Intelligent Question Answering System for HIPAA and Corporate Regulations

by

Vhinesh Ravi

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Supervised by

Dr. Carol Romanowski

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences Rochester Institute of Technology Rochester, New York

May 2017
Acknowledgments

I would like to take this opportunity to express my deepest gratitude to my project advisor, Dr. Carol Romanowski for her continuous support and guidance throughout the entire duration of the project. I would also like to thank my colloquium advisor, Dr. Reynold Bailey for his feedback and inputs during the milestone presentations which helped me in improvising my presentation skills.
Abstract

Intelligent Question Answering System for HIPAA and Corporate Regulations

Vhinesh Ravi

Supervising Professor: Dr. Carol Romanowski

Living in the world of data explosion where petabytes of data are being generated every minute, the necessity of putting this data to use is more than a necessity. Having said that, it is essential to know that a significant portion of all the existing data is unstructured, lying in the forms of raw text, documents, books etc. One way of putting this large unstructured data to use is by building a system for efficient search and retrieval of this unstructured data. The area of Question and Answering has been researched over a long period resulting in significant advancements in the area. In this project, a cognitive-based Question and Answering system is built using IBM Watson services for the HIPAA and Corporate regulation documents of the ARC of Monroe county.
## Contents

Acknowledgments ................................................................. ii

Abstract .................................................................................. iii

1 Introduction ........................................................................... 1  
  1.1 Background ......................................................................... 2  
    1.1.1 Unstructured Information Management - UIMA ............... 2  
    1.1.2 OpenCog ................................................................. 2  
    1.1.3 IBM Watson ........................................................... 2  
  1.2 Goal .................................................................................. 3  
  1.3 Motivation .......................................................................... 3  
  1.4 Overview ........................................................................... 4

2 Design and Approach ............................................................. 5  
  2.1 System Design ................................................................. 5

3 Implementation ....................................................................... 7  
  3.1 Data Collection .................................................................. 7  
  3.2 Data Cleaning and Preprocessing ......................................... 7  
  3.3 Retrieve and Rank Implementation ......................................... 11  
    3.3.1 Retrieve Component .................................................. 12  
    3.3.2 Ranker Component .................................................. 14  
  3.4 Testing the system ............................................................ 17

4 Analysis .................................................................................. 21

5 Conclusions ........................................................................... 22  
  5.1 Current Status ................................................................. 22  
  5.2 Future Work ................................................................. 22
List of Figures

2.1 Question and Answering system design ............................................. 5
3.1 Data Cleaning and Preprocessing steps ............................................. 7
3.2 Normal document converted to document with smaller paragraph ........... 8
3.3 Transformed document converted to answer unit ................................. 9
3.4 Answer units parsed to form new Answer unit with only required tags ...... 10
3.5 The script for calling the Data Conversion service using Python SDK ....... 11
3.6 The Python script for creating the SOLR cluster ................................ 12
3.7 The Python script for adding the answer units to collection .................. 13
3.8 Testing of retrieve component by asking a question ............................ 14
3.9 Sample of training set ....................................................................... 15
3.10 Python script for training the ranker ................................................ 15
3.11 Python script for testing the trained ranker ....................................... 16
3.12 Answer from Rank component ......................................................... 17
3.13 Answer from Retrieve component ..................................................... 18
3.14 Answer from Rank component ......................................................... 18
3.15 Answer from Retrieve component ..................................................... 19
3.16 Answer from Rank component ......................................................... 19
3.17 Answer from Retrieve component ..................................................... 20
Chapter 1

Introduction

The cognitive computing systems are a new class of systems that are designed to imitate the functionality of the human brain. The functioning of these systems is modeled after the functioning of the human brain. They are an amalgam of advanced multidisciplinary areas within computer science including Artificial Intelligence, Machine Learning, Intelligent Computing, Natural Language Processing, Speech Processing, and others[1]. These systems are highly intelligent and are capable of making decisions using the past experiences as well as the rationality of the context that it is dealing with. These systems are capable of learning from experiences that it encounters, just like how humans can learn from experiences. The decision making of these systems tends to improve over time based on the situations and context the system is dealing with.

The most interesting capability of these cognitive systems is the ability to handle all forms of data including structured, semi-structured and unstructured data. Among the mentioned three classifications, unstructured data contributes to the major chunk of data explosion and it also requires complicated methodologies to process them. The ability of cognitive systems to deal with this unstructured data makes them unique from other similar intelligent systems. This unique ability of cognitive systems to reason and understand the unstructured data allows us to create applications with capabilities to directly interact with humans, as we as humans are a natural source of this unstructured data.
1.1 Background

This section gives a brief overview of some of the popular frameworks for building cognitive systems designed to deal with large unstructured data.

1.1.1 Unstructured Information Management - UIMA

Unstructured Information Management[6] is a standard for doing natural language analysis.[5] It is an architectural framework for building applications that are capable of doing analysis, discovery, and delivery of useful relevant information to the input query. This framework remains as a base for many current successful complicated cognitive systems[5].

1.1.2 OpenCog

The OpenCog[7] is a framework for integrating various individual artificial intelligence methods and techniques to arrive at a coherent solution. The different parts of the human brain are capable of performing different functions, for a given problem, the human brain arrives at a coherent solution by combining the capabilities of its various parts[7]. The OpenCog addresses this problem of integration in cognitive systems, by providing a framework for combining the multiple intelligent agents to use the same architecture, knowledge base and state transition between them.

1.1.3 IBM Watson

The IBM Watson[2] is one of the most advanced cognitive based question answering systems ever developed. It takes the question in natural language form and provides back with the relevant answer. To highlight its capability, it defeated humans in the jeopardy game with a huge margin. This is phenomenal, considering the fact that it has to process a large volume of unstructured data and make a decision to identify the relevant answer among multiple possible answers. This capability of decision making distinguishes it from other existing keyword based search systems.
The Watson is one of the successful cognitive system developed following the UIMA framework. When a question is posed to Watson, it processes the question by undergoing multiple stages before retrieving an answer. Some of the major steps involved in processing the input query include "Question analysis, Query decomposition, Hypothesis generation, soft filtering, Evidence scoring, Merging and ranking, Answer and confidence”[9]. The main idea is that for a given question many different hypotheses are created and the relevant answers generated for each of the hypotheses are filtered by scoring techniques and evidence matching [8].

The IBM Watson services[2] are available to the public through its cloud platform called IBM Bluemix. IBM Bluemix allows users to avail a plethora of services including Watson services to custom build the user’s cloud applications based on their requirements.

1.2 Goal

The goal of the project is to create a closed domain question and answering system that takes input in the form of questions in natural language and provide the relevant answer back in natural language. The domain for the Q&A system is HIPAA and Corporate Regulations. The objective is to build the Q&A system by using Watson, through exploiting its ability to understand the unstructured data.

1.3 Motivation

Living in the age of data explosion, a large number of applications exploiting the potential of data has been developed. Considering the vastness of the unstructured data present, the knowledge and insights to be explored from them remain significant. Thus developing a cognitive question and answering system would be one such effort in bringing this large volume of unstructured data to use.
If satisfactory results could be obtained from this closed domain question and answering system, the model could be replicated to other domains where it could make a significant difference. The current version of this project is developed for HIPAA and Corporate Regulations data of the ARC of Monroe County. This system can be used by employees, contractors and students working in the ARC of Monroe County, aiding them in understanding the HIPAA regulations, as the nature of their work would require them to comply with HIPAA regulations.

1.4 Overview

The Organization of the report would be as follows; The chapter 2 would discuss the design and approach followed in creating the Q&A system. The chapter 3 would provide the implementation details for each of the major tasks involved in creating the Q&A system. The chapter 4 would provide an analysis of the results obtained and the chapter 5 would conclude the project report with the current work and the future work which has to be done.
Chapter 2
Design and Approach

2.1 System Design

Figure 2.1: Question and Answering system design

The above system model could be briefly summarized as below,

- The system receives an input query from the user.
• The knowledge base is already created and populated into the SOLR cluster.
• Now, based on the input query, the answer units are retrieved from the SOLR cluster.
• Then the retrieved answer units are ranked using the IBM Ranker service which is already trained using the training data.
• Now the top 10 relevant answer units are retrieved to the user ordered in terms of their relevancy.

The key tasks involved in devolved in developing a question and answering system are listed below,

**Data Collection:** The documents are collected based on the chosen domain in our case HIPAA and Corporate Regulation documents of the ARC of Monroe County.

**Data Cleaning and Preprocessing:** The collected documents should be cleaned and converted to the format that is digestible by the SOLR.

**Retrieve and Rank Implementation:** This is the heart of the system where first the retrieve component is implemented followed by the ranker component. Python SDK of Watson is used for the implementation purpose.

**Training and Testing the model:** Once the Ranker is created, it should be trained using the training data. Then the trained Ranker can be tested by using a testing set consisting of randomly created questions different from the training set.
Chapter 3

Implementation

3.1 Data Collection

The data set for the project is the HIPAA and Corporate Compliance documents. The dataset consisting of HIPAA and Corporate Compliance documents are collected from the ARC of Monroe county website [3]. There are about 77 pdf documents with raw text explaining the various rules, regulation, and guidelines to followed for the employees working in the Arc of Monroe county.

3.2 Data Cleaning and Preprocessing

The document Cleaning and Preprocessing require sequences of tasks to be followed before generating the answer unit, Which is the output of this phase. Figure 3.1 shows the major tasks involved in this phase.

![Figure 3.1: Data Cleaning and Preprocessing steps](image)

Figure 3.1: Data Cleaning and Preprocessing steps
The documents in the dataset are distinct from each other, where each document represents a unique policy. Though there is document wise distinction existing, there was a lack of segmentation or categorization existing within the document. Each of the document consists of large paragraphs of text with little or no categorization of the topics within. This lack of segmentation within the documents would result in poor performance of the system as it would be difficult for the Ranker to see patterns existing within data without proper distinction. This required each of the document to broken down into smaller paragraphs with a heading explaining what the paragraph is about.

Figure 3.2: Normal document converted to document with smaller paragraphs
The paragraph wise broken documents need to be converted to JSON format units called answer units. The Python script is written to read through all of the transformed documents and call the Watson Document Conversion service on each document, which generates answer unit for each paragraph in the document. Each answer unit generated has a unique answer id, title, content, media type, type and other fields. Figure 3.3 shows the document and its converted answer units.

![Figure 3.3: Transformed document converted to answer unit](image)

The answer units obtained through Document Conversion service needs to be parsed to extract essential tags, based on the requirements. A Python script is written to read through
each of the generated answer units and extract the required tags to form the new answer unit. Now the transformed answer unit contains only the following tags id, title, and text.

Figure 3.4: Answer units parsed to form new Answer unit with only required tags

The figure 3.5 shows the Python script that is used for reading through the documents and calling the data conversion service to convert each of the documents to answer units. Before using the Data Conversion service through Python, We have to register for IBM Data Conversion serve and obtain the keys which would be used in Python code while initializing.
3.3 Retrieve and Rank Implementation

Retrieve and Rank is one of the most advanced services in the Watson Developer Cloud that forms the heart of this question and answering system. Retrieve and Rank’s implementation is composed of two major components namely Retrieve component and the Rank component. The Retrieve component implementation is based on creating a SOLR cluster, which is an open source search platform with capabilities of indexing and search[4]. The second part is Ranker component, which is nothing but a learning to rank algorithm designed to re-rank the documents retrieved based on their relevance to the input query.
3.3.1 Retrieve Component

Before starting the implementation of the Retrieve component, We have to register for the Retrieve and Rank service and obtain the authentication keys. The Retrieve component has been implemented in Python using the Python SDK for the Watson Developer Cloud. Then the SOLR cluster has to be created by customizing the cluster size and cluster name.

```python
try:
    parsed_json = output
    clusters = parsed_json['clusters']
    for i in range(len(clusters)):
        cluster_json = clusters[i]
        if cluster_json['cluster_name'] == CLUSTER_NAME:
            found = True
            cluster_id = cluster_json['solr_cluster_id']
except:
    print ('Command:')
    print ('retrieve_and_rank.list_solr_clusters()')
    print ('Response:')
    print (output)

if found:
    print 'Cluster "{}" already exists with ID "{}"'.format(CLUSTER_NAME, cluster_id)
    print json.dumps(parsed_json, sort_keys=True, indent=4)
else:
    # Running command that creates a cluster
    output = retrieve_and_rank.create_solr_cluster(cluster_name=CLUSTER_NAME, cluster_size=CLUSTER_SIZE)

    try:
        parsed_json = output
        print json.dumps(parsed_json, sort_keys=True, indent=4)
        credentials[cluster_id] = parsed_json['solr_cluster_id']
        with open(credfile_path, 'w') as credfileUpdated:
            json.dump(credentials, credfileUpdated)
    except:
        print ('Command:')
        print ('retrieve_and_rank.create_solr_cluster()')
        print ('Response:')
        print (output)
```

Figure 3.6: The Python script for creating the SOLR cluster

Once the cluster is created, we have to configure the cluster by creating the configuration file. The configuration file tells the cluster how to interpret the input data so that it can create indexes accordingly.
The cluster creation is followed, up creating a collection into which the answer units can be uploaded. Now, the SOLR indexes the input answer units. This makes input documents added in the form of answer units available for searching.

```python
retrieve_and_rank = RetrieveAndRankV1(
    username=USERNAME,
    password=PASSWORD)

#please make sure your documents are in data/content/solrDocuments.json or modify the path below
SOLR_DOCUMENTS_PATH="/Users/vhineshravi/Desktop/Capstone/Code/json_new_data.json"

#Get Solr Client Handle
pysolr_client = retrieve_and_rank.get_pysolr_client(SOLR_CLUSTER_ID, COLLECTION_NAME)

#Add documents
with open(SOLR_DOCUMENTS_PATH) as data_file:
    data = json.load(data_file)
output = pysolr_client.add(data)

#Running command that index documents
try:
    print (output)
except:
    print ('Command:')
    print ('pysolr_client.add()')
    print ('Response:')
    print (output)
```

Figure 3.7: The Python script for adding the answer units to collection

The retrieve component is ready and would be able to retrieve answers for the input query but the answers retrieved may not be completely relevant to the input query as the answers are yet to be ranked. Figure 3.8 shows the testing of the retrieve component by asking a question ”Who is a privacy officer?”
3.3.2 Ranker Component

The Ranker Component is the one responsible for retrieving the relevant results as output. The first step in training the Ranker is to create a training set, which would act as a ground truth for the Ranker to learn and understand the patterns existing between the input query and the answer units. The training set is created by forming multiple questions addressing different sections within the dataset. The same questions are asked in three different ways, so that it would help the Ranker in understanding the relations existing between question and answer. Figure 3.9 shows the sample of training set that is created.
What happens if I violate the access to protected health information policy?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

Who implements the access to protected health information policy?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What is the scope of accounting of disclosures policy with regard to all agency staff?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

When does the accounting of disclosures policy apply to?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What is the statement of accounting of disclosures policy?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

Who can access the accounting of disclosures?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

How does accounting of disclosures work?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

How to implement accounting of disclosures?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What type of disclosure must be tracked?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What type of protected health information must be tracked?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What information must be provided in each disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What information is required for each disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What happens if accounting of disclosures is violated?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What happens if accounting of disclosures policy is violated?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What is the statement of accounting of disclosures policy for Management?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

How to request accounting of disclosures?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

How to follow up questions on requests regarding accounting of disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What is the response time for requests regarding accounting of disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

How long to wait for response for requests regarding accounting of disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What are the exceptions for accounting of disclosure?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

What are the threats that can be unacknowledged in disclosures?  
Rank: 0.6 | 0.6 | 0.656 | 0.67 | 0.691 | 0.695 | 0.748 | 0.753 | 0.852 | 1

Figure 3.9: Sample of training set

```python
# Checking if ranker with same name already exists
found = False
ranker_id = ""
output = retrieve_and_rank.list_rankers()
try:
    rankers = output['rankers']
    for i in range(len(rankers)):
        ranker_json = rankers[i]
        if ranker_json['name'] == RANKER_NAME:
            found = True
            ranker_id = ranker_json['ranker_id']
except:
    print('Command: ')
    print('list_rankers()')
    print(Reponse())
    print(output)
if found:
    print("Ranker " + RANKER_NAME + " already exists with ID " + ranker_id + ".")
    json.dump(output, output, sort_keys=True, indent=4)
else:
    # Running command that trains a ranker
    with open(TRAINING_DATA, 'rb') as training_data:
        output = retrieve_and_rank.create_ranker(training_data, training_data, name=RANKER_NAME)
    try:
        print(json.dump(output, output, sort_keys=True, indent=4))
        credentials[ranker_id] = output['ranker_id']
        with open(credPath, 'w') as credFileUpdated:
            json.dump(credentials, credFileUpdated)
    except:
        print('Commmand: ')
        print('cmd')
        print(Reponse())
        print(output)
```

Figure 3.10: Python script for training the ranker
The training set consists of a mapping between questions and the relevant answer ids along with relevancy score. The training set consists of about 200 question and along with their answer id mappings.

Once the training set is created, then training data needs to be created from the training set. Which is done by using the train.py script given as a part Retrieve and Rank service.

The Ranker is trained using the training data generated after which the Ranker would be available to take input questions and return the relevant answers. For each question posed to the Ranker, the Ranker would return top ten relevant answers ranked in the order of their relevancy.

The trained ranker can be tested using the script shown in the figure 3.11. In the figure 3.11, a question about medicare fraud is used as test question to test the trained ranker.

```
QUESTION= "What is medicare fraud?"
QUESTION = QUESTION.replace(" ","\&")
curl_cmd = 'curl -u "$s": "$s" "\%s/solr/$s/select?q=ranker_id=$s&qf=\%s&wt=\%s&fl=id,title,body" &&
(USERNAME, PASSWORD, SOLRURL, SOLR_CLUSTER_ID, COLLECTION_NAME, RANKER_ID, QUESTION)
process = subprocess.Popen(shlex.split(curl_cmd), stdout=subprocess.PIPE)
output = process.communicate()[0]
try:
    parsed_json = json.loads(output)
    print(json.dumps(parsed_json, sort_keys=True, indent=4)
except:
    print('Command: ')
    print(curl_cmd)
    print('Response: ')
    print(output)

{
  'response': {
    'docs': [
    {
      'body': "Consistent with the Federal False Claims Act statute (31 USC 3729), medicare fraud occurs when th e following takes place: \"0000\" knowingly presents, or causes to be presented, a false or fraudulent claim for payment or approval; \"0000\" knowingly makes, uses, or causes to be made or used, a false record or statement material to a fa lse or fraudulent claim; conspires to commit a violation as indicated in this definition; \"0000\" has possession, custod y, or control of property or money used, or to be used, by the Government and knowingly delivers, or causes to be deli vered, less than all of that money or property; \"0000\" is authorized to make or deliver a document certifying receipt of property used, or to be used, by the Government and, intending to defraud the Government, makes or delivers the rec eipt without completely knowing that the information on the receipt is true; \"0000\" knowingly buys, or receives as a pledge of an obligation or debt, public property from an officer or employee of the Government, or a member of the Arm ed Forces, who lawfully may not sell or pledge property; \"0000\" knowingly makes, uses, or causes to be made or used,

Figure 3.11: Python script for testing the trained ranker"
3.4 Testing the system

The testing was done by randomly creating 50 questions from the input dataset which forms the testing set of questions. The questions created were highly diverse ranging from questions similar to the training set and the questions completely dissimilar to the training set. The Retrieve component and the Ranker component are both tested by asking the same questions from the testing set. The performance of the Ranker is evaluated by comparing the relevancy of results returned by Ranker to the Retrieve component. Below are some of the questions and their corresponding answers that are retrieved by the Ranker Component and the Retrieve Component. Only the first answer retrieved is shown in both the cases.

**Question: ”Does arc reward employees?”**

**Answer: Rank Component**

"As part of The Arc\'s commitment to recognizing those who are exemplary in compliance with the corporate compliance plan, the following incentives may be used to encourage and reward the behavior or employees, volunteers, consultants, interns and students: Situational recognitions of staff contributions or assistance, including special awards. Handwritten notes of appreciation from supervisors, managers and/or the compliance director. Public recognition in The Arc\'s newsletter or community newspaper. Celebration of successes (i.e., a great audit) Performance reviews and positive feedback. Continuing education opportunities. Opportunities for career advancement. Serving as a verification of good services provided by a service provider. Continued use of a contractor's services."

Figure 3.12: Answer from Rank component
Answer: Retrieve Component

"body": ["The agency's Privacy Officer has general responsibility for implementation of this policy. Members of our medical staff and agency staff who violate this policy will be subject to disciplinary action up to and including termination of employment or contract with The Arc of Monroe County. Anyone who knows or has reason to believe that another person has violated this policy should report the matter promptly to his or her supervisor or the agency's Privacy Officer. All reported matters will be investigated, and, where appropriate, steps will be taken to remedy the situation. Where possible, The Arc of Monroe County will make every effort to handle the reported matter confidentially. Any attempt to retaliate against a person for reporting a violation of this policy will itself be considered a violation of this policy that may result in disciplinary action up to and including termination of employment or contract with The Arc of Monroe County."]

Figure 3.13: Answer from Retrieve component

Question: ”how to document medicaid fraud?”

Answer: Rank Component

"body": ["For clinical staff, services provided shall be documented according to the professional standards of the agency's clinical discipline in addition to agency policy. For all staff, including clinical staff, services provided shall be documented according to regulatory requirements, formal and approved treatment plans, goal methodologies, and/or program-specific policy and procedure. If, when mistakes are identified, management will provide necessary training and/or support to assist the staff person in being successful moving forward. Service documentation must be completed on or about the date when the service was provided, ideally by the end of the shift on the date when the service was provided. Management shall provide necessary counseling and discipline regarding this standard. Management believes that knowing or deliberate falsification of documentation may have occurred, management will contact the Quality Improvement Department. An investigation will occur consistent with the policy. If it is determined that official agency-related documentation was falsified knowingly or deliberately: For employees, the matter will be referred to the Human Resource department for final review, and the person's employment will be terminated. For contractors, students, consultants, and interns, the CDO or designee will be notified and will take formal actions to sever their relationship with the person. If it is determined that a program has engaged in abuse or misuse of Medicaid as defined by this policy, the Quality Improvement Department will work with the program director and/or senior administrator to ensure that formal steps are taken to correct the immediate situation and prevent recurrence moving forward."]

Figure 3.14: Answer from Rank component
Answer: Retrieve Component

Figure 3.15: Answer from Retrieve component

Question: ”What amounts to medicaid fraud?”

Answer: Rank Component

Figure 3.16: Answer from Rank component
Answer: Retrieve Component

Figure 3.17: Answer from Retrieve component
Chapter 4

Analysis

On comparing the returned answers from both Retrieve component and trained Ranker component, the answers obtained through trained Ranker component is more relevant to the questions asked. This confirms the quality of the answers returned is directly proportional to the quality of the training that is being given to the Ranker component. There were cases during testing, where the trained Ranker returned no response to a certain type of questions. On further analysis, it is found that the questions of the type "When?" is not clearly understood by the Ranker, Where the Ranker starts looking for answers with actual time represented in number rather than the situations. This can be improved by training the Ranker with more questions involving "When".
Chapter 5

Conclusions

5.1 Current Status

The current system is successfully implemented using the Retrieve and Rank service[2] for the HIPAA and Corporate Regulation domain. The system is trained with 200 training questions and its performance is evaluated using 50 test questions. Overall the performance of the system is satisfactory which is concluded from the analysis of the relevancy of answers returned from the trained Ranker and the Retrieve component. Though the accuracy of the Ranker is good, there are instances where the Ranker fails to return an answer for questions involving "When?", which is because the Ranker searches for actual time in number rather than the situations which are time by itself. The most relevant answer in some cases may not be the first one in the returned list.

5.2 Future Work

The application could be extended to other domains, enabling to create a question and answering system that is capable of handling questions on multiple domains. Currently, the system doesn’t support any user feedback on answers retrieved, in future, we could include a feedback loop which could be used to increase the relevancy of the answers retrieved dynamically, as the system would continuously learn from the user feedback. We could provide a user interface and host the application to public and let them use the system.
Bibliography


