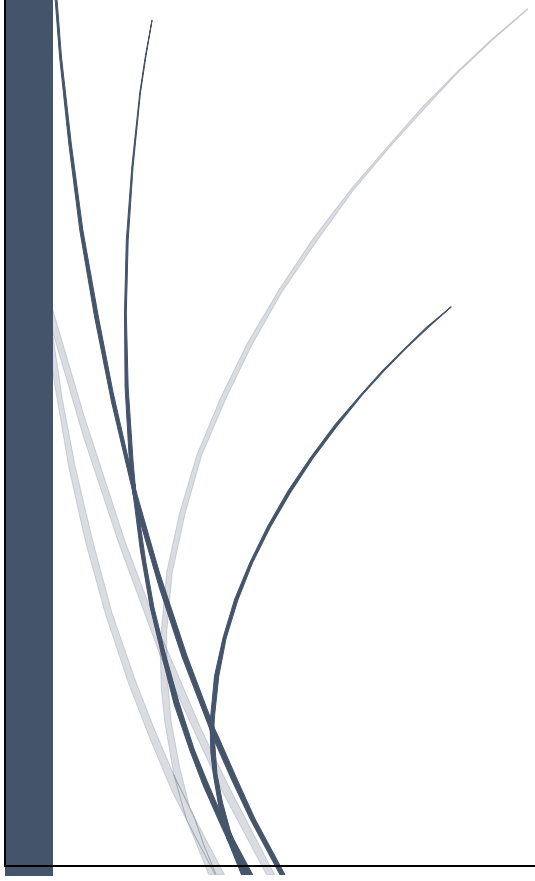
Brainizer
Intelligent System



Faculty of Computers and Information Sciences – Ain Shams University

Brainizer Intelligent System

June, 2015

No	Name	Department
1	Mohamed Ali Mohamed	Computer Science
2	Mahmoud Khaled Mahmoud	Computer Science
3	Ahmed Hani Ibrahim	Computer Science
4	Mohamed Gamal Soliman	Computer Science
5	Mahmoud Mohamed Afify	Computer Science

Contents

Acknowledgements

We wish to express our thanks to the Head of our Computer Science department **Prof. El-Sayed Mohamed El-Sayed El-Horbaty** for accepting our graduation project proposal, also we thank **Dr. Salma Hamdy** for helping us to make a good, clear and well-formatted proposal that were one of the main factors to accept our project.

As we are working on such a project that involves many researches, we needed some help and advices from experts who got involved and worked on the research field in general and on Machine Learning and Natural Language Processing specifically. **Dr. Mohamed Elhoseiny**, who was graduated from our college at 2006, helped us with his professional advices that helped when we stuck in a problem, **RSDE/ Ahmed Emad Moursy** who helped with his practical experience in the applied research field and provided us with many papers that used during implementation phases. And encourage us in every single step of our project.

We also very grateful to everyone who supported us.

Brainizer Intelligent System Team.

Abstract

We introduce a Hybrid Question Answering system based on large text knowledge-base and Information Retrieval engine. Knowledge-base is built according to some previous data obtained from answered questions. Our system allows the user to ask a question on a given paragraph or text and the system navigates him/her to the desired answer using the Information Retrieval engine.

Another feature of the system is that it retrieves relevant texts precisely using question types and synonymous expression dictionary. The system is about turning the machine to an "**Expert**", the expert who can make some analysis to a given text to extract some information; the machine makes use of this information to build its Knowledge Base.

The main idea of the system is to divide it to 3 main modules. The first module is about classify the user's question using Machine Learning and Swarm Intelligence algorithms to find the question class among 50 classes. The second module is the Information Retrieval engine that gets some passages from the paragraph that may have the answer of the question. The third and final module is about extracting the candidate answers from the passages using sentence's dependency tree and Pattern Learning / Matching and word vector space.

After we get the answer for the given question, we add the answer for the given question to the knowledge-base. Increasing the amount of the answered questions leads to turn the system to an Expert that could use such information and data to get the answer for other future questions without needing a resource that turns the system from a closed domain to open domain.

The system could be modified by inserting a module that is about system-user conversation. Conversation may be used to observe more information about the question, the question may have some ambiguous words that would effect on the question classification and answer extraction accuracy.

Chapter 1

Project Initiation

In this chapter, we introduce a general overview of the project. Before we begin implementing an application or turning an idea to a system, we should define and determine many things related to the idea itself.

In Computer Science, an implementation word means a realization of pre-specified technical functionalities and non-functionalities requirements of an algorithm or system and software modules. An Example for that is Web Browsers. We use web browsers to be able to access the World Wide Web (WWW), we can illustrate the idea of the web browsers that we want to make an application that helps us to access the World Wide Web, so, before we begin the implementation we want to get general idea of how to access the World Wide Web with some functional and non-functional requirements. After we reach the idea, we begin the implementation which will be an easy task if the idea is fully illustrated and handling many issues that may appear when implementing it.

In this chapter we illustrate the idea of our project and mention the main factors that we took into consideration before we began the implementation.

We can conclude this chapter in 6 main points or factors, *Problem Overview*, in which we discuss an overview of the problem and its definition with mentioning a case study, *Current / Existing systems*, in which we show the existing systems that do the same idea of our project with mentioning its functionalities, *Purposes / Goals of the project*, why did we choose this project?, what is new in our project that differs it from the other existing systems?, what is our target?, all of these questions will be answered in this section.

The remaining 3 points are *Stakeholder Analysis*, in which we discuss who can use our project and get benefit from it as the users are the main motivation for doing any application, *Risk Analysis*, in which we talk about the source data we

use or the user use in the system and how effect of that in the security level, the sixth and final point is about the *System Constraints* and what the user can do and can't do with the current version of the system.

1.1 Problem Overview

When human deals with paragraph of text, they don't only read it as a collection of words, they extract information from the text, for example, "**The weather in Cairo is rainy**", there are obvious information we can extract from this sentence:-

Place	Cairo
Weather	Rainy

We can make some questions based on this data, like "**what is the weather in Cairo**?" Based on the data we extracted before, the answer will be "**rainy**", but how about questions like "**where is Cairo**?", if we just deal with the shown information, you won't find the exact answer for this question.

One of the main differences between us and machines, is that when we deal with data, we can make some analysis on it. Back to our previous example, when you see the word "**Cairo**", your mind will do some analysis phases for it.

Your mind will check its knowledge-base (memory) to see if it contains more information about "**Cairo**" or not, if the information needed is found, it is retrieved with some data like "**Cairo is in Egypt!**" or other information which is somehow related to Cairo. This analysis is known as *Relation Extraction* which is an important topic in Natural Language Processing field. If there is no data about Cairo, you won't find an answer or details about it without an outer source. There are many outer sources the human could use to gather some information and increase his/her knowledge such as books, internet and TVs.

When humans get their data or knowledge source, the brain obtains information from it and saves it in its memory to be used later. Obtaining information from source, which could be some text, is known as Information Retrieval process which is also an important topic in Natural Language Processing.

All of these phases are handled by the human's brain by default, thank to Allah who has given us this complex system.

The machines can't do the same of these analysis phase because they don't have such a complex system to help them on such complex tasks. Nowadays, the researchers are trying to increase the complexity of the machines to be able to do these tasks, so, they worked on Relation Extraction and Information Retrieval and improve their accuracy by creating some applications and implementing these techniques.

Question answering system is one of these applications that deals with such a Natural Language Processing techniques. There are many papers and researches done and still doing about this application because till now they haven't reached to an optimal accuracy when getting an answers for an asked question. There are many existing Question Answering systems such as IBM Watson, Apple Siri and Amazon Evi.

Our project is a Question Answering, the user has the ability to add a resource (text, paragraph), on which he/she asks question on, and the system retrieves the answer from the paragraph.

1.2 Evaluation of current and existing systems

As we mention on the previous section, there are many existing applications and systems that concentrate on Question Answering such as IBM Watson, Siri and Evi. We will talk about IBM Watson and Siri.

Watson represents a first step into cognitive systems, a new era of computing. In addition to using programmatic computing, Watson has three

capabilities that make it truly unique: Natural Language Processing, Hypothesis generation and evaluation and Dynamic Learning. Watson brings these powerful capabilities together in a way that's never been done before, resulting in a fundamental change in the way businesses look at quickly solving problems. Watson is a natural language question answering system that does not use prepared answers, but determines its answers and associated confidence scores that are based on knowledge it has acquired.

Cognitive systems consists of:

1- Navigate the complexities of human language and understanding

2- Ingest and process vast amounts of structured and unstructured (big) data

3- Generate and evaluate countless possibilities

4- Weigh and evaluate responses that are based only on relevant evidence

5- Provide situation-specific advice, insights, and guidance

6- Improve knowledge and learn with each iteration and interaction

7- Enable decision making at the point of impact

8- Scale in proportion to the task

These systems apply human-like characteristics to conveying and manipulating ideas. When combined with the inherent strengths of digital computing, they can solve problems with higher accuracy, more resilience, and on a more massive scale. Watson is an example of a Cognitive System. It can tease

apart human language and identify inferences between text passages with human-like high accuracy at speeds far faster and scale far larger than any human. A rules-based approach would require a near-infinite number of rules to capture every case we might encounter in language.

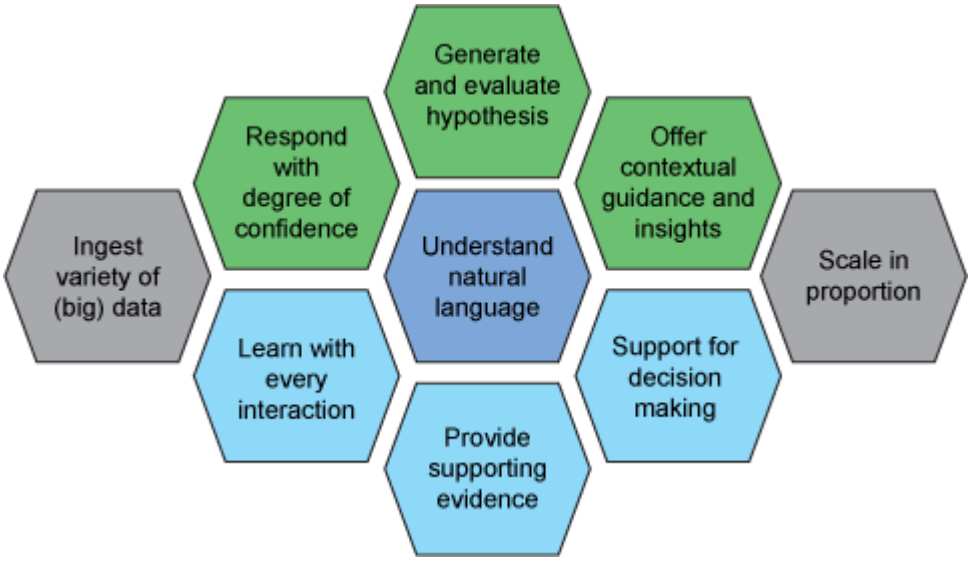


Figure 1.2.1

Another example is Apple's Siri application which mainly uses voice recognition and statistical approaches. The sounds of your speech were immediately encoded into a compact digital form that preserves its information.

The signal from your connected phone was relayed wirelessly through a nearby cell tower and through a series of landlines back to your Internet Service Provider where it then communicated with a server in the cloud, loaded with a series of models honed to comprehend language.

Simultaneously, your speech was evaluated locally, on your device.

A recognizer installed on your phone communicates with that server in the cloud to gauge whether the command can be best handled locally -- such as if you had asked it to play a song on your phone -- or if it must connect to the network for further assistance. (If the local recognizer deems its model sufficient to process your speech, it tells the server in the cloud that it is no longer needed: "**Thanks very much, we're OK here.**")

The server compares your speech against a statistical model to estimate, based on the sounds you spoke and the order in which you spoke them, what letters might constitute it. (At the same time, the local recognizer compares your speech to an abridged version of that statistical model.) For both, the highest-probability estimates get the go-ahead.

Based on these opinions, your speech -- now understood as a series of vowels and consonants -- is then run through a language model, which estimates the words that your speech is comprised of. Given a sufficient level of confidence, the computer then creates a candidate list of interpretations for what the sequence of words in your speech might mean.

If there is enough confidence in this result, and there is -- the computer determines that your intent is to send an SMS, Erica Olssen is your addressee (and therefore her contact information should be pulled from your phone's contact list) and the rest is your actual note to her -- your text message magically appears on screen, no hands necessary. If your speech is too ambiguous at any point during the process, the computers will defer to you, the user: did you mean Erica Olssen, or Erica Schmidt?

1.3 Purposes / Goals of the Project

There are always goals of the chosen project that determined before beginning the implementation phase. Our goal can be divided into two main goals: The target output of the system and our target from implementing such a project.

The target output of the system is to build and efficient core for an open-domain Question Answering system. Open-domain means that the user can ask anything about any subject, and the system retrieves the accurate answer for it. Also, make the system as an API that could be used in any other systems that want to build a Question Answering system too. To reach these targets, we tried to make an efficient, readable and well-designed code to make configuring it with systems an easy task.

Using Machine Learning and Natural Language Processing algorithms, we introduced a core for Question Answering system. We tried to get the best accuracy for question classification and answers extraction by using different approaches and ideas to get different results so that we determine the best methods for each task and module.

The other goal for choosing this project is that we (as students) want to enter the Applied Research field. We are living in the era of Artificial Intelligence, at which there are multiple papers published daily about new methods or approaches to introduce new solutions for the current problems the human faces to make their life easier. Applied Research field concerns on how to turn the theories about a problem solution into a realization applications and systems. So, we thought that it is a good thing to know more about this field, know how to perform a scientific research and choose the best solution for the problems we face during the implementation phases.

There is also a business advantage of this system. This system could be a seed for a startup that concerns on Machine Learning and Natural Language

Processing applications. We know that as programmers, our target is to produce products that users use in their daily life. This system could be a source for increasing the knowledge of the human by letting the user asks what he/she want to know, and the system gives back the answer for it.

Determining the goals of the system helps the project team to focus their attention on those requirements that make the greatest contributions to meet the specified goals. Things such as efficiency, readability, optimality and functionality make the team members choose their implementation methods very wisely.

1.4 Stakeholder Analysis

The project mainly concerns on the users need, if the system is available for them, they would use it for asking what they want to know then the system gives back the answer. So, the users can use the system for educational or entertainment purposes. Imagine that you have an application that can answer all of your questions, it would be fantastic thing!

As we mention that we concern on the user's need, we tried to make the system efficient and easy as possible to use by creating a friendly user-interface. Also, we are planning to add a voice recognition to the system to make the user uses his/her voice to ask a question without needing to type on the keyboard.

1.5 Risk Analysis

Our system uses TREC dataset for question classification training phase, TREC dataset is available for everyone who want to make any processing on it. So, there is now risk when we use this data as it is legal to use and anyone can use it as an open source data.

Also, we will add a login feature to make the user secure his/her data and knowledge-base.

1.6 Project Constraints

As it is a research project that we still trying to improve and add new features, it is expected that there are many constraints on the project. We tried to minimize the constraints on the input and output to make using the application easier for the user.

The current version of the system needs the user to enter a resource in which the answer is mentioned into it, in other words, the user should provide the text or the passage, and the answer for his/her question mustn't be outside the given resource.

The resource given to the application should be a text file, we still haven't handled HTML and XML files.

There is possibility that the user asks about something that is ambiguous to the system to identify, words such as "Titanic" that is not in the dictionary make the classification process more difficult. We planned to handle such case by adding a Conversation module, this module enable the system to ask the user about an ambiguous word to get more information about it. There is also another idea which make the system uses the internet to search for the ambiguous word and get the needed information about it.

The user must enter the question by typing on the keyboard, the voice recognition engine hasn't been added to the system.

Their system works only on PCs and Laptops, both of them should have Java environment to be able to run the system.

Chapter 2

Literature Review

2.1 Project Field Researches

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.

- *Closed-domain* question answering deals with questions under a specific domain (for example, 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. QA systems in the context of machine reading applications have also been constructed in the medical domain, for instance related to Alzheimer's disease

- *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.

The most researches in QA systems nowadays are focusing in many different approaches such as:

2.1.1 Linguistic approach

A question answering system requires understanding of natural language text, linguistics and common knowledge. Therefore, many of the previous researchers relied on artificial intelligence (AI) based methods that integrate natural language processing (NLP) techniques and knowledge base or corpus to build QA logics. The knowledge information is organized in the form of production rules, logics, frames, templates (represented with triple relations), ontologies and semantic networks, which are utilized during analysis of question-answer pair. Linguistic techniques such as tokenization, POS tagging and parsing were implemented to user's question for formulating it into a precise query that merely extracts the respective response from the structured database. However, deployment of a specific domain knowledge base poses portability limitation as a different application domain requires different grammar and mapping rules. Additionally, building an appropriate knowledge base is a time-consuming process, so these systems are generally applied to problems that have long-term information needs for a particular domain. Earlier QA systems around 1960s such as BASEBALL [1] and LUNAR [3] were merely natural language frontends for structured database query systems. The questions presented to these systems were usually analysed using NLP techniques to produce a canonical form, which was then used to construct a standard database query. Dialogue system viz., ELIZA [2] and GUS [4] also used structured database as the knowledge source. The key limitation of these systems is that the knowledge stored in the structured database was only capable of answering questions asked within the restricted domain. However, in recent works, this limitation of the knowledge base is accepted as the capability to provide a situation-specific answer. Clark et al. [7]

presented an approach for augmenting online text (dynamic manual) with knowledge-base question answering ability. This combined approach allows users to access not only response of routine questions but also of those questions that were unforeseen at the time of system construction. This specific feature of QA system is achieved through inference engine component. Some of the existing QA systems such as START [5], QA system by Chung et al. [18] and Mishra et al. [30] have acquired web as their knowledge resource. These systems apply their own heuristics to store information from Sanjay K. Dwivedi and Vaishali Singh / Procedia Technology 10 (2013) 417 – 424 419 web documents in the local knowledge database that has to be later on accessed and rely on linguistic techniques for answer generation. In addition with NLP techniques, some of the knowledge base QA systems rely on the rule based mechanism. After applying general purpose NLP techniques, rules are further built to identify question classification features. Quarc [8] developed by Rilloff et al., and Cqarc [23] developed by Hao et al., used heuristic rules that look for lexical and semantic clues in question to identify the question class. However, question class taxonomy may vary from one system to another. Some systems exploit general taxonomy for semantic classes like who, when, what, where and why type questions while some others utilize domain specific taxonomy. Table 1.1 shows the difference between NLP based and rule based QA systems.

2.1.2 Statistical approach

In the current research scenario, rapid growth in available online text repositories and web data has increased the importance of statistical approaches. These approaches put forward such techniques, which cannot only deal with the

very large amount of data but their heterogeneity as well. Additionally, statistical approaches are also independent of structured query languages and can formulate queries in natural language form. These approaches basically require an adequate amount of data for precise statistical learning but once properly learned, produce better results than other competing approaches. Furthermore, the learned statistical program or method can be easily customized to a new domain being independent of any language form. However, one of the major drawbacks of statistical approaches is that they treat each term independently and fail to identify linguistic features for combination of words or phrases. In general, statistical techniques have been so far successfully applied to the different stages of a QA system. Support vector machine (SVM) classifiers, Bayesian classifiers, Maximum entropy models are some techniques that have been used for question classification purpose. These statistical measures analyze questions for making prediction about users' expected answer type. These models are trained on a corpus of questions or documents that has been annotated with the particular mentioned categories in the system. One of the pioneer works based on the statistical model was IBM's statistical QA [9] system. This system utilized maximum entropy model for question/ answer classification based on various N-gram or bag of words features. Moschitti [17] had used Rocchio and SVM text classifiers for question and answer categorization and tested his approach on Reuters-21578. A Chinese question answering system developed by Zhang et al. [32] has also used SVM classifier based on the features of words, part of speech (POS), named entity and semantics. Quarteroni et al. [28] proposed an interactive QA system which implements SVM classifiers for question classification. Berger et al. [10] has investigated the prospects of applying

statistical methods to answer finding task in QA and discovered that these techniques performed quite well depending on the characteristics of the underlying data set– vocabulary size, the overlap between question and answers, and between multiple answers, etc. Statistical techniques such as N-gram mining, sentence similarity models and Okapi similarity measurement are applied to answer finding tasks in a QA system. These techniques analyze question and document based on various similarity features in order to determine the closeness of candidate documents or answers with respect to question. The notion of answer validation could also be implemented incorporating statistical approaches via relevance feedback mechanism. Information retrieval for this (IBM's QA) system used a two pass approach based on Okapi formula and expansion of queries based on TREC-9 QA corpus. Answer selection phase of this system relied on various heuristic distance metrics to search for an answer. Moschitti [17] implemented similarity measurement model for calculating the similarity score between query and documents or sentences from corresponding collections. The similarity model presented by Cai et al. [20], relied on a sentence similarity model to calculate the similarity between question and answer. This model accounted on different features such as keyword similarity, length similarity, order similarity and distance similarity of the keywords used in question and answer. A system developed by Soricut et al. [22] used a statistical chunker that implements a dynamic programming algorithm to chunk the natural language questions into chunks/phrases asked to the search engine and N-gram co-occurrence statistics for answer extraction. This system could answer complex questions and non-factoid questions too.

2.1.3 Pattern matching approach

This approach uses the expressive power of text patterns to replace the sophisticated processing involved in other competing approaches. For example, the question "**Where was Cricket World Cup 2012 held?**" follows the pattern "**Where was held**?" and its answer pattern will be alike "**was held at"**. Currently, many of the QA systems automatically learn such text patterns from text passages rather than employing complicated linguistic knowledge or tools viz., parser, named-entity recognizer, ontology, WordNet, etc. to text for retrieving answers. Simplicity of such systems makes it quite favourable for small and medium-size websites, which cannot afford complex solutions that require much time and rare human skills to install and maintain the system. Most of the patterns matching QA systems use the surface text patterns while some of them also rely on templates for response generation.

2.1.3.1 Surface Pattern

Based this approach extracts answers from the surface structure of the retrieved documents by relying on an extensive list of patterns. Answer to a question is identified on the basis of similarity between their reflecting patterns having certain semantics. These patterns are like regular expressions. Though designing such set of patterns requires a lot of human skill and time but the approach has shown high precision too. Initially, the surface pattern based method aim at finding answers to factual questions, as their answer is limited to one or two sentences. In order to design an optimal set of pattern, most of the recent surface pattern based system used method described by Hovy et al. [13]. They implemented an automatic learning method which used bootstrapping to build a large set of patterns starting only with a few examples of QA pair from the

web. Motivation behind their work is surprising strength of such patterns proposed by Soubbtin and Soubbtin [11] in the TREC- 10 Question answering evaluation track. Another related concept based on surface patterns is proposed by Zhang et al. [14] who augmented surface patterns with 'support' and 'confidence' measures from data mining community. This system showed very high precision but low recall. Greenwood et al. [16] integrated surface patterns with named entity tagger to generalize these patterns induced from free text. System developed by Cui et al. [25] used soft pattern matching based on bigram model and Profile Hidden Markov Model (PHMM) instead of regular expression-based hard matching patterns to identify answer sentences. Some other QA systems have also followed this approach to improve their question answering mechanism. Saxena et al. [24] used pattern matching as an alternative approach for difficult questions like acronym expansion questions, date of birth questions and location questions.

2.1.3.2. Template base

A template based approach makes use of preformatted patterns for questions. The focus of this approach is more on illustration rather than interpretation of questions and answers. The set for templates is built in order to contain the optimum number of templates ensuring that it adequately cover the space of problem, and each of its members represents a wide range of questions of their own type. Templates have entity slots, which are missing elements bound to the concept of the question that has to be filled to generate the query template to retrieve the corresponding response from the database. The response returned by query would be raw data, which is returned to the user.

System developed by Sneiders [15] also utilizes answer templates to pose the answer in a formatted manner. The basic principle followed by template based question answering system is much similar to the automated FAQ (Frequently Asked Questions) answering system that responds with pre-stored answers to user question but unlike static FAQs, the question templates are filled dynamically with parameters. One of such systems by Gunawerdena et al. [31] has been built to meet the needs of the close domain system to understand SMS language from a mobile phone in addition to natural language questions in English. This system used pre-processed text to identify best matched template-answer pair stored in database. Each of such templates is defined to match many different variants of the same question but not the question of same type making it too constrained. While for Question Assistant [15], Sneiders designed templates so that single template could cover a wide range of data instances relevant to the entity slot and almost all questions of its type. Entity slots within a question template represent concepts or entities contained in database and relationship between these concepts is represented by templates themselves. However, if fresh relationship has to be added, a new template is required. Another QA system making assistance of template based approach is proposed by Unger et al. [34]. He used this technique over RDF (Resource Description Framework) data utilizing SPARQL [38] template. Thus, given technique is quite adaptable to semantic web. A SPARQL template directly reflects the internal structure of the questions and maps natural language question to domain vocabulary. This system, however, also applies deep linguistic analysis to generate the SPARQL template as this template not only focuses on syntactical pattern but also on semantic understandings.

2.2 Context of our research

In our project we mainly focused on the three modules, We started each one of them from scratch in order to build an efficient and accurate core for a question and answering system. Our research context was all about finding an effective solutions for the three problems with the help of the latest researches in this field. Our research vision is to direct this core in the future into a startup that try to solve a real problem in our society. It was long term research not short term, the objective of the work of the first year is to reach 50% accuracy then we will try to improve that as far as possible.

2.3 Conclusion

A question answering system is a system that seeks answers to natural language questions from knowledge base text. Textual question answering systems are an important research problem because as the amount of natural language text in digital format grows all the time, the need for novel methods for pinpointing important knowledge from the vast text databases becomes more and more urgent. In addition to this, textual question answering systems form a well-defined framework with lots of existing
Evaluation data in which new methods can be developed and evaluated in a principled way. The separate sub problem of developing answer extraction methods for extracting answers from unstructured text is an interesting problem not only by itself but also because it is quite similar to many other problems, such as the problems of information extraction from text and of semantic annotation of text. Thus, the novel answer extraction methods may be generalized for these problems. As a matter of fact, the methods developed in this thesis may be expanded to any problem where the training
Data can be preprocessed into the required format. This format is quite general as it only consists of an input question and of its answer. Examples of problems that can be processed into this format are information extraction from text and

semantic annotation of text. In the case of information extraction from text, the template to be lled can be expressed as a set of natural language questions. The same applies to semantic annotation. For example, in order to annotate semantically all names of lms appearing in a text, the system is given as input both the question Name a lm. And a list of lms appearing in the text document collection that is to be used as the training data. The main contributions of this thesis are the development and evaluation of both a new type of answer extraction pattern and of two different methods for their automatic generation. The pattern matching based approach is chosen because of its language and application independence and because pattern matching based techniques have become one of the most successful methods in textual question answering over the last years. The answer extraction methods are developed in the framework of our own question answering system. Publicly available datasets in the English language are used as training and evaluation data for the methods. The methods developed are based on the well-known methods of sequence alignment and hierarchical clustering. The similarity metric used is based on edit distance. The new answer extraction patterns developed consist of the most important words in the question, part-of-speech tags, plain words, punctuation marks and capitalization patterns. The two new methods for creating answer extraction patterns are named the concatenation based method and the alignment based method. The performance of the answer extraction patterns and of the methods for generating them is measured indirectly through the performance of the patterns in the answer extraction task of a question answering system. The difference in performance between the concatenation based and the alignment based answer extraction pattern generation methods is not important when evaluated using the evaluation data. However, when evaluated using the training data and when taking into account only the rst answer candidate, the alignment based method performs better than the concatenation based one.

Chapter 3

System Analysis and Requirements

3.1 Problem Analysis

When we are talking about question answering system, there are 3 main words come to our mind: Question, Knowledge and Answer. That was the base of the analysis.

1- Question word:

We need to know what the question is asking about. For example "***where is Cairo?***" and "***what is the distance between earth and moon?***" for us we know that those questions are asking about LOCATION of CAIRO and DISTANCE between EARTH and MOON. To extract such information we need a module that extract the ***question category*** "**Class, i.e. location, numeric, name of entity ...**" and the ***keywords*** in that question. From here Question Processing Module came.

2- Knowledge word:

After knowing such information, we need to see what knowledge we have related to such category and keywords. In this system, the knowledge was represented in 2 forms: ***knowledge-base*** and ***paragraphs***. Knowledge-base is already prepared to retrieve the question with related knowledge. Regarding paragraphs, we need Information Retrieval to extract the knowledge within them, and that knowledge form is *group of passages*.

3- Answer word:

The extracted passages used to make candidates for answers, and based on these candidates we choose the best answer.

3.2 System Requirements

Working on such a huge system, it is needed to determine the requirements which helps on achieving the system goals. We can divide the system to two requirements, functional and non-functional requirements.

We can conclude the non-function requirements into 2 points, efficiency and user-friendly. When performing an application that deals with the user, the most important thing is to make the application easy to use, without making the user feels any complexity in the application. So, we provide a user-friendly application that has a simple GUI that shows the user what to do.

Efficiency is one of main factors for an application success, in our system we tried to make an efficient code that take the performance into consideration whenever it does any process or operation.

Efficiency not only in choosing the appropriate algorithms or techniques, but also choosing and making the system whole design. A well-designed code leads to improve the system performance.

3.3 System Full Architecture UML

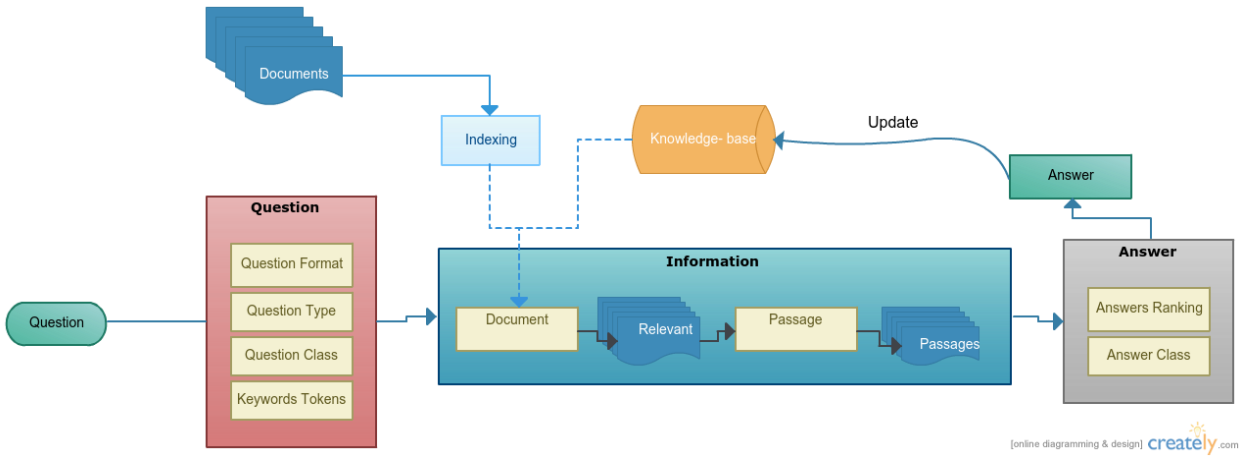


Figure 3.3.1

There are 3 main components "**modules**" in the system, *Question Processing*, *Paragraph Indexing* and *Answer Processing*. The system has 2 inputs, *user question* and *documents* that contain the answer of the question.

Question processing module is responsible to take the question and extract its class using classification algorithm (i.e. SVM, maxent, neural network) and keywords using question structure (i.e. parse tree, dependency tree), and send it to Paragraph Indexing module "**IR**".

Paragraph Indexing "**IR**" mission is to collect a group of passages that have Information related to question class and keywords, if there is not answer already saved in the knowledge base.

After extracting or collect related passages, Answer processing module start to rank answers based on those passages using pattern learning and dependency tree, and save the best answer in the knowledge-base.

Chapter 4

Modules Analysis and Design

4.1 System main modules

The system consists of 3 main modules, Question Processing, Information Retrieval Engine and Answer Extraction. The modules could be separated from each other because each one of the has its own task, but each module waits the output from the previous module, in other words, Question Processing module output is the input to the Information Retrieval Engine, and the output of the Information Retrieval Engine is the input to the Answer Extraction module.

This separation make working on each module a lot easier as the team members divided themselves to each module and each team works and searches for his assigned module or task.

We made an overview and an OOP design for each module, we used Design Patterns to make the design for each module, and this helps us to make a clear and non-complicated code. We will talk about each module on the next pages.

4.2 Modules Overview

4.2.1 Question Classification Module

This module focuses on getting the question class, it takes the input from the application GUI which is the question, and then it tries to find the question class. The class of question shows what the question is about. Our system works on 50 classes of question which is obtained from TREC dataset.

The module also extracts the question keywords, we made a data structure for each question that contains all words info because the question class and these keywords are the input for the next module (Information Retrieval).

The following figure illustrates the modules processes which will be discussed in details on the next chapter.

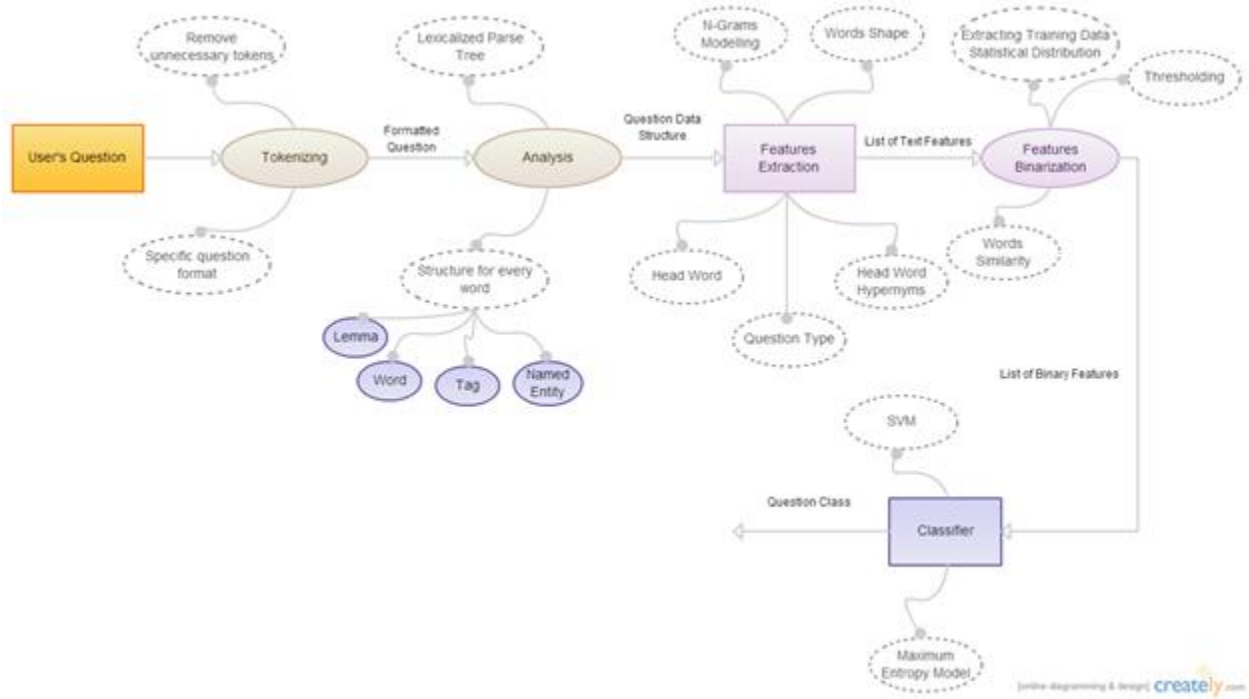


Figure 4.2.1

Also, we make an OOP design for this module, we mainly used 2 design patterns, Bridge Pattern and Singleton Pattern

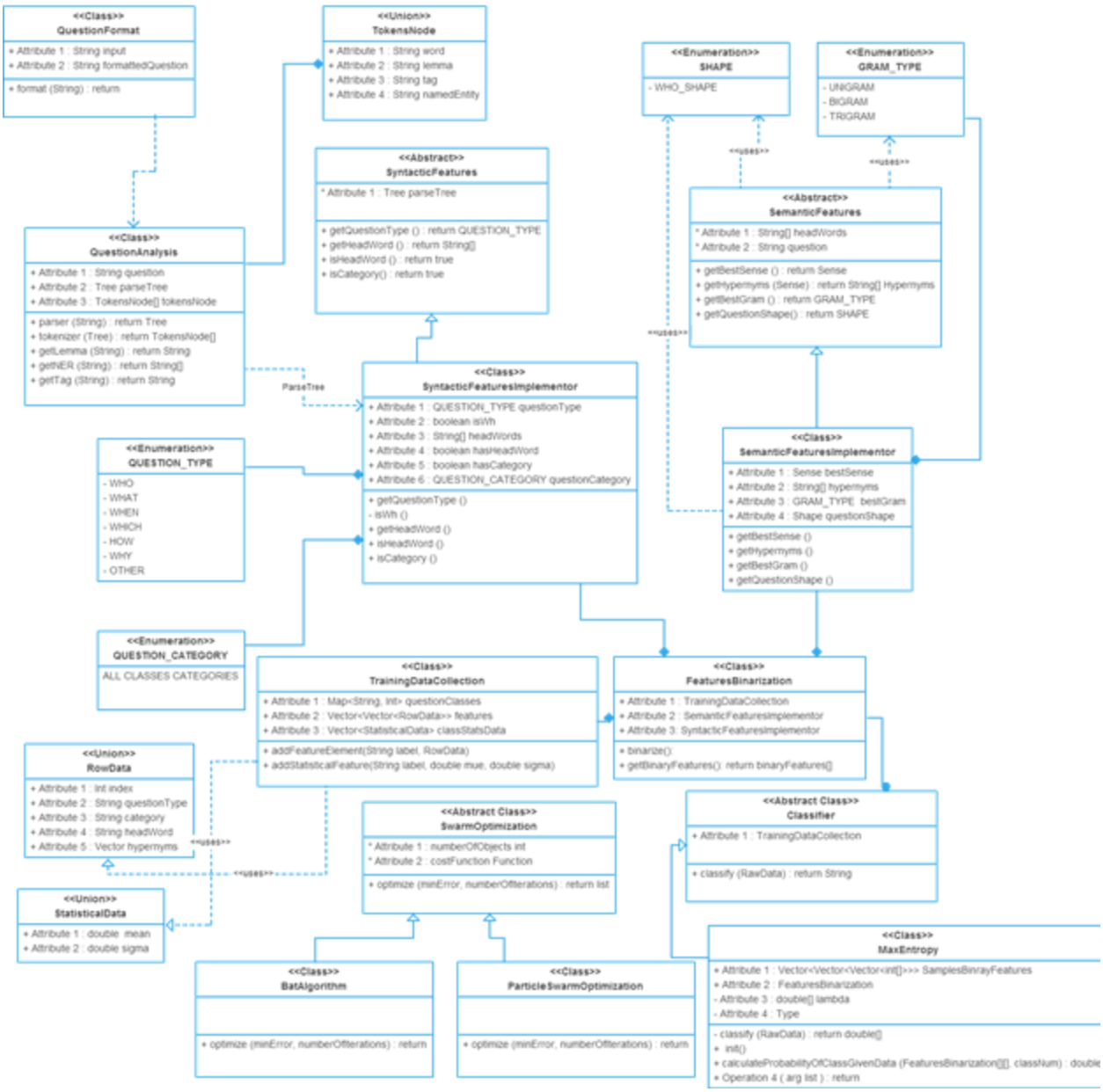


Figure 4.2.2

4.2.2 Information Retrieval Module (IR Engine)

The module is concerns on getting the relevant passages to the question keywords from the given paragraph. We can conclude this module by 2 main processes, the first one which is Knowledge-base operations such as Searching, Updating and Inserting, the second one is Paragraph Indexing which contains indexing and ranking operations.

The following figure shows the OOP design of this module.

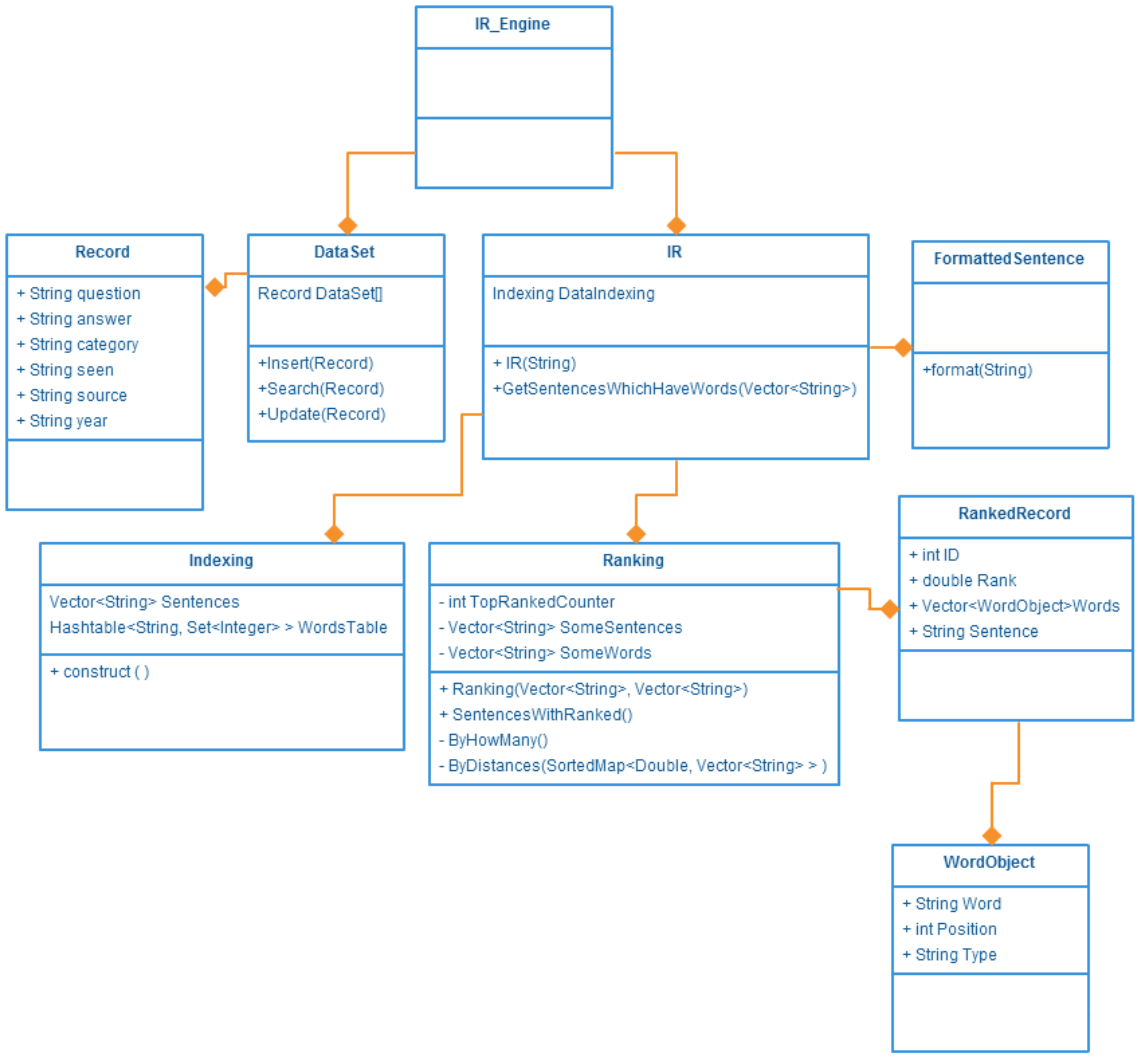


Figure 4.2.3

4.2.3 Answer Extraction Module

In this module we are trying to extract the most accurate candidates' answers and choose the top answer. The previous two modules providing us with very critical parameters for our processing which are:

- **Question Classification Module**: Question Type, Question Focus, Answer Type.
- **Information Retrieval**: Top 5 passages from the document which is 100% sure

that the answer exist at one of them and we trying to find it.

There are several approach for solving this problem and we mainly focused on two approach to find the most accurate answers which are:

1) **Extracting Answers based on Named Entity Tagger**.
2) **Extracting Answers based on Word vector space**.

Chapter 5

Implementation

5.1 Implementation Approach

In this chapter, we talk about the implementation phases for each module, the talk contains deep technicalities that has been used and tried on our system.

We also mentioned a lot of references that we have used during the implementation phases, so, you can check them if you want to find more details about something not clear in the illustration.

In Question Classification, you will see a detailed explanation about how to implement a classifier from scratch to classify a given question, also we will mention the features we used for classification. In Information Retrieval Engine, you will see how the system deals with knowledge-base and the extracted information from the passage. Finally, in Answer Extraction module you will see the different methods that could be implemented based on our experience for answers extractions.

5.1.1 Question Classification

An important module for any Question Answering system. To obtain an accurate classification for the question the key for that is to get the suitable features and use them into the classifier.

So, we illustrate the main features we used in this module. The features are presented as a bag of features that is feeded to the classifier. The most important feature that could be extracted from the question is the head word feature.

The head word is considered to be the most important word in the sentence, which indicates what the sentence want exactly, an example for the:

"**What is a group of turkeys called?**", the head word for this sentence is "**turkeys**", by obtaining this word, we can make a strong assumption that the question is looking for a HUMAN: group , another example: "**George Bush purchased a small interest in which baseball team**", the head word here is baseball which indicates that the question is about HUMAN: group.

To obtain the head word feature, we make a parse tree that turns the question to a dependency tree, then using the dependencies and relations between each word, we observe the head word. There accurate parsers are available such as Chaniak parser (Charniak and Johnson, 2005), Stanford parser (Klein and Manning, 2003) and Berkeley parser (Petrov and Klein, 2007).

Here is a pseudo-code for the head word extraction task

Algorithm 1 Question head word extraction

Require: Question q
Ensure: Question head word
1: **if** q.type == *when*|*where*|*why* **then**
2: return null
3: **end if**
4: **if** q.type == *how* **then**
5: return the word following word "how"
6: **end if**
7: **if** q.type == *what* **then**
8: **for** any aforementioned regular expression r (except HUM:desc pattern) **do**
9: if(q matches r)
10: return r.placehold-word
11: **end for**
12: **end if**
13: **if** q.type == *who* && q matches HUM:desc pattern **then**
14: return "HUM:desc"
15: **end if**
16: String $candidate$ = head word extracted from question parse tree
17: **if** $candidate$.tag starts with NN **then**
18: return $candidate$
19: **end if**
20: return the first word whose tag starts with NN

Figure 5.1.1

The second feature is the wh-word. As we know, we have only 7 wh-words that are used in questions, When, What, Who, Where, Which, Why and How. This feature is important when it is mapping with the previous head words. There are relations between some head words and the wh-words that both of them may appears together.

The third feature is the word hypernyms, Y is a hypernym of X if every X is a (kind of) Y, an example for that: "**What breed of hunting dog did the Beverly Hillbillies own?**" requires the knowledge of animal being the hypernym of dog. We use WordNet to know the hypernyms of each word of the question. In WordNet the words are organized into hierarchies form with hypernyms relations.

The previous features are the features that we focused on during the implementation phase, there are other features such as the N-grams and words shape, but we haven't used them.

After we get the features, we binaries the features so it can be feeded to the classifier. Features binarisation process is about turning the text features to numerical binary values. To do that, we have used the counting and words similarities approach.

We have a training data that contains the questions and their assigned classes, we extracts the previous features from the dataset questions, then make a new dataset that contains the featured questions labeled with their assigned class. Each class has their own mentioned head words and hypernyms. When the user asks a question, we get the features from it, then search of the features are mentioned before or not (counting process) if they are found with a specific class, we make the features of the question with this class equals to 1 for all features of the question. If the features not found with any class, we use the words similarities, if the question words features similar to some classes words features by a pre-determined coefficient, we make the features equals to 1 for all features of the question.

After we get the binary features vector, we feed the classifier with it. There are 2 main classifies that we used in this module, Maximum Entropy Model and Support Vector Machine. The reasons for using these classifiers is that they perform well with non-linear separable data. In Maximum Entropy we get the maximum probability of the question being with all classes. Maximum Entropy depends on the Parameter estimation (Lambda) which we will illustrate in the

upcoming pages. The second classifier is Support Vector Machine, the reason for using it is its Kernel Trick that can change the problem dimension space which could lead to a successful linear separation of the data.

Also, one of the main reasons for choosing these algorithms is the surveys we have done about the best Machine Learning algorithm for question classification task. The following figure shows the accuracy of using both of these algorithm on TREC dataset using the previous features.

		6 class		50 class	
		SVM	ME	SVM	ME
wh-word + head word		92.0	92.2	81.4	82.0
wh-word +	depth=1	92.0	91.8	84.6	84.8
head word +	depth = 3	92.0	92.2	85.4	85.4
direct hypernym	depth = 6	92.6	91.8	85.4	85.6
wh-word + head + indirect hypernym		91.8	92.0	83.2	83.6
unigram		88.0	86.6	80.4	78.8
bigram		85.6	86.4	73.8	75.2
trigram		68.0	57.4	39.0	44.2
word shape		18.8	18.8	10.4	10.4

Figure 5.1.2

The Maximum Entropy Classifier get the following

$$\log P(C \mid D,\lambda) = \sum_{(c,d)\in(C,D)} \log P(c \mid d,\lambda) = \sum_{(c,d)\in(C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

Figure 5.1.3

Lambda values are the parameter that we want to estimate to solve the problem. We want to get the lambda values that make the above formula value maximum. It is an optimization problem, we have a function and we want to get a best value for a specific parameter to make the whole function value maximum. To solve this task we implemented 2 different algorithms but both of them are considered to be a Swarm Intelligence algorithms. Bats algorithm and Particles Swam Optimization algorithm.

Both of these algorithms are trying to get the best lambda parameter to maximum the Maximum Entropy function returned value.

We haven't completed all of this module testing, but what we've implemented gives us a 65% accuracy which is somehow good as we know how to improve this accuracy.

*5.1.2 Information Retrieval Engine "**IR engine**"*

The Information Retrieval Engine for Brainizer is the second phase in processing. There were several features in Brainizer IR engine which designed to support Brainizer.

IR engine contains:

1. Knowledge-base.
    a. Insertion.
    b. Searching.
    c. Updating.
2. Paragraph processing.
    a. Indexing.
    b. Ranking.

Knowledge-base

Knowledge-base contains 30658 record. Each record contains question, answer, category, source, year of source and seen counter. Categories are Fine Arts, Geography, History, Literature, Mythology, Philosophy, Religion, Science, Social Science and Trash.

Example:
QUESTION:

Corroborating the suspicions generated by the possibly spurious Tanaka memorial, this 1931 incident saw the dynamiting of a section of Japanese owned railroad in Manchuria, providing a pretext for invasion.
ANSWER:
{Mukden} Incident.
Category:
History.

Proprietary data from www.naqt.com" **National Academic Quiz Tournaments**" are excluded. These questions (and questions from other categories) are publically available at:
https://code.google.com/p/protobowl/downloads/detail?name=shuffled.json

Paragraph processing

User will input a huge data like books or papers. Documents will be indexed according the keywords obtained from the Question Processing module Retrieve number of passages that contain the answer for the given question The output passages will be the input for the Answer Processing module.
Indexing:
Takes user input and do two things Prepare Sentences and Prepare Words Table.

Prepare Sentences:
It takes data and split them to sentences, then clear each one "**make them lowercase and remove Impurities**". Now we have all sentences and each one has index number.

Prepare Words Table:
It hash each word in each sentence with all sentences which contains it.

Ranking:
Take keywords from question and find sentences which contains these words. Ranking is based on three techniques.

1. Frequency.
2. Distance.
3. Position.

How many word from question word in this sentence, the distance between words and where each word in the sentence.

5.1.3 Answer Extraction

1) Extracting Answers based on Named Entity Tagger.

In this approach the extraction of the answer mainly based on Name entity recognition and pattern learning/matching and some enhancement to improve the extraction accuracy.

The accuracy of this approach mainly dependent on the right choice of the passages candidates, to make it much easier for the extractor to extract the most accurate answer, And here is a list of features that help in doing the mentioned task.

- **Question Word Matches**

This very simple feature just returns the number of words in the answer passage which also appear somewhere in the corresponding question (comparison is case-insensitive, but matching words must be otherwise identical). Arguably, this is a feature of the passage that is used in the initial IR filtering, and should be replaced by more fine-grained features

That couldn't be carried out in the initial filtering, but we retained this feature because its addition shouldn't hurt performance, and could still potentially be useful in some cases.

- Expected Answer Matches

For each question, we are supplied with an expected answer type in the form of a Named Entity Tag. This feature simply gives the number of named entities in the target passage that match the expected answer type. If the value for this feature is zero, then this answer passage is unlikely to be useful, since it contains no examples of the expected answer type at all. However, values of greater than one are not necessarily any more promising than passages with a single expected answer type match.

- **Punctuation**

This feature adds a score of 1 for each word/phrase of the expected answer type that is followed by a punctuation mark (and 0 when the expected answer is not followed by a punctuation mark) and then returns the total punctuation score divided by the number of expected answer type matches. This takes into account the relatively frequent case in which there is more than one match of the expected answer type in a candidate answer; regardless of the number of expected answer type matches, this feature gives the average number of expected answer type matches that are followed by punctuation. We were not sure exactly why this feature would be useful. Perhaps it can be justified by the fact that named entities followed by punctuation are often modifiers that yield key information. For example, in "**John Smith, born 1904,"** the commas indicate the fact that John Smith and 1904 are probably key figures in the passage. Despite lack of better justification, we decided to include this feature since Pasca et al. seem to have obtained good results using it.

- Question Words in Close Proximity

This feature is similar to the "**Question Word Matches**" feature described above, except that it only counts words in the question that also appear in the answer passage within a certain proximity of a word of the expected answer type. We ended up using the same "**closeness**" criteria from the Pasca paper, counting a word from the question if it was no farther than 3 words and one comma (and no stronger punctuation, like a period) from a particular word of the expected

answer type. Proximity to question words suggest that a candidate answer is likely actually be related to the question words, rather than just appearing in the same passage with them. Punctuation here encodes for the fact the words across phrase and sentence boundaries are less likely to be semantically related to each other. In answers with more than one word of the expected answer type, we consider the number of question words that are close to each individual word matching the expected answer type, and then return the average. This feature can be particularly useful for answers in which a large portion of the question is repeated in the answer.

- Question Words in Order

Here we return the maximum number of words from the question that occur together in the same order in the answer. We only consider uninterrupted strings of words from the question, skipping over any punctuation marks. High scores for this feature can indicate an answer that is especially on-topic and repeats part of the question verbatim. Note that the average expected score for this feature increases as answer passage length increases (which is not true of the features which are based on averages of certain properties of the answer). We assumed this would not be a significant problem since the passages are all approximately the same length.

- WordNet Similarity

It is important to use a feature which uses lexical knowledge in order to identify question text synonyms in an answer passage which gives that passage more relevance to the question than string text matching IR techniques would recognize. Given a measure of similarity between words sim(w1,w2) we define the WordSimilarity feature for an answer passage P and question Q to be [if we let QK be the set of keywords derived from pruning function words in Q]

WordSimilarity(Pi,Q) = 1/ |Pi||Q| X Summation(sim(wi,wj))

We utilized the perl module WordNet::Similarity in order to get similarity scores

derived from theWordNet taxonomy [3]. In particular we used the Resnik measure, which is given by the maximal information content of any common ancestor of two concepts. Formally, if we let c1, c2 be concepts and S(c1, c2) be the set of common ancestors, then the Resnik measure [6] is given by:

$$R(c1, c2) = \max -\log p(c)$$

Where the probability distribution over concepts is empirically estimated from a corpus and the concepts assigned to the words in it. For two words, sim(w1,w2) is given by the maximal Resnik measure over pairs of applicable WordNet concepts, or sys-sets corresponding to w1 and w2. It is important to note that many of the sim scores for a question and passage will be zero if either WordNet does not contain the concept or if the concepts share no common ancestors (the former is much more likely to be the case).

The previous steps guarantee that the chosen passages are accurate with high percent and here how our algorithm works:

- First we collect all the name entities in the passages which have the same entity of the expected answer type which we get from the question classification module.
- After that we sort these entities according to a comparison function that was basically a decision list heuristic. If an answer was from a passage that our passage classifier gave a confidence of more than 5% higher than the confidence given to the passage for the other answer, then the first answer was always Preferred. If the question was a "**When**" question and the one answer contained digits while the other didn't, the answer with digits was always preferred. The reason for this was somewhat of a hack: phrases such "**two hours ago**" or "**tomorrow**" were tagged in our passages as DATE entities, which are the expected answer type for most "**When**" questions, but almost all the "**When**" questions in TREC refer to fixed times in the past. Thus, digits were a good indicator that a candidate answer also refers fixed date or time period. Next, we preferred answers with fewer tokens over more tokens. The reason for this is that

if an answer has too many tokens, it is often a good indication that named the entity tagger went astray and added far too many tokens to a single entity.
Finally, if none of the other decisions applied, we chose the answer that had the largest count in co-occurrence with keywords on Google. Using this decision list as a comparison function, it was simple matter to sort the
Answers using a standard sorting algorithm and return the top 5 answers.

- We used a pattern matching techniques to detect the pattern of the questions and the answers of each question type and that helps at improving the accuracy significantly , This step is very helpful specifically with the questions which it's answers is NOT a named entity, Here are some examples on the list and factoid questions patterns:

- **"How Many" Questions**

How many questions are answered using patterns and not named entities since these questions?
Are looking for a count of a certain entity. We discovered that answers will appear Frequently in the tagged documents in the pattern "**[NP NUMBER ENTITY]**". For the question, "**How many hexagons are on a soccer ball?**" the entity to be counted is hexagons, so the pattern it is trying to match is [NP NUMBER hexagons]. This pattern is found in the following passage:

[PP After/IN] [NP all/DT] ,/, [NP a/DT buckyball/NN] [NP 's/POS structure/ NN] [PP of/IN] [NP 12/CD pentagons/NNS] and/CC [NP 20/CD hexagons/NNS] [VP is/VBZ] [ADVP just/RB] [PP like/IN] [NP that/DT] [PP of/IN] [NP a/DT classic/JJ soccer/NN ball/NN] ./.

This module will extract 20 hexagons and send that as a possible answer, along with the passage it is found in, to the answer ranker.
WordNet is also used to get synonyms for the entity that is being counted. For example, given the question, "**How many floors are in the Empire State building?**" the synset (floor, level, storey and story) of the word floor can be used to extract answers from the passage:

"**The Empire State building climbed to an unthinkable height of 102 stories in that city four years later**."

- **Names of People**

The OAK system tags people in four ways: 1. person, 2. last name, 3. female firstname and 4. Male first name. This presents a problem because the OAK tagger tags everything at once from a list of names. Some last names are missing from the list of tags, and other names are also names of cities, such as the name Paris. The answer extraction module needs to extract the full name of the person when it is available. Some examples of problems and their patterns for solutions are:

[NP hCITY DETROIT/NNP i] :/: [NP hMALE FIRSTNAME Juan/NNP i Gonzalez/NNP] [VP was/VBD] [ADVP back/RB] [PP in/IN] [NP the/DT starting/VBG lineup/NN] [PP after/IN] [VP missing/VBG] [NP three/CD games/NNS] [PP because/IN of/IN] [NP a/DT sore/JJ foot/NN] but/CC [VP may/MD be/VB sidelined/VBN] [PP after/IN] [VP aggravating/VBG] [NP it/PRP] [SBAR while/IN] [VP running/VBG] [NP the/DT bases/NNS] [ADVP when/WRB] [NP he/PRP] [VP hit/VBD] [NP a/DT triple/JJ] ./.

In this example, the name Juan is followed by the untagged last name Gonzalez. This can be fixed by the pattern "**h[A-Z]NAME [A-Za-z]NNP i [A-Za-z]/NNP**". Notice that it will also get last name, as well as the other two types. Some people have last names that are usually considered to be first names:

[NP P.S./NNP hMALE FIRSTNAME Kevin/NNP i hMALE FIRSTNAME Ryan/NNP i] [NP forwards/RB a/DT hCITY Sacramento/NNP Bee/NNP i] [VP clipping/VBG describing/VBG] [NP a/DT paper/NN sign/NN] [PP on/IN] [NP the/DT men/NNS] [NP 's/POS room/NN wall/NN] [PP in/IN] [NP the/DT state/NN] [NP Capitol/NNP] [NP 's/POS hFACILITY Legislative/ NNP Office/NNP i Building/NNP] :/: "/" [VP Please/VB wash/VB] [NP your/PRP$ hands/NNS] [PP before/IN] [VP touching/VBG] [NP legislation/

NN] ./. ”/”

This will present the pattern of “**(NAME TYPE NAME/NNP) (NAME TYPE NAME/NNP)**”.

- **Dates**

There are two types of date questions, one looking for a certain day that happens every year, and ones that are looking for a particular day. Some dates that are tagged by OAK do not fit into either of these categories and are considered relative dates. These include today, this year, this month, next week and midnight. These dates are not helpful in answering questions, and are eliminated right away. Also, answers to questions that are looking for a particular date should have a four digit year in them.

- **Quantities**

OAK tags each quantity it sees as a quantity, but does not tag quantities together that are of the same measurement if the quantities are of different units. For instance, it will tag the measurement 4 foot 2 inches as h PHYSICAL EXTENT 4 foot i h PHYSICAL EXTENT 2 inches i. We can extract the full quantity if we use a pattern that extracts more than one quantity when two quantities of the same measurement are together.

- **Other Types of Questions**

For the rest of the questions, possible answers are extracted by extracting the named entities associated with the answer type of the question. This is done by pattern matching with “**(NE X**)” where X is a possible answer if NE is a named entity tag corresponding the answer type of the question.

- **Definition Questions**

In fact finding questions, there is always a topic that is being sought. If the question is “**Who is X?**” (X being a name of a person), there will be different methods for finding facts for it, compared to a non-person entity that will be phrased as “**What is a Y?**” We have chosen to implement a method of pattern

finding to answer fact based questions.

- **Finding the pattern,** in finding the pattern procedure we proceed 3 steps:

Step 1: Manually Finding Facts

the fact sentences are found by forming a query from the topic, and retrieving relevant passages from our information retrieval system. NIST provides answers to past definition and other questions that can be used to form a query to try to get a specific fact. For the question "**Who is Aaron Copland?**"
The following sentence contains the fact that Aaron Copland is a composer.
His circle of friends included Picasso and Piet Mondrian, composer Aaron Copland, actors Charlie Chaplin and Paulette Goddard, and titans of U.S. industry such as the Fords and Rockefellers.

Step 2: Chunking Fact Sentences

The fact sentences are then chunked and patterns are observed. For example:
[NP His/PRP$ circle/NN] [PP of/IN] [NP friends/NNS] [VP included/VBD] [NP Picasso/NNP and/CC Piet/NNP Mondrian/NNP] ,/, [NP composer/NN Aaron/NNP Copland/ NNP] ,/, [NP actors/NNS Charlie/NNP Chaplin/NNP and/CC Paulette/NNP Goddard/ NNP] ,/, and/CC [NP titans/NNS] [PP of/IN] [NP U.S./NNP industry/NN such/JJ] [PP as/IN] [NP the/DT Fords/NNP and/CC Rockefellers/NNP] ./.

Step 3: Pattern Creation

Patterns are formulated from manual observations of the tagged sentences. The passage shows that information contained before the target, which is in the noun phrase, is a pattern to find out a fact about this target. Our current definition patterns include, with X representing facts and TARGET representing the subject of the fact:

[NP X TARGET]
[TARGET] , X (, or . or ;)

[TARGET] (is or are) X (. or ;)
X called [TARGET]
[BEGINNING OF SENTENCE] [TARGET], X, is X
[TARGET] and other [NP X]

- The next procedure is to **Rank the Answers**, This procedure gives a score to each possible answer, and returns the one with the highest score, or the answers with the highest scores if a list of answers is required. It is possible that the corpus will not contain the answer to the question. Because of this, answers should only be given if the system is sure that the answer is correct. If a list of answers is required by the question, this module will only pass on answers that are over a certain rank.

- **Detecting the answers patterns**, As stated in the previous section, many systems use lexical patterns to extract answers from the documents, in contrast to our approach of extracting named entities. Our system used the following patterns to rank answers.

- **Date of Birth and Date of Death**
  The date of birth and date of death of a person are sometimes put in brackets, after a person's name, e.g. "**PERSON (DATE - DATE)**". For example, the sentence "**Elvis Presley (1935-1977), James Dean (1931-1955) are the new men.**" contains such patterns.

- **"What" Location Questions**

When a location is the answer type of the question, and the question contains a preposition like in, on, from or near, those words will frequently appear before the location entity that is the answer. For questions that contain on and from, in can also appear before the answer.
Examples of these types of questions are:
What continent is Togo on?
What continent is India on?

An example of a sentence with this pattern, for the question "**What continent is Togo on**?" is:
The president, who returned home early on Tuesday after a three-day visit to Mali and Togo in West Africa, agreed to undergo the most intensive medical Check-up to date as a bid by the government and the African National Congress To refute damaging rumours that his health was deteriorating.

- **"When" Questions Ending in a Verb**

Some questions end with a verb that represents the particular action that is being asked about. Examples of these questions are:
When was Microsoft established?
When was the first Wal-Mart store opened?
When was Hiroshima bombed?
For these questions, an answer is usually found in the following pattern, "**VERB in (DATE)**", where VERB represents a verb with a stem that is a synonym of the last verb from the question. An example of a sentence with this type of pattern for the question, "**When was the first Wal-Mart store opened**?" is:
Supercenters, the first of which opened in 1988, had already transformed the $420 billion-a-year grocery business.

- **Who Action**

Who questions often contain the action of being. For example: **who was Khmer Rouge's first leader**? There are some questions that contain a physical action that are not being: who wrote "**Dubliners**"? These actions are represented in the passages that the answers are in, but might take different forms. For these types of questions, our system takes the pattern of the whole action. For example, the question "**who wrote"Dubliners**"?", has the action "**wrote** "**Dubliners**"", and is the pattern our system uses to rank.

- After ranking the candidate answers according to the mentioned features and patterns we **threshold the answer** with the highest rank.

- Finally after answering the question we make a procedure to validate our

answer which based on **the abductive theorem,** It have worked to improve our performance at the level of choosing between entities in an individual passage would have been the use of weak abduction to attempt to prove our answers by using the passages they came from as premises. This approach is in fact used by QA system developed by Pasca et al. which has been by far the most successful of the QA systems at recent TREC competitions. One method of using abduction costs to decide how a set sentences justifies a particular sentence as a conclusion is described by Hobbs and Stickel. We could use a similar method to determine how well an answer to a question is justified by the passage that the answer came from.

We considering implementing such a procedure as follows: First, we would take that candidate answer and use it to fill the gap in the question and turn it into a statement. In some cases this is easy. **For example, we can use "Roy Romer" to fill in the gap in "Who is the Governor of Colorado" and get "Roy Romer is the Governor of Colorado"**
As a statement. In other cases, however, more complex syntactic manipulations would be required. Once a statement has been generated from the question and answer, we could use it as a conclusion to be proved using the passage sentences as the premises. We considered using an abductive theorem prover currently being developed by Rajat Raina to do the actual proving, but eventually we decided against attempting to implement the whole procedure

2) Extracting Answers based on Word vector space.

Problem definition:

One of the biggest problem in the Natural language Processing field is:-
1) Selecting the features that represent the sentence in a digitized model that a computer can understand.
2) Unlike the image recognition Most of the features that can represent the sentence in a semantic manner is variable length size (eg. Parse tree, dependency

graph, etc...) Which is needed to be represented in a vectorized manner to be able to be used.

Solution:

To solve the previous stated problem we need first a way to represent a word in a digitized model, there are three basic common solutions to do so:-

1) N-grams:

Standard approach to language modeling

Task: compute probability of a sentence W

$$P(W) = \prod_i P(w_i|w_1 \dots w_{i-1})$$

Figure 5.1.4

And can be simplified to trigrams

$$P(W) = \prod_i P(w_i|w_{i-2},w_{i-1})$$

Figure 5.1.5

2) Word Classes:

It's one of the most successful NLP concepts in practice.

Similar words should share parameter estimation, which leads to generalization.

Example:

Class1 = (Yellow, green, blue, red)

Class2 = (Italy, Germany, Spain, France)

Usually, each vocabulary word is mapped to a single class (similar words share the same class).

There are many ways how to compute the classes – usually, it is assumed that similar words appear in similar contexts

It's an alternative way instead of just counting the words, we can use also counts of classes, which leads to generalization (better performance on novel data).

There are many ways how to compute the classes – usually, it is assumed that similar words appear in similar contexts.

3) Bag-of-Words (BoW):

Simple way to encode discrete concepts such as words, generally converts the sentence into a vector of frequency of N dimensions, where N is the vocabulary size.

```
vocabulary = (Monday, Tuesday, is, a, today)
Monday Monday       = [2 0 0 0 0]
today is a Monday   = [1 0 1 1 1]
today is a Tuesday  = [0 1 1 1 1]
is a Monday today   = [1 0 1 1 1]
```

Figure 5.1.6

Dimensional embeddings; the latter are then combined component-wise with an operation such as
Summation. The resulting combined vector is classified through one or more fully connected layers.

One of the disadvantages in this model it can cause a confliction in a sentence representation because of the loss of order of the words.
The loss of order problem can be solved by either bag-of-N-grams or Adjacency Matrix of bag of words.
Example:-

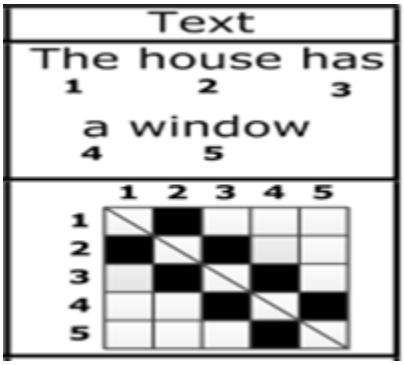


Figure 5.1.7

These are the main features that can be extended to give a high level of sentence representation (eg. Named Entity Recognition, Parse Tree, Text completion, Word embeddings)

Question Answering

After getting the top candidates from the Information retrieval based on sentence shallow processing, in order to get the right answer we should look at the problem from the other point of view from the Semantic perspective and this is the main focus of this module to answer the question
The main characteristics this problem to be able to answer the question is:
1) Recognizing context focus.
2) Question type focus.

1) Recognizing context focus:

1.1) Choosing the headwords of the context

The context headwords is the most important words of the sentence that controls the sentence context.
can be easily extracted and learned by a simple machine learning algorithm such as Support vector machine or Neural network based on the following features :-
1) Part of Speech Tagging:
Usually NN, JJ, VB, NNS, RB, VBN have high weights to be headwords.
2) Dependency graph:
The dependency relation between the current word and the headwords.
3) NER:
If the current word is a Named Entity.

This method should be applied to the question and the answer to extract the context head words.

1.2) Finding the similarity

Calculate the context similarity between the question and the answer.

To solve this problem, we should first solve its sub-problem which is finding the similarity between two words.
There's two common solutions to do so:-

1.2.1) Word Hypernyms

Each word has its hypernyms which is extracted from a dictionary (eg. Wordnet), Hypernyms of a single word share the same semantic meaning, this method is about to group them in a single class to be able to be used in the calculation of the similarity.
Very simple way if compared to the other methods but not very efficient.

1.2.2) Word Embedding

Usually known as Word embedding or word vectors, this is based on representing a word as a vector.
Word vectors can be trained on huge text datasets, they provide generalization for systems trained with limited amount of supervised data.
More complex model architectures can be used for obtaining the word vectors (neural net language model with multi-task learning (Collobert & Weston, 2008)).

1.2.2.1) Word vectors – linguistic regularities

Recently, it was shown that word vectors capture many linguistic properties (gender, tense, plurality, and even semantic concepts like "**capital city of**").
We can do nearest neighbor search around result of vector operation "**King – man + woman**" and obtain "**Queen**" (Linguistic regularities in continuous space word representations (Mikolov et al, 2013)).

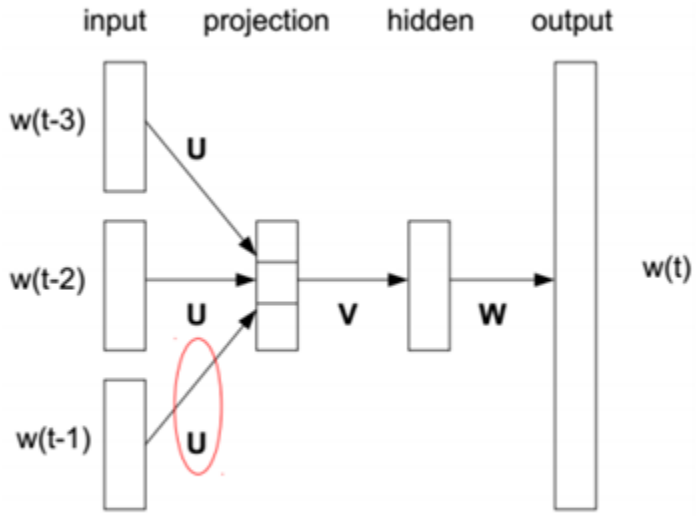
Tomas Mikolov, COLING 2014

Figure 5.1.8

1.2.2.2) Word vectors – various architectures

Neural net based word vectors were traditionally trained as part of neural network language model (Bengio et al, 2003).

· **Bigram NNLM**



Tomas Mikolov, COLING 2014

Figure 5.1.9

Neural Network Language Modeling models consists of input layer, projection layer, hidden layer and output layer, this model can be extended by adding more context to the bigram NNLM.

The projection layer will be explained in the next section.

· **CBOW**

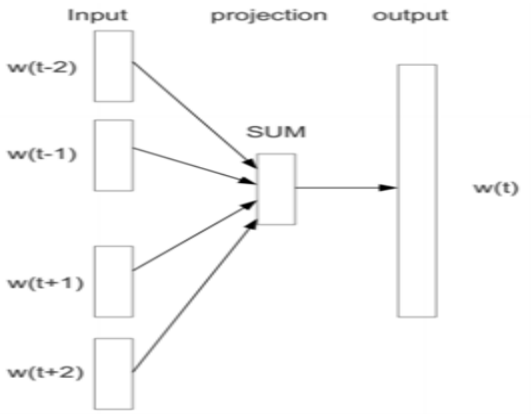


Figure 5.1.10

ontinuous bag-of-words model adds inputs from words within short window to predict the current word. This model differs from the NNLM in the following characteristics:-

-The weights for different positions are shared.

-Computationally much more efficient than normal NNLM.

-The hidden layer is just linear.

· **Skip-gram NNLM**

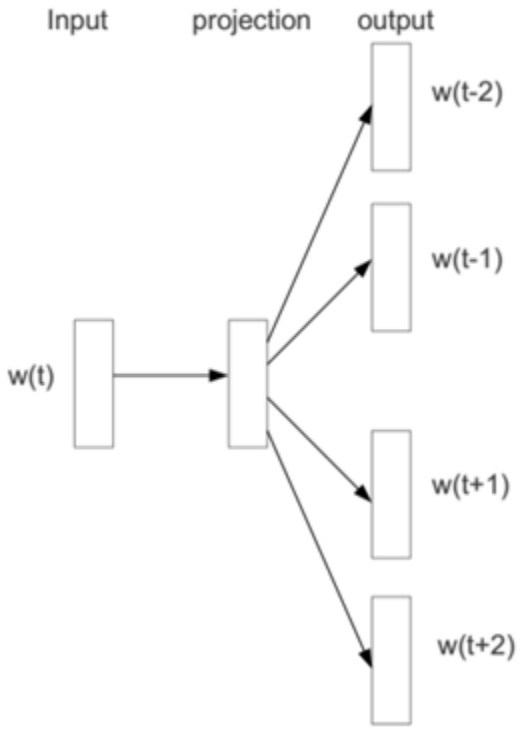


Figure 5.1.11

We can reformulate the CBOW model by predicting surrounding words using the current word, performance is almost similar to CBOW if both architectures trained for sufficient number of epochs.

The projection layer:

The projection layer maps the discrete word indices of an n-gram context to a continuous vector space.

Each neuron in the projection layer is represented by a number of weights equal

to the size of the vocabulary. The projection layer differs from the hidden and output layers by not using a non-linear activation function. Its purpose is simply to provide an efficient means of projecting the given ngram context onto a reduced continuous vector space for subsequent processing by hidden and output layers trained to classify such vectors. Given the one-or-zero nature of the input vector elements, the output for a particular word with index i is simply the i th column of the trained matrix of projection layer weights (where each row of the matrix represents the weights of a single neuron).

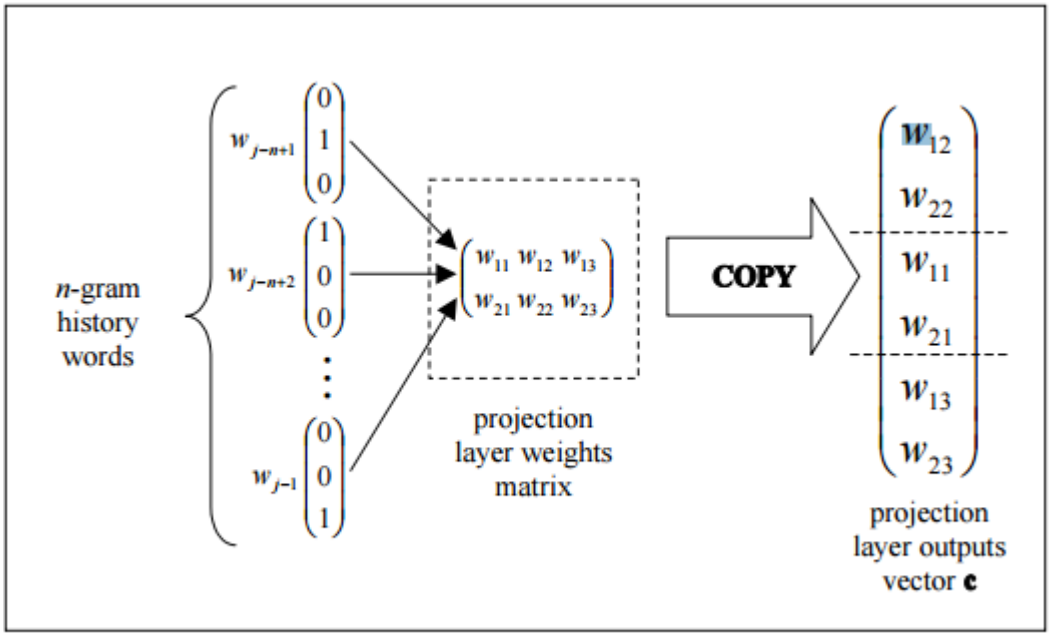


Figure 2 – Formatting the projection vector from an *n*-gram context

Figure 5.1.12

For an n-gram history of length h n 1 and a projection layer size of P neurons, the final output from the projection layer is a vector of real values of length h * P. The resulting projection vector is used as the input to the next layer in the network: the hidden layer.

1.2.1.3) Word vectors – training

Output layer size is very large – equal to the vocabulary size can be easily in order of millions. Two methods are used to solve this problem:-

1) **Negative sampling**
2) **Hierarchical softmax**

Before the training it's very useful to sub sample the frequent words (such as 'the', 'is', 'a', etc…) during training.
Also Non-linearity does not seem to improve performance of these models, thus the hidden layer does not use activation function.

- **Negative sampling**

Instead of propagating signal from the hidden layer to the whole output layer, only the output neuron that represents the positive class + few randomly sampled neurons are evaluated.
The output neurons are treated as independent logistic regression classifiers.
This method makes the training speed independent on the vocabulary size.

- **Hierarchical Softmax**

In traditional NNLM, we need to compute the conditional probabilities of all vocab words given a history:
p(w|history) $w \in Vocabulary$,
Finally perform a normalization (namely softmax). Obviously, such kind of operation requires huge computation, especially on the whole vocab.

To solve this problem this solution was proposed with the following processes:-
1) First, build a Huffman tree based on word frequencies. As a result, each word is a leaf of that tree, and enjoys a path from the root to itself;
2) Now, each step in the search process from root to the target word is a normalization.
For example, Root chooses left subtree with probability 0.7 and chooses right subtree with probability 0.3. This is a normalization step in this tree level.
Naturally, the final probability for finding the target word is the continuous multiplication of probabilities in each search step.
3) To compute the Hierarchical Softmax probability from this equation

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot {v'_{n(w,j)}}^T v_{w_I})$$

Figure 5.1.13

Some comments to understand this formula:-

1) The whole process is a continuous multiplication. Product ${v'_{n(w,j)}}^T v_{w_I}$ can be treated as a kind of "**similarity**" between history (also called "**Input**" or "**context**") and the jth ancestor. Note that all the leaves and non-leaves (i.e., ancestors) have initialized embeddings which will be updated gradually. So, above "**similarity**" is computed using embeddings. **Its intuition is that if given context is more similar with the ancestors of certain word, then the word has higher probability to be the target word.**

2) Given "**similarity**" score, Hierarchical Softmax converts it to two probabilities to search left subtree and right subtree. That's achieved by using $\sigma([n(w,j+1) = ch(n(w,j))] \cdot {v'_{n(w,j)}}^T v_{w_I})$. Note that the equation in the symbol "$[\cdot]$" is -1 or +1. It makes $\sigma(+x) + \sigma(-x) = 1$. This is a normalization step. The original sentence says "**let ch(n) be an arbitrary fixed child of n**". Hence, we can always think that ch(n) means the left child. So, if the j+1 ancestor of target word w is the left child of its j ancestor, then the probability of j choosing j+1 is $\sigma(+0.78234)$ (let's assume the similarity between context and ancestor j is 0.78234), otherwise, the probability is $\sigma(-0.78234)$.

3) Now, we know how to compute the probability of ancestor j choosing j+1 ($j \in \{1, 2, \cdots, L(w) - 1\}$). Multiplying them one by one equals to the normalized probability of target word given context or history.

1.2.2.4) Comparison of performance

Model	*Vector Dimensionality*	*Training Words*	*Training Time*	*Accuracy [%]*
Collobert NNLM	50	660M	2 months	11
Turian NNLM	200	37M	few weeks	2
Mnih NNLM	100	37M	7 days	9
Mikolov RNNLM	640	320M	weeks	25
Huang NNLM	50	990M	weeks	13
Skip-gram (hier.s.)	1000	6B	hours	66
CBOW (negative)	300	1.5B	**minutes**	**72**

Figure 5.1.14

1.2.2.5) Scaling up

The choice of training corpus is usually more important than the choice of the technique itself.

The crucial component of any successful model thus should be low computational complexity.

1.2.2.6) Examples

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Figure 5.1.15

1.2.2.7) Visualization using PCA

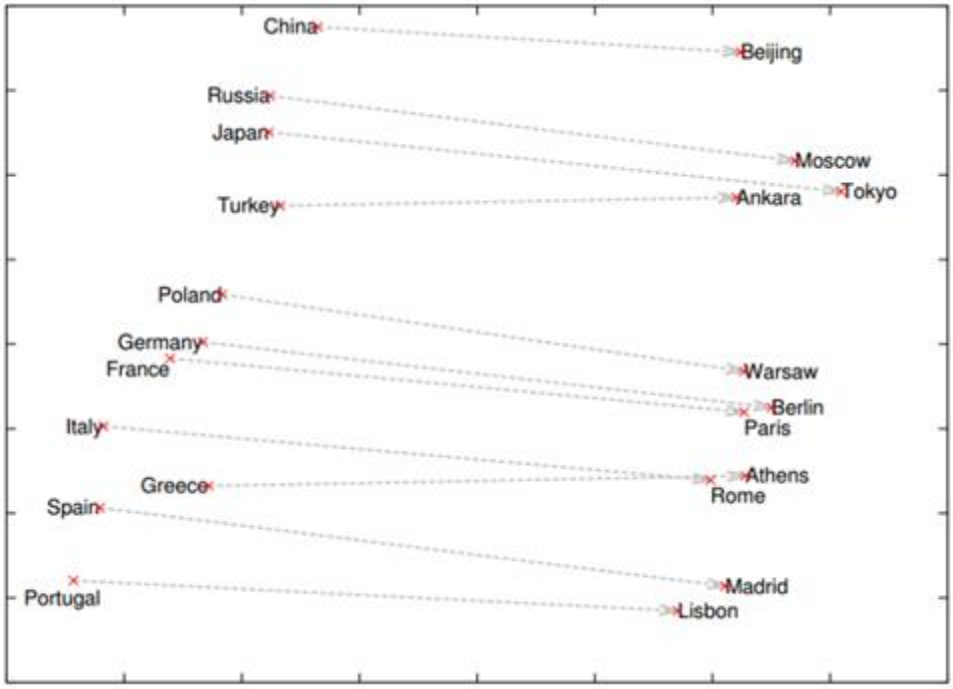


Figure 5.1.16

The same technique that's used to represent the words can be used to represent the sentence but it's very complicated and should have imaginary number of training data.

By the end of this module each word has a corresponding vector representing it in a semantic manner and sure of the question and answer share the same context.

2) Question type focus:

This is usually known as Question-Answer Pattern learning, Each question type share a set of words that's usually answered by (eg. When questions type the answers usually contains "**IN**")

Figure 5.1.17

This module is trained by giving a set of answers that answers the question type correctly and a set of answers that doesn't, every input data is labeled by the question type.

The features that are used in this classifier is Wi, Wi-1, Wi-2, the value of each word is extracted from the word embedding.

An alternative way to represent the features is based on the Part of Speech Tagging and Named Entity Recognition. This module can classify the answer correctly using simple Neural Network.

5.2 Programming Language

Implementing such system needs to make an OOP design. So, we determined to make the system using Java programming language for its support for OOP. And the other reason for choosing Java is that we have used Stanford Core-NLP API and other Java APIs for modules tasks, so it is better to make the whole system with Java without integrating multiple programming languages together.

The tools we have used in the system: Stanford Core-NLP API, WordNet API, and Words Similarity API

5.3 User Manual

We are working for a fully Java Doc that contains all classes and APIs we used during the implementation phases.

5.4 Completed / Uncompleted Parts

The completed parts are: Question Classification, Information Retrieval Engine and Answers extraction. But each one of them needs some maintenance to improve each one output accuracy.

The uncompleted part are: Adding Voice Recognition Engine, Adding a system-user conversation module.

Chapter 6

Testing

User Input:

Arabic (, ( ) or) is a Central Semitic language, thus related to and classified alongside other Semitic languages such as Hebrew and the Neo-Aramaic languages. In terms of speakers, Arabic is the largest member of the Semitic language family. It is spoken by more than 280 million people as a first language, most of whom live in the Middle East and North Africa, and by 250 million more as a second language. Arabic has many different, geographically-distributed spoken varieties, some of which are mutually unintelligible. "**Arabic language**." Encyclopaedia Britannica. 2009. Encyclopaedia Britannica Online. Retrieved on 29 July 2009. Modern Standard Arabic is widely taught in schools, universities, and used in workplaces, government and the media.

Modern Standard Arabic derives from Classical Arabic, the only surviving member of the Old North Arabian dialect group, attested in Pre-Islamic Arabic inscriptions dating back to the 4th century. Versteegh, 1997, p. 33. Classical Arabic has also been a literary language and the liturgical language of Islam since its inception in the 7th century.

....
”http://www.ark.cs.cmu.edu/QA-data/ “

Question:

Where are the Western Arabic numerals used?

Correct answer:

North Africa

Question processing output:

Head word:

western

Key words:

western
arabic
numerals

Question type:

Country Location

Ranking Output:

97.6 %
in most of present-day north africa the western arabic numerals 0 1 2 3 4 5 6 7 8 9 are used.

63.966666666666676 %
however in egypt and arabic-speaking countries to the east of it the eastern arabic numerals are in use.

54.96666666666667 %
to handle those arabic letters that cannot be accurately represented using the latin script numerals and other characters were appropriated.

53.366666666666674 %
arabic calligraphy has not fallen out of use as calligraphy has in the western world and is still considered by arabs as a major art form calligraphers are held in great esteem.

45.06666666666668 %

central asian arabic spoken in uzbekistan tajikistan and afghanistan is highly endangered hassaniya arabic spoken in mauritania some parts of mali and western sahara hejazi arabic spoken in hejaz western saudi arabia judeo-arabic dialects maltese spoken on the mediterranean island of malta is the only one to have established itself as a fully separate language with independent literary norms.

33.33333333333334 %
arabic or is a central semitic language thus related to and classified alongside other semitic languages such as hebrew and the neo-aramaic languages.

32.06666666666666 %
maghrebi arabic heavily influenced by berber in pronunciation and includes moroccan arabic algerian arabic algerian saharan arabic tunisian arabic and libyan arabic and is spoken by around 45 million north africans in morocco western sahara algeria tunisia libya niger and western egypt it is mostly difficult for speakers of near eastern arabic varieties to understand.

17.866666666666674 %
traditionally there were several differences between the western north african and middle eastern version of the alphabetin particular the fa and qaf had a dot underneath and a single dot above respectively in the maghreb and the order of the letters was slightly different at least when they were used as numerals.

Other questions and correct answers for this example:

- Is Arabic a Central Semitic language? Yes
- Is Arabic classified alongside Semitic languages? Yes
- How many people speak the Arabic language? 280 million people.
- When was Arabic calligraphy invented? Many styles were developed after 786.
- Where is Arabic spoken? The Middle East and North Africa

- Why is Arabic related to Islam? Arabic is the liturgical language of Islam
- Why does Arabic heavily influence European languages? Arabic was a major vehicle of culture in Europe, and the Arab and European civilizations are geographically close.
- Is Arabic the largest member of the Semitic language family? yes
- Does Modern Standard Arabic continue to evolve like other languages? yes
- Is Hassaniya Arabic spoken in Mauritania? yes
- What is the only variety of modern Arabic that has acquired official language status? maltese
- Egyptian Arabic is spoken by how many in Egypt? 76 million
- Where are the Western Arabic numerals used? North Africa
- Where are the Western Arabic numerals used? present-day North Africa
- The most active Academies of Arabic Language are found where? damascus and cairo
- Hassan Massoudy is a master of what genre? Hassan Massoudy
- Hassan Massoudy is a master of what genre? Arabic calligraphy

Chapter 7

Conclusion and Future Improvements

7.1 Most Important Points of the System

The methods that are used in the system is generalized not specific to a single domain.
The performance of the system is strongly related to the size of the training data set.

The answer is extracted based on deep learning algorithms which correctly represent and understand the sentence in a semantic manner.

7.2 Possible Enhancements and Extensions

Instead of just representing the words as vectors of their semantic meaning using word embedding, much more efficient way is to represent the whole sentence as a vector in a corresponding similar space, by doing so the system would be able to answer all the questions a human being can answer it. Autoencoders Neural Network, this neural network is proposed in order to save the knowledge in a very similar way the human brain does.

Conclusion

In our system, we tried to make an effective application that deals with Machine Learning and Natural Language Processing algorithms. We may not reached to a high accuracy compared with large systems such as Siri and IBM Watson. But making such a system deeply benefits us and opened our eyes to the world’s progress to make applications that compete with human's brain!

References

[1] Green BF, Wolf AK, Chomsky C, and Laughery K. Baseball: An automatic question answerer. In Proceedings of Western Computing Conference, Vol. 19, 1961, pp. 219–224.
[2] Weizenbaum J. ELIZA - a computer program for the study of natural language communication between man and machine. In Communications of the ACM, Vol. 9(1), 1966, pp. 36-45.
[3] Woods W. Progress in Natural Language Understanding - An Application to Lunar Geology. In Proceedings of AFIPS Conference, Vol. 42, 1973, pp. 441–450.
[4] Bobrow DG, Kaplan RM, Kay M, Norman DA, Thompson H, and Winograd T. Gus, a frame-driven dialog system. Artificial Intelligence, Vol. 8(2), 1977, pp. 155-173.
[5] Katz B. Annotating the World Wide Web using natural language. In Proceedings of the 5th RIAO conference on Computer Assisted Information Searching on the Internet,1997, pp. 136-159.
[6] Voorhees EM. The TREC-8 question answering track report. In Proceedings of TREC-8, 1999, pp. 77-82.
[7] Clark P, Thompson J, and Porter B. A knowledge-based approach to question answering. In Proceedings of AAAI'99 Fall Symposium on Question-Answering Systems, 1999, pp. 43-51.
[8] Riloff E and Thelen M. A Rule-based Question Answering System for Reading Comprehension Tests. In ANLP /NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems, Vol. 6, 2000, pp. 13-19.
[9] Ittycheriah A, Franz M, Zhu WJ, Ratnaparkhi A and Mammone RJ. IBM's statistical question answering system. In Proceedings of the Text Retrieval Conference TREC-9, 2000.
[10] Berger A, Caruana R, Cohn D, Freitag D, and Mittal V. Bridging the lexical chasm: statistical approaches to answer-finding. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000, pp. 192- 199.
[11] Soubbotin MM and Soubbotin SM. Patterns of Potential Answer Expressions as Clues to the Right Answer. In Proceeding of the TREC- 10, NIST, 2001, pp. 175-182.
[12] Kwok C, Etzioni O and Weld DS. Scaling question answering to the Web. ACM Transactions on Information Systems (TOIS), Vol. 19(3), 2001, pp. 242-262.
[13] Ravichandran D and Hovy E. Learning surface text patterns for a question answering system. In proceeding of 40th Annual Meeting on Association of Computational Linguistics, 2002, pp. 41-47.
[14] Zhang D and Lee WS. Web based pattern mining and matching approach to question answering. In Proceedings of the 11th Text REtrieval Conference, 2002.
[15] Sneiders E. Automated question answering using question templates that cover the conceptual model of the database. In Natural Language Processing and Information Systems, Springer Berlin Heidelberg, 2002, pp. 235-239.
[16] Greenwood M. and Gaizauskas R. Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03), 2003, pp. 29-34.
[17] Moschitti A. Answer filtering via text categorization in question answering systems. In Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, 2003, pp. 241-248

[18] Chung H, Song YI, Han KS, Yoon DS, Lee JY, and Rim HC. A Practical QA System in Restricted Domains. In Workshop on Question Answering in Restricted Domains. 42nd Annual Meeting of the Association for Computational Linguistics (ACL), 2004, pp. 39-45.
[19] Ramakrishnan G, Chakrabarti S, Paranjpe Dand Bhattacharya P. Is question answering an acquired skill?. In Proceedings of the 13th ACM international conference on World Wide Web, 2004, pp. 111-120.
[20] Cai D, Dong Y, Lv D, Zhang G, Miao X. A Web-based Chinese question answering with answer validation. In Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering, pp. 499-502, 2005.
[21] Whittaker E, Furui S and Klakow D. A Statistical Classification Approach to Question Answering using Web Data. In IEEE International Conference on Cyberworlds, 2005, pp. 8-pp.
[22] Soricut R and Brill E. Automatic question answering using the web: Beyond the factoid. In Journal of Information Retrieval-Special Issue on Web Information Retrieval, Vol. 9(2), 2006, pp. 191-206.
[23] Hao X, Chang X, Liu K. A Rule-based Chinese question Answering System for reading Comprehension Tests. In 3rd International Conference on International Information hiding and and Multimedia Signal Processing (IIH-MSP 2007), Vol. 2, 2007, pp.325-329.
[24] Saxena AK, Sambhu GV, Kaushik S, and Subramaniam LV. Iitd-ibmirl system for question answering using pattern matching, semantic type and semantic category recognition. In Proceedings of the TREC, Vol. 2007, 2007.
[25] Cui H, Kan MY and Chua TS. Soft pattern matching models for definitional question answering. In ACM Transactions on Information Systems (TOIS), Vol. 25(2): 8, 2007.
[26] Xia L, Teng Z, and Ren F. An Integrated Approach for Question Classification in Chinese Cuisine Question Answering System. In IEEE second International Symposium on Universal Communication, 2008, pp. 317-321.
[27] Lee YH, Lee CW, Sung CL, Tzou MT, Wang CC, Liu SH, Shih CW, Yang PY and Hsu WL. Complex question answering with ASQA at NTCIR-7 ACLIA. In Proceedings of NTCIR-7 Workshop Meetings, Entropy, 1, 10, 2008.
[28] Han L, Yu ZT, Qiu YX, Meng XY, Guo JY and Si ST. Research on passage retrieval using domain knowledge in Chinese question answering system. In Proceedings of IEEE International Conference on Machine Learning and Cybernetics, Vol. 5, 2008, pp. 2603-2606.
[29] Quarteroni S, and Manandhar S. Designing an interactive open-domain question answering system. Natural Language Engineering, Vol.15(1), 2009, pp. 73-95.
[30] Mishra A, Mishra N, Agrawal A. Context-aware restricted geographical domain question answering system. In Proceedings of IEEE International Conference on Computational Intelligence and Communication Networks (CICN), 2010, pp. 548-553.
[31] Gunawardena T, Lokuhetti M, Pathirana N, Ragel R, Deegalla S. An automatic answering system with template matching for natural language questions. In Proceedings of 5th IEEE International Conference on Information and Automation for Sustainability (ICIAFs), 2010, pp. 353-358.
[32] ANSWER EXTRACTION FOR SIMPLE AND COMPLEX QUESTIONS, SHAFIQ RAYHAN JOTY Bachelor of Science in Computer Science and Information Technology Islamic University of Technology (Bangladesh), 2005
[33] CS224 Final Project: Answer Extraction Aria Haghighi, Guy Isley, Alex Williams

Links:

http://www.coling-2014.org/COLING%202014%20Tutorial-fix%20-%20Tomas%20Mikolov.pdf
https://yinwenpeng.wordpress.com/2013/09/26/hierarchical-softmax-in-neural-network-language-model/
http://cs.nyu.edu/~fergus/teaching/vision_2012/9_BoW.pdf
http://www.cs.cmu.edu/~mfaruqui/talks/nn-clab.pdf
https://perso.limsi.fr/bg/pageWebPHP/fichiers/GrauKO02.pdf
http://www.cs.berkeley.edu/~klein/cs294-19/SP08%20cs294%20lecture%2023%20--%20question%20answering%20(6PP).pdf
http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction-Slides.pdf
http://nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf
http://nlp.stanford.edu/courses/NAACL2013/
http://www.socher.org/index.php/DeepLearningTutorial/DeepLearningTutorial
http://smir2014.noahlab.com.hk/paper%209.pdf
http://arxiv.org/pdf/1412.1632.pdf
http://nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf
http://lxmls.it.pt/2014/socher-lxmls.pdf

Libraries:

http://nlp.stanford.edu:8080/corenlp/process
https://code.google.com/p/word2vec/

List of figures: