks if it is in the form of a question, and generates the Question-Triplet if it identifies the input as a question. This triplet correlates with the list of triplets mentioned earlier in the mini inverted index that have a list of probable occurrence positions. This information is further used to form a meaningful answer to the original question.

Chapter 2 discusses the components used by the Question Answering (QA) System in various manners. These components are the essential and necessary source of information for the Question Answering (QA) System. Chapter 3 discusses the detailed overview of the proposed Question Answering (QA) System, along with the factors playing an important role in affecting the overall accuracy of the system. Chapter 4 gives an overview of the design and implementation strategy employed for the QA System. It also describes about other components used in the Question Answering (QA) System, such as the Part of Speech tagger, Parse Tree Generation, Triplet Extraction, and Question Generation. Moreover, this chapter also discusses the method used to look up a query entered by the user. Chapter 5 discusses the behavior of the Question Answering (QA) System as a stand-alone system, as well as the system that is integrated with the Yioop system. Finally, this chapter also discusses areas of possible improvements and future work.

CHAPTER 2

BACKGROUND

This chapter discusses the overview of the processes running in the background as a part of the crawling process. In order to generate a meaningful response to the user query, the Question Answering (QA) system tries to understand the question and retrieve the relevant data from the World Wide Web in an efficient manner. However, this data include a lot of unnecessary data containing special characters, such as html tags, xml tags, hyperlinks, meta-data of webpages, and many more. This data is not relevant to the actual information. All of the above has to be processed in a well-structured and articulated manner so that the Question Answering (QA) System gets to process the valid raw information.

One of the prominent features of the Yioop system is that the user can have control over providing/restricting the source URLs so that the results returned belong to the domain of the mentioned source. This feature can be accessed, edited and managed using the Manage Crawler option of the Yioop system's admin portal.

2.1 Components of Yioop

The crawling process in Yioop has four main components:

1. Queue Server
2. Fetcher
3. Summarizer

4. Mini Indexer

2.1.1 Queue Server

The Queue Server is responsible for maintaining the priority order of the URLs to be processed next in the queue.

2.1.2 Fetcher

The Fetcher of the Yioop System fetches each individual URL from the Queue Server and downloads all the information from the respective URL. As a part of information refinement, processes like Stemming and Stop Word Removal are applied to the downloaded information. Stemming is the process of reducing a word to its origin [12]. For example, the word “Running” is stemmed from the word “Run,” thus, in the current process, the Stemmer would convert the word “Running” to “Run.” The Stop Word removal is a process which removes the most common words from the sentence, i.e. *the, a, which, who, what, is*, etc. This processed data is then fetched by the Summarizer for the next level of its own processes.

2.1.3 Summarizer

The data is ready to be given to the Summarizer once all the basic preprocessing is completed. The Summarizer gets hold of the <meta> tags of the HTML page. It appends the contents of the first four <p> and <div> tags, which are further appended with the contents of the , <td>, <dd>, <dt> and <a> tags until it reaches a maximum predefined threshold value [HTMLPROCESSOR::MAX_DESCRIPTION_LEN (2000)] [5]. The order of precedence of the items added is from the tag with the most characters

to the tag with the least. The HTML processor can choose to obtain the same result by using a centroid summarizer. In this case, it removes all the tags and bifurcate the document into sentences, after which the average sentence vector is calculated while ignoring the stop words. The sentence vectors comprise the terms and the value for each term. Here, the value represents the likelihood of the term existing in any sentence of the document. After this, the distance between the centroid and each sentence is calculated. The sentences that have the least distance are added to the summary until it reaches the limit of 2000.

2.1.4 Mini Indexer

The mini inverted index is created from the summary generated by the Summarizer. The general format of the mini inverted index is as follows:

```
term_id_1 => ...  
term_id_2 => ... ...  
term_id_i => ((summary_map_1, (positions in summary 1 that term i appeared) ),  
(summary_map_2, (positions in summary 2 that term i appeared) ),  
...)  
...
```

The term ID shown above has a size of 20 bytes. The terms might represent a singular word or several terms or phrases. The first 8 bytes of the MD5 hash of the first word in the phrase/word are the first 8 bytes of the term ID. The byte, following the first 8 bytes, indicates whether the term is a word or a phrase. If the term is a word, then the

remaining bytes are used to identify the page type. However, if it is a phrase, the remaining bytes encode various length hashes of the remaining words in that phrase. The summary map provides the offset of the occurrence of the phrase/word in the summary. The summary is viewed as a single string, combining words extracted from the URL, appended with the summary title and appended with the summary description to calculate the position of the term. The number of words is counted from the start of the string. Phrases start at the position of their first word [5]. For example, two summaries consisting of only words, no phrases are shown below:

Summary 1:

URL: <http://test.yioop.com/>

Title: Fox Story

Description: The quick brown fox jumped over the lazy dog.

Summary 2:

URL: <http://test.yioop2.com/>

Title: Troll Story

Description: Once there was a lazy troll, P&A, who lived on my Discussion board.

The generated mini-inverted index is as in the below form:

```
(  
  [test] => ( (1, (0)), (2, (0)) )  
  [yioop] => ( (1, (1)) )  
  [yioop2] => ( (2, (1)) )  
  [fox] => ( (1, (2, 7)) )  
  [stori] => ( (1, (3)), (2, (3)) )  
  [the] => ( (1, (4, 10)) )  
  [quick] => ( (1, (5)) )  
  [brown] => ( (1, (6)) )  
  [jump] => ( (1, (8)) )  
  [over] => ( (1, (9)) )  
  [board] => ( (2, (16)) )  
)
```

The above representation is the posting list. The terms in this list are known as postings. These terms are already in the processed form before adding them into the mini-inverted index. To get the actual summarizer output and the mini index generated from this summarizer, Yioop has provided a nice web interface. The user can test it in a few easy steps, as follows:

1. User can login to Yioop as an Admin.
2. After the log in, click on the “Page Option”. (Refer Fig.1)

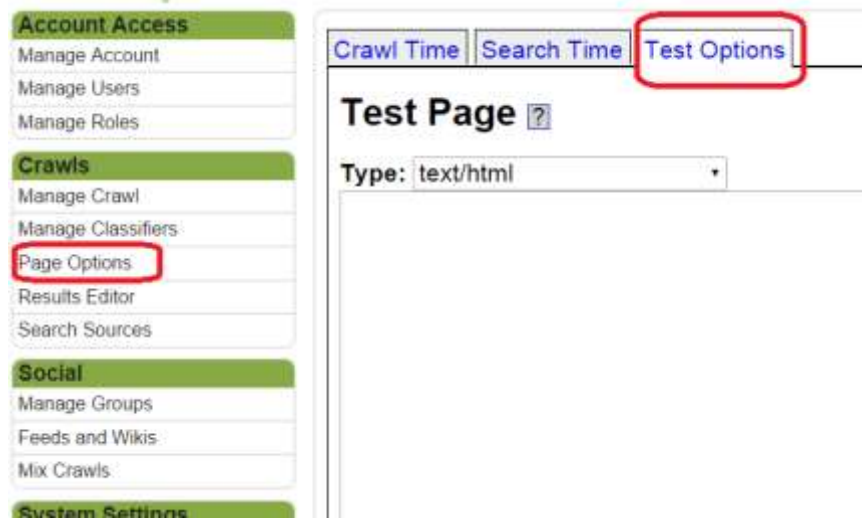


Fig. 1 Web interface to test Summarizer and Mini Index

3. In the Crawl section, select “Test Option” from the right pane. (Refer Fig.1)
4. Open any web page, for instance, the https://en.wikipedia.org/wiki/Barack_Obama page.
5. Right click on the page and select “view source”.
6. Copy the whole source file to the Yioop’s “Test Page” text window and click on the “Test Process Page”.
7. User will be able to see the output on the same page as below. (Refer Fig. 2 & 3)

After page rules applied

```
Array
(
  [ENCODING] => UTF-8
  [URL] => http://test-site.yioop.com/
  [IP_ADDRESSES] => Array
    (
      [0] => 1.1.1.1
    )

  [HTTP_CODE] => 200
  [MODIFIED] => 1448913409
  [TIMESTAMP] => 1448913409
  [TYPE] => text/html
  [HEADER] => page options test extractor
  [SERVER] => unknown
  [SERVER_VERSION] => unknown
  [OPERATING_SYSTEM] => unknown
  [LANG] => en
  [JUST_METAS] =>
  [ROBOT_METAS] => Array
    (

  [TITLE] => Hypertext Transfer Protocol - Wikipedia, the free
encyclopedia
  [DESCRIPTION] => 4 HTTP Authentication. 5 Request methods. 5.1 Safe
methods. 5.2 Idempotent methods and web applications. 10 Request message.
11 Response message. 12.2 Server response. The server, which provides
resources such as HTML files and other content, or performs other functions
on behalf of the client, returns a response message to the client. HTTP
provides multiple authentication schemes such as Basic access
authentication and Digest access authentication which operate via a
challenge-response mechanism whereby the server identifies and issues a
challenge before serving the requested content. HTTP provides a general
framework for access control and authentication, via an extensible set of
challenge-response authentication schemes, which can be used by a server to
challenge a client request and by a client to provide authentication
information. For example, WebDAV defined 7 new methods and RFC 5789
```

Fig. 2 Generated summary by Summarizer

Words and positions extracted to index from summary

```
Array
(
  [test site yioop hypertext transfer protocol wikipedia the free
encyclopedia] => Array
  (
    [0] => 0
    [cond_max] => 5
  )

  [site yioop hypertext transfer protocol wikipedia the free encyclopedia
4] => Array
  (
    [0] => 1
    [cond_max] => 5
  )

  [yioop hypertext transfer protocol wikipedia the free encyclopedia 4
http] => Array
  (
    [0] => 2
    [cond_max] => 6
  )

  [hypertext transfer protocol wikipedia the free encyclopedia 4 http
authent] => Array
  (
    [0] => 3
    [cond_max] => 10
  )

  [transfer protocol wikipedia the free encyclopedia 4 http authent 5]
=> Array
  (
    [0] => 4
    [cond_max] => 9
  )
)
```

Fig. 3 Example of mini Index on the Summary

The Question Answering (QA) system comes into the picture after generating phrases from summaries. These phrases are statements from the summary. The Question Answering (QA) system takes all the terms greater than 3 words. It iterates over each term and applies the Part of Speech tagger to get the context of the statement. After that, it generates the parse tree out of these tagged terms, which will be further explained in Chapter 4. This generated tree becomes the input to the triplet extraction process. These triplets are used to create the Question triplets, which are then stored to the Yioop index.

Fig. 4 describes the data flow information from queue server to the Yioop index. It also mentions the position of the Question Answering (QA) system in the entire process.

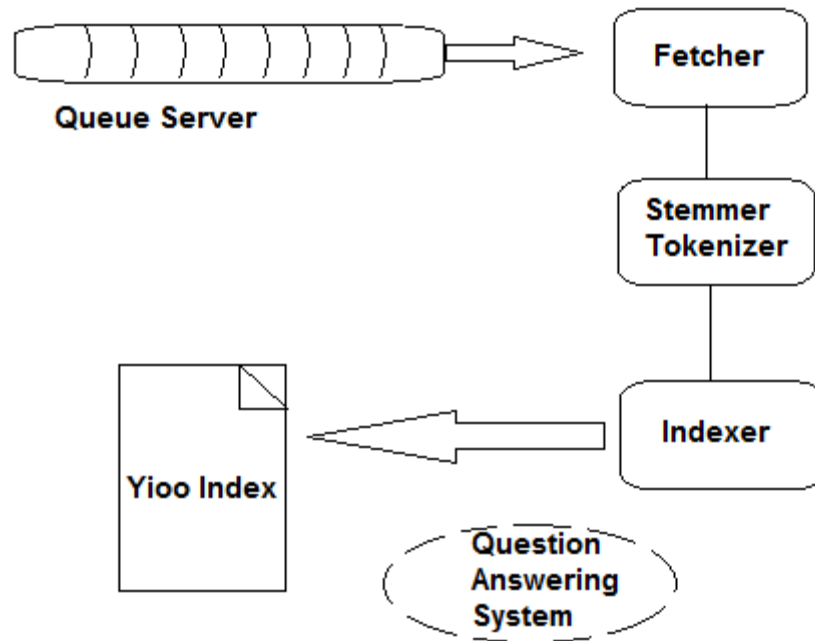


Fig. 4 Overview of information flow

As seen in Fig. 4, the Question Answering (QA) System is positioned between the indexer and the Yioop index. During run time, as soon as the user enters a query, it goes through a number of predefined processes before the actual index lookup. It also checks for the control words during this stage. The control words are used to select an index or a mix to use. Such control words may be m: or i: terms, they can also be other commands like “*raw*” and “*no.*” The *raw* command tells Yioop what level of grouping to use, whereas *no* conveys not to use a standard processing technique. It also checks for the query semantics. Keywords in the query are matched to this and then rewritten to some other query terms. For example, a query term existing in the domain name form is

rewritten to the meta word form, site: domain. Hence, only the pages from the domain are returned by the query [5]. It also applies the same stemmer and stop words removal used at the time of storing the inverted index, at the crawl time.

Yioop builds an iterator object from the resulting terms after executing the above steps. These terms are used to iterate over summaries and link entries containing all the terms. The Question Answering (QA) System comes into the picture only when the query type is a question and converts it to the Question triplet structure. This structure is similar to the one stored in the inverted index during the crawl process.

CHAPTER 3

QUESTION ANSWERING (QA) SYSTEM OVERVIEW

The Question Answering (QA) System will be a feature added to the Yioop Search Engine that would enable the search engine to respond to a question entered by a human in natural language. In order to build an efficient and accurate Question Answering (QA) System, one can take several approaches that can be used to fetch and store the data from the World Wide Web. One of the main challenges faced while developing a Question Answering (QA) system is the vast amount of data, leading to data redundancy, which can affect the accuracy of the Question Answering (QA) system. Another challenge faced is the variations encountered in the natural language entered by humans. The fundamental processing of a Question Answering (QA) System starts with the retrieval of the documents and ends when the sentence is structured into some kind of a template, along with the relation to that particular structure. This, in the end, can be used to respond to the query submitted to the search engine in the form of a question. The following subsections discuss in detail the approaches and the challenges faced by a developer of the basic Question Answering (QA) system.

3.1 Representation of a Question Answering (QA) System:

One of the major challenges of building such a Question Answering (QA) System is obtaining high accuracy. The primary reason behind low accuracy is the highly unstructured information on the World Wide Web. This haphazard structuralism can be due to linguistic barriers, poor grammatical skills, or even the tone of the text.

A good Question Answering (QA) system should be able to interpret and process, keeping these factors in mind, and it should store valuable information after the removal of unnecessary redundancies. It is also important to parse the unstructured document and store it efficiently. Efficient storage has a direct impact during the question lookup stage, as the user expects the answer to be shown within a fraction of a second. Following are the features that can lead to an efficient Question Answering (QA) system:

1. A speedy and accurate information retrieval scheme
2. A good and structured passage retrieval method
3. An accurate information extraction method that extracts the information from the passage statements.
4. An efficient storage of extracted information in an efficient way for a faster retrieval
5. An efficient way to interpret the user query correctly based on its semantics

The Question Answering (QA) System can be broadly classified into two major phases, namely:

1. The Pre-Computed Processing
2. The Real-Time Processing

Pre-Computed Processing involves parsing several thousands of HTML or XML pages and storing the necessary information in the form of triplets.

Real-Time Processing involves taking the input from the user in the form of a question and creating Question Triplets from this. It also responds to the user within a short span of time by extracting the information from the previously stored information.

3.2 Information Extraction approaches and its challenges

There are several approaches that one can use in order to build a Question Answering (QA) System. Several are briefly explained below.

3.2.1 The Bag of Words Approach

The Bag of Words approach is one of the most basic approaches used in the Question Answering (QA) System. Once the query is received from the user, the count of each term appearing in the query is noted down after stemming and stop words removal phase. The look up to the inverted index (Posting List) is purely based on the words in the query. However, this approach can easily fail with even a slight variation in the user query. An example of this might be the query, “Whom did Oswald kill?” It is imperative that the user wants to know who was killed by Oswald. However, with the approach of Bag of Words, the resultant output could be “Ruby killed Oswald” or “Oswald killed Ruby.” It is difficult to determine which would be the correct output of the query, as the important terms in the query, i.e. “Oswald” and “Kill” are both present in the resultant answers as well. Thus, this approach fails in this situation.

3.2.2 The Triplet Extraction Approach

The above approaches discussed in Section 3.2.1 fails to respond to the semantic variations of the user query. In order to overcome this, a better approach is to understand

the dependency between the subject and the object in the sentence. This can be achieved through a triplet extraction approach. Consider, for example, the sentence “The big man ate the dog” and the “The big dog ate the man”. Here, the triplet can be formed as [MAN (Subject), ATE (Predicate), DOG (Object)] for the first sentence while another triplet can be formed as [DOG (Subject), ATE (Predicate), MAN (Object)] for the second sentence. Now, in order to guess the question related to either of these sentences, one can replace either the subject, relation, or object in any of the triplets. For example, if the question asked is WHO ATE THE DOG? The answer can be easily derived by considering the question triplet as [WHO (Question), ATE (Predicate), DOG (Object)]. By matching this with the triplet of the first sentence, i.e. [MAN (Subject), ATE (Predicate), DOG (Object)], the answer can be derived as “The big man ate the dog.”

An improvement to this approach involves adding the synonyms of the Predicate in the triplet. For example, consider the sentence, “The first internal combustion engine was built in 1867.” Here the triplet is formed as [INTERNAL COMBUSTION ENGINE (Subject), BUILT (Predicate), 1867 (Object)]. Now, the synonyms of the predicate “built” can be “invent,” “create,” “constructed,” etc. If these synonyms are added to the list of predicates that can be suitable for this triplet, then this sentence can answer the question, “When was the internal combustion engine invented?” Without this additional feature, the Question Answering (QA) System would not have been able to map this question to its appropriate answer.

The triplet extraction approach is quite useful and is the one being used for this project. However, as with any other approach, this also has a few flaws. This approach

fails when the interpretation of the question is highly vague in nature. For example, if the query is “Where did the game Croquet originate?” The Question Answering (QA) System would not be able to provide the user with a definitive answer because the Wikipedia page for the game of croquet does not give a very definitive sentence answering the above query. The Wikipedia page for croquet has statements like:

- i) The first explanation is that the ancestral game was introduced to Britain from France during the...
- ii) The second theory is that the rules of the modern game of croquet arrived from Ireland during

This makes it difficult to answer such a question.

A good Question Answering (QA) System needs to take care of the ordering of the words in the question. For example, “What is Bill Gates’ net worth?” and “What is the net worth of Bill Gates?” are trying to ask the same question. Similarly, “When was Whatsapp acquired by Facebook?” and “When did Facebook acquire Whatsapp?” are two questions with the question words changed. These questions can be easily handled with the Triplet Extraction Approach [4].

However, the Triplet Extraction Approach fails when the question is directed towards a person or a thing. For example, if the query is “Who acquired Whatsapp?” the answer should be Facebook, however, since the Question Answering (QA) System does not know whether the question is asking for a person’s name or an entity’s name, it fails to give the correct answer. This problem can be solved by using the labeling technique. In this technique, the Question Answering (QA) System labels the subject or the object as a person, thing, entity, or any other category. This labeling technique can be helpful in

answering questions such as “Who acquired Whatsapp?” because the system now knows to look for an entity rather than a person in order to answer this question. Since Facebook would be labeled as an entity, the system would respond with “Facebook acquired Whatsapp.”

The Question Answering (QA) System needs to be able to evolve by collecting responses from the user. This can be done with a mechanism that can store the user feedback as a “Correct answer” or “Incorrect answer” after the result is returned by the Question Answering (QA) system. If the user marked the answer as incorrect, then the system needs to note it down and improve the answer to that query in the future.

The approach used for this project is the Triplet Extraction Approach, which seemed to be the best approach out of all that were found during the research.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF THE QUESTION ANSWERING SYSTEM

This chapter discusses more about the approach chosen for the Yioop Question Answering (QA) System. It includes both design and implementation of the approach. The discussion is divided into two main categories: 1. Role of the Question Answering (QA) System during the crawl time and 2. Role of the Question Answering (QA) System during the query execution time. Each part has its own processing components, which will be discussed in further detail.

4.1 Role of the Question Answering (QA) System during crawl time:

As mentioned in Chapter 2, the primary objective of the crawler is to crawl on the documents retrieved from the World Wide Web. It downloads the contents of the page, runs the summarizer on it, and produces concise information in the form of the summary. This summary is then processed by the mini indexer. It then creates an inverted index along with the information of an offset or the position of occurrence of words/phrases in the respective pages. The form of the obtained inverted index has already been discussed in the section 2.1.4.

The inverted index is accumulated and appended to the Phrased-List. This list consists of phrases extracted from the summary that contain only the stemmed words. These phrases do not contain the stop words. Now, there might be several phrases that have a single or double word as the phrase. There is a check imposed on the word count

of a phrase that makes sure that the phrase has at least 3 words. This check is necessary for the successful creation of a triplet to take place. By definition, a triplet is a combination of three words wherein the first word is the Subject, the second word is the Predicate and the third word is the Object. Thus, it becomes obvious that a phrase that has less than 3 words should not be considered for creating a triplet. The Question Answering (QA) System preserves the offset information, along with the phrase list created by the mini indexer. This offset information is assigned to the triplet generated from the respective statement or phrase. Once the triplet containing the subject, predicate and object is created, it looks something like this:

[<Subject>, <Predicate>, <Object>]

This triplet is then used to create a question triplet. The creation of Question Triplet helps to save run time while processing of the query parsing. The generated question triplets take the following forms:

[<Question>, <Predicate>, <Object>]

[<Subject>, <Question>, <Object>]

[<Subject>, <Predicate>, <Question>]

In order to better understand this, consider the example: “Alice chased the rabbit”

The triplet derived from this piece of information would be:

[Alice (Subject), chased (Predicate), rabbit (Object)]

Whereas the question triplets formed using this sentence would be:

[Who (Question), chased (Predicate), rabbit (Object)] → Who chased the rabbit?

[Alice (Subject), chased (Predicate), what (Question)] → What did Alice chase?

In order to create such question triplets, the sentence goes through four major components in the system, as shown below.

- 1) Part of Speech Tagger
- 2) Parse tree generation
- 3) Triplet Extraction
- 4) Question Triplet Generation

As shown above, first and foremost is the Part of Speech tagger, followed by the Parse Tree Generator, Triplet Extraction, and finally, the Question Statement Generation.

4.1.1 Major components of the Question Answering (QA) System

The following section gives a detailed overview about each component:

Part Of Speech (POS) Tagger

A part-of-speech tagger, or POS-tagger, processes a sequence of words and attaches a part of speech tag to each word. This process is also called a grammatical tagging or a word-category distribution. It is based on the relation of the word to its adjacent words. The ideal POS tagger should handle a complex statement and tag them with proper tags. However, this creates difficulty. There are numerous cases where the

POS tagger fails to tag the words properly, leading to incorrect tagging of words. A POS tagger has at least 10 tags to start off with, which are described in the table below.

TABLE 4.1 Part of Speech Tags [6]

Tag	Part of Speech	Example
ADJ	Adjective	new, good, high, special, big, local
ADP	Apposition	on, of, at, with, by, into, under
ADV	Adverb	really, already, still, early, now
CONJ	Conjunction	and, or, but, if, while, although
DET	Determiner, article	the, a, some, most, every, no, which
NOUN	Noun	year, home, costs, time, Africa
NUM	Numerical	twenty-four, fourth, 1991, 14:24
PRT	Particle	at, on, out, over per, that, up, with
PRON	Pronoun	he, their, her, its, my, I, us
VERB	Verb	is, say, told, given, playing, would
.	Punctuation	marks . , ; !
X	Other	ersatz, esprit, dunno, gr8, university

These are some of the basic tags that are used by the POS tagger. However, the number of tags can go up to 50-150, depending on the complexity of the sentence found in the information source.

There are two basic approaches that are employed for POS tagging:

1. Supervised Tagging
2. Unsupervised tagging

Supervised tagging includes a set of known words that are already mapped to a particular part of speech. For example, in supervised tagging, “is”, “was”, and “were” would always be tagged as verbs. The algorithm for this technique was developed by Ken Church and Steven DeRose [7].

Unsupervised tagging is type of intelligent tagging that tags the words based on pattern matching. It observes the use of each word and its place in the sentence, then decides what to tag to each word.

There are several well-known algorithms that are used for POS tagging, such as Viterbi algorithm [8], Brill Tagger algorithm [9], and Constraint Grammar algorithm [10].

Yioop uses the Brill tagger algorithm for POS tagging. It is considered to be an inductive method of tagging, which falls under the category of supervised tagging. This method first assigns the tags to the whole set of triplets that were derived from the data set. If the query contains the same word as found in the initial triplet, then the word is directly assigned the tag. If the word is not found in the predefined list, then the word is given the tag of a Noun during the first attempt. During the second iteration, an attempt is made to correct the initial tag of a noun based on a set of predefined rules. The iteration continues till the best results are achieved.

For example,

Statement “They refuse to permit us to obtain the refuse permit”, a part of speech tagger will tag the statement word as [(‘They’, ‘PRP’), (‘refuse’, ‘VBP’), (‘to’, ‘TO’), (‘permit’, ‘VB’), (‘us’, ‘PRP’), (‘to’, ‘TO’), (‘obtain’, ‘VB’), (‘the’, ‘DT’), (‘refuse’, ‘NN’), (‘permit’, ‘NN’)].

Another statement “Alice chased the rabbit” can be tagged as [(‘Alice’, ‘NP’), (‘chased’, ‘V’), (‘the’, ‘Det’), (‘rabbit’, ‘N’)] where NP is a proper noun, V is a verb, Det is a determiner, N is a Noun.

The part of speech tagger helps with recognizing the context at a basic level by tagging words which can be either an adjective or verb. This information is required for the Tree generation.

Parse Tree Generation

This component will have the output statement from the POS tagger, which also has all the tags associated with each word. In order to determine the relationship between these tags, there are many information chunking techniques available.

In English, most of the statement follows a syntactical structure of Noun Phrase (NP) + Verb Phrase (VP). To identify and extract the noun phrase from the sentence, the basic grammar rule followed is (Refer [15] for tag set):

NP: {<DT|JJ|NN.*>+} #extract the sequences of DT, JJ, NN

So for the statement “The little yellow dog barked at the cat”, part of speech output is: the-DT little-JJ yellow-JJ dog-NN barked-VBD at-IN the-DT cat-NN. Here, by applying the grammar rule, it is easy to extract the “The little yellow dog” as a noun phrase for our statement. To cover more in-depth details and a complex structure, we can also apply one more grammatical step when extracting the Noun phrase: adding the preposition rule to get the information following the noun phrase. For a Question Answering (QA) system in Yioop, this rule is considered as a part of the Noun Phrase only. However, it can be easily expanded by following the grammar given below:

PP: {<IN><NP>} # Extract prepositions followed by NP

In order to discuss the verb phrase extraction, a simple verb phrase follows the grammatical rule, as below (Refer [15] for tag set):

VP: {<VB.*><NP|PP|CLAUSE>+\$} # Extract verbs and noun phrases

As shown above, this rule consists of a verb and a noun phrase. So for the Question Answering (QA) System in Yioop, it is simplified to (Refer [15] for tag set):

VP: {<VB.*><NP>\$} # Extract verbs and noun phrases

Here, a verb will help in identifying the relationship between the subject and the object. It conveys the dependency between the subject and the object. Identifying the verb will help the use of synonyms or a chain of words in the future. For example, consider a statement like “Roman engineers built the first combustion engine.”

The output of the part of speech tagger is as follows:

Roman-NNP engineers-NNS built-VBN the-DT first-NNP combustion-NN engine-NN

If we apply the extraction rules presented above, it gives us “built” as the verb in our Verb phrase. So keeping a chain of synonyms like built -> invented -> manufactured can help increase the search and lookup quality for the information. Further in Verb Phrase, the second part will be the Noun Phrase with the same grammar rule presented above. The extracting process can either be carried out via regular expression or by the recursive decent parser, which can be represented by a tree structure.

Regular expression can cover most of the cases but might generate errors for some complex recursive structure of statements. Another approach is to extract using a recursive decent parser based on the grammar rules presented above. For Yioop, it makes use of the recursive decent parser. It does so by recursively parsing a statement and assigning it to the appropriate parent. Each leaf of the node is a word of the statement. The path to the leaf from the root gives the grammar rule that assigns the word to the appropriate parent.

The above creates the basic structure of a tree from the sentence and creates the base for the Triplet Extraction process. In order to create the triplet extraction algorithm explained in the paper, it requires a parse tree generated from the statement wherein each leaf node is labeled with a word from the statement. A Treebank is a text corpus where each sentence belonging to the corpus has a syntactic structure added to it.

There is a method of how to construct the tree from the sentence in the paper titled TRIPLET EXTRACTION FROM SENTENCES [4].

The summary of the set of rules identified and implemented in this Question Answering (QA) system is as below [11][15].

CLAUSE: {<NP><VP>}	# Extract NP, VP
NP: {<DT JJ NN.*>+}	# Extract sequences of DT, JJ, NN
VP: {<VB.*><NP PP CLAUSE>+\$}	# Extract verbs and their arguments

Consider a simple example statement “Alice chased the rabbit.” Part speech tags this statement as:

Alice-NNP chased-VBN the-DT rabbit-NN

The tree generated by applying the above set of rules is as below:

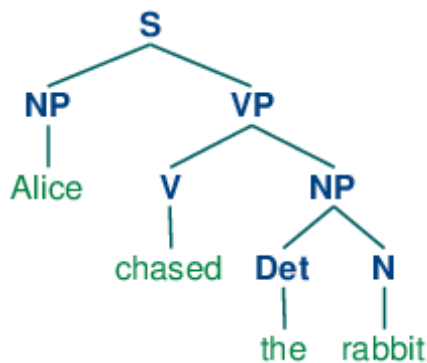


Fig. 5 Generated Parsed Tree

Another complex example statement that can be covered by the recursive decent parser is: “A rare black squirrel has become a regular visitor to a suburban garden.” The generated tree for the statement is as below:

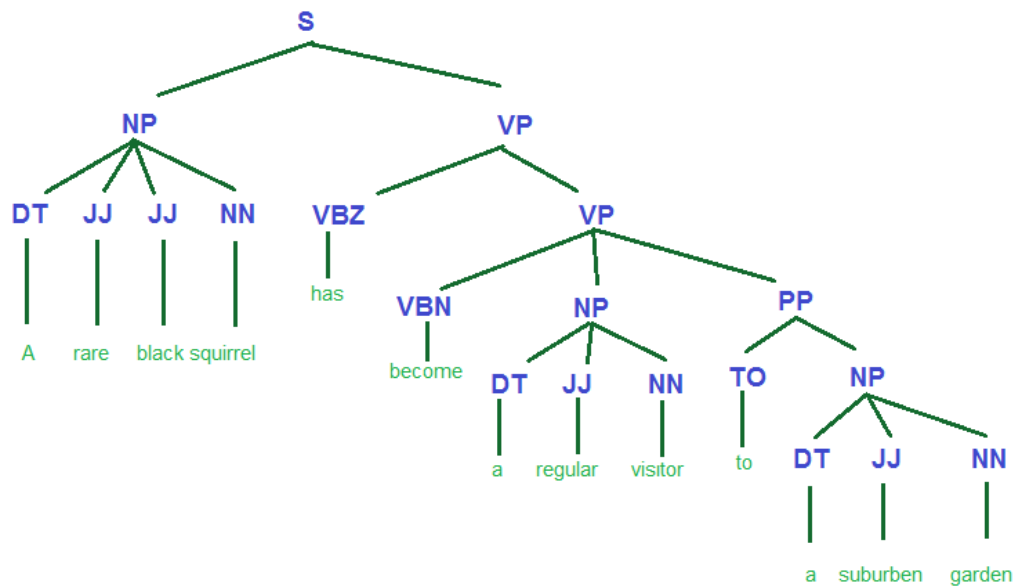


Fig. 6 Parse tree generated from the complex sentence

The advantage of recursive decent parser is the ability to cover complex statements. It requires a bit of modification to cover special cases. In Yioop, if the tree generation from the statement fails, it falls back and processes the next phrase in the list. This helps with discarding the erroneous triplets.

Triplet Extraction from the TREE

This component takes the parsed tree generated above as the input and attempts to get the triplet in the form of [SUBJECT – RELATION – OBJECT]. The purpose of the

triplet extraction is to identify the context in terms of how the subject is related to the object.

This algorithm follows the same approach presented in the paper [6]. It is broadly divided into 3 forms. Extracting the Subject from the tree gets the main noun from the statement. Extracting the Predicate from the tree gets the dependency from the statement. Extracting the object from the tree gets the object on which the subject depends.

function TRIPLET-EXTRACTION(sentence) returns a solution, or failure

result \leftarrow *EXTRACT-SUBJECT(NP_subtree)*

U EXTRACT-PREDICATE(VP_subtree)

U EXTRACT-OBJECT(VP_siblings)

if result \neq *failure* *then return result*

else return failure [4]

function EXTRACT-ATTRIBUTES(word) returns a solution, or failure

```
// search among the word's siblings
if adjective(word)
  result ← all RB siblings
else
  if noun(word)
    result ← all DT, PRP$, POS, JJ,
              CD, ADJP, QP, NP siblings
else
  if verb(word)
    result ← all ADVP siblings

// search among the word's uncles
if noun(word) or adjective(word)
  if uncle = PP
    result ← uncle subtree
  else
    if verb(word) and (uncle = verb)
      result ← uncle subtree

if result ≠ failure then return result
else return failure [4]
```

function EXTRACT-SUBJECT(NP_subtree) returns a solution, or failure

```
subject ← first noun found in NP_subtree
subjectAttributes ← EXTRACT-ATTRIBUTES(subject)

result ← subject UsubjectAttributes

if result ≠ failure then return result
else return failure [4]
```

function EXTRACT-PREDICATE(VP_subtree) returns a solution, or failure

```
predicate ← deepest verb found in VP_subtree  
predicateAttributes ← EXTRACT-ATTRIBUTES(predicate)  
  
result ← predicate UpredicateAttributes  
  
if result ≠ failure then return result  
else return failure [4]
```

function EXTRACT-OBJECT(VP_sbtree) returns a solution, or failure

```
siblings ← find NP, PP and ADJP siblings of VP_subtree  
  
for each value in siblings do  
    if value = NP or PP  
        object ← first noun in value  
    else  
        object ← first adjective in value  
  
    objectAttributes ← EXTRACT-ATTRIBUTES(object)  
  
result ← object UobjectAttributes  
if result ≠ failure then return result  
else return failure [4]
```

The above algorithms are explained part by part, as discussed below.

Subject Extraction

In the parse tree generated, we search for the first level of Noun phrase from the tree to extract the subject, as each statement is assumed to be made up of Noun Phrase + Verb Phrase. Subject extraction looks only into the Noun Phrase subtree. By reaching the Noun phrase, the process performs the search to extract all the children of the Noun Phrase subtree. The main noun can be tagged under the parent having the label NN, NNP, NNPS, NNS. These labels are based on the grammar rule presented above. While

considering a statement in a natural language, the noun can be followed by different attributes. In our case, it can be of form of determiner, an adjective that complements the information attached to the noun. This information needs to be stored along with the actual noun, as it helps increase the match. In some cases, a noun might have a different context than it should. In those cases, the information in the form of the attributes helps to identify the context of the statement. In major cases, if the subject is pointed to the person, it can be used to answer the question asked by “WHO.” For example, when a statement is like: “Barack Obama is the president of the United States,” a triplet extraction technique identifies “Barack Obama” as a subject. This subject can be the answer of an expected question such as: “Who is the president of the United States of America?”

Predicate Extraction

The goal of the predicate extraction is to determine the relation between the subject and the object. It portrays how the subject is dependent on the object. This can be found in the verb phrase subtree of the statement. It looks for the deepest verb descendent of the verb phrase. This gives the second element of the triplet. They are marked with a parent that has one of the values from VB, VBD, VBG, VBN, VBP, VBZ. There can be several attributes attached to the verb as well. These attributes are extracted and stored along with the predicate. The verb has a minor impact in providing the specific information, but it plays a major role in sensing the context of the statement. There can be many instances where a different subject is related to the same object. But to determine this relationship, the information in the predicate turns out to be a deciding factor.

Object Extraction

The third part of the tree generation is the object extraction. Similar to a subject in deciding the nouns of the statement, the object from the statement also plays a major role, as discussed below. As mentioned in section 4.1.1, objects are found in the noun phrase of the verb phrase of the statement as the sentence is divided into a noun phrase and a verb phrase. The verb phrase is further divided into a verb phrase and a noun phrase. This noun phrase part is the one that provides the object to the triplet, along with the attributes. Most of the questions that can be asked by what/where/who can be answered by the object part. The attributes of the object also hold importance for returning the correct results. For example, “The bank of the river” and “The bank on the river” have the attribute values “OF” and “ON” that can change the meaning of the statement.

A complete example of Triplet Extraction

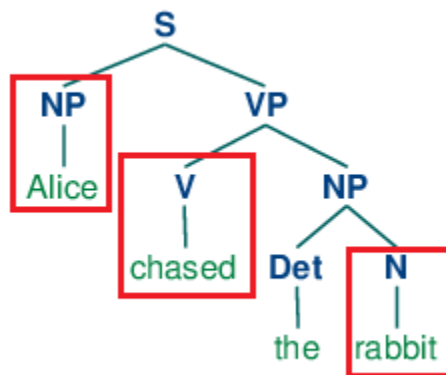


Fig. 7 Triplet Extraction from the Tree

Fig. 7 shows the tree generated from the implementation of the following sentence: “Alice chased the rabbit.” Part-of-Speech tagger tags this statement as

[("Alice", NP), ("chased", V), ("the", Det), ("rabbit", N)]. Applying a triplet extraction algorithm to this, it extracts [ALICE – CHASED – RABBIT] as [SUBJECT-PREDICATE-OBJECT] triplet.

As previously discussed, the subject, object, and predicate can have attributes. To make use of these attributes, the Question Answering (QA) System generates two versions of any triplet. The first one is the RAW triplet, whereas the second is the FEATURED triplet. The FEATURED triplet contains more information regarding the triplets, thus, it helps with matching the query accurately. Due to this, during query time, the FEATURED triplet takes a higher precedence.

For example, consider a statement: "The big man ate the dog."

Triplet generation generates two variations of triplets as follows:

Raw triplet: [man – ate – dog]

Feature triplet: [the big man - ate – the dog]

Question triplet generation from Triplet

Storing the information on the index is highly dependent on the generation of questions from the triplet. The question triplet formed is stored in Yioop's index. The purpose of generating the Question Triplet is to guess the questions that can be asked by the user. However, a triplet can be interpreted as various forms of questions. For example, triplets like "[CHRIS - RUNS – IN THE BACKYARD]" and "[CHRIS – RUNS -IN THE MORNING]", would lead to questions such as "Where does Chris run?"

And “When does Chris run?” In order to accurately answer such questions, the triplet can be assigned tags that differentiate them on the basis of time, place, person, etc. Thus, for the triplets mentioned above, if the OBJECT [IN THE MORNING] is assigned a tag of time and the OBJECT [IN THE BACKYARD] is assigned the tag of place, it will be easier to achieve an accurate result.

The Question Answering (QA) system takes a basic approach towards generating question triplets. The generated triplet has the format of [SUBJECT – PREDICATE - OBJECT]. The user query, which will be in the form of a question, is expected to have at least two of the triplet attributes. That is, a question will have either a combination of the [SUBJECT, PREDICATE] or [SUBJECT, OBJECT] or [PREDICATE, OBJECT].

Consider the statement: “George Washington is the first president of USA.” After applying POS tagger, parse tree generator, and the triplet extraction on this sentence, the resulting triplet is [GEORGE WASHINGTON – IS – THE FIRST PRESIDENT OF USA]. The question triplet generated from the triplet has 3 variations, such as [GEORGE WASHINGTON – IS - QUE], [GEORGE WASHINGTON – QUE - THE FIRST PRESIDENT OF USA] and [QUE – IS - THE FIRST PRESIDENT OF USA]. Here, “QUE” is any identifier that is used to identify the Question tag. Thus, the expected questions from the statement can be either “Who is the first president of USA?” where “Who” is the QUE identifier and the question contains the PREDICATE and the OBJECT, or “Who is George Washington?” which contains the SUBJECT and the PREDICATE.

The question triplet generated above also contains the information of the offset of the original source statement in the summary. Thus, the above flow can be summarized as:

1. User asks the question “Who is George Washington?”
2. The query parser converts it to [QUE – IS – THE FIRST PRESIDENT OF USA].
3. Yioop looks up this triplet in the index.
4. Once a match is found, the source information is returned by using the offset information.

4.2 Role of Question Answering (QA) System during Query Time:

During query time, the user is expected to ask a question in natural English. At the time of query parsing, a check has been imposed to identify whether the query entered by the user is a question or not. Question Answering (QA) system starts the query processing only if it identifies the input as a question. This identification involves checking whether a query entered by the user starts with any “WH-” question. If it does not, then the Question Answering (QA) system query parser is skipped for that query.

The natural language used in the question leads to several forms of the same. Question Answering (QA) System tries to handle such variations. For example, a simple question like “Who founded Apple?” can have many variations such as “Who was the founder of Apple?”, “Who were the founders of Apple?” and many others.

In order to process this question, the Question Answering (QA) System applies the POS tagger. The POS tagger tags the question words like WDT, WP, WP\$, or WRB. As of now, the Question Answering (QA) System handles the WHO question in a different manner than it does WH+ questions.

CHAPTER 5

EXPERIMENTS

This chapter discusses the experiments carried out on the developed Question Answering System. These experiments were divided into two main parts:

- 1) Experiments on a standalone Question Answering (QA) system
- 2) Experiments on an integrated Question Answering (QA) system

5.1 Experiments on a standalone Question Answering (QA) system

The experiments on a standalone Question Answering System give an idea about the system's performance under ideal conditions. Here, an assumption is made that the statements provided to the system are valid.

One of the few experiments carried out on the system is for a simple statement like “Kim Kardashian is a social media personality.” The Question Answering (QA) system generates a question triplet, as shown below (Refer Fig. 8). This triplet can be helpful in answering questions like “Who is Kim Kardashian?” Another such statement was “Narendra Damodardas Modi is the Prime Minister of India”. The question “Who is the Prime Minister of India?” can be easily answered by the triplet stored at the Yioop index that pertains to the statement (Refer Fig. 9). A more complex example statement like “Chris accepted the position of Vice Chairman of the University” would give the Predicate as “Accepted.” So, the question “Who accepted the position of vice chairman of the University?” can be addressed by the triplet generated below (Refer Fig. 10). The

same theory applies to the statement “The little yellow dog barked at the cat”. The triplets were successfully generated (Refer to Fig. 10). Thus, Triplet generation is a process that is very useful for storing relations and dependencies to the index.

The concept can also be applied to question statements. For example, a question statement like “Who is Kim Kardashian?” generates the triplet as [KIM KARDASHIAN – IS – QUE] after query processing. This triplet is the same as the one stored on the Yioop index.

Example 1:

Kim Kardashian is a social media personality.

Part Of Speech Tagger Output:

Kim–NNP Kardashian–NN is–VBZ a–DT social–JJ media–NNS personality–NN.

Generated tree:

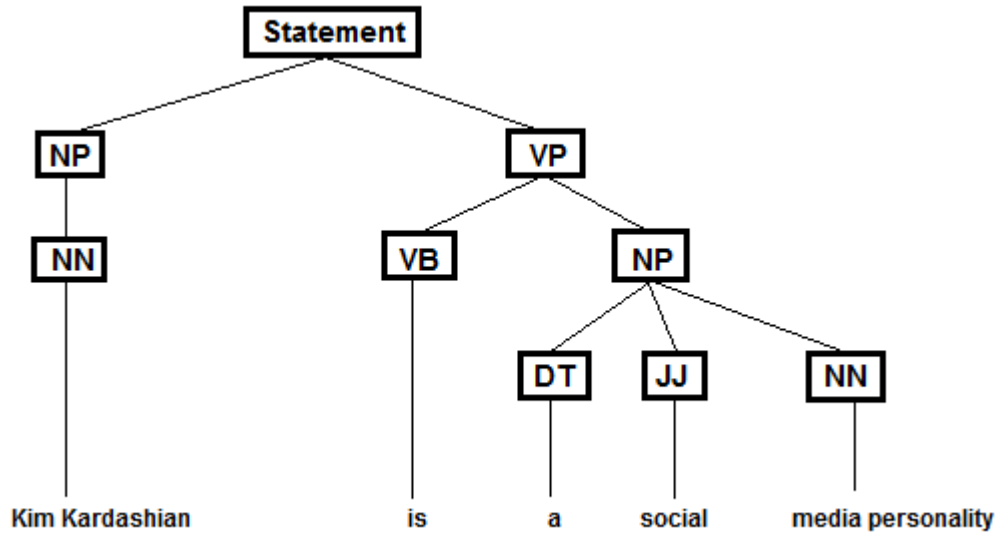


Fig. 8 Generated Tree from the statement

Triplet Extracted:

[KIM KARDASHIAN – IS – A SOCIAL MEDIA PERSONALITY]

Question Generated from the statement:

[“QUE”-IS-A SOCIAL MEDIA PERSONALITY]

[KIM KARDASHIAN – “QUE” – A SOCIAL MEDIA PERSONALITY]

[KIM KARDASHIAN – IS – “QUE”]

Example 2:

Narendra Damodardas is the prime minister of India

Part of Speech Tagger Output:

Narendra-NN Damodardas-NNS is-VBZ the-DT prime-JJ minister-NN of-IN India-NNP

Generated tree:

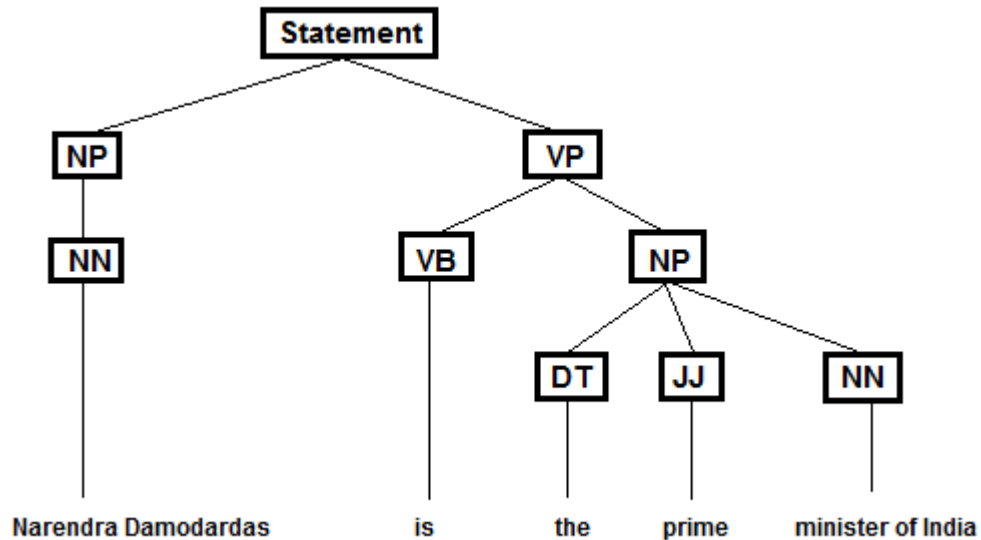


Fig. 9 Generated Tree from the statement

Triplet Extracted:

[Narendra Damodardas - is - the prime minister of India]

Question Triplets:

[“QUE” - is - the prime minister of India]

[Narendra Damodardas - “QUE” - the prime minister of India]

[Narendra Damodardas - is - “QUE”]

Example 3:

Chris accepted the position of vice chairman of the University

Part of Speech Tagger Output:

Chris-NNP accepted-VBN the-DT position-NN of-IN vice-NN chairman-NN of-IN the-DT University-NN

Generated tree:

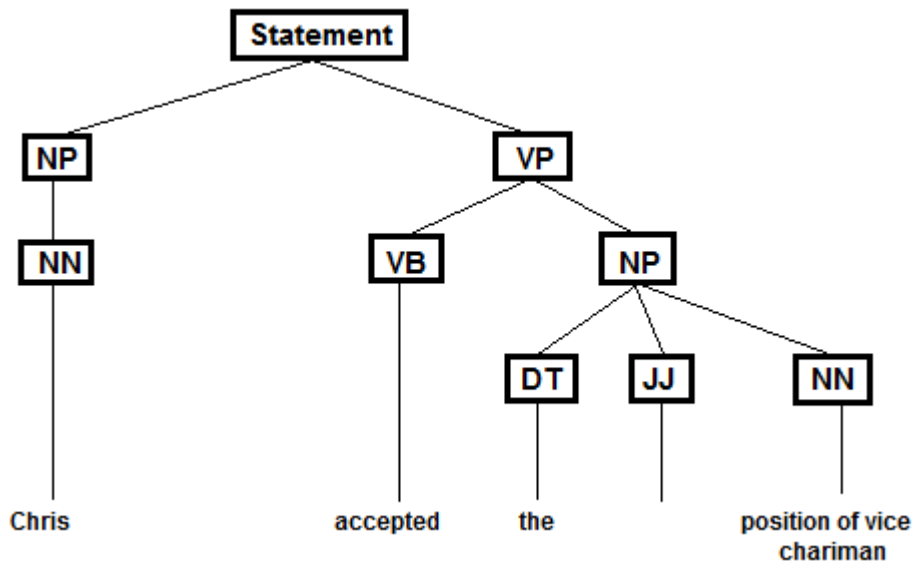


Fig. 10 Generated Tree from the statement

Triplet Extracted:

[CHRIS – ACCEPTED – THE POSITION OF VICE CHAIRMAN]

Question Triplets:

[“QUE” – ACCEPTED – THE POSITION OF VICE CHAIRMAN]

[CHRIS – “QUE” – THE POSITION OF VICE CHAIRMAN]

[CHRIS – ACCEPTED – “QUE”]

Example 4:

The little yellow dog barked at the cat

Part Of Speech Tagger Output:

The-DT little-JJ yellow-JJ dog-NN barked-VBD at-IN the-DT cat-NN

Generated Tree:

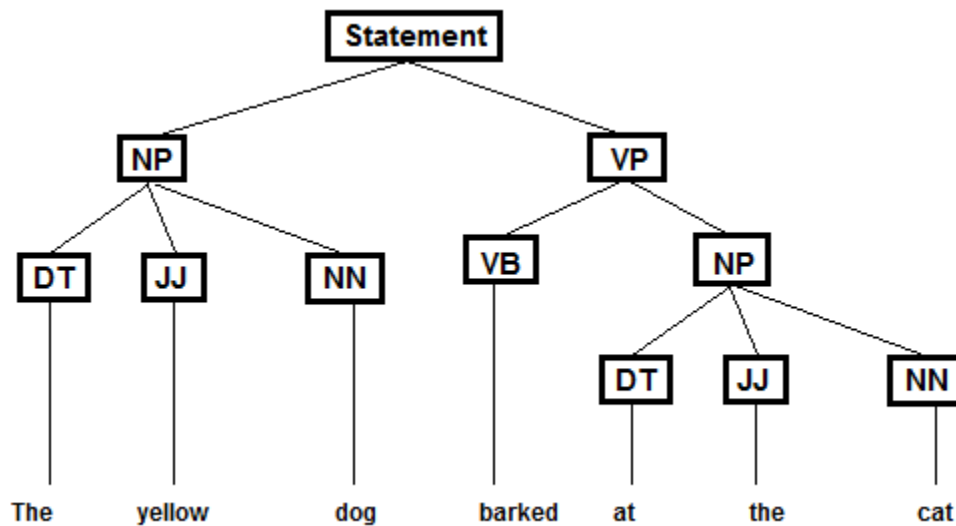


Fig. 11 Generated Tree from the statement

Triplet Extracted:

[THE LITTLE YELLOW DOG – BARKED – AT THE CAT]

Question Triplet:

[“QUE” – BARKED – AT THE CAT]

[THE LITTLE YELLOW DOG – “QUE” – AT THE CAT]

[THE LITTLE YELLOW DOG – BARKED – “QUE”]

Example 5 (Question Statement):

“Who is Kim Kardashian?”

Part Of Speech Tagger Output:

Who-WP is-VBZ Kim-NNP Kardashian-NN.

Triplet Generated:

[KIM KARDASHIAN – IS - WHO] =>[KIM KARDASHIAN – IS - QUE]

5.2 Experiments on an integrated Question Answering (QA) system

The experiments carried out after integrating the Question Answering (QA)

System with Yioop are divided into two parts:

1. Experiments during the crawl time
2. Experiments during the query time

5.2.1 Experiments during crawl time

To test the Question Answering system during crawl time, one needs to follow these steps:

- 1) Login to Yioop as an admin
- 2) Open the web interface to test the Mini Indexer through page option, as shown in Fig. 1.
- 3) Copy the source of any webpage
- 4) Click the test process page.
- 5) Check the section “Words and positions extracted to index from summary” in the output

The output provides the triplets generated from the summary, along with the offset information.

Consider an example: the Wikipedia page of “Kunal Nayyar,” which can be found at https://en.wikipedia.org/wiki/Kunal_Nayyar. Copy the source of the web page to Yioop’s test page option. After clicking the test source page, the snapshot of the section in the output is as below:

```

)
[he qqque st columba s school] => Array
(
  [0] => 87
  [cond_max] => 3
)
[he attend qqque] => Array
(
  [0] => 87
  [cond_max] => 3
)
[qqque attend st columba s school in new delhi] => Array
(
  [0] => 86
  [cond_max] => 6
)
[he qqque st columba s school in new delhi] => Array
(
  [0] => 87
  [cond_max] => 3
)
[he qqque third] => Array
(
  [0] => 52
  [cond_max] => 3
)
[he is qqque] => Array
(

```

Fig. 12 Generated Question Triplet at crawl time

A few of the triplets generated from the summary include [HE – QQQUE – ST COLUMBA S SCHOOOL IN NEW DELHI] and [QQQUE – ATTEND – ST COLUMBA S SCHOOL IN NEW DELHI]. These triplets can be used to answer questions like, “Who attended St. Columba’s school in New Delhi?” or “Which school did Kunal Nayyar attend?” The “QQQUE” is the identifier used in the same context of the QUE tag explained earlier. Take as our next example the web page: “https://en.wikipedia.org/wiki/Parul_Institute_of_Engineering_and_Technology”. Copy

the source of this webpage. After running the test process page, a list of triplets, along with offset information, will be provided as below:

```

    [0] => 60
  )
[parul univers is qqque] => Array
(
  [cond_max] => 14
  [0] => 60
)
[parul univers qqque locat in vadodara] => Array
(
  [cond_max] => 14
  [0] => 60
)
[qqque is approv] => Array
(
  [0] => 84
  [cond_max] => 3
)
[parul institut qqque approv] => Array
(
  [cond_max] => 24
  [0] => 85
)
[parul institut is qqque] => Array
(
  [cond_max] => 24
  [0] => 85
)
[qqque is approv by all india] => Array
(
  [cond_max] => 24
  [0] => 85
)
[parul institut of engin technologi qqque approv by all india] =>
Array

```

Fig. 13 Generated Question Triplet from the crawling

The generated triplets are [PARUL INSTITUT – IS - qqque] and [qqque – IS – APPROV BY ALL INDIA], along with the offset information.

5.2.2 Experiments during query time

Considering the example of Parul University’s webpage, as mentioned above. One of the Question Triplets that was stored to the Yioop’s index was [PARUL

UNIVERSITY – qqque – LOCAT IN VADODARA]. If the question “Where is Parul University located?” is asked, Yioop gives the response shown in Fig. 16.

5.3 Observations after Integration

In order to test the system after the integration, start the crawler by specifying urls in the seed site section located under the Crawl Options. For testing purpose, crawler option is given a seed site as https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. The index is then set after crawling over 451 websites. Fig. 15 shows the existing behavior of Yioop, which provides a list of urls along with the description as an answer to the question. Whereas Fig. 16 shows the behavior of Yioop, which provides answer to the question entered by the user with minor error.

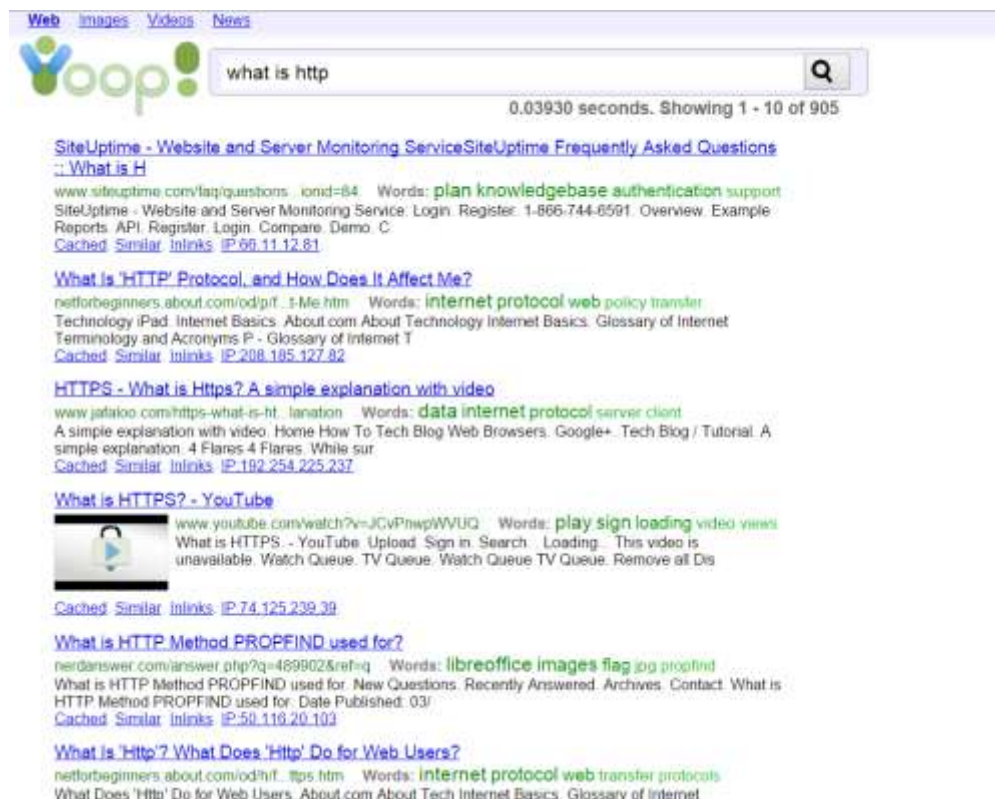
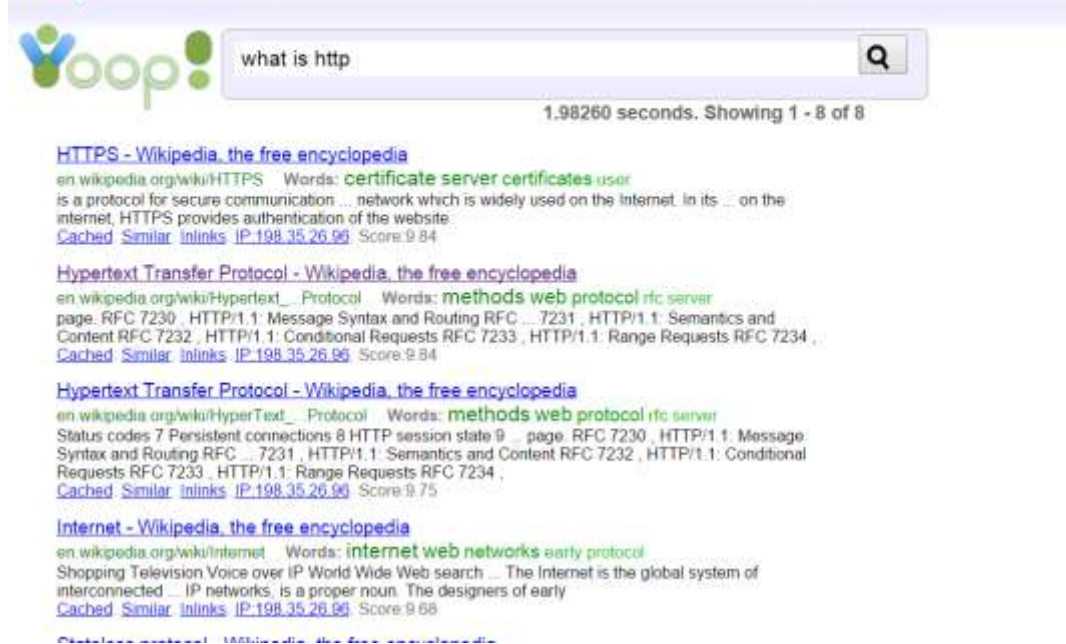


Fig. 14 Results before integration

Answer:a protocol for secur commun
Answer:a stateless protocol
Answer:a stateless protocol
Answer:the languag us on the web
Answer:an exampl of a stateless protocol layer
Answer:a stateless protocol
Answer:a stateless protocol
Answer:rpc



The screenshot shows a search engine interface with the Yioop logo on the left. A search bar contains the text "what is http" and a magnifying glass icon. Below the search bar, it displays "1.98260 seconds. Showing 1 - 8 of 8". The search results are listed below, each starting with a blue link to a Wikipedia page:

- [HTTPS - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/HTTPS Words: certificate server certificates user
is a protocol for secure communication ... network which is widely used on the Internet. In its ... on the internet, HTTPS provides authentication of the website.
Cached Similar Inlinks IP:198.35.26.96 Score:9.84
- [Hypertext Transfer Protocol - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Hypertext_... Protocol Words: methods web protocol rfc server
page. RFC 7230 , HTTP/1.1: Message Syntax and Routing RFC ... 7231 , HTTP/1.1: Semantics and Content RFC 7232 , HTTP/1.1: Conditional Requests RFC 7233 , HTTP/1.1: Range Requests RFC 7234 ,
Cached Similar Inlinks IP:198.35.26.96 Score:9.84
- [Hypertext Transfer Protocol - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/HyperText_... Protocol Words: methods web protocol rfc server
Status codes 7 Persistent connections 8 HTTP session state 9 ... page. RFC 7230 , HTTP/1.1: Message Syntax and Routing RFC ... 7231 , HTTP/1.1: Semantics and Content RFC 7232 , HTTP/1.1: Conditional Requests RFC 7233 , HTTP/1.1: Range Requests RFC 7234 ,
Cached Similar Inlinks IP:198.35.26.96 Score:9.75
- [Internet - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Internet Words: internet web networks early protocol
Shopping Television Voice over IP World Wide Web search ... The Internet is the global system of interconnected ... IP networks, is a proper noun. The designers of early
Cached Similar Inlinks IP:198.35.26.96 Score:9.68
- [Stateless protocol - Wikipedia, the free encyclopedia](#)

Fig. 15 Results after integration

The implication of the integration has its major effect on the performance of the crawler. The triplet generation process takes place before storing processed data to Yioop's index that results in more processing time. The generated triplet from each document results in more space. Fig. 17 shows the time difference of crawling process for individual pages.

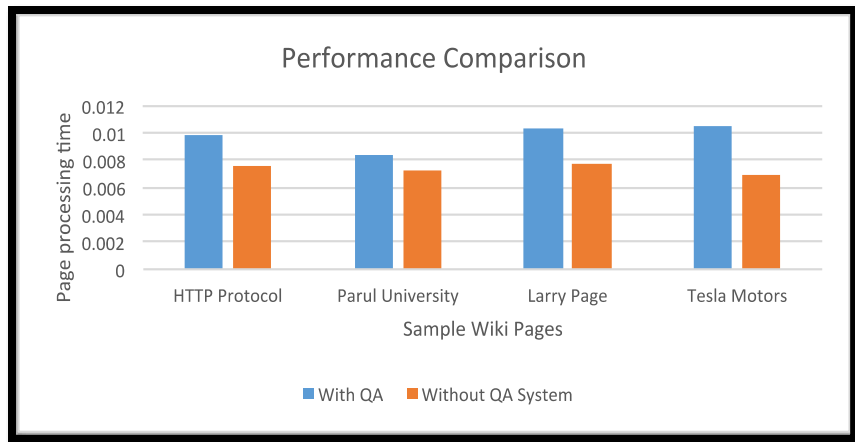


Fig. 16 Page processing time before/after the integration

The approach that is considered uses more space but it does not alter the query time performance. Question Answering System stores the triplet on Yioop's index, so that whenever a question is asked, it looks up directly and provides the answers from the mapping of the triplets.

CHAPTER 6

CONCLUSION

In this project, a new module called the Question Answering System is developed for Yioop. This module is responsible for storing information in a manner where it can be later used in answering questions asked by the user. This system employs a different approach to processing queries written in the form of a question.

The Question Answering (QA) system processes information available on the World Wide Web. It tries to handle the variations of natural language in the questions asked by the user and stores the result in Yioop's Index. This system also takes care of any question that may require a specific answer. This system will help Yioop to process questions asked by the user.

Currently, the components of the Question Answering system are unable to handle a wide range of variation in the query. Future work can be done to improve each component so that the overall efficiency of the system increases.

REFERENCES

- [1] Question answering. (2015). Retrieved from http://en.wikipedia.org/wiki/Question_asking
- [2] Ibm.com, 'What is IBM Watson?', 2015. [Online]. Available: <http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html>. [Accessed: 30- Nov- 2015].
- [3] Start.csail.mit.edu, 'The START Natural Language Question Answering System', 2015. [Online]. Available: <http://start.csail.mit.edu/index.php>. [Accessed: 30- Nov- 2015].
- [4] Rusu, D., Dali, L., Fortuna, B., & Grobelnik, M. & Mladenic, D. (N.D.). *Triplet extraction from sentences*, pp. 8-12, 2007.
- [5] Seekquarry.com, 'Resources', 2015. [Online]. Available: <https://www.seekquarry.com/p/Resources>. [Accessed: 09- Sep- 2015].
- [6] Umiacs.umd.edu, 'BrillTagger', 2015. [Online]. Available: <http://www.umiacs.umd.edu/~jimmylin/downloads/brill-javadoc/edu/mit/csail/brill/BrillTagger.html>. [Accessed: 13- Mar- 2015].
- [7] S. DeRose, 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics*, vol. 14, no. 1, pp. 31-39, 1988.

- [8] G. Forney, 'The viterbi algorithm', Proceedings of the IEEE, vol. 61, no. 3, pp. 268-278, 1973.
- [9] Phpir.com, 'Part Of Speech Tagging', 2015. [Online]. Available: <http://phpir.com/part-of-speech-tagging/>. [Accessed: 30- Nov- 2015].
- [10] N. Lindberg and M. Eineborg, 'Learning Constraint Grammar-style disambiguation rules using Inductive Logic Programming', Aclweb.org, 2015. [Online]. Available: <http://www.aclweb.org/anthology/P98-2128>. [Accessed: 16- Jul- 2015].
- [11] Nltk.org, '7. Extracting Information from Text', 2015. [Online]. Available: <http://www.nltk.org/book/ch07.html>. [Accessed: 01- Apr- 2015].
- [12] Nlp.stanford.edu, 'Stemming and lemmatization', 2015. [Online]. Available: <http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>. [Accessed: 02- May- 2015].
- [13] Katz, B., Lin, J., Loreto, D., Hildebrandt, W., Bilotti, M., Felshin, S., Marton, G. & Mora, F. (2003). Integrating web-based and corpus-based techniques for question answering. *In proceedings of the twelfth text retrieval conference* (pp. 426-435)
- [14] C. Manning, 'Text-based Question Answering systems', 2015. [Online]. Available: <http://web.stanford.edu/class/cs224n/handouts/cs224n-QA-2013.pdf>. [Accessed: 08- Sep- 2015]

[15] Umiacs.umd.edu, 'BrillTagger', 2015. [Online]. Available:

<http://www.umiacs.umd.edu/~jimmylin/downloads/brill->

[javadoc/edu/mit/csail/brill/BrillTagger.html](http://www.umiacs.umd.edu/~jimmylin/downloads/brill-javadoc/edu/mit/csail/brill/BrillTagger.html). [Accessed: 13- Dec- 2015].