USING REASONING TO DECIDE THE ANSWER TO A DECISION QUESTION INVOLVING AN OPINION ADJECTIVE

A Thesis Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Huijie Li

May 2017

USING REASONING TO DECIDE THE ANSWER TO A DECISION QUESTION INVOLVING AN OPINION ADJECTIVE

Huijie Li

APPROVED:

Dr. Kam-Hoi Cheng, Chairman

Dr. S. H. Stephen Huang

Dr. Rajkumar Vedam
Schneider Electric USA, Inc

Dean, College of Natural Sciences and Mathematics

USING REASONING TO DECIDE THE ANSWER TO A DECISION QUESTION INVOLVING AN OPINION ADJECTIVE

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Huijie Li

May 2017

Acknowledgements

I would like to give special thanks to my advisor Dr. Kam-Hoi Cheng for his patient instructions in my two years' master study. He guided me to the areas of AI and NLP, which really inspire my interest. His supervision on the thesis work also provided the novel insights into the problems. I really appreciate his time for teaching me, which gave me the correct direction to focus my efforts. It is my great honor to have a chance to learn from him and work with him.

I am very grateful to my committee members, Dr. S. H. Stephen Huang and Dr. Rajkumar Vedam. Thanks a lot for their helpful advice and shared knowledge about my master thesis work. I appreciate the chance to be an intern under Dr. Rajkumar Vedam. He taught me how to be a professional engineer in the industry. He has had a very positive impact on how I should work in the future.

I also want to say thanks to my friends: Yi Jiang, Santosh KC, Fan Yang, and Yifan Zhang. Thanks to them for their patience towards my questions. My thesis work is to simulate human thinking. Thanks for them for sharing their experience about logical reasoning and natural language processing.

USING REASONING TO DECIDE THE ANSWER TO A DECISION QUESTION INVOLVING AN OPINION ADJECTIVE

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Huijie Li

May 2017

Abstract

Communication ability is an important feature of humans. Artificial Intelligence is to simulate human learning, understanding, and thinking processes by a computer program. A Learning Program System (ALPS) has been developed enabling computers to understand knowledge, to learn English grammar, and to read English sentences. Here, we intend to improve the communication ability of ALPS. The focus of this thesis is to understand and answer the decision question involving an opinion adjective. Humans can learn notions easily, like "importance". However, it is difficult for a computer to understand notions well because notions are abstract and difficult to quantify. This thesis analyzes the human's learning process of notions and creates a logical inference algorithm to simulate the human's thinking process.

A human learns a new notion by learning the various criteria required to justify the use of an adjective of the notion. The criteria include both examples and important factors. We created a knowledge-based method to simulate the process of learning a notion with various criteria. In our knowledge-based method, we abstract knowledge components of a notion: scales, factors, and examples of each factor. Given the target, the adjective, and the reference that represent a decision question, our inference algorithm first matches the target with the examples. If not matched, it then measures the similarity between the target and the examples based on the reason why each example is considered to be an example. After that, a heuristic search will search the factors. Our program will give a "Yes" if the target can satisfy any factor; otherwise, it will give a "No". In addition, an associated reason is also provided to justify the answer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction in AI and NLP

Artificial Intelligence (AI) has been developing for more than 60 years. It has progressed greatly, including areas in language and image recognition, robotics, logic systems, and deep learning. This thesis intends to use AI technologies to enhance Natural Language Processing (NLP) in A Learning Program System (ALPS). In this section, we introduce brief histories of AI and NLP. Our solution is inspired from successful methods.

AI research started at a workshop where researchers discussed how to simulate human intelligence in computers in 1956 [10]. It has progressed immensely in four stages. After that workshop, several groups were founded for preliminary AI research. They applied AI in programs such as checkers, and they also created a reasoning system called Logic Theorist (LT) which can prove mathematical theorems. Based on these simple applications, they contributed a couple of advanced

AI applications: General Problem Solving (GPS) and NSS chess player. Meanwhile, some other significant AI research achievements arose in machine learning, pattern recognition, and AI applications design. For example, LISP, a main AI programming language was invented [10]. Inspired by the progress, AI research has experienced several main development stages [10].

The first stage (1952-1969): The definition of AI was proposed. Many achievements like game players, GPS, and LISP appeared. However, some applications failed because of the limited inference reasoning ability. In this stage, people attached great importance in creating methods to solve special problems and ignored general knowledge representation and retrieval.

The second stage (1960s-1970s): The knowledge-based system (KBS) and expert systems appear. In this stage, logical reasoning and knowledge base combination could solve more complex problems. AI was applied in more real-life cases, like language recognition, disease treatment, and chemistry analysis.

The third stage (1970s-1980s): The neural network, which can be used in optimization problems, brought significant improvements to many areas like pattern recognition, prediction, and intelligent control.

The fourth stage (1980s-1990s): Web technology, especially the Word Wide Web (WWW), has made revolutionary progress. AI developed from a single intelligent agent to a multiple-agent system. The AI in this stage was applied in dynamical networks and multi-objective optimization, which has actively influenced people's lives in various areas.

AI has many applications in its development, and NLP is one of them. Natural

Language Processing (NLP) enables computers to communicate with humans [1]. AI technologies have been used in NLP work since the late 1960s with a focus on NLP's knowledge representation [4]. The knowledge representation and logic reasoning from AI technologies can solve the language-understanding problems in NLP.

1.2 Methods Review in AI and NLP

To develop a natural language framework in NLP, three core functions should be achieved: reading, understanding, and writing [2]. Although numerous methods have developed to enable computers to reach these functions, NLP faces major challenges in understanding complex structure sentences and ambiguous language [3]. Therefore, the key to enhance NLP is to improve its ability to understand. Currently, NLP applications try to understand language from two main parts [12]:

1. Syntax Analysis: Check the syntax of input sentence; parse and restructure the sentence based on grammatical rules.

2. Semantic Analysis: Generate and represent the basic meaning of the input sentence.

Syntax defines the grammar rules that organize words in a sentence. Syntax analysis is inspired by how humans learn a language: humans first learn individual words then linguistic fundamentals. After that, humans practice to parse and make up sentences so that they can use the language fluently. Similarly, Syntax Analysis methods (such as the Context-Free Grammar) recognize grammar rules that organize words to a sentence. However, with only a set of grammar rules, computers cannot always correctly understand complex sentences.

To understand the meanings of complicated sentences, semantic analysis methods are developed. One of the approaches used in the semantic analysis is logic reasoning. It represents the knowledge from sentences and expresses the logical structure in the represented knowledge.

In this thesis, we intend to use logic reasoning as the main approach in semantic analysis. Our logic reasoning methods were developed in a knowledge-based system. A knowledge-based system mainly has two types: Knowledge Base (KB) and Inference Engineer. The KB represents sentences with particular formal language. Like a memory in the knowledge-based system, KB can be used for providing knowledge about facts for an Inference Engineer. The KB should be able to efficiently retrieve and maintain the knowledge from sentences, in order to represent a large number of sentences. The Inference Engineer uses the information in KB to infer some new facts with logic reasoning. The Inference Engineer is the learning and reasoning part of the knowledge-based system. The relation between the knowledge-based system and reasoning system is shown in Figure 1.1.

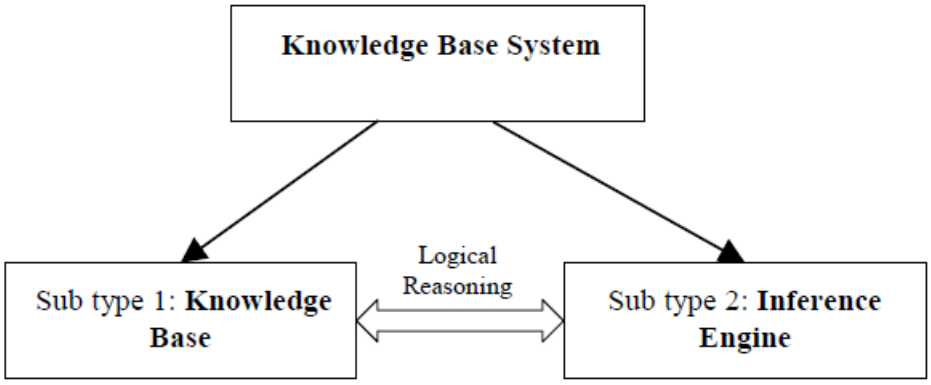

Figure 1.1: Knowledge base system introduction.

An example of the knowledge-based system application is an Expert System.

It contains a knowledge base in specific domains and an inference engine for logical reasoning. The Expert System is designed to collect and learn from experts' intelligence to solve problems in various domains [13]. The Expert System has great value in areas where experience and experts' knowledge are precious and rare, like the medical field. The knowledge-base of an Expert System builds a knowledge database from experts' knowledge in certain domains. The Inference Engine is like a brain in the Expert System. The rule interpreter and logic reasoning are used to implement the Inference Engine. The Inference Engine can understand queries from users and then uses logical-reasoning chains to draw final conclusions for users [13]. Expert System is a successful application of a knowledge-based system. The success of the Expert System underlines the importance of logic reasoning. The logical-reasoning chain should be robust and flexible to use the expert's knowledge in other cases in the same domain.

As we pointed out before, logic reasoning is the central part in the Inference Engineer of a knowledge-based system. Logic reasoning imparts intelligence to computers, which simulates the process of human inference. Logic reasoning tries to implement some procedures into different methods including basic logical-reasoning methods (deduction, induction, abduction), First Order Logic (FOL), and production system and frame. The basic logical-reasoning methods are introduced below.

Deduction applies one or more premises to generate a conclusion for a certain case [7]. It is used to infer a conclusion for a particular case from the general principle. Typical reasoning systems developed from deduction are Inference Engines and Theorem Provers. Following is a simple example for deduction:

Rule 1: IF it rains,

THEN the ground will be wet.

Fact Information: It is raining now.

Conclusion: The ground is wet.

This example shows basic parts of deduction, rules, fact information, and conclusion. The rules and fact information are stored in the knowledge base. Each rule has two parts: primes (IF part) and assertion (THEN part). After matching primes with the fact information, the conclusion can be drawn from the stored assertion parts.

Induction is the reverse of deduction. Deriving a general principle from some given special cases [8], induction is mainly used in neural network and pattern recognition. In this case, we know some facts will generate a conclusion based on those collected facts.

A simple induction example:

Case 1, The first observed car has four wheels.

Case 2, The second observed car has four wheels.

Case 3, The third observed car has four wheels.

Case n,

General conclusion: All cars have four wheels.

Essential parts in induction are shown in this simple example, including observed cases and the conclusion. To implement induction, we should observe or collect several facts. From these facts, we can predict or summarize a general principle, although the conclusion may not be exactly right. Also, similar to deduction, the

conclusion in one induction can be used as the prime case and generate another conclusion with another inductive reasoning chain.

Abduction is similar to induction; it also infers a general principle from observed facts. However, abduction is a way to infer a principle that is the best explanation for the observation. In another word, the conclusion drawn from abduction is the most likely explanation for the observation. Abduction is frequently used in the Expert System.

An example for abduction:

Hypothesis 1: Streets are wet. It has rained. (Most likely)

Hypothesis 2: Streets are wet. The sprinkler is on. (Perhaps)

Hypothesis 3: Streets are wet. The buckets leaked. (Low possibility)

In this example, if we have the fact that streets are wet, we would like to find an explanation for it. After reviewing this fact in the knowledge base, we can get multiple hypotheses. Then we will choose the most likely hypothesis as the conclusion of this abduction.

Deduction, Induction, and Abduction are basic logical inference methods. With these methods, we can implement prediction, analysis, and a reasonable assumption. A high-level summary of these methods is shown in Figure 1.2.

With the introduced logical reasoning methods, computers can make a simple response to the input sentences. However, to solve different kinds of practical problems, more sophisticated interpretations should be added besides the logical reasoning. In order to interpret more complex knowledge in real cases, more structures and items are added to the basic reasoning methods. With the introduced logical reasoning

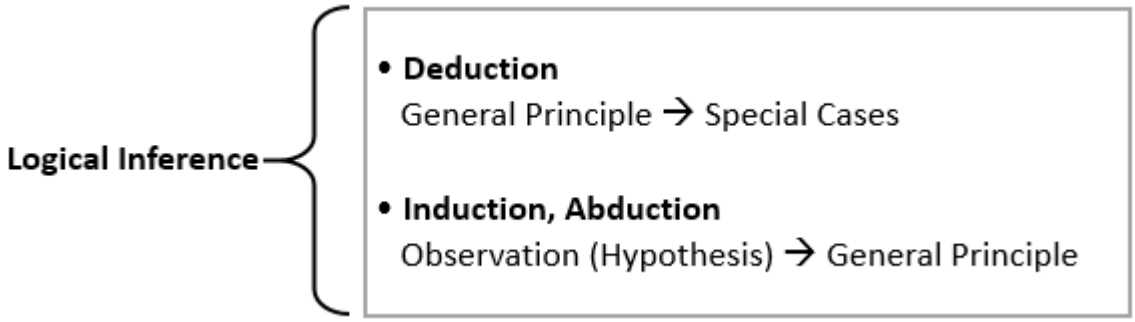

Figure 1.2: Summary of logical inference methods.

methods, computers can make simple responses to input sentences. However, to solve different kinds of practical problems, more complex interpretations should be added besides the logical reasoning, like First Order Logic, Production System, and Frame In complex interpretations, more structures and items are added to the basic reasoning methods.

First Order Logic (FOL) is a deduction method with quantifiers, functions, and relation symbols proposed by Batwise in 1977 [4]. It can support grouping, building relations, and adding connections. It enhances the formal logic expression because it adds variables, the range of variables, and relations between groups in representation. Thus, the semantic meaning of the sentence can be well represented. The usage of these items can be explained in some examples. Let us consider the statement "Every human is mortal". With the universal quantifier, variables, and connectives, we can have:

$$\forall x(human(x) \rightarrow mortal(x))$$

This expresses: "All x, if x is human, then x is mortal". From this example, we can see that we can easily express and understand the inference structure and inference rules inside the statement by FOL. FOL has great advantages in mathematical

expressions. It is also the main method for expressing knowledge which contains axioms and rule inferences [4].

The main difficulty in NLP is ambiguity. Researchers continue to think of better methods to describe or express the meaning of language. To make the computer understand the language better, it requires proper formal representation for each sentence from natural language. Also, it requires the ability of connection and correlation between different sentences. The disambiguation can be achieved by explanations from other related sentences. Production System is designed to well represent the language and make a logical connection between different production rules.

Production System, proposed by Chomsky in 1956, is a well-known method because it simulates human logical thinking as a structured procedure. Generally, a human analyzes a problem by using the learning experience and knowledge to draw some internal decisions, and then a human recursively applies some suitable rules in the internal decisions to arrive at a final decision. A Production System uses three main components: database, a set of rules, and a control system to simulate a human-analysis procedure. This database is different from a regular database concept. It is a collection of known facts, intermediate status, and final conclusions. The form of a rule in the set of rules is

$$Ci \rightarrow Ai$$

The Ci is the condition part. The Ai is the action part. The whole meaning is IF $\langle$Condition Ci$\rangle$, Then $\langle$action Ai$\rangle$. Condition Ci determines whether a rule should be triggered, and Action Ai determines what should be done after triggering of this rule. Control system decides when the rules should apply to the database. Also, it

needs to solve the conflicts when several rules are applicable at the same time. The basic procedure of a production system is shown below with pseudo code.

Input: Database, a Set of Rules, Control System, and User Interface

Output: result for the user's request

Match the conditions in rules with User Question

WHILE(hasMatchedRules)

IF numbers of matched rules >1

Conflict Resolution : Choose the most suitable rule to apply

Apply : apply the chosen rule; update the database

IF find the answer for user's question

Stop;

END

Production System has two different inference methods: Forward Inference Chaining and Backward Inference Chaining. Forward chaining is a data-driven inference, which starts from an initial state, and then matches the conditions (IF$\langle ... \rangle$), and execute the actions (THEN$\langle ... \rangle$). After running the recursive procedure, we can arrive at the final result for the user's request. Backward chaining is a goal-driven inference. It starts with the final result and then looks for matched conditions as sub-goals. This recursive procedure runs until initial facts are obtained.

Another traditional method is Frame, a structured data collection. Developed from semantic networks by Marvin Minsky in 1975 [4]. The Frame has properties and values and different frame blocks linked with a value, which fits the requirements of deduction reasoning. Based on the Frame theory, Marvin Minsky built a learning

Table 1.1: Knowledge representation methods and applications

AI Stage	Stage 1, 2 (1952-1970s)	Stage 3 (1970-1980s)	Stage 4 (1980-1990s)
Knowledge Representation	Production Rule IF<...> THEN<...>	First Order Logic (FOL)	Frame
Representative AI Application	Game Player, Expert System	Theorem Prover, Neural Network	Word Sense, Database, Paragraph Reading

machine, the first semantic network simulator, with 40 agents and a success reinforcement system. Because of Frame's advantages in hierarchy and encapsulation, it has become one of the most popular human cognition theories [30].

In general, the knowledge representation methods were developed with the technology revolution in AI. With different knowledge representation methods, more and more different AI applications are created. The summary of development is shown in Table 1.1.

Derived from Frame theory, Dr. Kam Hoi Cheng created A Learning Program System (ALPS) based on an Object-Oriented Paradigm [10]. The goal of ALPS is to learn all kinds of knowledge and communicate with humans in a natural language. The decision question answering in this thesis is built beyond the ALPS [20] platform. ALPS is a knowledge learning system, which simulates human learning, thinking, and communicating procedures. It already has some basic components of natural language processing, including learning the English language [20], parsing and understand simple English sentences [21, 29], learning grammar [22], word-sense disambiguation [23], and question answering [19]. The current functions within ALPS provide a solid foundation for new function development. In this thesis, we have

developed a knowledge-based logical inference to answer a question with a notion adjective. Our implementation uses ALPS existing classes which built a reliable and flexible natural language framework [2]. Our solution is designed in accordance with the object-oriented principle [25], which facilitates extending new interfaces or new functions. It also simulates human learning and growing procedures, meaning that an increasing knowledge can be learned and memorized while more functions can be added with enough knowledge for inference and actions.

1.3 Motivation and Problem Definition

In ALPS, Question Answering is a key basic component. Question Answering (QA) is to automatically decide the answer to the human's input question, which is one of the key problems in AI and NLP applications. A better solution for QA can significantly improve the human-computer interaction. However, computers cannot answer humans questions well, especially open-domain questions. The potential to provide a better solution for QA motivates us to improve sub-problems in the QA area.

A whole QA system consists of four parts: input sentences parser, knowledge representation, information retrieval (IR), and answers generation from retrieved information. In general, three methods can be used in developing a QA system, an IR based method, a knowledge-based method, and a hybrid method [15]. In the following discussion, We will define each method and analyze its merits and limitations with well-known examples.

(1) IR-based QA: An IR-based QA system first detects keywords in input question, then it retrieves the most related documents, and finally it extracts the information for generating answers and ranks the answers to the user. Famous IR-based QA applications include IBM Watson and Google.

(2) KB-based QA: A KB-based QA system needs a knowledge representation step. With the representation, text knowledge can be structured to a logical representation. Answers are generated by logic searching through the knowledge base. A famous KB-based QA is Apple Siri.

(3) Hybrid QA: A Hybrid QA system is a combination of IR, KB, and NLP linguistics methods. It uses a knowledge representation method in KB-based QA, and it also uses IR technology to find matched resources. Eventually, it will evaluate the relevance of an input question and give a rank of possible answers. A famous Hybrid QA application is IBM Watson.

Google is an IR-based QA system. It generates answers for each input searching query by retrieving a lot of documents, web pages, and datasets. Google provides answers for the searching query by ranking the linked resources. For some fact-based questions that have only one or limited correct answers, it can answer with high accuracy. The pipeline of IR-based QA applications, like Google, is modeled as the form below:

Question Recognition (recognizing the type of question) $\longrightarrow$ Question Rewriting (filtering out the keywords and highlight the categories of question) $\longrightarrow$ IR Searching (extracting the related documents) $\longrightarrow$Answers ranking (ranking N-best matched answers)

The IR-based QA application, like Google, has huge advantages in retrieving matched resources from structured data and unstructured data. Also, it can give an excellent rank for the retrieved information with PageRank algorithms. However, it is not real AI by just providing the links that may answer the user's questions. Besides, the IR-based QA application is difficult to explain reasons of the ranking methodology because the ranking needs complex mathematics. These algorithms do not simulate human thought and human communications very well. Humans tend to answer a question directly with the most suitable sentences extracted from memorized knowledge. Moreover, in communications, humans do not need complex advanced mathematics to help them make sentences. The kids without mathematics knowledge can communicate and answer the questions. These flaws motivated us to develop a method that simulates the human logical reasoning procedures for QA.

Another pioneer KB-based QA application is Apple Siri. Compared to Google, Siri focuses on increasing the machine's communication ability. It can give helpful, direct, and short answers for some simple daily questions, such as "Do I need to bring an umbrella in my way to campus now?" and "What is Xs phone number?" The questions that Siri can answer correctly have clear semantic categories, such as time, location, and person. The high-level model of KB-based QA is formed as:

Representing Question (reorganizing the question with name-entity form) $\longrightarrow$ Searching Answers (only searching in structured domain data in KB) $\longrightarrow$ Generating answer with certain logical reference rules (answering question and explaining the reference logics)

Apple Siri is a very thrilling product because it can communicate with humans.

It performs very well in answering simple daily fact-based questions, giving recommendations, and operating the iPhone with input queries. However, it is clear that Siri's accuracy still needs to be improved. The fact-based questions can be parsed clearly by machines with the name entity method [30]. After recognizing the question type and question keywords correctly, Siri can generate answers with the simple reasoning in the knowledge base. Siri highly relies on the third-party dataset. For example, Siri will execute reasoning in the Yelp dataset when asked for the restaurant related questions, and Siri will go to Wolform Alpha when asked for some general fact questions. Such heavy dependence on many famous third party datasets brings some disadvantages. When Siri answers questions from users, it will use existing logics inside the data of third parties. Think about how users ask Siri for the way to a library. After parsing the input, Siri will connect to Google map, and next Google map will answer the shortest route to the users as the default. However, this procedure does not simulate human-reasoning very well. Human-reasoning procedure for this kind of question starts with recognizing the different situation and defining a different purpose of the question. It is possible that people ask the route to a library to borrow a rare book. In this situation, just answering the nearest library route is not suitable. The disadvantages of Siri motivated us to improve the simulation of the human-inference procedure. A better simulation of the human's inference could improve the QA products accuracy.

IBM Watson is a complex open domain QA application, which is a significant hybrid QA product [16]. In 2011, it defeated humans as the winner in the Jeopardy Game. Watson is different from Google and Apple Siri because it must give only one

answer with the highest confidence to each question. It cannot provide lots of links related one question, and it needs much higher accuracy than Siri. Otherwise, it cannot beat the human experts. To follow the rules of the Jeopardy Game, Watson is not allowed to connect to the internet, which requires Watson to store more data in the machine. The pipeline of Watson's working procedure was summarized [16] and shown in Figure 1.3 as a representation of hybrid QA applications.

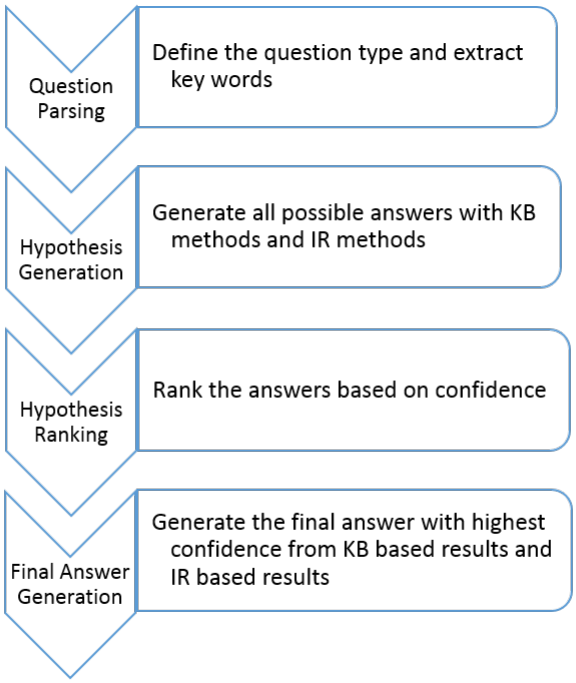

Figure 1.3: Pipeline of hybrid QA application.

IBM Watson has a huge advance over the expert systems in the last decades. The success comes from several aspects: (1) ability to quickly search big data, (2) the use of wise rules and strategies inference, (3) a good combination of many technologies in QA, and (4) the ability to understand unstructured text data. Additionally, the success of IBM Watson DeepQA can be used in other domains whose features are

similar to the Jeopardy Game, like health, science, and education. However, it is not as robotic as Apple Siri. It needs human experts to define some rules and strategies in advance. In other words, Watson can only behave well in Jeopardy. It cannot easily change to other problems without human experts defining the smart rules in other domains. Apart from this, it is too sensitive for question input, because Watson's parsing strategies only focus on Jeopardy-style question information [17]. Thus, Watson is not a general AI QA system. If it were to be extended to other domains, it would need a large import of intelligence from human experts.

From the analysis of current well-known QA products, we can see that each QA product has its limitations. The purpose of this thesis is to develop a more general QA reasoning method with following the three features: first, it can answer the question with a direct answer; second, it can deal with general and frequently asked questions during conversations; lastly, it should be easy to change control parameters that modify answering reasoning strategies in different situations. In general, We are motivated in creating a method that can improve QA accuracy in general conversions and improve its robotics in various situations.

With these motivations, We focus on the problem of understanding and answering a decision question involving an opinion adjective. The decision question, to confirm given information in question, asks whether a subject or a target has certain aspects [18]. There are many different types of decision questions based on its purpose. One type of decision question is to ask whether the target is a classification, like “Is Zoologist a Scientist”. Another type of decision question includes an adjective, like “Is John an important human”. In addition, some decision questions

may be nonsense, like “Is the weight of John colorful” or “Is computer an important book”. In this thesis, we also intend to check the sensibility of a question and detect those decision questions that are nonsense. Both the “Is target classification” and nonsense-type questions are well solved with logic reasoning methods [18, 19].

In this thesis, we develop a method to answer the decision question with an opinion adjective, like “Is John an important human”. In this sentence, “important”, the opinion adjective, is named as description because it is to describe a human's opinion or thought. This sub-problem under decision questions consists of a target, an opinion-describing adjective, and a corresponding noun. The answer for this sub-problem would be “Yes” or “No” followed by a brief explanation. The answer is decided by the comparison between general common knowledge for this question and the related information retrieved from the target. For example, we know that humans often determine whether a person is important or not by his or her occupation. Thus, to answer “Is John an important human”, we will retrieve John's occupation from his profile. And then we would compare his occupation with the common sense knowledge about an important occupation to give a final decision. It is meaningful to solve this kind of decision question since it is a direct and effective way to provide feedback about the characteristics of a target. For example, we can ask “Is this review is a positive review” when we are shopping online rather than reading all words in every review. Currently, products on Amazon or Yelp have thousands of reviews, which are meaningful to customers in decision making. However, it is hard for customers to read all reviews. Therefore, if a customer service machine can answer users's question directly or briefly summarize all reviews, the online shopping

experience would improve. Also, it is an advancement of the QA system since it can answer some questions with an opinion. In [19], a method to answer fact-based decision questions is developed. In [18], they solve the problem and justify the sensibility of a decision question. However, answers for opinion decision questions are not based on the target satisfying the criteria of the adjective. The work in this thesis extends both of them to answer an opinion decision question, and base the decision on the target satisfying a criterion to use the adjective.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the analysis of data related to the problem. Specifically, the details of knowledge about notion are presented. Chapter 3 gives the details of our Logical Inference Algorithm. The algorithm uses the organized data about notion to infer the answer to a decision question with a notion adjective. Chapter 4 describes the experiment to test the algorithm in different cases. Chapter 5 gives a brief description of our work, conclusion, and future work.

Chapter 2

Data Collection for Notion Knowledge

In this chapter, we will discuss the knowledge needed in ALPS to answer decision questions involving an opinion adjective. Since ALPS only uses logic inference to solve the problem rather than the statistical methods, it is not necessary for ALPS to prepare huge training data sets. We will use an example to decide the necessary knowledge and data needed to be collected. Figure 2.1 shows a paragraph describing the biography of Grace Hopper as an example to reveal what knowledge is needed and stored.

In Figure 2.1, useful knowledge about different aspects of the target can be extracted from the raw text. Each aspect and its value may be stored as a structured pair in the knowledge base of the computer system. The stored knowledge can easily be retrieved when needed.

Since data organization decides the accuracy of the inference procedure, it is

Q: Is Grace Hopper an important person?

Retrieved Paragraph in Grace Hopper Biology [24]:

" **Grace Hopper** was born on December 9, 1906, in New York City. She was one of four women in a doctoral program of ten students, and her doctorate in mathematics was a rare ***accomplishment*** in its day. In 1949 she joined the Eckert-Mauchly Computer Corporation as a Senior Mathematician. **Grace Hopper** received many ***awards*** and for her ***accomplishments***. In 1969, she was awarded the first ever Computer Science Man-of-the-Year Award from the Data Processing Management Association. In 1973, she became the first person from the United States and the first woman of any nationality to be made a Distinguished Fellow of the British Computer Society. "

Structured Information extracted from paragraph:

<*Target*: **Grace Hopper**>, <*Education*: doctoral>, <*Accomplishment*: doctorate in mathematics>, <*occupation*: senior mathematician >, <*Awards*: Naval Ordnance Development Award, Distinguished Fellow, ...>

Figure 2.1: An example on the knowledge collected from a biography.

important to analyze the required data related to a notion to be stored in ALPS. We summarize the required data by analyzing human's logic inference procedure. Humans decide the answer to an opinion decision question based on remembered examples in the brain and retrieved knowledge about relevant aspects of the target. With memorized examples, humans can match examples with the target in the question. The knowledge about different aspects of the target provides the support for logic inference. People's inference process normally starts with matching the target

with examples. It is simple to match them exactly. If examples are not matched exactly, people still match target with examples by similarity. To decide whether the target is similar to an example, we need the knowledge about the reasons why each example is correct. If people cannot get the answer by comparing examples, people will continue inference by using other aspects of the target. The aspects are retrieved based on people's criteria. Therefore, in ALPS, we need to organize a list of factors as people's criteria for the notion.

To answer an opinion question, if the target is an example known by a person, then the answer can be readily determined to be "yes". For example, if Grace Hopper is known to be an important human, then the answer is immediately available. However, if the target is not a known example, then the person will need to make the decision based on his/her own criteria. Since the criteria of one person can be different from another person, each person may have his/her own opinion. Each criterion usually spells out that the target should have a particular value for a specific aspect in order to satisfy that criterion. In this thesis, although an opinion may have many criteria, we will only be concerned with opinions that can be decided even when only one criterion is satisfied. For example, if one of the criteria for an important human is to have an important award, then Grace Hopper would be considered important because she has received several important awards. There is no need to continue even though she may have satisfied other criteria to be an important human.

Accordingly, our solution will first build some basic knowledge on an opinion of a notion. These include a list of criteria and a list of examples in order for the opinion to be true. To answer the opinion question, our solution will first check

whether the target is one of the memorized examples. If the target is an example, then the answer to the question is "yes". Otherwise, our solution will then check the list of criteria, going through them one by one. Once a criterion is verified to be satisfied by the target, our solution will stop and give a "yes" answer. If the target fails to satisfy any criterion, our solution will give a "no" answer.

2.1 The Notion Class

As pointed out earlier, this thesis focuses on how to decide the answer to an opinion of a notion using ALPS as our platform. To answer this kind of question, we need to know how to represent knowledge with an adjective of the notion that will express an opinion, what relevant knowledge is needed for the inference, and how to organize those data in ALPS knowledge base. ALPS organizes data into different classes to express an opinion of a notion. ALPS also requires relevant knowledge of that notion. The class, notion, is developed to store relevant knowledge. For each notion, it has many adjectives that can be used. Some adjectives convey a positive sense of the notion, while some convey a negative sense. For example, the notion of importance, the adjective important conveys positively for the notion importance, while the adjective trivial conveys negatively. We will use an object of the class scale, a class of ALPS, to store all the adjectives that can be used to express an opinion of that notion. The relation between notion and scale in shown in Figure 2.2.

The adjectives in a scale are maintained as a linear list. Since there may be several adjectives describing the same sense of a notion each with a different degree of emphasis, insertions of additional adjectives into the scale are allowed. Three

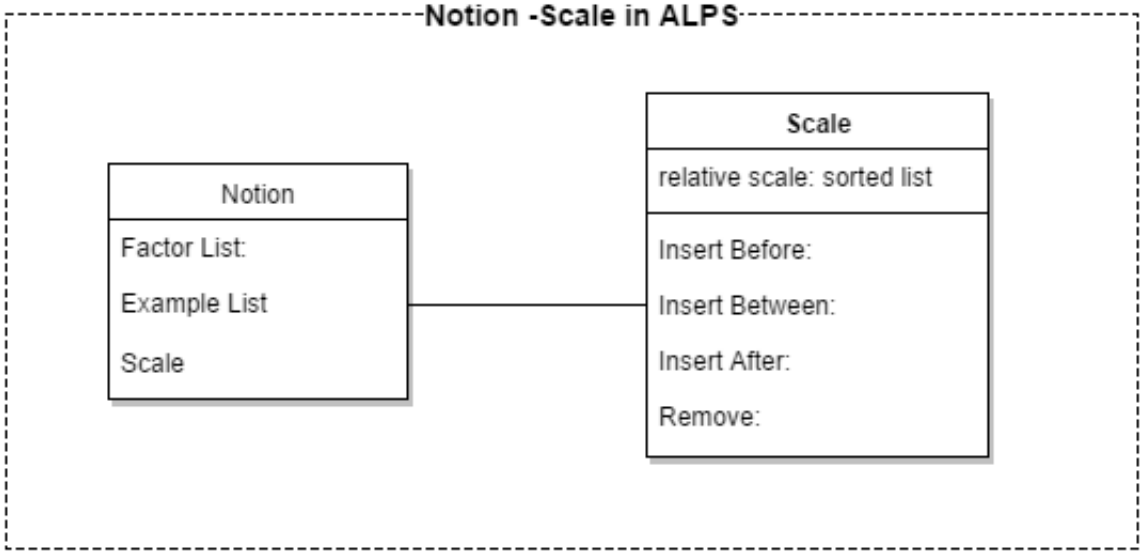

Figure 2.2: Relation between notion and scale in ALPS.

operations: "insert between", "insert before", and "insert after" are provided by the class to accomplish these actions. The use of a scale object allows the system to properly maintain the relationship among the various adjectives describing the notion. It also allows the system to decide whether a given word is an adjective for the notion.

2.2 Factors List and Examples List

Humans may be asked different kinds of questions. One kind of questions is a factual question. The answer to this kind of questions is based entirely on some known facts, and is simply a "yes" or "no" answer without the need to give an explanation or a reason. For example, for the question "Is John an Oscar winner", the answer is "yes" if John has won an Oscar award, and "no" if he did not. Another

kind of questions is an opinion question, e.g., “Is John a charming man”. Each human may have his/her own opinion, and that may be the same or different from another human's opinion. In addition, even if the opinion is the same, different humans may have different reasons to support their opinions. Therefore, a human will normally answer the question and followed that by an explanation. As a result, the answer to an opinion-related question needs to be generated by reasoning from multiple data sources [26]. The explanation could be generated through the reasoning linkage in the found data source that satisfy a criterion.

Given a sentence which is a decision question, the parser in ALPS can parse it correctly and determine its type easily [18]. Several logical components can be identified, namely, the target, the description, and the reference. For example, for the question “Is John an important human”, the target is “John”, the description is “important”, and the reference is “human”. If the reference is omitted in a decision question, ALPS will use either the category or the parent category of the target as the reference. For example, for “Is John important”. “human” is used because it is the category of John. Once these logical components have been identified, one may start to decide the answer to the question.

As pointed out earlier, a human can answer questions about a notion only after that notion has been learned and understood. A notion is considered learned and understood if the following two conditions are satisfied. First, adjectives for the notion must be known. Second, one must be able to decide whether an adjective has been used correctly or not. Knowing the adjectives can easily be achieved by storing a scale that contains a list of adjectives usable for the notion. To know if

an adjective has been used correctly, one way is to check with known examples. For instance, the adjective "important" in "John is an important human" is considered to be used correctly if John is known as an example of "important human". Another way is to verify the factor that justifies the use of the adjective. For example, if one factor for an important human is an important award, then John can be considered to be important if he has received a Nobel prize. Note that the adjective "important" can also be used to describe the aspect "award". Obviously, the examples and factors for "important human" are different from "important award". For instances, some examples of "important human" are Albert Einstein and Grace Hopper. Some factors required to be an "important human" may include having some great accomplishment or having received some important awards. On the other hand, Nobel prize and Academy awards are examples of an important award, while a factor to be an important award is a broad recognition. Thus, for each aspect an adjective can describe, the notion needs to store two additional lists, one for the examples and the other for the factors.

We will collectively call the list of examples and the list of factors as the criteria. A criterion can only be added if the adjective is known to be usable for the notion. In other words, an adjective for the notion has to be taught before it can be used for an aspect. Since humans learn knowledge over time, additional examples and factors can be added.

The list of factors is maintained as a linear ordered list. The reason is that humans normally consider some factors are more important than other factors although any one factor would be sufficient to justify the use of the adjective. Consequently,

the functions that add additional factors are "insert before", "insert after", and "insert between". This implies that at least one factor must be taught initially so that additional factors can be added later.

The list of examples is stored as a map provided by C++ using the example name as the key so examples can be recalled based on their names. Since we want to provide a reason to justify our answer, if the answer is decided by using an example, we need the reason to be available. Besides just storing the names of the examples, we also need to store the reason why the example is correct. However, based on the same fact, humans may give different reasons. For example, given the fact that Albert Einstein discovered the general theory of the relativity, there are many possible valid reasons that a person may give to justify that Albert Einstein is important. These include, "He had great achievement", "He discovered the general theory of relativity", and "He made a scientific discovery". The first reason is a high-level reason, which uses one of the factors for an important human, namely "have great achievement". The second reason is a low-level reason, which uses the detail fact directly. The third reason is a middle-level reason, which uses a category that is a special case of a high-level reason and it includes the detail fact as an example.

Out of the three levels of reason, we choose to store both the high-level and middle-level reasons. The high-level reason is an adequate justification if one is not interested in the detail. For example, one would be satisfied that a person is important if he/she has great achievement. As for the middle-level reason, it may be used to decide on other cases which are similar to the current example. For instance, one can decide that Newton is also important, similar to Albert Einstein, since both have

scientific discoveries; Einstein discovers relativity while Newton discovers gravity. By finding similarity between the target and the known examples, we may be able to draw the correct conclusion without the need to go through the list of factors. When the problem domain is open, there is no universal method to find similarity. Some researchers use the past and known reasoning to offer clues for handling unknown cases [27]. We propose to use category and hierarchical relationships to find similarity.

Chapter 3

Algorithms Implementation

3.1 Object-Oriented Design in ALPS

ALPS [20] is designed using Object-Oriented design (OOD) methodology. Currently, ALPS is able to learn a subset of English grammar and use it to parse and understand sentences based on the learned grammar. Given a thought, it can also write an English sentence according to the learned grammar. An example of items in ALPS is an implementation of *Class*, *ClassDef*, which encapsulate attributes and methods of a group of similar objects. *Classes* are basic and important units in ALPS since all objects are instantiated from classes. The class in ALPS represents the knowledge of a unit in the natural language. Each class in ALPS is named with a noun, including the knowledge for a category, such as descriptions, aspects, operations, and structured knowledge. The Unified Modeling Language (UML) is used for modeling classes' structures in a software design. A simple example of UML for ALPS design is shown in Fig.3.1.

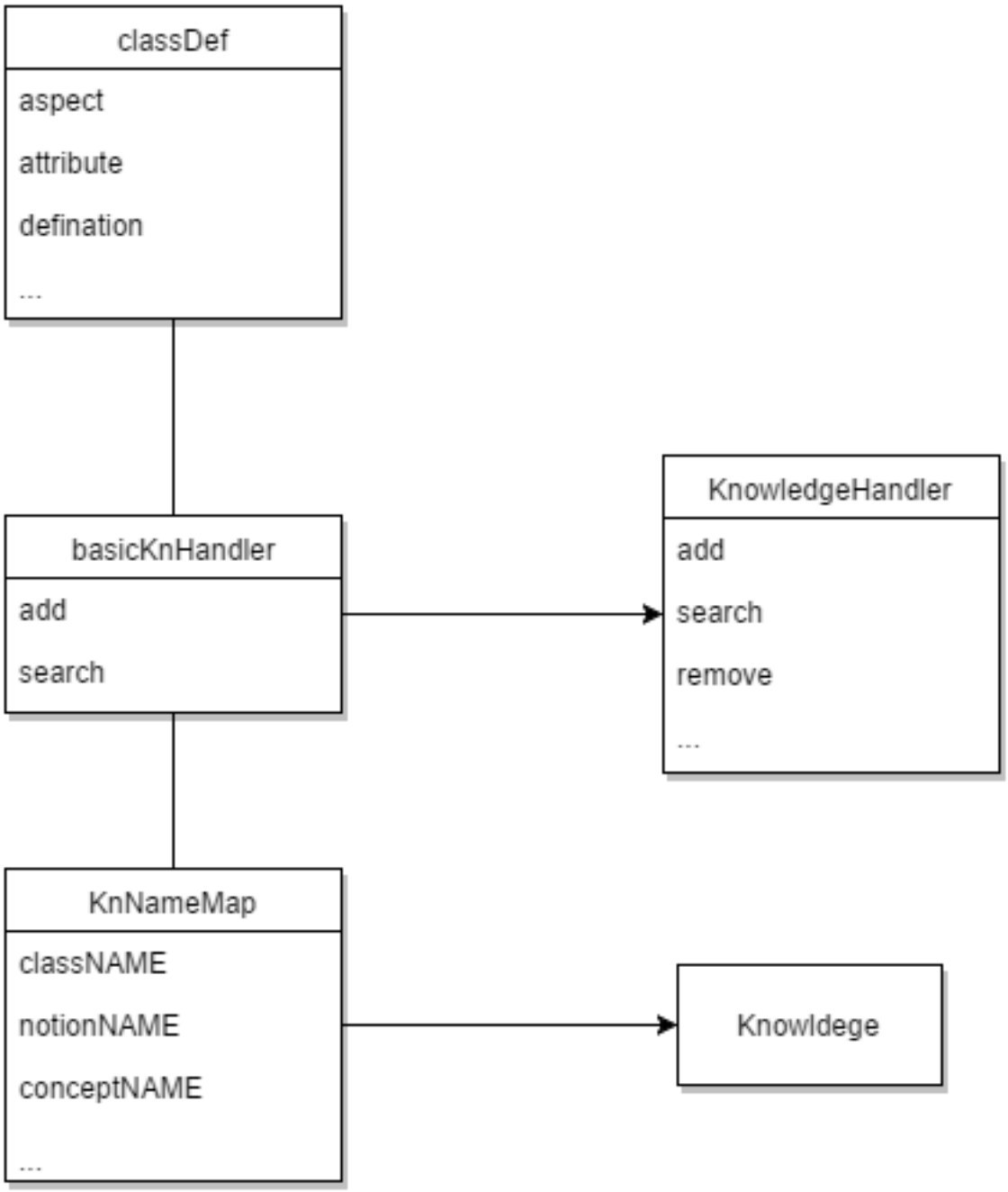

Figure 3.1: A simple example of UML for ALPS.

Fig.3.1 shows the related components in ALPS to add and search a class. The *ClassDef* is used for creating a class. The *basicKnHandler* maintains instances of a class. It can add a new class instance to the respective list. The *KnNameMap* stores a special kind of knowledge to a list.

The base class in ALPS is a class called knowledge. Every kind of knowledge to be learned by ALPS is modeled by a child class of knowledge. For example, the class concept is to model knowledge about physical concepts that can be measured. The knowledge base class provides several polymorphic functions to be used for all knowledge subclasses. Some functions allow the addition of new knowledge to the knowledge base of ALPS. The *create* function allows objects of that class to be created based on information provided. One *add* function adds the newly created knowledge object to an appropriate object responsible for maintaining all knowledge objects of that class. Several other *add* functions allow knowledge to be added to an object of a class. Based on the provided information, hierarchical relationships among different knowledge objects may be established. The *change* function changes an existing knowledge about an object of the class. Several *search* functions are provided to find sub-knowledge within an object using different amounts of provided information. The *isA* function allows ALPS to decide whether the hierarchical relationship between two pieces of knowledge is true or not.

In ALPS, a language class is provided to allow the system to learn natural languages, An object of the language class has been created to learn English. It allows details of the language to be acquired by the system, such as the grammatical structures, roles, and rules of the language. By using the learned grammar, ALPS is

able to parse English sentences and translated those into thoughts. Different kinds of sentences have been translated into different kinds of thoughts. For examples, a declarative sentence has been translated into a declaration, while a decision question has been translated into a decision question. Based on the different kinds of thought, ALPS will respond differently. For instance, if the given sentence is a question, the thought created will first attempt to find the answer, then create a thought for the answer, and finally, will use the learned grammar to produce a response sentence. To facilitate the recognition of the knowledge corresponding to the given words in the English sentence, every subclass of knowledge has to implement the polymorphic function, *parse*. The *parse* function of each class is responsible for matching the given English words to the knowledge objects where those words are the names of the objects.

3.2 Notion, Factors List, and Examples List Implementation

In this section, we discuss the methods to implement different components in teaching notion knowledge to ALPS. To implement whole notion learning and answering process with those items in ALPS, we need to create a user interface for collecting data from users. Next, with OOD principle, different types of data should be stored by different collections and with different data structures in C++. Moreover, after organizing the data, the answer to a user's input question should be generated from inference through data. Consequently, users can teach ALPS the

knowledge about a notion and their opinions in different aspects through the user interface. After that, ALPS can use the learned knowledge to infer answers to the user's questions.

The notion user interface is one component in ALPS that allows users to communicate with ALPS for their own opinions about notions. Each user can teach ALPS notion knowledge based on their special understanding. The user interface for the notion class contains multiple options for users to teach notion-related knowledge to ALPS. The options include "create notion", "add scale", "add factor", "add additional factor", and "add additional example", which are designed based on the earlier analysis for notion class data collection. The options are added to ALPS with ioObj class in ALPS. The ioObj class in ALPS is designed for collecting information from users to perform operations on knowledge objects. With the ioObj class, notion names and notion criteria can be collected into ALPS. As for "add factor", a list of related factors and examples should be taught by the users. Therefore, we need to choose another class, getSeqIoObj class for information collection. The getSeqIoObj class, inherited from ioObj class, can read an input string to ALPS, which meets the requirements for adding a list of factors and examples.

Factors list is one key component of the notion class which contains the criteria for one description of the notion and the reference. For example, under the notion "importance", one factor list may be for "important human", which contains "important" description and "human" aspect. The user needs to teach ALPS a list of factors that are regarded as important criteria to justify this classification. As pointed out earlier, the factors list should store factors in the order from the most important

factor to the least important factor. Therefore, after getting the factors from users, we choose to use one class, knNameList in ALPS, to store the collection of factors. The knNameList is designed to store elements in a total-order sorted list, which fits our requirements for factors list. To insert the factors from users to one instance of the knNameList class, we can use the methods developed in knNameList, such as "insert after", "insert between", and "insert begin". When inferring the answer to a question, the knNameList class also provides retrieval methods for searching a piece of knowledge, like "search before" and "search after". The retrieval methods can help us to traverse the factors list in the correct order.

The examples list is another knowledge component in the notion class for users to teach ALPS knowledge about the notion. Examples in the notion are used for matching with the target in the question, which is required to be retrieved efficiently. Consequently, the knNameMap class in ALPS is a suitable collection for storing the list of examples. The knNameMap class can efficiently get access to the knowledge information with the knowledge's name as index. As pointed out earlier, besides storing the unique name of each example, we also need to teach ALPS the reason the example is correct. Since we need to store the name of an example and its associated reason, the add function should contain at least two items, the key to the knowledge and the contents of the knowledge. In knNameMap class, the add function is constructed by a name to locate the knowledge and a pointer to the contents of the knowledge. Consequently, for each example, we can create a piece of knowledge with the example's name and reason. The add method from knNameMap class can insert the generated knowledge to ALPS with the examples name as the key and a

pointer to the created knowledge. When retrieving a stored example, we can also easily get its reason.

Apart from implementing each component in notion class, it is necessary to check the validity of users-teaching inputs for building a reliable knowledge base. One validation is to check whether the teaching is sensible or not for the notion. In this validation, we need to search for the existence of the adjective in the notion. If the search for the adjective fails, an unknown exception is thrown, and the teaching is not added to ALPS. For examples, for the notion "importance", "important human" can be added because "important" is a known adjective for the notion, "importance"; however, "beautiful human" cannot be taught since "beautiful"does not exist in the notion of "importance". Another validation is to check whether the factors in a factors list have already been taught because ALPS cannot use unknown knowledge. This validation operation is done during the inference procedure since users can continue to add new factors to the notion before answering the question. When inferring an answer, linkages between different factors can be constructed. To link with a new factor, our methods validate if this factor has been taught. If the factor is an unknown factor, our methods will stop further heuristic inference from this factor.

In general, the whole notion class and the related implementation extends the ALPS' question-answering ability. With this implementation, ALPS can answer a decision question with an opinion adjective. The implementation uses different kinds of classes to organize the data and search for knowledge while inferencing. The attributes and operations in the notion class are summarized in Fig 3.2.

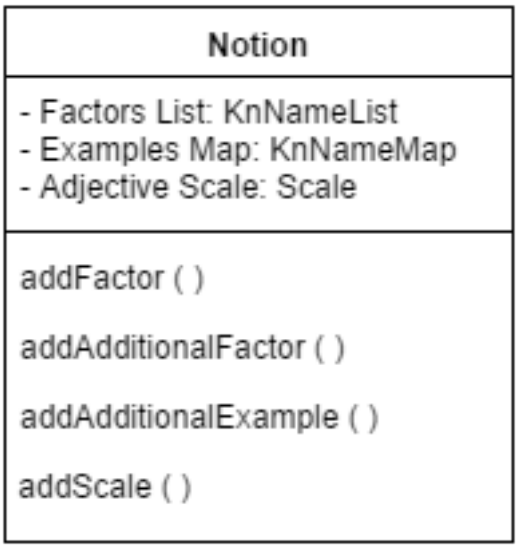

Figure 3.2: The UML for the notion class.

3.3 Logical Inference Algorithm

3.3.1 The Reason for Logical Inference

ALPS intends to simulate human deduction thought and judgment. We use the term *notion* to describe abstract concepts which humans use to explain their own judgment. Individuals have different criteria for the same notion. For example, people will have different opinions about happiness, since it is more related to their own personal history, values, and goals. Some people tend to put salary in the first place when considering happiness, while others attach more importance to their beliefs. Thus, abstractness is difficult for a machine to apply simple statistics to make a judgment, humans have rather different criteria for the same notion.

To solve this issue, we extend the prior work of classification inference [19] and design a process to better simulate human logical judgment of a notion. The human

judgment process of a notion is based on a set of criteria, including related factors and examples. The factors reflect human values, goals, or personalities. The examples are human knowledge pools. The criteria are implemented by the factors and examples lists. The factors list is designed to be the trigger to search for representative criteria of a specific notion. The examples list is to provide reliable and specific cases for each factor. For example, ALPS can tell Thomas Edison is an important person if the user believes the "important invention" is one of the factors for "important human"; and the user knows that Edison's light bulb invention is one example of the important inventions. However, if the user believes "important invention" should not become a factor of "important human", then Edison, in this case, cannot be regarded as an important human. In another case, the user believes the "important invention" is a factor for "important human", but Edison's light bulb is not an example of "important invention". Then the system will go to the next level of the user's criteria, the influential result, a factor of the "important invention". If the light bulb invention had an influential result, the light bulb invention would be regarded as an important invention. Consequently, Thomas Edison is identified as an important human based on this user's criteria. We believe that this process can better imitate human judgment system. The factors list will be the logical line to trigger the knowledge pool in each individual judging scenario, and the program will follow the factors list to search the corresponding criteria.

A simple example for the notion judgment in ALPS is shown in Figure 3.3 and Figure 3.4. In Figure 3.3, we can see an example of factors linkages between different notions. Factors can link with each other because one factor can be a feature of

another factor. Under each factor, it contains a set of learned examples. In Figure 3.4, a special inference example is shown. With the profile of the target Picasso and the factors linkages file in Figure 3.3, the searching traversal paths for justifying whether Picasso is an important human are generated in Fig 3.4. As shown in Figure 3.4, if we cannot find the content of one factor, we will stop and prune this branch. The meaning of a factors list under a criterion is that those factors can serve as features of this criterion. If the target's profile has no content for the aspect of this criterion, it is nonsense to continue searching the criterion's factors list.

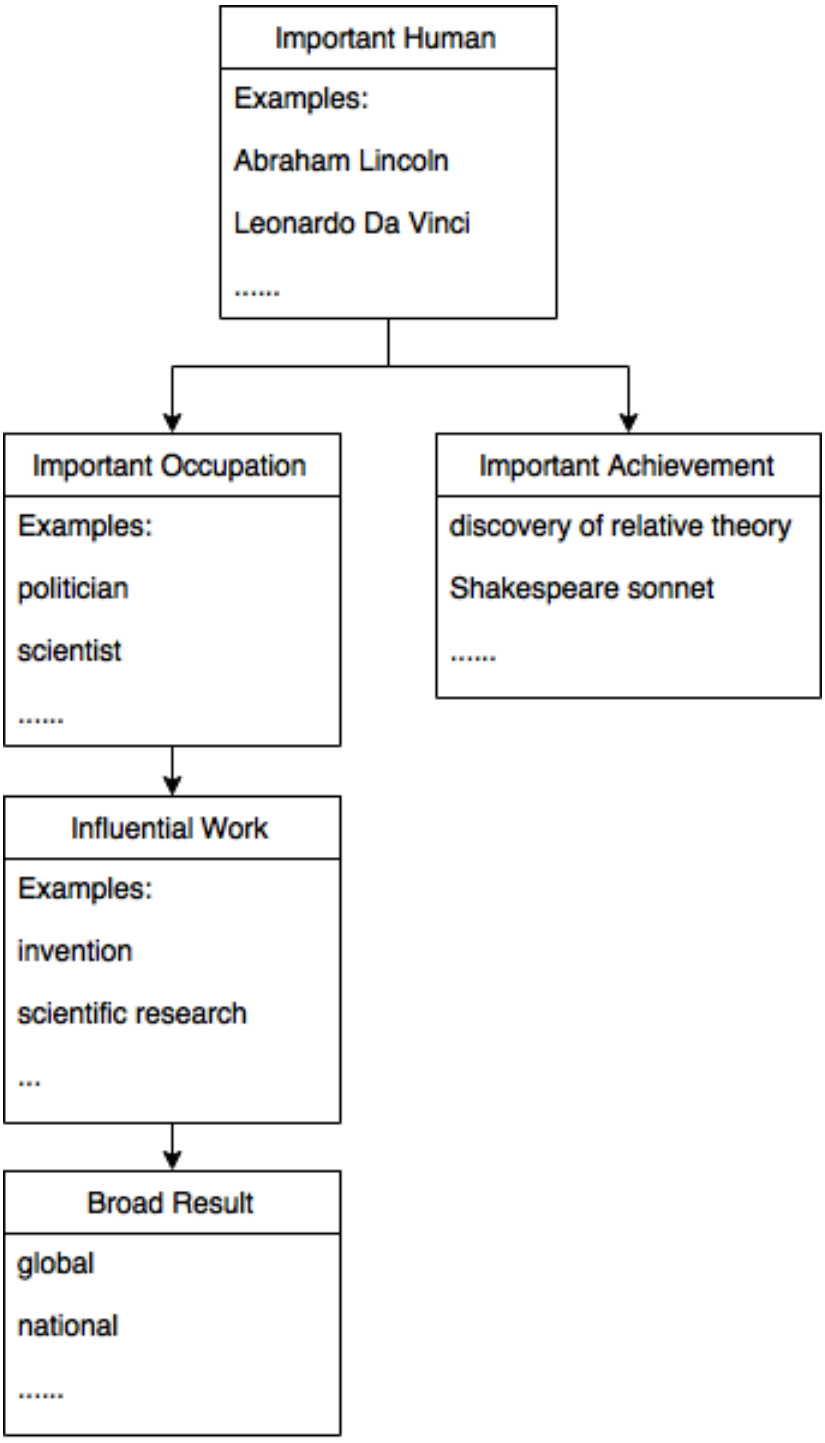

Figure 3.3: An example of factors linkages.

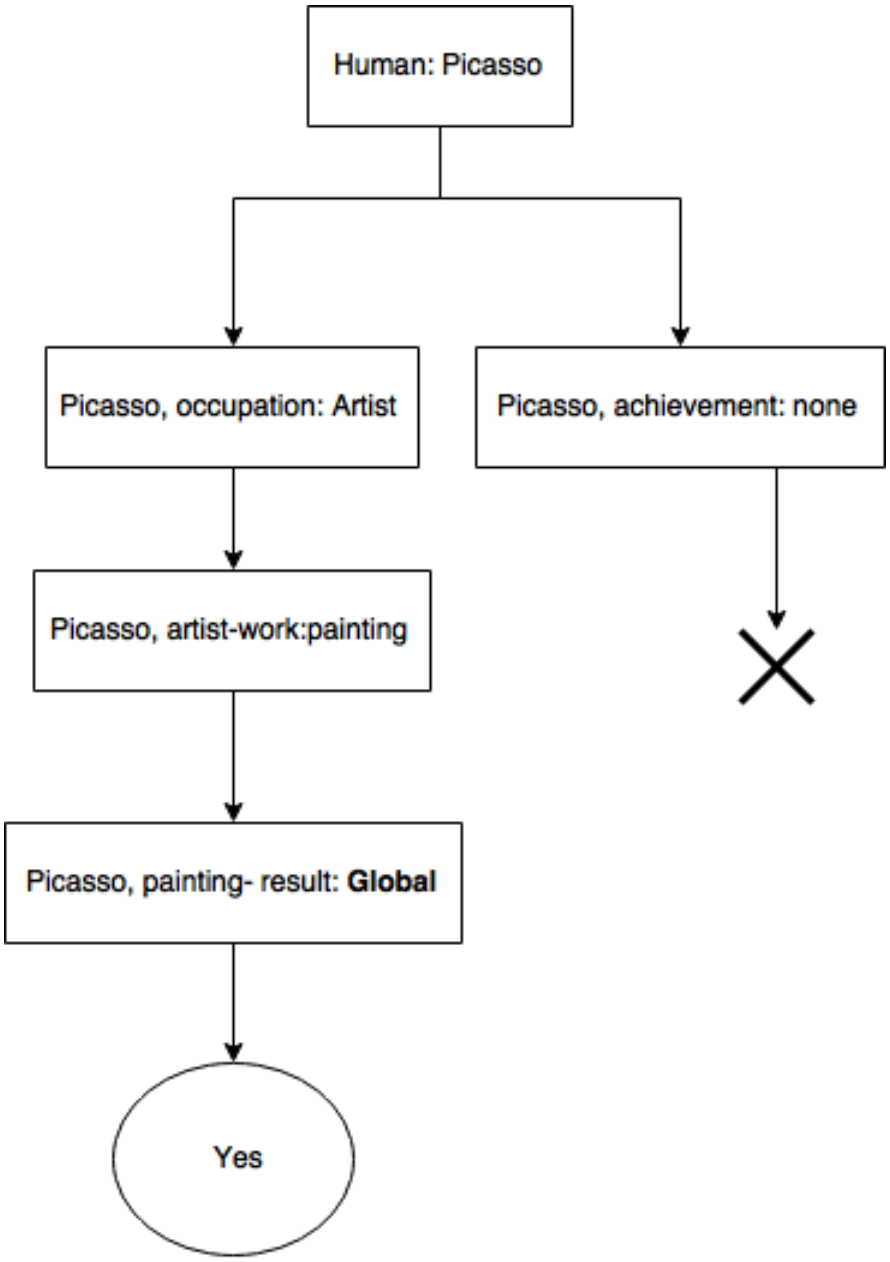

Figure 3.4: An example of searching procedure with factors linkages.

3.3.2 The Logical Inference Algorithm

In this thesis, the logical inference process follows a tree searching structure. The tree traversal is based on linkages between different factors. In each factor, a set of supporting examples will be stored. For example, the important achievement factor can have the "innovation of quantum computer", the "discovery of relative theory", and the "painting of Mona Lisa" as examples. Each example will contain a reason explanation. The answer is generated by inference through factors lists and examples lists. In general, the answer can be reached in two ways: a direct match or an example's categories match. The different inference methods are summarized in Table 3.1.

Table 3.1: Summary of the inference methods

Answer Inference Type	**Simple Explanation**	**Example** (Q: *Is target an important human*)
Direct Match	The search target is in the examples list	Abraham Lincoln is in examples. "Yes" answer.
	The special significant case has common sense knowledge	Edison invented the light bulb. Common sense: Light bulb invention has a tremendous effect. "Yes" answer.
Example's categories match	Match based on hierarchy information	Einstein has a discovery in physics. Hierarchy: Physics is a science. Category: Science. "Yes" answer.
	The category matches directly	Gandhi's occupation is a politician. One category in an important occupation is a politician. "Yes" Answer.

A direct match will occur if the search target or the search target's aspect already exists. The example's categories match occurs if one or more examples are similar to the search target. As shown in Table 3.1, the similarity between examples and search target is measured if the search target can match an example category or the search target's hierarchy can match an example category. For example, if the knowledge base does not have the "discovery of gravity field" as an example under important

achievement but it has the "discovery of magnetic field", the "discovery of gravity field" will also be regarded as the important achievement as both of them belong to the discovery of a science category. If the target cannot match with examples in both ways, the inference process will go to the next criteria level. If all criteria have been traversed and without a "Yes" decision, the algorithm will return "No".

The logical inference algorithm in our solution uses a Breadth First Search (BFS) schema. The algorithm is shown in Table 3.2. The criteria are traversed level by level with BFS. While linking to one factor in the traversal process, a comparison between the search target's respective aspect and supporting examples under this factor will be executed. Moreover, to avoid connecting to nonsense factors in the traversal process, each linked factor will be checked for its sensibility.

3.4 Answer Generation

A "Yes" or "No" answer is not enough in communication. Humans often explain reasons for their decisions. This is one of the major challenges for developing an opinion question-answering system [26]. In this thesis, our solution not only provides a "Yes" or "No" answer, but provides a supporting reason to the answer. Humans can have many different explanations for the same decision. The exact explanation depends on which criterion is satisfied by the target. Since the criteria used by different humans may be different, they may disagree on their answers. It is also possible that they arrive at the same answer but with different reasons to support their answers. The explanation provided by our solution depends on how the answer arrives. We use four different cases of explanations.

Table 3.2: Algorithm: Logical Inference Algorithm in Decision Question Answering

Algorithm 1: Logical Inference Algorithm

Data: target' profile, criteria and example list of notion, basic knowledge learned in ALPS

```
Initialization;
Create empty queue Q;
CheckSensible (rootFactor);
Q.enqueue (rootFactor);
while Q is not empty:
 currentFactor = Q.dequeue();
 checkSensible (currentFactor);
     for each example : currentFactor
      if (Match(target, example))
       return true;
      if (Similarity(target, example))
              return true;
     end
     for each factor : currentFactor
      if (target has aspect of factor)
              Q.enqueue(factor)
      else
              continue;
     end
end
return false;
```

The first case is the target of the question or its relevant aspect is one of the stored examples in the knowledge base. For example, assume that Abraham Lincoln is an example of "important human" and is known by ALPS. Then the answer to the question "Is Abraham Lincoln an important human" is "Yes", and the reason provided by our system will be "Abraham Lincoln is the exact example of important human". This reason will be given for all situations when our solution decides the answer by matching an example in the example list.

The second case is the target and its relevant aspects are not examples in the knowledge base, but a certain aspect of the target can match the reason for an example. In other words, one aspect of the target belongs to one category of the reason for an example. In this case, our system can give a reason that the target has similarity with an example. Assume that John is not an example of important human nor his occupation of politician an example of important occupation. As a result, neither the target nor a relevant aspect of the target matches the examples. In addition, assume that one example of an important human is Abraham Lincoln with the reason that his occupation is a politician. To answer the question "Is John an important human", our system compares John with examples and finds out that Johns occupation matches with the occupation of Abraham Lincoln. Thus, we can stop inference here and give a "Yes" decision with the reason that John is similar to the example Abraham Lincoln because their occupations match. This is a simple and direct example for using similarity to get a "Yes" answer.

The third case corresponds to the situation such that the given question is nonsense. In this case, our solution gives a "No" decision rather than going through

the logical inference linkage. For example, assume that Tom is a dog, but the question is "Is Tom an important human". Since the target "Tom" is a dog but not a human, thus the answer is "No" because Tom is not a human. In [18,23,28], they give a deep analysis of nonsense sentences, but we focus on generating an explanation here.

The fourth case corresponds to the situation such that the target does not satisfy any criteria to be an important human. Since no criteria are satisfied, our solution will search through all the criteria without getting a "Yes". As a result, the answer is "No" and the reason given is that we cannot find an aspect of the target to satisfy the criteria.

Chapter 4

Experiments

To demonstrate our algorithm works correctly, we created a simplified user interface to simulate the input of a decision question: "Is a target an adjective category", The user interface simply asks for the target, the adjective, and the category. For example, if the question is "Is John an important human", then the input for the target, adjective, and category is John important and human, respectively. We will show several examples to decide whether a given target is an important human or not. Before testing each example, the knowledge about the notion "importance" such as the criteria to be an important human and the relevant knowledge about the targets are assumed to exist in the knowledge base of ALPS.

The first example demonstrates that a "yes" answer is obtained by a direct match with an existing example in the knowledge base. The adjective is important, the category is human, and the target is Abraham Lincoln. The given input corresponds to the question, "Is Abraham Lincoln an important human". Since Abraham Lincoln is an example of an important human in the knowledge base, the answer is "yes".

The second example also demonstrates a match with an existing example but the match is indirect. The target is not a direct example of the question, but an aspect of the target matches an example of a factor. Assume that Marie Curie is not an example of an important human, but she has an award of a Nobel Prize. Assume further that one factor for an important human is to have an important award, and an example of the important award is Nobel Prize. For the question "Is Marie Curie an important human", although Marie Curie is not an example of an important human, she is still important because Nobel Prize is an example of the important award, which is a factor to be an important human.

Our third example corresponds to that case that neither the target nor any aspect of the target matches an example, but one aspect of the target matches with the reason of an example. Assume that Abraham Lincoln is an example of an important human, and the reason is that his occupation is a politician. In addition, although an important occupation is a factor to be an important human, a politician is not known to be an example of important occupation. Now assume that Gandhi is not an example of an important human, and his occupation is a politician. For the question "Is Gandhi an important human", although neither Gandhi nor politician is an example, Gandhi is still an important human since he is similar to an example of an important human, Abraham Lincoln. They are similar because they have the same reason, namely the same occupation.

Our fourth example is similar to our third example except that the occupation of the target is not exactly the same as the occupation of an example, but instead it is a subcategory. Assume that the occupation of John is governor, and assume that

governor is known by ALPS as a child category of politician. Using this hierarchical relationship, our algorithm concludes that John is important because John is similar to Abraham Lincoln.

Our fifth example shows that we can decide a "Yes" answer based on a unique and significant detail of the target. The information about the special detail can directly match a criterion. Consider the question "Is Picasso an important human". Assume that the knowledge base has no examples of an artist or criteria about famous artists, but an article that has a tremendous effect can satisfy a criterion to be an important human. In addition, assume that the knowledge base of ALPS has the knowledge that Guemica, a picture from Picasso, has a tremendous effect. Even though we cannot find direct matching or similarity of Picasso with any examples of important human, our program can still give a "Yes" answer when the reasoning line finds the special detail of Guemica.

Finally, we demonstrate an example that the "No" answer is decided. Assume that Tom is a human who does not satisfy any of the given criteria of an important human. Our algorithm correctly gives a "No" answer.

Chapter 5

Discussion

In this thesis, we discuss a method to answer a decision question with an opinion adjective. We solve this problem by addressing it to several sub-problems: the knowledge representation of opinion or notion, data organization in the knowledge base, logical inference through all the criteria. Notion knowledge representation is developed based on several classes and collections in ALPS. Data organization is inspired by human thinking and inferring procedures. By summarizing the general data that humans need in inference, we organize the data in a respective format. Logical inference algorithm simulates human inference. It can search through the criteria linkage and justify whether it matches or is similar to criteria and examples that can answer the question. Finally, the questions could be answered correctly with sensible criteria, enough examples, and some common sense knowledge learned in ALPS.

Our experiments show the evidence that our solution is a reliable part of ALPS in answering opinion OR decision questions. Different users can teach ALPS their

own opinions. Our experiments cover several common teaching styles. Our system can answer test questions correctly with different teaching files. Moreover, our system can operate error-checking in users' notion-teaching procedures and users' questions to avoid an incorrect inference.

Our method is a complementary solution for the statistics-based method. It can solve some problems that statistics methods cannot. For example, in a recommendation system application, statistics methods give recommendations based on training in huge data set. However, our method can analyze each user's shopping criteria and justify whether one product is suitable for the user. Our method can provide recommendations without lots of data.

There are still issues that need to be solved. For example, in the sentence with more than one adjective, the problem will be a combination of logic or and logic and problem. For example, the question could be “Is John a charming and important man”. To answer this question, we need to apply the logic or method to make a final decision.

Bibliography

[1] N. J. Nilsson. "The quest for artificial intelligence"; Cambridge University Press, 2009.

[2] W. Faris, and K. Cheng. "Performance of a Natural Language Framework"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 98-104, 2014.

[3] A. Andrenucci, and E. Sneiders. "Automated question answering: Review of the main approaches"; Information Technology and Applications, Third International Conference on. Vol. 1. IEEE, pp: 514-591, 2005.

[4] E. Cambria, and B. White. "Jumping NLP curves: a review of natural language processing research"; Computational Intelligence Magazine, IEEE, pp: 48-57, 2014.

[5] K. Cheng. "The representation and inferences of hierarchies"; Proc. IASTED International Conference on Advances in Computer Science and Technology, pp: 269-273, 2006.

[6] K. Mahesh, and S. Nirenburg. “Knowledge-Based Systems for Natural Language Processing”; New Mexico State University, Computing Research Laboratory, MCCS-96-296, 1996.

[7] M. Kohlhase. “Artificial Intelligence: Automated Reasoning”; Van Nostrand’s Scientific Encyclopedia, 2002.

[8] P. R. Cohen, and A. F. Edward. “The handbook of artificial intelligence”; Vol. 3, Butterworth-Heinemann, 2014.

[9] C. J. MacLellan. “A Generative Theory of Problem Solving”; First Annual Conference on Advances in Cognitive Systems, pp: 1-18, 2012

[10] S. Russell, et al. “Artificial intelligence: a modern approach”; Vol. 2, Prentice hall, 2003.

[11] A. Voronkov. “Logic for Programming, Artificial Intelligence, and Reasoning”; Springer Berlin Heidelberg, 2010.

[12] J. F. Allen. “Natural language understanding”; Benjamin Cummings, Second Edition, 1995

[13] T. KP. “A review on knowledge-based expert system: concept and architecture”; IJCA Special Issue on Artificial Intelligence Techniques-Novel Approaches & Practical Applications, no. 4, pp: 19-23, 2011.

[14] A. Konar. “Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain”; CRC press, 1999.

[15] Y. Liu, J. Bian, and E. Agichtein. "Predicting information seeker satisfaction in community question answering"; Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp: 283-290, 2008.

[16] D. A. Ferrucci. "Introduction to 'this is watson'"; IBM Journal of Research and Development 56.3.4 pp: 1-1, 2012.

[17] M. C. McCord, J. W. Murdock, and B. K. Boguraev. "Deep parsing in Watson"; IBM Journal of Research and Development 56.3.4 pp: 3-1, 2012.

[18] K. Cheng. "Answering Decision Questions involving an Adjective"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp. 264-270, 2016.

[19] K. Cheng. "Using Multiple Methods to Infer Classification with An Incrementally Expanding Knowledge Base"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 221-227, 2015.

[20] K. Cheng. "An object-oriented approach to machine learning"; International Conference on Artificial Intelligence, pp: 487-492, 2000.

[21] E. Ahn, W. Faris, and K. Cheng. "Recognizing the Effects Caused by an Action in a Declarative Sentence"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 149-155, 2009.

[22] W. Faris, and K. Cheng. "An Object-Oriented Approach in Representing the English Grammar and Parsing"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 325-331, 2008.

[23] w. Faris, and K. Cheng. "A Knowledge-Based Approach to Word Sense Disambiguation"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 111-117, 2013.

[24] K. W. Beyer. "Grace Hopper and the invention of the information age." The MIT Press, 2009.

[25] K. Li, R. G. Dewar, and R. J. Pooley. "Object-oriented analysis using natural language processing"; Linguistic Analysis, pp: 75-76, 2005.

[26] H. Yu, and V. Hatzivassiloglou. "Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences"; Proceedings of the 2003 conference on Empirical methods in natural language processing, pp: 129-136, 2003.

[27] I. Watson, and M. Farhi. " Case-based reasoning: A review"; Knowledge Engineering Review, 9.4, pp: 327-354, 1994.

[28] R. Mihalcea, and D. I. Moldovan. "An Iterative Approach to Word Sense Disambiguation"; FLAIRS Conference, pp: 219-223, 2000.

[29] K. Cheng. "Representing Definitions And Its Associated Knowledge In A Learning Program"; Proceedings on the International Conference on Artificial Intelligence (ICAI), pp: 71-77, 2007.

[30] D. Nadeau, and S. Sekine. “A survey of named entity recognition and classification”; Lingvisticae Investigationes, 30.1 pp: 3-26, 2007.