NATURAL LANGUAGE TECHNIQUE FOR WEB DRIVEN INTERFACE

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Shashin S. Patel
Summer 2012
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the
Thesis of Shashin S. Patel:

Natural Language Technique for Web Driven Interface

Joseph Lewis, Chair
Department of Computer Science

Carl Eckberg
Department of Computer Science

Peter Blomgren
Department of Mathematics and Statistics

July 2, 2012
Approval Date
DEDICATION

I dedicate this thesis work to my dear family, for their constant encouragement and their willing to provide unconditional support every time I needed.
ABSTRACT OF THE THESIS

Natural Language Technique for Web Driven Interface

by
Shashin S. Patel
Master of Science in Computer Science
San Diego State University, 2012

Natural Language Processing is a domain of research which becomes backbone for Question and Answering application. This interface is retrieving answers to questions, which retrieves from application backend. It leads us to new domain of developing a system which has potential to become useful application.

This application comes with higher relatively between Questions and answers. It also gives interactive user interface so that user can easily communicate with it. I took the approach of designing a question answering system that is based on tagging and chunking algorithm, keyword fetch and statistical approach to match keywords classification. Question classification extracts useful information from the question about how to answer the question. Tagging extracts useful information, which will be used in finding the answer to the question. We used different approach to tag the documents. Currently our system classifies the questions using manually developed rules.

I also investigated Natural Language Processing algorithms which can use various methods to answer questions and come up with implementation of Maximal Likelihood algorithm. This thesis also includes investigations into modules of a question answering system and gives insights into how to go about developing a question answering system based on tagging and chunking.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT ..........................................................</td>
</tr>
<tr>
<td>LIST OF FIGURES ...............................................</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS .............................................</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION ..................................................</td>
</tr>
<tr>
<td>1.1 Question Answering .................................</td>
</tr>
<tr>
<td>1.2 Web Driven Interface .................................</td>
</tr>
<tr>
<td>1.2.1 Purpose and Objectives .........................</td>
</tr>
<tr>
<td>1.2.2 Challenges Faced in the Existing Scenario ...</td>
</tr>
<tr>
<td>1.2.3 Proposed System .................................</td>
</tr>
<tr>
<td>1.3 Application Background and Their Relation with Framework</td>
</tr>
<tr>
<td>1.4 Limitation of Study .................................</td>
</tr>
<tr>
<td>2 BACKGROUND AND LITERATURE REVIEW ...............</td>
</tr>
<tr>
<td>2.1 Background .................................................</td>
</tr>
<tr>
<td>2.2 Approaches for Question Answering ...............</td>
</tr>
<tr>
<td>2.2.1 Content-based Methods Using Combination of NLP and IR</td>
</tr>
<tr>
<td>2.2.1.1 NLP and Information Retrieval ................</td>
</tr>
<tr>
<td>2.2.1.2 Natural Language Processing ...................</td>
</tr>
<tr>
<td>2.2.1.3 Information Retrieval ............................</td>
</tr>
<tr>
<td>2.2.1.4 QA System Architecture .......................</td>
</tr>
<tr>
<td>2.2.2 Link Analysis Approach .........................</td>
</tr>
<tr>
<td>2.2.2.1 Page Rank Algorithm ............................</td>
</tr>
<tr>
<td>2.2.2.2 Hyperlink-Induced Topic Search Algorithm ...</td>
</tr>
<tr>
<td>2.2.3 Statistical Approach .........................</td>
</tr>
<tr>
<td>2.2.3.1 Bayesian Algorithm ............................</td>
</tr>
<tr>
<td>2.2.3.2 Maximum Likelihood Algorithm ................</td>
</tr>
<tr>
<td>2.3 Understanding- Natural Language Toolkit ..........</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1. Question-answering system. .................................................................10
Figure 4.1. Retrieved answer for GRE question. .......................................................28
Figure 4.2. Retrieved answer for GPA question. .......................................................29
Figure 4.3. Retrieved answer for prerequisite question. .........................................31
Figure 4.4. Upload file error. ..................................................................................32
Figure 5.1. Python manage.py runserver 9000. .....................................................33
Figure 5.2. Create super user. ................................................................................34
Figure 5.3. User interface – home page. .................................................................35
Figure 5.4. Login to Gmail account. .......................................................................36
Figure 5.5. Browse text file. ..................................................................................37
Figure 5.6. Previously asked question list. .............................................................38
Figure 5.7. Admin login page - Django framework. ..................................................39
Figure 5.8. Admin page – Django framework. .........................................................40
Figure 5.9. Uploaded text file- Django framework. .................................................41
Figure 5.10. Previously asked question list - Django framework. .........................42
Figure 5.11. Chunk of text- Django framework. .....................................................43
Figure 5.12. (a) Categories created (b) how to create a category – Django framework.44
Figure 5.13. Assign category to question - Django framework. .............................45
Figure 5.14. Observer question - Django framework. ............................................45
Figure 5.15. Super user list-Django framework. .....................................................47
Figure 5.16. (a) Delete super user (b) confirm delete. ............................................48
Figure 5.17. Retrieved answer. ............................................................................51
Figure 5.18. If question or answer doesn’t exist. ....................................................52
I sincerely thank my thesis advisor, Dr. Joseph Lewis, for his Guidance, unrelenting support, and helping me whenever I needed his help during my research work. Without his help it would not have been possible for me to accomplish this task. I also like to thank Professor Carl Eckberg and Professor Peter Blomgren for being the readers of this thesis and for giving their valuable suggestions.
CHAPTER 1

INTRODUCTION

1.1 QUESTION ANSWERING

Computer Science has a wide range of innovation that makes human work easily. Time has seen significant improvements in the usability and effectiveness of computer science technology. As a discipline, computer science spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software. Databases were developed to replace paper filing systems, which were difficult to manage. Faster processors are being developed for computers, so calculations that would take human days take only seconds for a computer to complete. Artificial Intelligence (AI) is an area of computer science concerned with operations that use intelligence to support decision making. Question answering is one of the fields where human tasks are being made easier by AI.

Question Answering system is mainly useful for classification of Questions and answers which is normally performed by human by indexing the collection of documents with an information retrieval system. Information retrieval systems, also known as search engines, are successful in retrieving documents, based on a query, from a collection of documents. Then when a question needs to be answered, a query will be created to retrieve documents relevant to the question. Finally, each document retrieved would be manually read until an answer to the question is found, or all the documents have been read. This method is time consuming, and a correct answer could easily be missed, by either an incorrect query, resulting in missing [1].

This application approached to natural language question and returns the answer from the set of answers, if the answer can be found. Answer retrieval, rather than document retrieval. Currently, there is a web site such websites -which do such things but not at the level which we needed. This Application will handle all such criteria like domain specific requirement and easily approachable application-user interface.
This application uses information retrieval with Natural language processing and uses that it will extract keywords from question and after it will display relevant answer. In information retrieval and natural language processing (NLP), question answering (QA) is the task of automatically answering a question posed in natural language. To find the answer to a question, a QA computer program may use either a pre-structured database or a collection of natural language documents (a text corpus such as the World Wide Web or some local collection).

QA research attempts to deal with a wide range of question types including: fact, list, definition, How, Why, hypothetical, semantically constrained, and cross-lingual questions. Search collections vary from small local document collections, to internal organization documents, to compiled newswire reports, to the World Wide Web.

Question Answering System can be categorized in two as follows:

- Closed-domain question answering deals with questions under a specific domain (for example, educational, medicine or automotive maintenance), and can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies. Alternatively, closed-domain might refer to a situation where only a limited type of questions are accepted, such as questions asking for descriptive rather than procedural information.

- Open-domain question answering deals with questions about nearly anything, and can only rely on general ontologies and world knowledge. On the other hand, these systems usually have much more data available from which to extract the answer [2].

1.2 WEB DRIVEN INTERFACE

This thesis attempts to provide an automatic process of find nearly best answers for asked questions for the closed domain (educational). Currently we have various applications in this domain but this thesis provides optimal and clear implementation of algorithm with additional facilities to the faculty or administrator.

As mentioned earlier it will be an implementation in Closed-domain question answering for educational scenario and addresses the following questions.

1. What could be the possible Questions- Answers set?
2. What likely answers needed to find answers for asked Questions?
3. Do we have any existing techniques for fulfill the requirements?
4. What approach and techniques needed to accomplish the task?
In summary, the challenge of this research is therefore to design and develop and effective and efficient automatic approach to evaluate the quality of supplied answers, to propose the best answer to the question.

1.2.1 Purpose and Objectives

- It is a separate body.
- The motto of the application is to help any educational organization to automate QA module.
- This application uses various kinds of methodology.
- Keyword extraction
- Tagging and NP/Verb Group chunking
- Structural Approach – Finding question relevant Answer

1.2.2 Challenges Faced in the Existing Scenario

- Lack of getting most likely answer to asked question.
- Unnecessary time is wasted for educational organization to reply back to Questions asked.
- Difficult and time consuming to reply back each and every person.
- Lots of human intervention which increases the chances of error in some form or other.
- No systematic provision for categorize the set of questions in domain based on various parameters which could have helped immensely in tracking the progress of various trainings.
- Thus, less reliability of the existing way of working.

1.2.3 Proposed System

- The application which is being made by us is a fully computerized and automated version.
- The entire workflow within the educational organization is kept the same which is the requirement.
- The terminology used in the applications is also the same so that the users would not have any problem in accustoming to the new system.
1.3 APPLICATION BACKGROUND AND THEIR RELATION WITH FRAMEWORK

This application simplify domain related questions and answers including with authorization. Application is based on specially implemented algorithm which gives the refined result to the users. A conventional frequently asked questions website deals with providing the answers to the questions which are general among people.

Main reason to implement this application to provide the simplicity to domain related simplicity for users. It can be useful to any domain for dealing with common questions and answers with having facility to upload the data which relates to frequently asked question for authorized users. It is really important as it eases users’ searched information.

Unlike any other FAQ website, this tool uses natural language processing to process the user questions and retrieve the answer. This website provides a new dimension of creating a frequently asked question website. It focuses on the natural language processing which is the backbone of this application that involves text processing and data chunking concepts. A natural language processing deals with processing and analyzing the human language to communicate with machine. It is very unusual for other FAQ websites to have such features.

This application has been developed with the technologies like Django web framework. Django is python programming language related web framework that reduces the work of creating and building a web page. Google App Engine is also another technology which does a free hosting of the website. Natural Language processing is the key implementation for the tool.

This application extracts Answers with the help of specially designed algorithm like maximum likelihood algorithm. This Algorithm helped to process the keywords and keywords frequency. Then it helps in determines the most frequently used keywords in a text chunk and calculate their frequencies. These frequencies help in determining if the question exists in question-answer table. If the question exists, the answer is retrieved.

1.4 LIMITATION OF STUDY

This study does not discuss about features of web frameworks. It does not describe everything about these frameworks and their capabilities in writing applications. Test results
in this study have been performed related to Natural Language Processing and back end side algorithmic implementation. Discussion about Front end Technologies will be covered by Research Paper by Udaykanth Sripada. Limiting this study allows more focused on Natural Language Processing and its application. Generalized study is enough for basic need of choosing a framework.
CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Background

Question Answering System aims to present the user with a short answer to a question rather than a list of possibly relevant documents. As discussed before, question answering system can be categorized into two as described below:

- Closed domain QA: Dealing with questions under a specific domain
- Open domain QA: Dealing with questions about basically everything
- Question Answering System:
  - Analyze and classify the question
  - Using information retrieval techniques, construct a subset of available relevant documents
  - Analyze the retrieved documents, search for wanted entities
  - If found, return the entity as a response, i.e. extract answer [3]

Education domain is a closed domain where structure of people who are related directly or indirectly to each other through common relation or interest. Two fundamental elements in domain are Staff/Faculty and student.

Students should be ones writing questions asking the questions as far as this staff/faculty concerned. Student does ask thoroughly challenging thoughtful questions. The staff/faculty can act as a guide for yet more challenging ideas.

2.2 Approaches for Question Answering

Question Answering can be classified into 3 types of approaches based on their features:

1. Content-based methods using combination of NLP and IR
2. Link analysis methods
3. Statistical methods
2.2.1 Content-based Methods Using Combination of NLP and IR

Content based approach is a method to find the answer to question which is selected from the database or the web, based on the meaning of the question. Natural Language processing and Information retrieval are two major techniques used for content-based approaches

2.2.1.1 NLP AND INFORMATION RETRIEVAL

The history of NLP generally starts in the 1950s, although work can be found from earlier periods. In 1950, Alan Turing published his famous article "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence.

This criterion depends on the ability of a computer program to impersonate a human in a real-time written conversation with a human judge, sufficiently well that the judge is unable to distinguish reliably — on the basis of the conversational content alone — between the program and a real human.

Up to the 1980s, most NLP systems were based on complex sets of hand-written rules. Starting in the late 1980s, however, there was a revolution in NLP with the introduction of machine learning algorithms for language processing. This was due both to the steady increase in computational power resulting from Moore's Law and the gradual lessening of the dominance of Chomskyan theories of linguistics (e.g. transformational grammar), whose theoretical underpinnings discouraged the sort of corpus linguistics that underlies the machine-learning approach to language processing [4].

“Some of the earliest-used machine learning algorithms, such as decision trees, produced systems of hard if-then rules similar to existing hand-written rules” [4].

“Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to the features making up the input data” [4].

The caches language models upon which many speech recognition systems now rely are examples of such statistical models. Such models are generally more robust when given unfamiliar input, especially input that contains errors (as is very common for real-world
data), and produce more reliable results when integrated into a larger system comprising multiple subtasks [4].

2.2.1.2 **Natural Language Processing**

Natural language processing (NLP) is concerned with the interactions between computers and human (natural) languages. Instead of using Boolean logic, the user simply can type in a question as they would communicate with a human. One of the most challenging problems in computer science and search is to develop computers that can understand natural languages and return one right answer.

This "Human-Like Language Processing" is considered as one of the prime concepts of Artificial Intelligence. Some of the functionalities involved in Natural Language Processing are:

1. **Parts of Speech Tagging:** When we provide a sentence, we can determine the parts of speech in the sentence using this. For example the word "bank" can be understood in two ways, one being the river bank and second one being the bank for saving your money.

2. **Parsing:** A sentence can be analyzed in multiple ways. Parsing helps in minimizing the dis-ambiguity in the sentence.

3. **Semantic Analysis:** This functionality determines the possible meaning of the sentence by focusing on the interactions provided in the sentence. For example, break has two meaning. The meaning of the break in "I want to take a break" depends on the rest of the words in the sentence. Similarly in the sentence "I will break this" has different meaning and the sentence can be understood based on the words used in the sentence.

4. **Tokenization:** It is process of breaking the sentence into words, paragraphs, symbols and tokens. Tokenization helps in further processing of the text such as parsing.

5. **Question Answering:** It is a process of providing automated answers to the questions given in natural language. A list of matching answers to the question will be provided. [“This thesis aims for implement of this domain”]

6. **Text Chunking:** Text chunking consists of dividing a text in syntactically correlated parts of words. For example, the sentence “He reckons the current account deficit will narrow to only # 1.8 billion in September.” can be divided as follows: [NPHe] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September].

7. **Named Entity Recognition:** Given a sentence, recognizes named entities such as place, people, names and the type of each entity. When a sentence is given, the elements in the text are categorized under these entities.
8. **Stemmer**: It is a process of reducing derived words to their stem. A stem is basically the root of the word. For example the word banking is reduced to the word "bank" to make it more understandable.

9. **Keyword Extraction**: Emphasizes on extracting keywords from a given sentence. Suppose if we want to find a sentence from a given check of texts, we need to get the most frequent keywords used in the text. By doing so, we can retrieve the sentence. There are different tools available that make use of Natural Language Processing. They are:
   1. NLTK using python
   2. OpenNLP using Java
   3. Stanford NLP using Java
   4. MontyLingua using python or Java.

   Among all of them Python is very strong with NLP. Python is heavily used in industries, research and education throughout the world. Python is often considered as an efficient language due to its productivity, quality and maintainability of the software. Some of the Natural Language Processing programs are written in python. It provides basic classes for representing data relevant to natural language processing; standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, and text classification; and standard implementations for each task that can be combined to solve complex problems.

   Natural Language Processing focuses on developing efficient algorithms that process texts and make their information accessible to applications. As we know there are plenty of such algorithms available. Get-Answer uses a mixture of such NLP processing with machine learning algorithms [4].

### 2.2.1.3 INFORMATION RETRIEVAL

- Information Retrieval includes the information types which consist of Text \ Voice \ Image \ Structured data \ Rules \ Program \ Animation \ Video etc.

- Types of Information we need of Vague or precise

- Types of query language Ambiguous or exact

- Types of matching Exact or approximate

Putting all combinations together, we only have a subset of all possibilities

Information retrieval is:

- Text data,
• vague information need,
• imprecise matching,
• exact or an ambiguous query language

But there is more to text management than retrieval, indexing, routing, Classification, Extraction & Summarization. Acquisition (OCR), spell Checking, critiquing, compression, encryption, editing and formatting. All are part of text management

It is important to realize that IR is an inexact application ... people tolerate, even expect, to have non-relevant documents retrieved ... this is unlike most other applications of computing ... MT, KBS/expert systems, etc. Indexing and Retrieval, with a bit of clustering perhaps, were the standard IR applications for a long time but now the others, routing, classification, abstraction/summarization, are increasingly important, due to demand.

Application areas for text retrieval.

Traditionally in libraries and in legal domain (searching past case histories) and patent applications ... now searching news stories, encyclopedias, office applications, network resource discovery, etc. [3].

2.2.1.4 QA SYSTEM ARCHITECTURE

Figure 2.1 shows a basic architecture of a Question-Answering system.
The Question analysis modules include extraction of keywords for the query, removal of stop words and wh-words and normalization. The keywords may be expanded using such relevant algorithm. Once question is asked then it will extract the keywords from the question using topia lib. topia lib for tagging, since NLTK taggers use Numpy which uses C-precompiled modules and not allowed by Google app engine. While fetching the keywords it also fetch the stop word file which has data about all the verb, noun phrase.

2.2.2 Link Analysis Approach

Link Analysis approaches as its name suggests, analyses the linkages between the actors in a network to determine the relationships amongst them. These Relationships can be used to derive rank prestige in network analysis. Two types of algorithm mostly used in link analysis approach: PageRank Algorithm and second is the HITS algorithm and its variations.

2.2.2.1 PAGE RANK ALGORITHM

The PageRank algorithm is the most popular method to rank / weight websites. The algorithm was developed and published by Sergey Brin and Larry Page at the Stanford university end of the 90s [5]. The new idea was the introduction of a factor which was only generated from the linking structure. Neither the content nor the URL plays a role (such parameters are called off-page factors). Moreover, there is no difference between internal and external links. PageRank can be understood as the importance of a website. The ranking algorithm is a function of on- and off-page factors. On-page factors are for example the title, description, headings and plain text. An off-page factor apart from PageRank is the anchor text (of incoming links).

Google used some modified algorithm. At the beginning the algorithm leads to better results because the algorithm was hard to manipulate. However, publishing the algorithm as well as showing the PageRank in the toolbar leads to more attempts to manipulate it, e.g. by selling and buying links (PageRank). Also other engine created comparable algorithms, i.e. algorithms that based mainly on the linking structure and anchor text. This leads to an approach of the results for the leading search engines [5].
2.2.2.2 Hyperlink-Induced Topic Search Algorithm

Hyperlink-Induced Topic Search (HITS) (also known as hubs and authorities) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. It was a precursor to PageRank. The idea behind Hubs and Authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that it held, but were used as compilations of a broad catalog of information that led users directly to other authoritative pages. In other words, a good hub represented a page that pointed to many other pages, and a good authority represented a page that was linked by many different hubs [6].

2.2.3 Statistical Approach

Statistical approaches for determining the quality of answers can be simple as using statistical analysis on non-textual features such as frequency of question answer length and answer acceptance ratio. These approaches can be also be complex as the use of machine language. Several probability-based classification and clustering techniques such as Bayesian, Markov chain and Maximum likelihood have also been applied to solve QA tasks.

2.2.3.1 Bayesian Algorithm

Bayesian probability is one of the different interpretations of the concept of probability and belongs to the category of evidential probabilities. The Bayesian interpretation of probability can be seen as an extension of logic that enables reasoning with propositions whose truth or falsity is uncertain. To evaluate the probability of a hypothesis, the Bayesian probabilistic specifies some prior probability, which is then updated in the light of new, relevant data [7].

2.2.3.2 Maximum Likelihood Algorithm

The method of maximum likelihood corresponds to many well-known estimation methods in statistics. For example, one may be interested in the heights of adult female giraffes, but be unable due to cost or time constraints, to measure the height of every single
giraffe in a population. Assuming that the heights are normally (Gaussian) distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish this by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable (given the model).

In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects values of the model parameters that produce a distribution that gives the observed data the greatest probability (i.e., parameters that maximize the likelihood function). Maximum-likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems. However, in some complicated problems, difficulties do occur: in such problems, maximum-likelihood estimators are unsuitable or do not exist [8].

2.3 UNDERSTANDING- NATURAL LANGUAGE TOOLKIT

Natural Language Toolkit or, more commonly, NLTK is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. NLTK includes graphical demonstrations and sample data. It is accompanied by extensive documentation, including a book that explains the underlying concepts behind the language processing tasks supported by the toolkit.

NLTK is ideally suited to students who are learning NLP or conducting research in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. The NLTK Project is led by Steven Bird [9].

Using NLTK [10]. The NLTK website contains excellent documentation and tutorials for learning to use the toolkit. It would be unfair to the authors, as well as to this publication, to just reproduce their words for the sake of this article. Instead, I will introduce NLTK by showing how to perform four NLP tasks, in increasing order of difficulty. Each task is either an unsolved exercise from the NLTK tutorial or a variant thereof. Therefore, the solution and analysis of each task represents original content written solely for this article.
Some of the task which is done through NLPTK is:

- Predicting Words
- Discovering Part-Of-Speech Tags
- Word Association

Python and the Natural Language Toolkit (NLTK) allow any programmer to get acquainted with NLP tasks easily without having to spend too much time on gathering resources. This article is intended to make this task even easier by providing working examples and references for anyone interested in learning about NLP.

The Natural Language Toolkit is a suite of program modules, data sets, tutorials and exercises, covering symbolic and statistical natural language processing.

NLTK is written in Python and distributed under the GPL open source license. Over the past three years, NLTK has become popular in teaching and research.

### 2.4 Comparing Content Based, Link Analysis and Statistical Approach

After study of all approaches for Question Answering system it’s concluded that content based approach has its domain specific limitation so it suffers from low recall. The answer to a question is also limited to predefined categories. Link analysis approach overcome these problem but further study need to be done over statistical approach which utilizes the features of specific QA portals.

Content based method which relies on NLP and IR are most widely used in closed domain application it has some limitation in terms of interpretation of human language to answer the question. Link analysis based approach can be way to get best suitable for open domain application and statistical approach is using various techniques, algorithm and one of them –maximum likelihood algorithm with content based approach is used in proposed application.

So Get-Answer tool comes with approach which uses content based approach with NLP and IR along with statistical approach. It concludes with that: Natural Language Processing and Information retrieval deals with the data manipulation and statistical approach deals with the getting the most relevant approach from the system.
2.5 Natural Language Processing and Information Retrieval Approach

Part-of-speech tagging is the process of assigning a part-of-speech like noun, verb, pronoun, preposition, adverb, adjective or other lexical class marker to each word in a sentence. POS tagging plays an important part in search techniques and are easier to use. It mainly consists of two steps:

- **Tokenization:** This means splitting the words of the sentences. For example, let’s say we have a sentence like “heat the water in a large vessel”. The tokenized version of this sentence would be [heat, the, water, in, a, large, vessel].

- **Tagging:** After tokenizing the sentence we take each word and tag then with their respective parts of speech. Using the same example, the tagged words would look something like this [Heat/NNP][the/DT] [water/NN][in/IN][a/DT][large/JJ][vessel/NN]. For this we have used a python library called Topia.termextract which uses linguistic tools such as POS tagger. We can use this library in the following way:

```python
from topia.termextract import tag
from topia.termextract import extract
tagged_tokens = tagger.tokenize(text)
tagged_tokens = tagger(text)
```

Here text will be the words from the sentences in the text document. Text Chunking is a process of dividing input text into non-overlapping segments. It is also called as partial parsing. It helps in assigning partial syntactic structure to a sentence. The main advantage of text chunking is it can be much faster than full parsing, more robust and suitable to application that involves information extraction or question answering.

A chunk is non-recursive that means it cannot contain another chunk of same category. Chunking is usually done on tagged text. So after performing POS tagging, text chunking comes into the picture so that the readers do not see a solid block of text.

Let us consider an example, [the/DT little/JJ cat/NN sat/VBD on/IN the/DT mat/NN] is a tagged text. When text chunking is applied to this text the output will be:

```
(S: (NP: ’The’ ’little’ ’cat ’)
 ’sat’ ’on’
 (NP: ’the’ ’mat’))
```

While performing text chunking we need to consider NP-Chunking. NP-Chunking involves in identifying the chunks that contain noun phrase (NP). NP-Chunks usually are smaller than complete noun phrases.
POS tagging forms essential information provider for NP-Chunks. In order to create NP-Chunks, we need a grammar that defines how the sentences need to be chunked. For this we can use rules that include regular expressions.

The rule can say that an NP chunk should be formed whenever the chunker finds an optional determiner (DT) followed by any number of adjectives (JJ) and then a noun (NN). This can be represented as \(<DT>\)? \(<JJ>*\)<NN>.

This whole algorithm works in the following manner: the application filters the keywords and calculates keywords frequencies according from the text during text upload. This information is saved in a database in a Text-Chunk table. When a question is asked it gets keywords with frequencies for the question and then uses maximum likelihood algorithm to find text chunk based on keywords frequencies.

### 2.6 Statistical Approach

To get keywords we have used text tagging and chunking algorithms. To find the best answer we have used maximum likelihood estimation. For each text chunk, calculate the similarity measure of the question and text chunk based on keyword frequency. The similarity measure is inspired by maximum likelihood estimation. It relates to Bayes theorem, as Bayes theorem is the basis for all probabilistic methods. According to this thesis project it is defined as:

Let us consider that we have chunks “C1,...,Cn” and need to decide whether the item “t” (text) is related to it or not.

\[ \text{Likelihood (Ci)} = \text{Prob} \left( t \mid Ci \right). \]

Let us assume that each chunk “Ci” has frequency of word “w”: Fi (w).

Probability of frequency of word “w” in each chunk “Ci” is also represented as:

“Prob (w|Ci)”.

In text “t” we represent as “n1” occurrences of “w1” (word1), n2 of w2 etc.

\[ \text{Likelihood} = \text{Prob} \left( t \mid Ci \right) = \text{Product of all w in t} \times \left( \text{Prob (n occurrences of w | Ci)} \right) \]

\[ = \text{Product for all w in t} \left( \text{Prob (w|Ci)} \times n \right) \]

So, \( \log(\text{likelihood}) = \text{sum by } w(n^*\log(\text{Prob}(w|Ci))) = \text{sum by } w(n^*\log(Fi(w))) \)

So we need to maximize by:

Ci likelihood, or minimize sum by_
\[ w(n \log(Fi(w)) = \text{sum\_by\_w}(n \log(1/Fi(w)) \]

This is the same as minimize \[ \text{sum\_by\_w}(\text{freq\_w\_in\_t} \log(1/Fi(w)) \]

Since \[ \text{freq\_w\_in\_t} = \frac{n\_for\_w}{\text{sum\_by\_w\_in\_t}(n\_for\_w)} \]

### 2.7 Conclusion

This thesis addresses the problem of recommending the answer automatically from several answers provided by many users on system. It focused on the content based and statistical approach to accomplish this task. The Content based method which relies on NLP and IR are most widely used approaches in this area.

A Question is asked in the format of human language and it tries to convert it to machine understandable language. The aims to find answer to the question based on the relevance of the question keywords.
CHAPTER 3

ANALYSIS AND IMPLEMENTATION

3.1 GET ANSWER - GENERAL SCHEME

This application is quite different from other basic and commercial FAQ sites as it allows the user to update Get-Answer knowledge by uploading the text file with relevant information. Most of the FAQ websites out there have basic concept of extracting information by maintaining a database using technologies such as MySQL, php and produce the answers using sql queries. But this application does not use any such technologies. Instead it uses Natural Language Processing concepts which include parts of speech tagging, information extraction and text chunking etc. to process the text and extract the answers.

General scheme of this application can be explained as follows:

1. After opening the application, fetch the question given by the user.
2. Next we get keywords from the question and calculate the frequency of the keywords.
3. Based on the keyword frequency, we check if the question is in Question Answer table, if it is there we get the corresponding answer. End.
4. If the question is not there in Question Answer table, we use the text chunk table to find the most similar texts using most likelihood estimation algorithm and answer it.
5. End.

3.2 TECHNOLOGIES USED

Get-Answer application is written in Python. Since Python being the most powerful language that supports most of the natural language processing issues it has been used. As a part of this a web framework called Django is implemented. It comes with an object-relational mapper in which you describe your database layout in Python code. The application is deployed on Google Application Engine (GAE) free application hosting. Along with this Djangononrel libraries have been used to adopt Django code for Google App Engine.

The application also can run on any standalone PC with Python installed. In this case the development webserver is used. To run development webserver just run “python
manage.py runserver 9000” and you can see the application by address http://localhost:9000. You can replace 9000 by any other port number you like.

3.3 Files

Implementation of algorithm using python for find relevant information from previously uploaded data is described below.

All algorithms for get_answer/getanswer are coded here:

- **text_processor.py**: Functions related to text processing: tagging, chunking and calculating keywords frequencies.
- **upload.py**: Functions related to text upload.
- **get_answer.py**: Code related to get answer from user question.
- **views.py**: Handlers for urls of the application (mapping between these handlers and urls is done in root file: urls.py.
- **models.py**: Django models defined for this application (i.e. tables in the database).
- **def.py**: intended for general for the app constants. Now contains only DEBUG which controls if Log is written during app work.

Note: Application Algorithm Implementation focuses on **text_processor.py and get_answer.py**

3.3.1 Processing of Question in text_processor.py

- **text_processor.py**: get keywords from text and calculate frequencies of keywords.
- We have implemented Tagging and simple chunking is used to extract keywords.
- Use topia lib for tagging, since NLTK taggers use Numpy which uses C-precompiled modules and not allowed by google appengine.
- We use stopword_file which has predefined verb noun adjectives etc.

```python
# Class to store frequency info of a keyword

class WordFreq:
    # w is keyword or its frequncy string code if f is None
    # frequency code if "<keyword>:<freq>:<log_inverted_freq>"
    def __init__(self, w, f=None):
        if f == None:
            params = w.split(";")
            #LOG.write(w)
```
self.word = params[0]
self.freq = float(params[4])
self.log_inv_freq = float(params[2])
else:
    self.word = w
    self.freq = f  # text freq, or mean for category
    self.log_inv_freq = -log(f)  # log inverted mean freq for category

# Get description: for debugging purposes
def get_desc(self):
    return "%s freq:%s log_if:%s" % (self.word, \
        str(self.freq), str(self.log_inv_freq))

# Get keyword frequency string code "<keyword>:<freq>:<log_inverted_freq>"
def encode_to_string(self):
    return "%s:%s:%s" % (self.word, str(self.freq), str(self.log_inv_freq))

3.3.2 Text File Processed within the Application

The text processing within the application is done only when we upload some text. In
this application, there is a tab which enables the user to upload a text file. The text file could
be a paragraph chunk or just questions and answers. The following steps give a brief idea of
how the text file is uploaded.

- **Once the user uploads a text file, it is read into memory file.**
  In the code below file indicates the text file:
  def upload_get_memory_file(file):
      memfile = StringIO.StringIO()
      # reads each chunk
      for chunk in file.chunks():
          # writes it into a memory file.
          memfile.write(chunk)
      memfile.seek(0)
Then text is split into paragraphs by considering empty line occurrences. If there are empty line occurrences, we consider it as end of the chunk and add it to the chunk list.

def upload_get_chunks_from_text(file):
    chunks = []
    current_chunk = 
    line = file.readline()
    while line:
        # check if a new paragraph begins - empty line
        if not line.strip():
            if current_chunk:
                # Empty line - end of chunk - add chunk to list
                chunks.append(current_chunk)
                current_chunk = 
            else:
                # If line is too long - cut it to max size of chunk
                if len(line) > CHUNK_LEN_MAX:
                    line = line[:CHUNK_LEN_MAX]
                # If line length is too large - leave for a new chunk
                if current_chunk and \
                    len(current_chunk) + len(line) > CHUNK_LEN_MAX:
                    chunks.append(current_chunk)
                    current_chunk = line
            else:
                current_chunk = current_chunk + line
        line = file.readline()
    # Don't forget to add last chunk
    if current_chunk:
        chunks.append(current_chunk)
    return chunks
• Each paragraph is then added to TextChunks table in database.
  In this process we check if it a question answer chunk and filter the keywords
frequency from the chunk. Later these frequencies are stored in the database and the
final chunk data.
  
```python
def upload_save_chunks(chunks, text):
    chunk_ind = 1
    for c in chunks:
        # Check if question-answer chunk
        save_question_answer(c)
        # Get keywords frequency list from chunk
        wfl = get_keywords_from_text(c)
        # Encode keywords frequencies as a string to be put to database
        keywords = word_freq_list_to_string(wfl)
        # Save chunk data to database
        TextChunk.objects.create(chunk=c, text=text, text_order=chunk_ind,
                                 keywords=keywords)
        # Increment chunk order counter, so we remember order of chunks in text
        chunk_ind += 1
    return 0
```

• If paragraph is question-answer block, question and answer are saved in
Question Answer table in database
  
```python
def save_question_answer(text):
    text = text.strip()
    if not text.startswith(QUESTION_ANSWER_STR):
        return 0
    # Find question end = end of line
    first_line_end = text.find("\n")
    # If end of line not found, there is no answer - do nothing
    if first_line_end< 0:
        return 0
```
return 0

# Get question string
first_line = text[:first_line_end+1]
first_line = first_line.replace(QUESTION_ANSWER_STR, " ")

# Get unified version of question
first_line = unify_question(first_line)

# Add unified question to QuestionAnswer table in database
QuestionAnswer.objects.create(question=first_line,
answer=text[first_line_end+1 : len(text)])
return 0

3.4 Implementation of Maximum Likelihood Algorithm

Code Snippet:

def get_max_likelihood_similarity(wfl1, wfl2):
res = 0

# For each keyword in question find the same keyword in chunk
for wf1 in wfl1:
    found = False
    for wf2 in wfl2:
        if wf1.word == wf2.word:
            found = True
            res += wf1.freq * wf2.log_inv_freq
            break
    if not found:
        res += wf1.freq * 7.0
return res
We need to find minimum (not maximum since we inverted frequency in log above) for all text chunks. If minimum similarity is less than threshold, then return text chunk with this similarity. Otherwise return “I don’t know :(”

3.5 User Interface

This application also needs to have nice user interface which deals with user. It has been created by considering the design mentioned in the thesis topic “FAQ Application” by Udayakanta Sripda which are related to natural language processing. It is one of the concepts of communicating with design specification. The Designing that the thesis covers are Django framework and Google App Engine. In depth detail about the design implementation has been mentioned in the thesis book.

3.6 How to Run the Application on PC?

- Open setting.py file within the “getanswer” folder. Look for the variable DEBUG (line 12) in the file and change its value to “True”.
- From command line go to root folder of the application where you will find a file named “manage.py” and run command: python manage.py runserver 9000
- If you are running the application for the first time you will be asked to create a super-user. Just provide a name<space> e-mail-ID <space> password.
- You can create super-user later by running command python manage.py createsuperuser
- After a while you will see a message, something like "your app is running on http://localhost:9000". You can change 9000 to whatever port number you prefer.

3.7 Deploying the Application on Google Application Engine

The following steps explain how to deploy the application using Google App Engine.

- Before deploying the application we need to first create the application on Google. For this open appengine.google.com in a browser and sign in with your Gmail account. If you do not have a Gmail account one needs to be created.
- After signing into appengine.google.com click on “Create application”. Choose a name for the application. It will then provide the address as <application_name>..appspot.com. Thus your app will have address say for example, get-the-answer.appspot.com.
- After creating the application on Google, we need change the name of the application in app.yaml file in “get_answer” root folder (first line of the file).
- Change the value of the variable “DEBUG” as True in settings.py file and run development version first by running the command python manage.py runserver 9000. Then press <Ctrl+Break> and quit development version.

- Once you have deployed the application change the variable DEBUG to False in settings.py file and run the command "python manage.py deploy". That's all. The application will ask for Gmail account e-mail and password - put the one you used when created application on Google.
CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 WHAT THE APPLICATION DOES?

Get-Answer is the name of the application. It is an FAQ site mainly built for school purposes but in general can be used by anyone. The application is made in such a way that any commoner can easily use it. It allows the user to type a question he wants to get answer and the application gives best guessed answer. This application allows the user to update Get-Answer knowledge by uploading the text file with relevant information. This application does not use any such technologies. Instead it uses one of the Artificial Intelligence concepts called Natural Language Processing which include parts of speech tagging, information extraction and text chunking etc. to process the text and extract the answers.

4.2 APPLICATION DOMAIN AND EXPERIMENT

This application is domain specific to Education system. We have tried to resolves the possibly input from this domain.

This Application tries to simplify the manipulation of information which is going to be domain specific. Experiments were conducted to test proposed content based approach with NLP and IR and maximum likelihood algorithm.

4.2.1 Upload a Text File

Let’s say we have text file name GRE.txt which contains following information

4.2.1.1 GRE

The Graduate Record Examination (GRE) consists of two separate tests: the General Test and the Subject Test in psychology. The General Test is composed of three parts--verbal, quantitative, and analytical writing.

Store as text chunks in Get-Answer Tool:

nnp/test:0.142857142857:1.94591014906|nnp/general:0.0952380952381:2.35137525716|jj/analytical:0.047619047619:3.04452
Figure 4.1 shows your question “what is GRE” and retrieved answer.

4.2.1.2 GPA

A grade point average (GPA) is a calculated average of the letter grades you earn in school following a 0 to 4.0 or 5.0 scale.

Figure 4.2 shows your question “what is GPA” and retrieved answer.

4.2.1.3 PREREQUISITE

Something that is prerequisite, as a course that is required prior to taking an advanced course.
Figure 4.1. Retrieved answer for GRE question.
Figure 4.2. Retrieved answer for GPA question.

A grade point average (GPA) is a calculated average of the letter grades you earn in school following a 0 to 4.0 or 5.0 scale.
Store as text chunks in Get-Answer Tool:

nn/course:0.25:1.38629436112|nn/prerequisite:0.25:1.38629436112|nn/something:0.125:2.07944154168|vbd/advanced:0.125:2.07944154168|vbg/taking:0.125:2.07944154168|vbn/required:0.125:2.07944154168

Figure 4.3 shows your question “what is prerequisite” and retrieved answer.

**4.2.2 How it Works?**

After having text file uploaded it will first stores the text file as a text chunks and apply the tagging and chunking algorithm to store in database.

Now let’s user ask the question like:

What is GRE?

It is analyze by first doing keyword frequency and separate the stop word classification and then apply maximum likelihood algorithm to match that keyword frequency.

**4.3 What if Size of Text File is Large in Size?**

Currently we have restricted our application with text file size up to 1MB. We can modify the size as per our requirement.

So if we upload the text file which has size more than 1MB Application won’t let user to upload.

Figure 4.4 shows uploading a file more than 1MB.
Figure 4.3. Retrieved answer for prerequisite question.
Figure 4.4. Upload file error.
CHAPTER 5

SCREENSHOTS

Figure 5.1 shows the application is launched using the command “python manage.py runserver 9000.”

![Python manage.py runserver 9000](image)

Figure 5.1. Python manage.py runserver 9000.

Figure 5.2 shows you can also create a super user to access your localhost admin page where you can modify the contents of the text uploaded.

Figure 5.3 shows the home page of the application. The user provides a question in the text box next to “your question” and clicks the answer button.

Figure 5.4 shows that before uploading the text the super user needs to log in using Gmail account.

Figure 5.5 shows that the super user can choose the text file to be uploaded.

Figure 5.6 shows that whenever a question is asked by a user, it will be displayed in new questions page. Similar questions asked multiple times, will be displayed only once.

Figure 5.7 shows the admin login page. Note: the FAQ application using NLP algorithms is courtesy of Udayakanth Sripda.
Figure 5.2. Create super user.

Figure 5.8 shows the Django admin website where you can see different tables. Log entries, Questions Answers, Question Categories, Questions, Text Chunks and texts are tables.

Figure 5.9 shows when you click on Texts, you will observe the text files which are uploaded from the application along with the date.

Figure 5.10 shows when you click on Questions, it will show you the list of questions asked by the users.

Figure 5.11 shows when you click on Text Chunk, the uploaded text file is broken into a chunk. Later the text is POS tagged and keyword frequency is calculated which is displayed in keywords sections.

Figure 5.12(a) shows the list of categories created. Figure 5.12(b) shows how to create a category.

Figure 5.13 shows that after creating a category, one can assign this category to the corresponding question in the Question table. Also you can mark the question as important by simply checking.

Figure 5.14 shows that one can observe how many times the question is asked and whether the question is important or not. If the question is not marked as important then the important column has a red symbol.
Get Answer Tool

<table>
<thead>
<tr>
<th>Get Answer</th>
<th>Upload text</th>
<th>New questions</th>
<th>Updates</th>
</tr>
</thead>
</table>

Your question: [Input field]

Answer

Welcome to the Get Answer Tool - automatic answering software.

Get Answer tries to find best matching answer for user question. Get Answer uses NLP and machine learning algorithms to find the answer. It is based on text information, uploaded to the system. If you decide that answer is not correct or system does not know the answer - you can upload the text with relevant information. Next time Get Answer will answer it for sure.

To use Get Answer just type your question and click "Answer". Using links below one can get answers for some common questions. And we have answers for special questions:

- "what are the last new questions in Get Answer?"
- "what are the last updates to Get Answer?"

Figure 5.3. User interface – home page.
Figure 5.4. Login to Gmail account.
Figure 5.5. Browse text file.
Figure 5.6. Previously asked question list.
Figure 5.7. Admin login page - Django framework.
Figure 5.8. Admin page – Django framework.
Figure 5.9. Uploaded text file - Django framework.
Figure 5.10. Previously asked question list - Django framework.
Figure 5.11. Chunk of text- Django framework.
Figure 5.12. (a) Categories created (b) how to create a category – Django framework.
Figure 5.13. Assign category to question - Django framework.

Figure 5.14. Observer question - Django framework.
Figure 5.15 shows the admin page also shows the list of authenticated super users.

Figure 5.16(a) shows the admin page allows you to delete the super users. It confirms before deleting the super user as seen Figure 5.16(b).

Figure 5.17 shows when the user gives a question in the text box, the answer is retrieved as shown.

Figure 5.18 shows if the question or the answer does not exist, the application gives “I don’t know” as the answer.
Figure 5.15. Super user list-Django framework.
Figure 5.16. (a) Delete super user (b) confirm delete.
Figure 5.17. Retrieved answer.
Figure 5.18. If question or answer doesn’t exist.
CHAPTER 6

SUMMARY OF REVIEW TECHNIQUES

Question-Answering–It has obtained great requirements in recent years. Question-Answering tools are based around the quality of answers. Currently, the best answer to the question is chosen through a manual process which requires time and efforts from the both the end. An automatic process of choosing the best answer and give categories will allow to save time and give analysis of frequently asked questions. It is also very easy in terms of maintainability.

6.1 MAIN FINDINGS

This research presents an innovative approach to select the answer from many different answers from given backend. The research determines the optimized result for users’ by utilizing the likelihood of keywords to related content of answers.

Algorithm -Tagging and chunking which defines the keywords and store the data in terms of chunks and table. Alongside Maximal Likelihood algorithm find the maximal likely related answers by comparing the keywords from question to previously stored chunks and table.

The proposed content method makes use of this application, an online QA system as possible expert source. It allows administrator various features like authenticate users, adding the new answers, define the category of Questions.

A Series of experiments were conducted to test the effectiveness of the proposed algorithm. From the experiments on the proposed algorithm Tagging and chunking comes with sharp and easily implemented algorithm for Question Answering Tool.

6.2 USEFULNESS

This research can be used to directly for users who want to automatic approach to their structure. It can improve the efficiency of the process of work in terms of question and answer.
A feature of this research also give flexibility to user for categorize the Questions as needed and also get an analysis of the frequently asked Questions and proposed answers. Therefore, it is possible for the proposed methods to have wide application to QA systems. On the other hand quality of answer is determined by the content of quality of an answer such as relevance and in formativeness.

6.3 Future Work

There are number of limitation in this research one limitation is data collection is in small amount of data for performing implementation. Another limitation is no method is implemented for check quality of question and uploaded text. It makes sense that best Quality Question should get Quality Answer and no such Question should allowed by Application. This application is closed domain so it can be expanded for any domain framework and we can get answer which is more precise to question

As a Future work Questions needs to be analyzed and related Quality answers should be given to users. It would also be desirable to test and compare the performance and results of existing application in same domain for real world use.
REFERENCES


