

QUESTION ANSWER SYSTEM ON MEDICAL DOMAIN

Project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Naman Arora (121324)

Under the supervision of

Dr. Rajni Mohana

to



Department of Computer Science & Engineering and
Information Technology

Jaypee University of Information Technology Waknaghat,

Solan-173234, Himachal Pradesh

CANDIDATE’S DECLARATION

I hereby declare that the work presented in this report entitled “**Question Answer System on Medical Domain**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2015 to May 2016 under the supervision of

Dr. Rajni Mohana.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Naman Arora

121324

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Rajni Mohana

Assistant Professor (Senior Grade)

Department of Computer Science and Information Technology

ACKNOWLEDGEMENT

I owe my profound gratitude to my project supervisor Dr. Rajni Mohana, who took keen interest in my project work titled “**Question Answer System on Medical Domain**” and guided me all along, till the completion of my project by providing all the necessary information for developing such a good system. The project development helped me in research and I got to know a lot of new things in my domain. I am really thankful to her.

Dated:

Naman Arora

121324

Table of Content

Contents

List of figures.....	v
List of abbreviations.....	vi
Abstract.....	vii
Introduction	1
Introduction	1
Problem statement	2
Objectives	2
Methodology.....	2
Part of speech tagging	3
Question focus	3
Website crawler.....	4
GUI design.....	5
File access.....	6
Text summarizer	6
Organization.....	7
Literature Survey.....	8
Question Answering System	8
Natural Language Processing.....	9
Natural Language Toolkit.....	10
Python.....	11
Text Summarization	12
Wamp server.....	12
PHP Admin	12
Apache	13
SQL Server and database	14
System Development.....	15
Software Requirements	15
Hardware Requirements.....	15
Software Model	15
Algorithm	18

Implementation	20
Website crawler	20
Tokenizing and tagging sentences	23
GUI design	24
Text Summarizer	25
Final module	26
Performance Analysis	28
Unit Testing	28
POS tagger unit testing	28
Text summarizer unit testing	29
Final module with GUI unit testing	31
Black box testing	33
Python code testing using cProfile	35
Conclusions	36
Conclusion	36
Future Scope	36
References	37
Appendices	38

List of figures

Contents

Figure 1: Part of speech tagging	3
Figure 2: Text files generated using website crawler	4
Figure 3: GUI design.....	5
Figure 4: Text summarizer	6
Figure 5: Tokenizing and tagging of text using NTLK	10
Figure 6: Naming entities in NLTK.....	11
Figure 7: phpMyAdmin	13
Figure 8: Apache	13
Figure 9: MySQL.....	14
Figure 10: Flowchart of the project	19
Figure 11: MedIndia website for data retrieval	21
Figure 12: List of files for doctors in different medical domains	23
Figure 13: Interactive GUI design.....	31
Figure 14: Working of project	32
Figure 15: Black box testing approach.....	33
Figure 16: Interactive project design	34

List of abbreviations

NLP: Natural Language Processing

NLTK: Natural Language Toolkit

GUI: Graphical User Interface

TS: Text Summarizer

QA: Question Answer

PoS: Part of Speech

IR: Information Retrieval

HDD: Hard Disk Drive

CPU: Central Processing Unit

URL: Uniform Resource Locator

ABSTRACT

Searching any information on internet gives us a lot more than the intended. The user then has to crawl through all the data and find the desired output which is a tedious process. This project is all about developing a QA System on medical domain that will take in user question and give only the desired out and nothing extra.

QA System is an application that uses Natural Language Processing integrated with Python. The website crawler that is used to fetch data from the website is made in PHP and fetches data and saves it into the text files.

The main reason for developing this project is to make the user experience pleasing and simple by providing easy and relevant information about the major diseases and doctors.

The project will be developed in NLTK integrated with Python which will provide us the above functions.

CHAPTER-1 INTRODUCTION

1.1 Introduction

Question Answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language. [1]

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*. These keywords are then used in the question focus to determine the answer type.

Since the questions posed by the humans are in the natural language, there has to be a tool to convert such language into the machine understandable form. Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation. There are various toolkits that are available for this computer and human interaction one of which is Natural Language Processing Toolkit (NLTK)

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

1.2 Problem Statement

Natural language processing has been a researchable topic in the recent past and hence there are various domain specific tools coming up to cater the same. Medical domain is one such area where QA systems can help the user in getting their queries answered.

Toolkits like NLTK have now come up that help the developer process the input query and tag the relevant words for further processing. The major task is to use this toolkit in developing a QA system for medical domain.

1.3 Objectives

Thoroughly understand the basics of Python and NLTK and thereafter develop a tool on medical domain that takes in user input or query and processes it to give the relevant answer from the database. The user query can be a question posed in correct grammatical form in English language.

1.4 Methodology

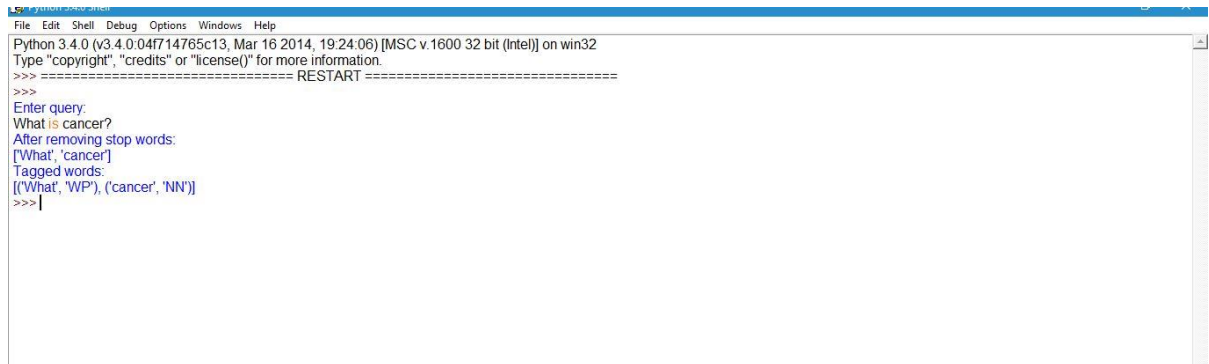
The development of the project involved the development and integration of the following independent modules:

- Part of speech tagging
- Question focus
- Website Crawler
- GUI design
- File Access
- Text Summarizer

1.4.1 Part of speech tagging

The first and the foremost module in the development of the QA system was the POS tagging module. In this module, the input sentence is split into words (tokenized) and each word is then tagged for its respected part of speech.

Let us say the user inputs, “**What is cancer?**”



```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:24:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter query:
What is cancer?
After removing stop words:
[What, cancer]
Tagged words:
[('What', 'WP'), ('cancer', 'NN')]
>>>|
```

Fig 1. Part of speech tagging

The screenshot above shows the tagging of the words “what” and “cancer” after removal of “is” which was a stopword.

1.4.2 Question focus

We identify the following question types which give a fairly clear indication of the type of answer: when, who, where, whom, why, describe, and define. For example, the answer to a question is usually a person or a group of people. Other keywords or question words are less clear about the expected answer type: what, which, how, and name. For example, consider the following three what questions: What time is the train arriving? What city is the train stopping at?, and What is the name of the driver of the train? This problem can be solved by defining concept called question focus. The question focus is a phrase in the question that disambiguates it and emphasizes the type of answer being expected. For example in the three questions above

the question foci are shown. In the first two cases, the question focus tells us directly that a time and a city are being looked for. In the third case we know that driver is a type of person and hence that a person's name is being sought. For the purpose of this system the question focus is defined as the first noun group that is not a word "name" if the question word is of an ambiguous type.[2]

1.4.3 Website crawler

Crawling a website to get the relevant data is the task of website crawler. The same is implemented in the project using PHP for the purpose of database creation. The data of doctors is extracted from the website and is stored in the text files with related areas of expertise. The data of all the cardiologists is stored under the Cardiology text file.

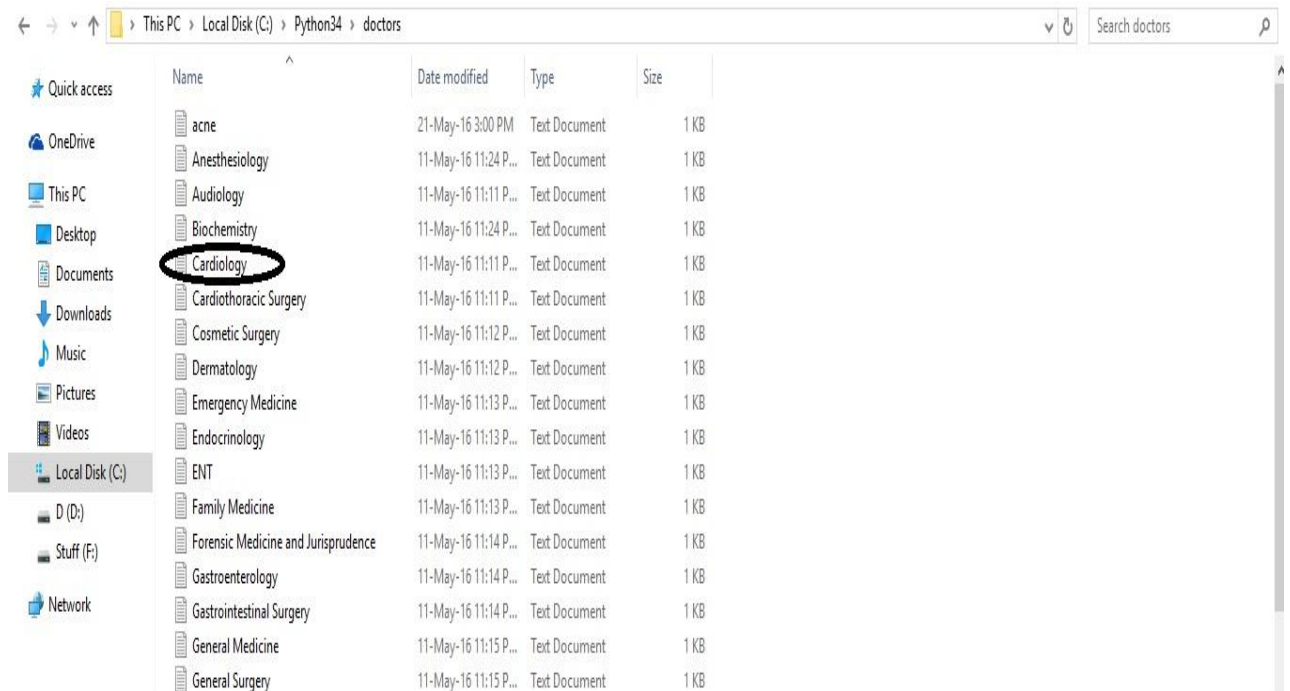


Fig 2 Text files generated using website crawler

1.4.4 GUI design

Designing the Graphical User Interface (GUI) was one of the module which involved the attractive button, text field interface for the user. The GUI development was done using Python PyQt4 which is the Python wrapper around the QT framework for creating graphical user interfaces, or GUIs.

Glimpse of the GUI is shown below:

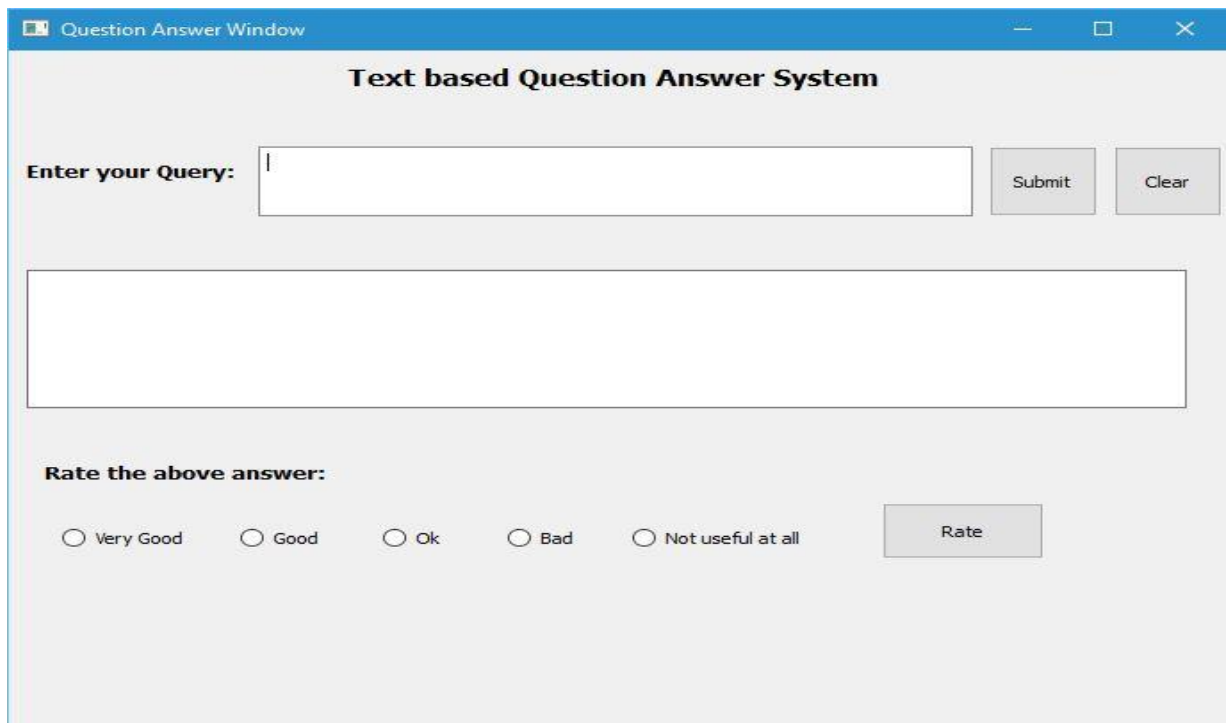


Fig 3 GUI design

1.4.5 File Access

File Access module is a test module that takes in the tagged user query, matches it with the preexisting strings in python dictionary and then opens and fetches data from the text files. This module is developed with the sole purpose to check the I/O operations in files and checking the same with the tagged words.

1.4.6 Text Summarizer

Text summarizer module is separately developed to summarize the data in the documents. It takes in a string of data and finds the most relevant data from this string based on score computed for each sentence. The scores for each of the sentences are compared and the ones with the highest scores are retained in the summarized documents. The extra unnecessary data is hence trimmed off to give the best out of the string.

The working of Text Summarizer is shown below for the disease “acne”

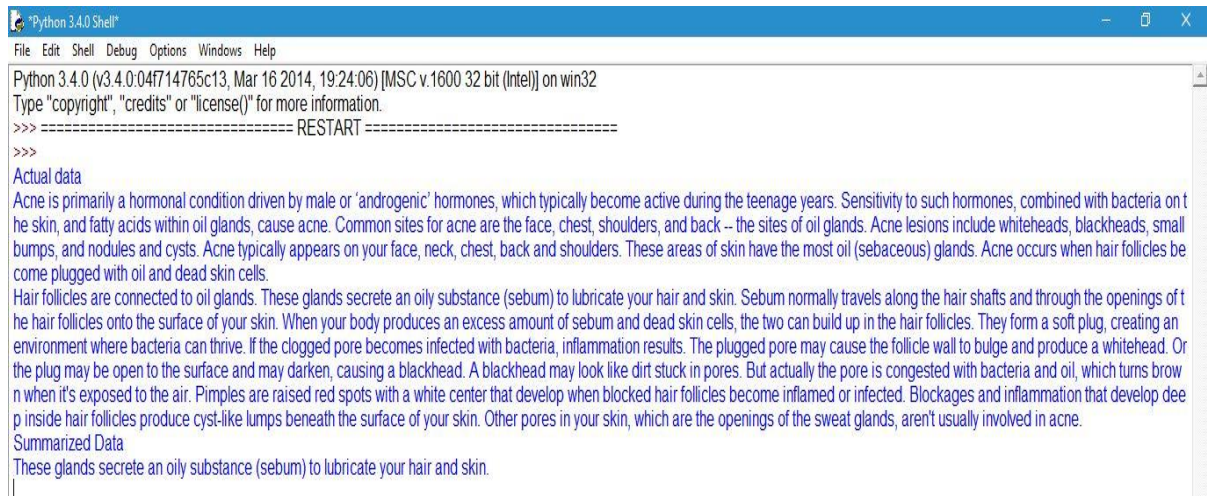


Fig 4 Text summarizer

1.5 Organization

The project is organized in the form of modules that are integrated to form the final module, or the complete project.

The different modules listed in the previous sections are organized in a way so that the overall project uses each of the independent module to run the overall tool of Question Answering in medical domain.

Firstly, website crawler was developed to generate the database. Once the database for the doctors and diseases was prepared, the second development phase included part of speech tagger. POS tagger as discussed tokenizes and tags the words in the sentence.

Text summarizer was developed next to facilitate the extraction of relevant text from the documents. In spite of listing the whole document, extracting the most relevant data based on score is done using this TS.

GUI was developed after TS where in the user can post and get his queries answered. In case the file or the disease queried doesn't exist in the database, a message is flashed to tell the user about the same.

The integration of all the modules and its organization is done using the iterative model.

CHAPTER – 2 LITERATURE SURVEY

2.1 Question Answering System

Question answering can be viewed as a sophisticated information retrieval (IR) task where a system automatically generates a search query from a natural language question and finds a concise answer from a set of documents. In the open domain factoid question answering task systems answer general questions like *Who is the creator of The Daily Show?*, or *When was Mozart born?*[3]

The question answering task has two reference inputs: the corpora to be used to extract the relevant answers and the question itself. Each of these inputs must be analyzed in a manner that makes the question-answer matching semantically relevant, easy to understand and potentially traceable. This matching process implies that we represent both questions and candidate answers (or whole corpus) in a homogeneous semantic representation that can be processed by information systems.

Since information retrieval is the first stage of question answering, its performance is an upper bound on the overall question answering system's performance. IR performance depends on the quality of document indexing and query construction. Question answering systems create a search query automatically from a user's question, through various levels of sophistication. The simplest way of creating a query is to treat the words in the question as the terms in the query. Some question answering systems apply linguistic processing to the question, identifying named entities and other query-relevant phrases. Others use ontologies to expand query terms with synonyms and hypernyms.

IR system recall is very important for question answering. If no correct answers are present in a document, no further processing will be able to find an answer. IR system precision and ranking of candidate passages can also affect question answering performance. If a sentence without a correct answer is ranked highly, answer extraction may extract incorrect answers from these erroneous candidates. Research shows that there is a consistent relationship between the quality of document retrieval and the overall performance of question answering systems.

Information retrieval (IR) for question answering consists of 2 steps: document retrieval and passage retrieval. Approaches to passage retrieval include simple word overlap, density based passage retrieval, retrieval based on the inverse document frequency (IDF) of matched and mismatched words, cosine similarity between a question and a passage, passage/sentence ranking by weighting different features, stemming and morphological query expansion, and voting between different retrieval methods. As in previous approaches, we use words and phrases from a question for passage extraction and experiment with using exactly matched phrases in addition to words. Similarly, we assign weights to sentences in retrieved documents according to the number of matched constituents.

As the search query is entered, the named entities are extracted from it and processed using various techniques mentioned. The technique used in this project is a simple word overlap which involves the string matching for the tagged words with that defined in the code. Once it is matched, the file related to that particular keyword is extracted and the data is fetched and displayed.

2.2 Natural Language Processing

Natural Language processing is a branch of computer science, artificial intelligence and linguistics concerned with the interactions between computers and human (natural) language. Natural language is any language that humans learn from their environment and use to communicate with each other. Whatever the form of the communication, natural languages are used to express our knowledge and emotions and to convey our responses to other people and to our surroundings.

Natural languages are usually learned in early childhood from those around us. Natural language processing is the collection of techniques employed to try and accomplish that goal. The field of natural language processing (NLP) is deep and diverse. Natural language processing (NLP) is a collection of techniques used to extract grammatical structure and meaning from input in order to perform a useful task as a result, natural language generation builds output based on the rules of the target language and the task at hand.

2.3 Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. [4]

NLTK is a broad-coverage toolkit that provides a simple, extensible, uniform framework for assignments, projects, and class demonstrations that needs processing of natural language. It is well documented, easy to learn, and simple to use. NLTK is unique among computational linguistics tools in its combination of three factors. First, it was deliberately designed as courseware and gives pedagogical goals primary status. Second, its target audience consists of both linguists and computer scientists, and it is accessible and challenging at many levels of prior computational skill. Finally, it is based on an easy-to-learn and easy-to-read programming language supporting rapid development and literate programming.

Simple things that can be done using NLTK include, tokenizing and tagging of sentences.

Let us say we have a sentence, “At eight o'clock on Thursday morning Arthur didn't feel very good.” Tokenizing and parts of speech tagging through NLTK is shown below.

Tokenize and tag some text:

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Fig 5. Tokenizing and tagging of text using NTLK.

Now let us identify the named entities in the following sentence.

Identify named entities:

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [(('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
            ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN')),
           Tree('PERSON', [(('Arthur', 'NNP')]),
              ('did', 'VBD'), ('n't', 'RB'), ('feel', 'VB'),
              ('very', 'RB'), ('good', 'JJ'), ('.', '.')])])
```

Fig 6. Naming entities in NLTK

2.3 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.

The project on QA system used python because of the following reasons:

- Python offers a shallow learning curve; it was designed to be easily learnt. [5]
- Python code is exceptionally readable, with transparent syntax and semantics.
- As an interpreted language, Python is suitable for interactive exploration.
- Python's light weight object oriented system makes it easy to encapsulate data and methods in classes.
- Python's recently added generator syntax makes it easy to create interactive implementations of algorithms. These interactive implementations can be used to step through" the algorithm, examining how its state changes as the algorithm progresses.
- Python's extensive standard library provides a great deal of power, when needed.

2.4 Text Summarization

Text summarization is the technique, where a computer summarizes a text. A text is entered into the computer and a summarized text is returned, which is a non-redundant extract from the original text. Automatic text summarization is based on statistical, linguistical and heuristic methods where the summarization system calculates how often certain key words. The key words belong to the so called open class words. The summarization system calculates the frequency of the key words in the text, which sentences they are present in, and where these sentences are in the text. It considers if the text is tagged with bold text tag, first paragraph tag or numerical values. All this information is compiled and used to summarize the original text.

2.5 Wamp server

WAMP is a Windows OS based program that installs and configures Apache web server, MySQL database server, PHP scripting language, phpMyAdmin (to manage MySQL database's), and SQLite Manager (to manage SQLite database's). WAMP is designed to offer an easy way to install Apache, PHP and MySQL package with an easy to use installation program instead of having to install and configure everything yourself. WAMP is so easy because once it is installed it is ready to go. You don't have to do any additional configuring or tweaking of any configuration files to get it running. [6]

2.5.1 PHP Admin

Allows you to change or add users and for making new databases phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the World Wide Web. PhpMyAdmin supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc), while you still have the ability to directly execute any SQL statement.[6]

Features:

- Intuitive web interface
- Import data from CSV and SQL
- Creating PDF graphics of your database layout
- Administering multiple servers
- Searching globally in a database or a subset of it
- Creating complex queries using Query-by-example (QBE)



Fig 7 phpMyAdmin

2.5.2 Apache

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

The Apache HTTP Server was launched in 1995 and it has been the most popular web server on the Internet since April 1996. It has celebrated its 20th birthday as a project in February 2015.[6]



Fig 8 Apache

2.5.3 SQL Server and database

SQL Server is a relational database management system from Microsoft that's designed for the enterprise environment. SQL Server runs on T-SQL (Transact -SQL), a set of programming extensions from Sybase and Microsoft that add several features to standard SQL, including transaction control, exception and error handling, row processing, and declared variables.[6]



Fig 9 MySQL

CHAPTER – 3 SYSTEM DEVELOPMENT

3.1 Software Requirements

- Python 3.4
- Wamp server
- Natural Language Toolkit
- Latest version of sublime text (text editor)
- Google Chrome or any browser

3.2 Hardware Requirements

- HDD: 200 MB free space
- Memory: 512 MB
- CPU: 2.2 GHz processor or more

3.3 Software Model

SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. [7]

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software development process.
- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

A typical Software Development life cycle consists of the following stages:

- Stage 1: Planning and Requirement Analysis
 - Stage 2: Defining Requirements
 - Stage 3: Designing the product architecture
 - Stage 4: Building or Developing the Product
 - Stage 5: Testing the Product
-
- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
 - **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
 - **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
 - **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
 - **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released.[7]

The QA system development used **prototype model** due to enhancements in the prototype that were made and change in the requirements every now and then. An early working prototype was delivered to which certain modifications were made as and when required. Using prototyping as the development strategy allowed to better understand the software requirements.

In prototyping, evolutionary prototyping was used as it is based on building actual functional prototypes with minimal functionality in the beginning. Using this technique only well understood requirements are included in the prototype and the requirements are added as and when they are understood.

What is Software Prototyping?

- Prototype is a working model of software with some limited functionality.
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is the stepwise approach to design a software prototype:

- **Basic Requirement Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.
- **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These

features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.

- **Review of the Prototype:** The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.
- **Revise and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like, time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

3.4 Algorithm

An algorithm is a procedure or formula for solving a problem. It involves the step by step approach to solve a particular problem using unambiguous steps.

Step by step algorithm for building the QA system is listed below:

1. Input a query from user
2. Tokenize and tag the query into respective parts of speech.
3. Match these POS with predefined rules
4. If matched, use the tagged words to open the file
5. If found, fetch the data
6. Apply text summarization
7. Print Data
8. If not matched or found, terminate.

Apart from algorithm design, GUI building and testing also played a major role in the project development.

The flow chart for the above mentioned steps is listed on the next page.

Flowchart:

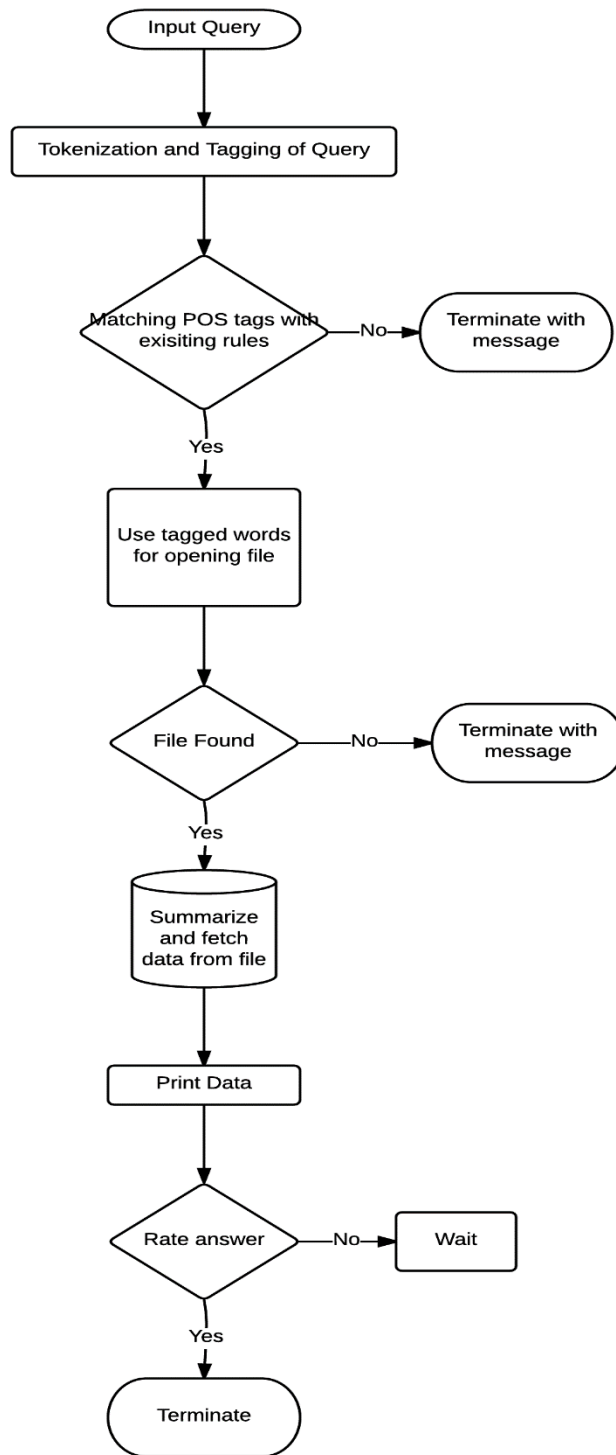


Fig 10 Flowchart of the project

3.5 Implementation

The implementation of the project involved various pieces of code.

3.5.1 Website crawler

```
$target_url = 'http://www.medindia.net/healthcare-directory/healthcare-directory.asp';  
$raw_data = file_get_contents(preg_replace('/\s|\n|\t/', '', $target_url));
```

The above code takes in the target url which is the URL of the website that is used to fetch the data. The function `file_get_contents` is used to get the contents of the website as in the source page.

The content in the form of HTML tags is hence stored in the variable named `$raw_data`.*

```
$file_name = str_replace("/", "_", $file_name);  
$file_name = str_replace(" ", "_", $file_name);  
$file_created = "doctors/" . $file_name . ".txt";  
$f = fopen($file_created, 'w');
```

The above code creates text files with name of the diseases as extracted from the source code and are stored in the variable named `$file_created`.

Each medical area eg. Cardiology, Neurology, etc. are created using this and names of the doctors are written inside the files.

For complete code on website crawler, refer Appendix A

The URL used on the previous page is the target URL which is used to create text files for different fields of medicine.

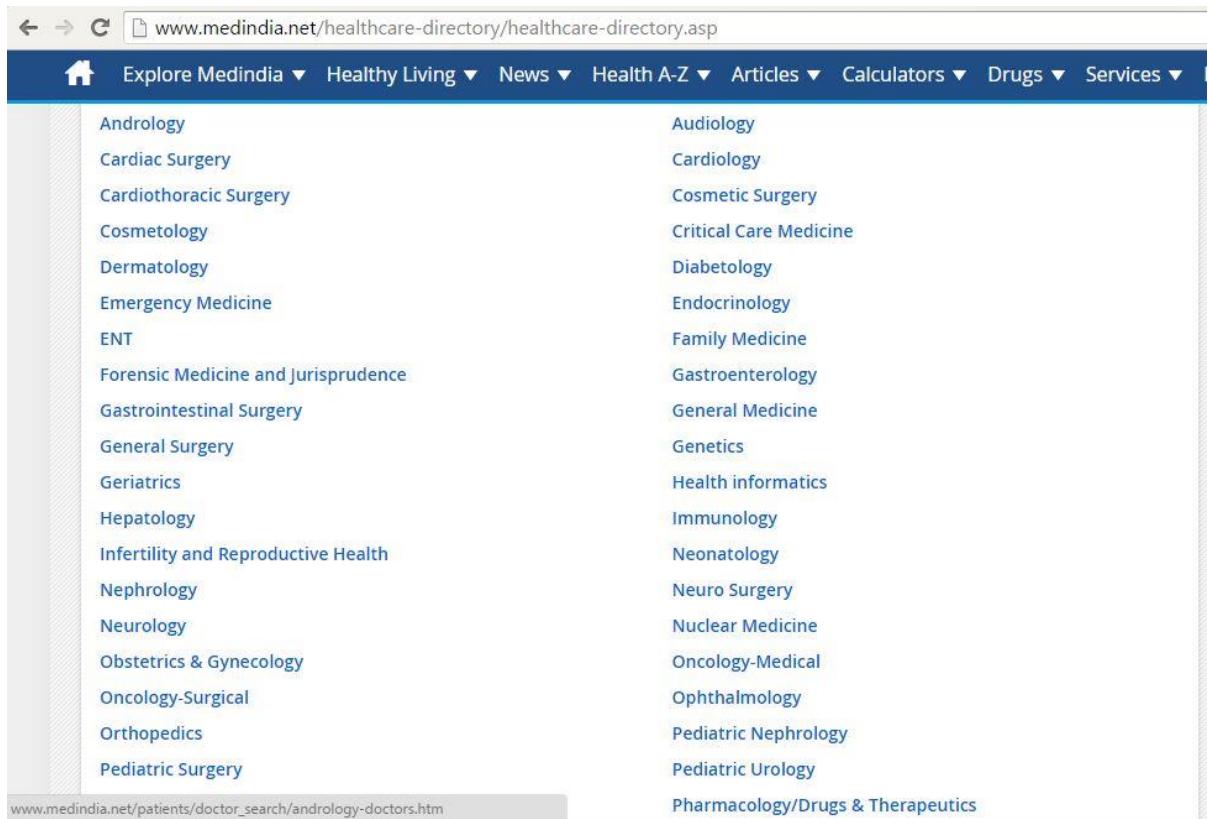


Fig 11 MedIndia website for data retrieval

The above image shows different fields that are then created as text files with names of doctors.

```

$path_doctor_name = 'http://www.drdata.in'.'/'.$disease_link;
echo $path_doctor_name."<br>";

$temp_file_doctors_names = file_get_contents($path_doctor_name);

fwrite($f, $temp_file_doctors_names);
fclose($f);

```

The data about the doctors is fetched from the link and stored in the files created. The variable `$temp_file_doctors_names` contains the names of the doctors as fetched from the URL.

The same is then written to the file.

```

for($i=1;$i<$total_links;$i++)
{
    $second_explode = explode('</option>', $first_explode[$i]);
    $disease_link = $second_explode[0];

    $third_explode = explode('">', $first_explode[$i]);
    $fourth_explode = explode("</button>", $third_explode[1]);
    $file_name = $fourth_explode[0];
}

```

The code creates files iteratively for the total number of links that are stored in the variable `$total_links`. Each disease file is hence created using the for loop.

On running the above piece of code, text files are created with names of the diseases and contains the names of the doctors inside.

Anesthesiology	11-May-16 11:24 PM	Text Document	1 KB
Audiology	11-May-16 11:11 PM	Text Document	1 KB
Biochemistry	11-May-16 11:24 PM	Text Document	1 KB
Cardiology	11-May-16 11:11 PM	Text Document	1 KB
Cardiothoracic Surgery	11-May-16 11:11 PM	Text Document	1 KB
Cosmetic Surgery	11-May-16 11:12 PM	Text Document	1 KB
Dermatology	11-May-16 11:12 PM	Text Document	1 KB
Emergency Medicine	11-May-16 11:13 PM	Text Document	1 KB
Endocrinology	11-May-16 11:13 PM	Text Document	1 KB
ENT	11-May-16 11:13 PM	Text Document	1 KB
Family Medicine	11-May-16 11:13 PM	Text Document	1 KB
Forensic Medicine and Jurisprudence	11-May-16 11:14 PM	Text Document	1 KB
Gastroenterology	11-May-16 11:14 PM	Text Document	1 KB
Gastrointestinal Surgery	11-May-16 11:14 PM	Text Document	1 KB
General Medicine	11-May-16 11:15 PM	Text Document	1 KB
General Surgery	11-May-16 11:15 PM	Text Document	1 KB
Genetics	11-May-16 11:15 PM	Text Document	1 KB
Geriatrics	11-May-16 11:15 PM	Text Document	1 KB
Health informatics	11-May-16 11:16 PM	Text Document	1 KB
Hematology	11-May-16 11:24 PM	Text Document	1 KB
Hepatology	11-May-16 11:16 PM	Text Document	1 KB
Immunology	11-May-16 11:16 PM	Text Document	1 KB

Fig 12 List of files for doctors in different medical domains

3.5.2 Tokenizing and tagging sentences

Input sentence by the user is tokenized and tagged into respective parts of speech using the following piece of code.

```

stopset = set(stopwords.words('english'))
sentence = input("Enter query:\n")
punctuations = '?\.\@|'
trimmed=""
for char in sentence:
    if char not in punctuations:
        trimmed = trimmed + char
sent_words = word_tokenize(trimmed)
tokens = [w for w in sent_words if not w in stopset]
print ('After removing stop words:')
print (tokens)

tagged_sent = ()
tagged_sent = pos_tag(tokens)
print ('Tagged words:')
print(tagged_sent)

```

Stopset is the set of all the words in the English language that when removed do not affect the meaning of the sentence. Punctuations is a string of symbols that need to be removed from the input text before processing. Tokens contains the list of words after removing the stop words. Tagged_sent contains the list of tagged words with their respective parts of speech.

3.5.3 GUI design

GUI is developed to better interact with the user and provide him a pleasant experience.

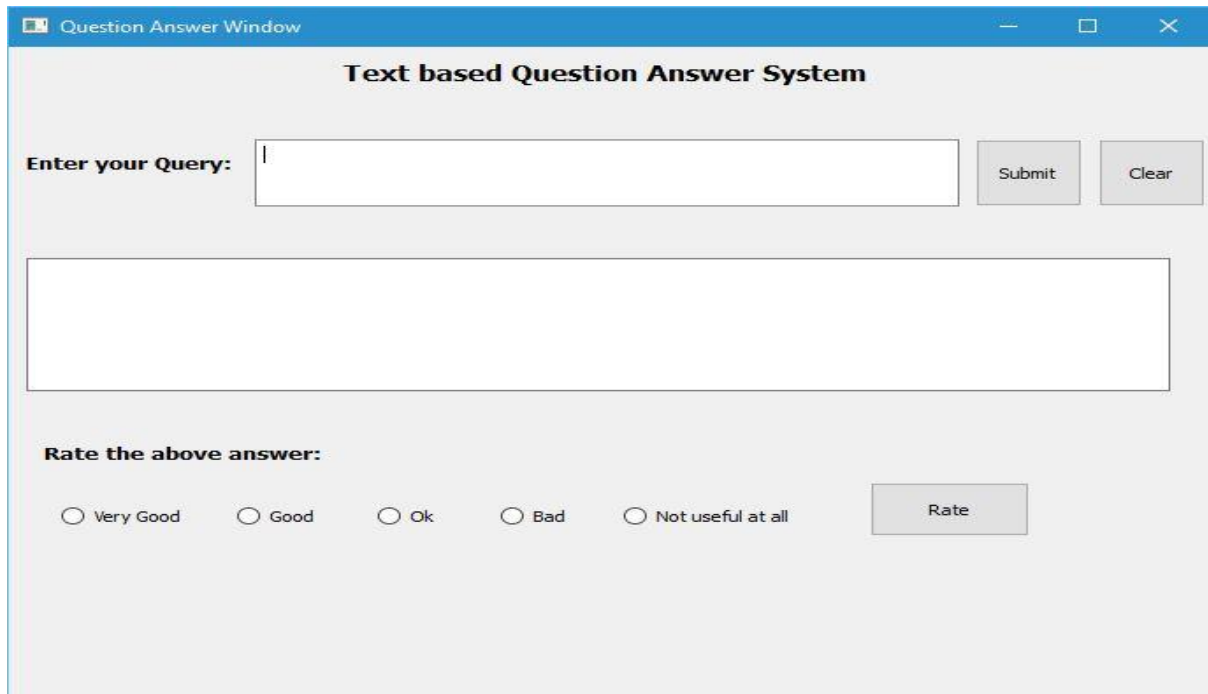
```

self.centralwidget = QtGui.QWidget(QuestionAnswerWindow)
self.centralwidget.setObjectName(_fromUtf8("centralwidget"))
self.query_text = QtGui.QPlainTextEdit(self.centralwidget)
self.query_text.setGeometry(QtCore.QRect(140, 70, 401, 51))
self.query_text.setObjectName(_fromUtf8("query_text"))
self.text_based_qas = QtGui.QLabel(self.centralwidget)
self.text_based_qas.setGeometry(QtCore.QRect(190, 10, 298, 19))
self.text_based_qas.setObjectName(_fromUtf8("text_based_qas"))
self.enter_your_query_label = QtGui.QLabel(self.centralwidget)
self.enter_your_query_label.setGeometry(QtCore.QRect(10, 80, 117, 16))
self.enter_your_query_label.setObjectName(_fromUtf8("enter_your_query_label"))
self.answer_text_box = QtGui.QTextBrowser(self.centralwidget)

```


The above code is made using the help of PyQt tool of python. The simple drag and drop functionality makes it easy for the developer to design the interface.

The GUI looks like:



3.5.4 Text Summarizer

The text summarizer module takes in a string block and tokenizes it into the corresponding sentences. The sentences are then further tokenized into words, these are then passed into a defined function “compute_score”.

Each word is then checked for its importance using the function “is_important” and if not, is discarded. Hence, the best sentence is returned as an output.

```

def summarize_block(block):
    """Return the sentence that best summarizes block"""
    if not block:
        return None
    sents = nltk.sent_tokenize(block)
    word_sents = list(map(nltk.word_tokenize, sents))
    d = dict((compute_score(word_sent, word_sents), sent)
             for sent, word_sent in zip(sents, word_sents))
    return d[max(d.keys())]

```

3.5.5 Final module

The final module of QA system is about the string matching and data fetching. Once the relevant data is fetched, it is then displayed to the user. The user then rates the answer. Based on the rating the database is updated to reflect the same.

```

rules = {
    0 : "WP NN ",
    1 : "NNP NN ",
    2 : "NN ",
    3 : "NN NN ",
    4 : "WP NN NN ",
    5 : "WP NNP ",
    6 : "NNS VBP ",
    7 : "WRB JJ NN ",
    8 : "WP VBZ NN ",
    9 : "NNS VBD ",
    10 : "WRB NN VBD ",
    11 : "WRB NNS VBD ",
    12 : "NNS NN ",
    13 : "WP VBD NN ",
    14 : "WP VBZ VBP ",
    15 : "WP VBP NN ",
    16 : "NN NNS VBP ",
    17 : "WP JJS NN NN ",
    18 : "WP JJS NNS VBP "
}

```

The previous code is the python dictionary with predefined rules to match the tagged string. If matched, the if statements following this code are executed and data from the files is fetched. The if statements check for the index and if matched computes the path of the file. The path is checked inside the try statement and if it doesn't exist, the except statements are executed.

```
if index == 0:
    NN_is = [word for word,pos in tagged_sent if pos == 'NN']
    path = os.path.join('define', '{}.txt'.format(NN_is[0]))

    try:
        with open(path) as searchfile:
            file_contents = searchfile.readlines()
            file_contents_str = ''.join(file_contents)
            print_data = text_summ.summarize_block(file_contents_str)
            searchfile.close()
    except (IOError, NameError, FileNotFoundError) as e:
        error_flag = 1;
        #print(print_data)
```

The code above checks for the definition of a particular disease in the database which is stored in the NN_is[0] variable. If the disease is found the string data is fetched and stored in the print_data variable after applying text summarization. This string data is then displayed in the GUI.

CHAPTER 4 – PERFORMANCE ANALYSIS

4.1 Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. [8]

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.

Let us now analyze the unit testing for each of the modules available.

4.1.1 POS tagger unit testing:

POS tagger takes in a sentence as the input and tags it for the respective parts of speech.

- Test case 1:

Input: What is asthma?

Expected output: “What as WP” and “asthma as NN”

Actual output:

```
Enter query:  
What is asthma?  
After removing stop words:  
['What', 'asthma']  
Tagged words:  
[('What', 'WP'), ('asthma', 'NN')]  
>>> |
```

- Test case 2:

Input: What are the causes of insomnia?

Expected output: “What as WP”, “causes as VBZ” and “insomnia as NN”

Actual output:

```
Enter query:
What are the causes of insomnia?
After removing stop words:
['What', 'causes', 'insomnia']
Tagged words:
[('What', 'WP'), ('causes', 'VBZ'), ('insomnia', 'NN')]
>>>|
```

4.1.2 Text summarizer unit testing:

It takes in a string of characters as input and gives out the best sentence as output.

- Test case 1:

Input: Asthma is a chronic disease involving the airways in the lungs. These airways, or bronchial tubes, allow air to come in and out of the lungs. If you frequently experience shortness of breath or you hear a whistling or wheezy sound in your chest when you breathe, you may have asthma — a chronic condition that causes inflammation and narrowing of the bronchial tubes, the passageways that allow air to enter and leave the lungs. If people with asthma are exposed to a substance to which they are sensitive or a situation that changes their regular breathing patterns, the symptoms can become more severe.

Expected output: If you frequently experience shortness of breath or you hear a whistling or wheezy sound in your chest when you breathe, you may have asthma — a chronic condition that causes inflammation and narrowing of the bronchial tubes, the passageways that allow air to enter and leave the lungs.

Actual output:

Enter data:

Asthma is a chronic disease involving the airways in the lungs. These airways, or bronchial tubes, allow air to come in and out of the lungs. If you frequently experience shortness of breath or you hear a whistling or wheezy sound in your chest when you breathe, you may have asthma — a chronic condition that causes inflammation and narrowing of the bronchial tubes, the passageways that allow air to enter and leave the lungs. If people with asthma are exposed to a substance to which they are sensitive or a situation that changes their regular breathing patterns, the symptoms can become more severe.

After summarizing:

If you frequently experience shortness of breath or you hear a whistling or wheezy sound in your chest when you breathe, you may have asthma — a chronic condition that causes inflammation and narrowing of the bronchial tubes, the passageways that allow air to enter and leave the lungs.

>>> |

- Test case 2:

Input: Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites.

Respiratory infections, such as the common cold.

Physical activity (exercise-induced asthma)

Cold air.

Air pollutants and irritants, such as smoke.

Strong emotions and stress

Sulfites and preservatives added to some types of foods and beverages, including shrimp, dried fruit, processed potatoes, beer and wine

Expected output: Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites.

Actual output:

```
Enter data:  
Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites.  
Respiratory infections, such as the common cold.  
Physical activity (exercise-induced asthma)  
Cold air.  
Air pollutants and irritants, such as smoke.  
Strong emotions and stress  
Sulfites and preservatives added to some types of foods and beverages, including shrimp, dried fruit, processed potatoes, beer and wine  
  
After summarizing:  
Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites.  
>>>
```

4.1.3 Final module with GUI unit testing:

This module is the final project module that contains the GUI, takes in string query input and processes it to output the relevant answer to the query.

- Test case 1:

Input: What do you mean by acne?

Expected output: These glands secrete an oily substance (sebum) to lubricate your hair and skin.

Actual output:

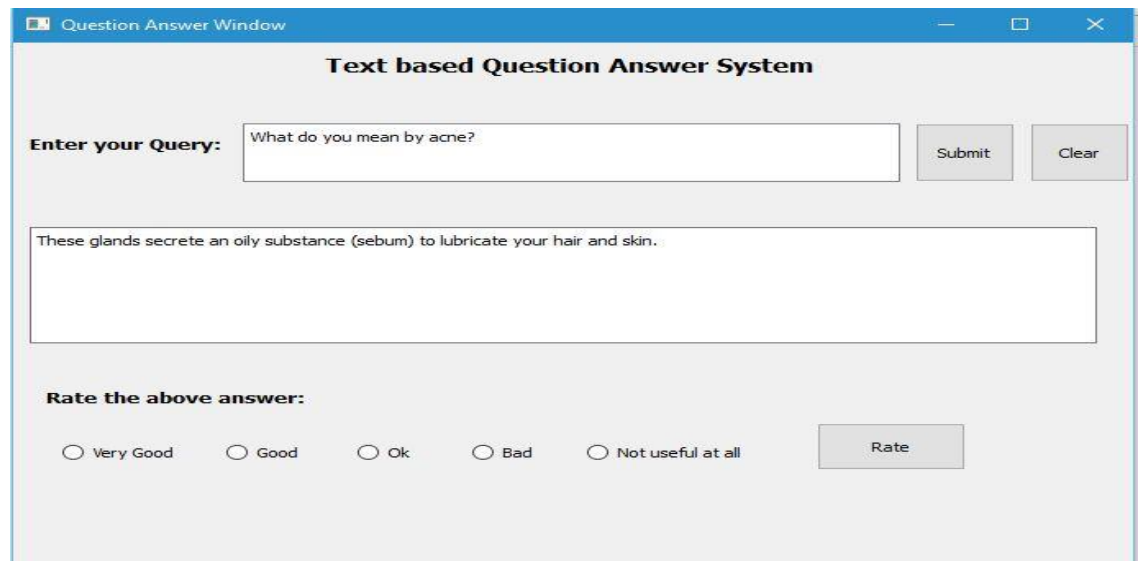


Fig 13 Interactive GUI design

- Test case 2:

Input: symptoms of asthma

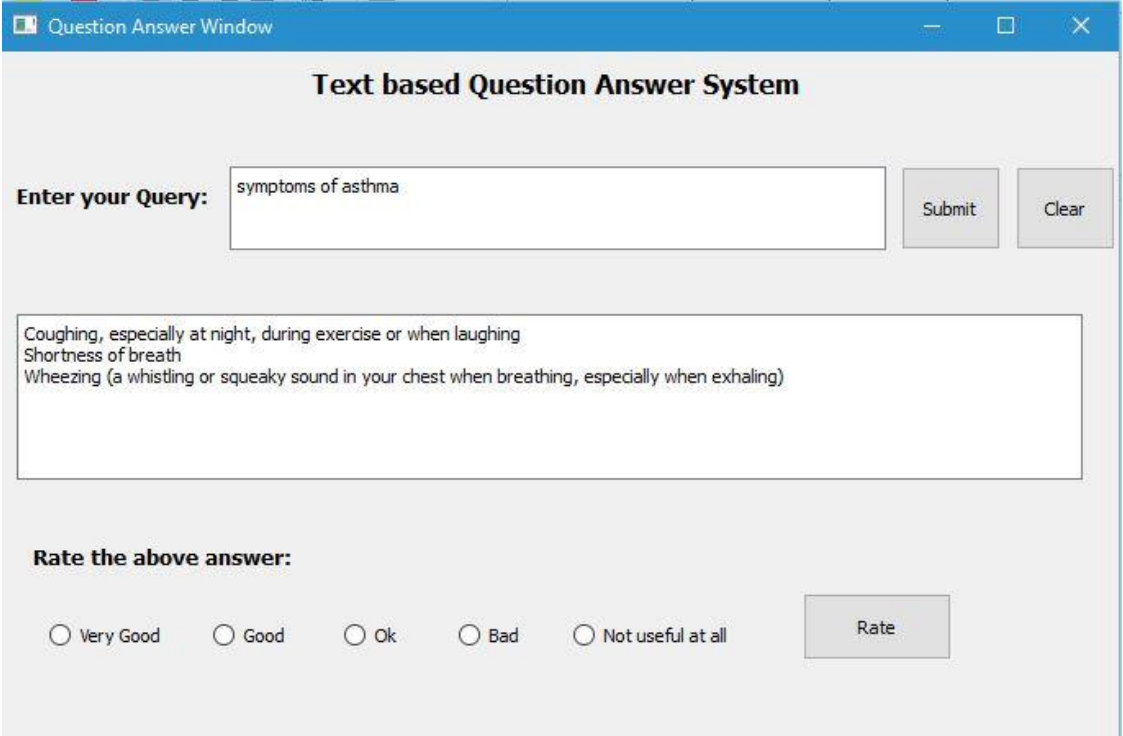
Expected output:

Coughing, especially at night, during exercise or when laughing

Shortness of breath

Wheezing (a whistling or squeaky sound in your chest when breathing, especially when exhaling)

Actual output:



The screenshot shows a web application window titled "Question Answer Window" with a subtitle "Text based Question Answer System". It features a text input field labeled "Enter your Query:" containing the text "symptoms of asthma". To the right of the input field are "Submit" and "Clear" buttons. Below the input field is a large text area displaying the expected output: "Coughing, especially at night, during exercise or when laughing", "Shortness of breath", and "Wheezing (a whistling or squeaky sound in your chest when breathing, especially when exhaling)". At the bottom, there is a "Rate the above answer:" section with five radio button options: "Very Good", "Good", "Ok", "Bad", and "Not useful at all", followed by a "Rate" button.

Fig 14. Working of project

4.2 Black-box testing:

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. [9]

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

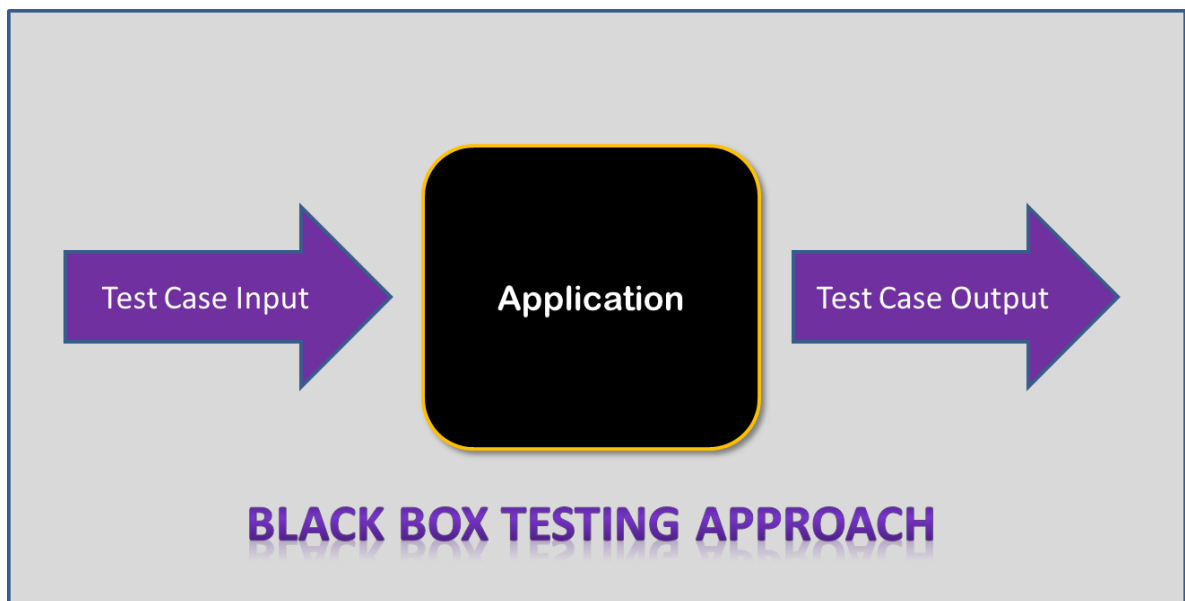


Fig 15. Black box testing approach

Test cases:

- Test case 1:

Input: what are the symptoms of asthma.

Expected output:

Coughing, especially at night, during exercise or when laughing

Shortness of breath

Wheezing (a whistling or squeaky sound in your chest when breathing, especially when exhaling)

Actual output:

Question Answer Window

Text based Question Answer System

Enter your Query:

Coughing, especially at night, during exercise or when laughing
Difficulty breathing
Wheezing (a whistling or squeaky sound in your chest when breathing, especially when exhaling)

Rate the above answer:

Very Good Good Ok Bad Not useful at all

Fig 16. Interactive project design

4.3 Python code testing using cProfile:

Profiling a Python program is doing a dynamic analysis that measures the execution time of the program and everything that compose it. That means measuring the time spent in each of its functions. This will give you data about where your program is spending time, and what area might be worth optimizing. [10]

Since Python 2.5, Python provides a C module called cProfile which has a reasonable overhead and offers a good enough feature set.

cProfile testing output:

```
-----
>>>
    3 function calls in 0.000 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1   0.000   0.000   0.000   0.000 <string>:1(<module>)
     1   0.000   0.000   0.000   0.000 {built-in method exec}
     1   0.000   0.000   0.000   0.000 {method 'disable' of '_Isprof.Profiler' objects}
```

The output above shows that the overall time spent in executing the function is 0.000 seconds which tells us that the file indexing and searching time is NULL.

CHAPTER - 5 CONCLUSIONS

5.1 Conclusions

The primary goal of this project is to develop a text based question answer system on medical domain for learning purpose. The aim is achieved meeting the project requirements. The tool developed answers the user queries regarding the definition, causes, symptoms, and list of doctors for a particular disease.

The tool developed passed all the relevant test cases and generates the information that best fits the query.

5.2 Future Scope

The project on text based question answering on medical domain can be further enhanced to find the list of best doctors in a particular area, locality or city. Since the project only provide answer to a limited number of diseases, it can be enhanced further to cover a larger spectrum of diseases. The feedback mechanism of the GUI can be better used to update the database.

REFERENCES

- [1] Wikipedia contributors, (2016, May 22) “Question Answering”, [Online]. Available: http://en.wikipedia.org/wiki/Question_answering.
- [2] Richard J Cooper and Stefan M Ruger, “A Simple Question Answering System”, TREC CDs and TIPSTER.
- [3] Svetlana Stoyanchev, and Young Chol Song, and William Lahti, “Exact Phrases in Information Retrieval for Question Answering”, pp 9-16, August 2008
- [4] Steven Bird, Ewan Klein, and Edward Loper, Natural Language Processing with Python, O’Reilly Media, 2009.
- [5] Edward Loper. 2004. NLTK: Building a pedagogical toolkit in Python. In PyCon DC 2004. Python Software Foundation.
- [6] Talha Asif, (2016, February 5) “Wamp server tutorials”, [Online]. Available: <http://www.wamptutorials.blogspot.in/>.
- [7] Tutorialspoint Team, (2016, February 16) “Tutorials point”, [Online]. Available: http://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm
- [8] Wikipedia contributors, (2016, March 30), “Unit Testing”. [Online]. Available: http://en.wikipedia.org/wiki/Unit_testing.
- [9] Wikipedia contributors, (2016, April 19) “Black box testing”. [Online]. Available: http://en.wikipedia.org/wiki/Black-box_testing.
- [10] Doug Hellmann, (2016, May 15), “PyMOTW”. [Online]. Available: <http://pymotw.com/2/profile/>.

APPENDICES

Appendix – A: Website Crawler code

```
<?php
error_reporting(0);

$target_url = 'http://www.medindia.net/healthcare-directory/healthcare-directory.asp';
$raw_data = file_get_contents(preg_replace('/\s|\n|\t/', '', $target_url));
$raw_data_file = 'doctors_raw_data.txt';
$check = file_put_contents($raw_data_file, $raw_data);
if($check)
    echo "Doctors Raw Database prepared!<br>";

$first_explode = explode('<option value=""', $raw_data);

$total_links = count($first_explode);

for($i=1;$i<$total_links;$i++)
{
    $second_explode = explode('</option>', $first_explode[$i]);
    $disease_link = $second_explode[0];

    $third_explode = explode('">', $first_explode[$i]);
    $fourth_explode = explode("</button>", $third_explode[1]);
    $file_name = $fourth_explode[0];

    $file_name = str_replace("/", "_", $file_name);
    $file_name = str_replace(" ", "_", $file_name);
}
```

```
$file_created = "doctors/" . $file_name . ".txt";
```

```
$f = fopen($file_created, 'w');
```

```
$path_doctor_name = 'http://www.drdata.in/' . $disease_link;
```

```
echo $path_doctor_name . "<br>";
```

```
$temp_file_doctors_names = file_get_contents($path_doctor_name);
```

```
fwrite($f, $temp_file_doctors_names);
```

```
fclose($f);
```

```
$fifth_explode = explode('<td data-title="Name" valign=', $temp_file_doctors_names);
```

```
$total_doctors = count($fifth_explode);
```

```
echo $total_doctors . "<br>";
```

```
}
```

```
?>
```

Appendix – B: Text Summarizer

```
from __future__ import print_function

import codecs
import nltk
from nltk.corpus import stopwords
import re
import string
import sys

_IS_PYTHON_3 = sys.version_info.major == 3

stop_words = stopwords.words('english')

LOWER_BOUND = .20

UPPER_BOUND = .90

def u(s):
    if _IS_PYTHON_3 or type(s) == unicode:
        return s
    else:
        # not well documented but seems to work
        return codecs.unicode_escape_decode(s)[0]
```



```

def is_unimportant(word):
    return word in [',', '!', ';', ] or "\" in word or word in stop_words

def only_important(sent):
    return filter(lambda w: not is_unimportant(w), sent)

def compare_sents(sent1, sent2):
    if not len(sent1) or not len(sent2):
        return 0
    return len(set(only_important(sent1)) & set(only_important(sent2))) / ((len(sent1) + len(sent2)) / 2.0)

def compare_sents_bounded(sent1, sent2):
    cmpd = compare_sents(sent1, sent2)
    if cmpd <= LOWER_BOUND or cmpd >= UPPER_BOUND:
        return 0
    return cmpd

def compute_score(sent, sents):
    """Computes the average score of sent vs the other sentences (the result of
    sent vs itself isn't counted because it's 1, and that's above
    UPPER_BOUND)"""
    if not len(sent):
        return 0
    return sum(compare_sents_bounded(sent, sent1) for sent1 in sents) / float(len(sents))

```

```
def summarize_block(block):  
    if not block:  
        return None  
    sents = nltk.sent_tokenize(block)  
    word_sents = list(map(nltk.word_tokenize, sents))  
    d = dict((compute_score(word_sent, word_sents), sent)  
            for sent, word_sent in zip(sents, word_sents))  
    return d[max(d.keys())]
```