Solving Winograd Schema Challenge : Using Semantic Parsing, Automatic Knowledge Acquisition and Logical Reasoning

by

Arpit Sharma

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved October 2014 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Joohyung Lee
Heather Pon-Barry

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

Turing test has been a benchmark scale for measuring the human level intelligence in computers since it was proposed by Alan Turing in 1950. However, for last 60 years, the applications such as ELIZA, PARRY, Cleverbot and Eugene Goostman, that claimed to pass the test. These applications are either based on tricks to fool humans on a textual chat based test or there has been a disagreement between AI communities on them passing the test. This has led to the school of thought that it might not be the ideal test for predicting the human level intelligence in machines.

Consequently, the Winograd Schema Challenge has been suggested as an alternative to the Turing test. As opposed to deciding the intelligent behavior with the help of chat servers, like it was done in the Turing test, the Winograd Schema Challenge is a question answering test. It consists of sentence and question pairs such that the answer to the question depends on the resolution of a definite pronoun or adjective in the sentence. The answers are fairly intuitive for humans but they are difficult for machines because it requires some sort of background or commonsense knowledge about the sentence.

In this thesis, I propose a novel technique to solve the Winograd Schema Challenge. The technique has three basic modules at its disposal, namely, a Semantic Parser that parses the English text (both sentences and questions) into a formal representation, an Automatic Background Knowledge Extractor that extracts the Background Knowledge pertaining to the given Winograd sentence, and an Answer Set Programming Reasoning Engine that reasons on the given Winograd sentence and the corresponding Background Knowledge. The applicability of the technique is illustrated by solving a subset of Winograd Schema Challenge pertaining to a certain type of Background Knowledge. The technique is evaluated on the subset and a notable accuracy is achieved.

To Mom and Dad

ACKNOWLEDGEMENTS

I would like to thank many people for their support and guidance which made this project possible. First of all, thanks to my advisor, Dr. Chitta Baral, who guided me at every step in the project with his experience and helped me in making sense out of numerous confusing scenarios. Also thanks to him for providing me financial support during the project. Thanks to my committee members, Dr. Joohyung Lee and Dr. Heather Pon-Barry, who offered guidance and support whenever i needed. Thanks to the Arizona State University for providing me an opportunity to work with such great minds and technology. Thanks to my colleagues in Dr. Baral's lab who encouraged me and corrected me at times with their knowledge of subject matter and experience. Without them this project would not have been possible. And finally, thanks to my parents, and numerous friends who endured this long process with me, always offering support and love.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table | Page

LIST OF FIGURES

Chapter 1

INTRODUCTION AND MOTIVATION

There are three main components of the thought process that defines the intelligent behavior in humans. First is to semantically parse the input received either in the form of text or speech. Second is to gather all the commonsense knowledge about the input via previous experiences. Third and last one is to reach a conclusion or extract a piece of conclusive information by using the other two components. Natural Language Understanding is a field of Artificial Intelligence in which computer scientists try to simulate this kind of intelligent behavior in machines.

Several attempts have been made in past decades to realize this goal. And a number of tests were defined along the way to check their genuineness. Turing test, proposed by Alan Turing in 1950, is the most famous among such tests. Since then, a number of computer programs have been developed which claimed to pass the test. The initial programs such as Eliza computer therapist (1966), PARRY and Cleverbot Mauldin (1994) and other chatter bots used the bag of words based algorithms to chat with humans and made them believe that they are chatting with a real human being. Eugene Goostman ADASHCHIK (2014) is another program developed in 2001 and claimed to pass the Turing test in June 2014 by deceiving 33% of judges.

Another criteria for testing the intelligence in machines could be the Captcha. It consists of a set of numbers or letters printed on the screen and the task is to identify them. The letters and numbers are distorted such that its not straight forward to identify them. Humans are able to identify the digits after careful analysis. The test can be used to identify a machine from a real person but the test is more of a measure of ability of an observer to recognize the patterns and not really of understanding something.

But recently there has been a shift of interest from the Turing test. This might have been because of the systems developed to deceive the machine in that particular test(i.e. chatting with a machine to determine if it is a human) or this test might not be a true judge of demonstration of an intelligent behavior. Considering these, a new test called The Winograd Schema Challenge, proposed by Hector J. Levesque Levesque *et al.* (2011), has been widely accepted as an alternative to the Turing test. Furthermore, walking along the path, the company Nuance have started an yearly competition to solve Winograd Schema Challenge.

1.1 The Winograd Schema Challenge

The Winograd Schema Challenge is a Question Answering test based on a sentence and a question related to the sentence. The schema consists of pairs of a sentence and a question such that the answer to the question can be achieved by resolving a definite pronoun or possessive adjective to one of its two co-referents in the sentence. The co-referents belong to same gender (i.e. both are either objects, males or females) and they have a number agreement between them (i.e. both are either singular or plural). This property makes the resolution harder because the pronoun can not be resolved just by considering the gender and the number of pronoun and matching it with each of the entities present in the sentence. The sentence also contains a *"special word"* which when replaced by another word (*alternate word*), the answer to the question also changes. There are 141 sentences in Winograd Schema corpus and each contains a special word and an alternate word, making the total count of Schema sentences as 282. An example of a sentence from Winograd Schema and its corresponding alternate sentence is demonstrated below.

Sentence: *The man couldn't lift his son because he was so* $\boldsymbol{heavy}_{special\_word}$

Question: *Who was heavy ?*

Answer: *son*

Sentence: *The man couldn't lift his son because he was so* $\boldsymbol{weak}_{alternate\_word}$

Question: *Who was weak ?*

Answer: *man*

One of the motive behind this challenge is to simulate human-like reasoning in machines. A machine can be said to have such a behavior if it reaches a conclusion from a situation like humans do or in words of Hector Levesque Levesque (2014) "...we also want to avoid as much as possible questions that can be answered using cheap tricks (aka heuristics)...". For example, in the sentence *"The fish ate the worm because it was tasty"*, if we try to find the evidence in a large corpus about existence of a *"tasty fish"* or a *"tasty worm"* then fish would be the answer based on evidence support, which is wrong in this case because this sentence require some background knowledge that something that is eaten might be tasty.

From the above example, it can be observed that there is a need of sentence specific background knowledge to answer the question. With this idea, this work proposes a mechanism to semantically represent the given sentence, automatically extract the background knowledge about the given sentence and perform logical reasoning on them to get the answer to the given question. Its applicability is demonstrated by categorizing the Winograd Schema corpus into five broad categories (explained in later Chapter) and solving two sub-types of one of the categories.

1.2 Related Works

Mitkov (1997) mentions that the pronoun resolution depends on several factors like gender and number agreement between entities, structure of the text, semantic weaving of events, entities and relations between them. There has been a number of works which use statistical techniques, for example Hobbs (1976) uses the distance measure between the pronoun and the possible antecedents as a feature. Ge *et al.* (1998), Raghunathan *et al.* (2010), Broscheit *et al.* (2010), Ng and Cardie (2002) train on a big corpus using features like gender and number agreement between pronouns and other entities present in the text. Lee *et al.* (2011) furthermore adds semantic features by making use of WordNet and Freebase to find correlation between entities or to relax the string matching criteria to find more probable answer. The above techniques have been proved effective in cases where resolution can be accomplished by considering the features like gender, numbers, syntactic structure of sentences but these techniques fail where background knowledge about the events and entities in the text is required. There are a few works which attempt to use lexicalized feature sets with syntactic features. These include Rahman and Ng (2012), Gilbert and Riloff (2013), Bengtson and Roth (2008), Rahman and Ng (2011).

Rahman and Ng (2012) develops a system which attempts to solve the problem of pronoun resolution on a corpus which requires some sort of reasoning on it. The system is evaluated by developing a big corpus of 941 sentence pairs like Winograd Schema except the fact that they are a bit relaxed in terms of interweaving of semantics and also there is redundancy in the corpus. For example the test set alone contains two sentences : *John shot Bill and he died.* and *The man shot his friend and he died.* In both of these sentences the reasoning is same, so if the system is able to answer one of them correctly then other should also be correctly answered. The system uses a number of statistical techniques. One of them is the use of Google, a set of queries is created using the sentence based on the two antecedents and Google search results are compared. The idea underneath it is to find the more probable candidate to have a property or perform an action. This technique is helpful in case of sentences like, *Lions eat zebras because they are predators* (From Rahman and Ng (2012)) because one of the queries *"Lions are predators"* is essentially more common than *"zebras are predators"*. But if the sentence is changed to *Lions ate zebras because they were hungry*, the system will fail using this technique or if it passes then it is by shear luck because ideally both *"Lions were hungry"* and *"zebras were hungry"* must be equally likely, when no additional text is provided. This technique is the top contributor in the overall accuracy of the system (33% when using single-feature co-reference model). One reason behind such a behavior is the high coupling of sentences with this technique which disagrees with an important property of Winograd Schema Challenge i.e. it should not be Googlable Levesque *et al.* (2011).

Another technique used by the paper is based on the usage of Narrative Chains Chambers and Jurafsky (2008). Narrative chains are sequences of partially ordered set of events (or verbs) centered around a common actor. *own-s spin_off-s plan-s purchase-s sell-s operate-s buy-s announce-s pay-s build-s acquire-s merge-s* is one such chain which represents the knowledge that someone who owns may spin off, plan, purchase, sell, operate, buy, announce, pay, build, acquire and merge (in respective order). The chains store knowledge about the probable actions performed by an entity over the course of time but one of the major issue with them is their very limited number. Furthermore, many times the resolution of pronoun depends on the properties and not events, like in the *"Lions"* example mentioned above. The Narrative Chains would not be of any help here because they contain only verbs (events) and not the attributes of the entities participating in those verbs. Another issue with Narrative Chains is that they do not represent the relation between events. For example the sentence *Ed punished Tim because he tried to escape* is expressed by a chain containing *punish-o escape-s*. But if the sentence is changed to *Ed punished Tim so he tried to escape*, still it can be represented by the chain *punish-o escape-s* whereas in first case the escape event occurred before punish event and vice versa in second case. It contributes to 30% of the overall accuracy (with single-feature co-reference models). The reason for its high accuracy is also the high coupling between this technique and the corpus.

Another work by Peter Schuller Schuller (2014), demonstrates a graph based technique and performs experiments on 4 out of 141 Winograd Schema pairs. It converts the given Winograd sentence to a dependency graph using Stanford dependency parser and then manually creates a background knowledge dependency graph which is required to answer the question. The main contribution is to formalize a way to combine both given sentence dependency graph and the manually created background knowledge dependency graph in using relevance theory and then use Answer Set Programming (ASP) Gelfond and Lifschitz (1988), Baral (2003) to extract the answer. As mentioned above, this paper illustrates the application of the technique on 4 pairs of Winograd Schema to show the usability of this technique. Extraction of Background knowledge is one of the main components of reasoning process. It is manually encoded into the system proposed by Schuller (2014).

Considering above mentioned motivation, limitations and drawbacks, a technique has been formulated, which simulates the human like thought process in solving such problems. It includes Semantic Parsing, Automatic Background Knowledge Acquisition and Logical Reasoning.

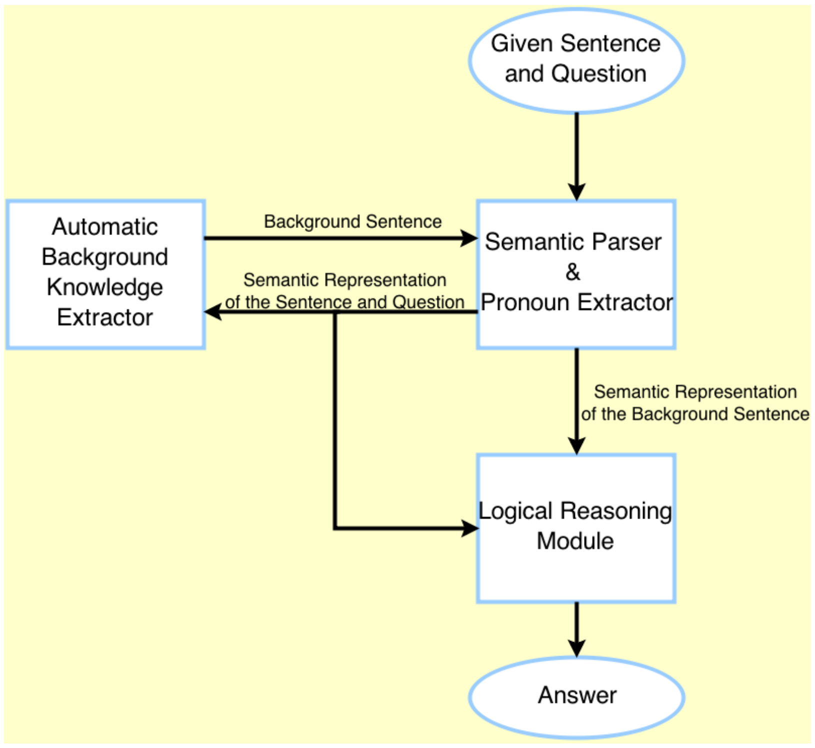

Given Sentence and Question
Automatic Background Knowledge Extractor
Background Sentence
Semantic Parser & Pronoun Extractor
Semantic Representation of the Sentence and Question
Semantic Representation of the Background Sentence
Logical Reasoning Module
Answer

Figure 1.1: Work-Flow Diagram for the Pronoun Resolution System

1.3 An Overview of the Approach

As previously mentioned, the approach is based on the thought process of human beings. It includes three basic component modules, namely, Semantic Parser and Pronoun Extractor (SPPE) to parse a Winograd Schema sentence into a graph based semantic representation and extract the *pronoun to be resolved* from, the given Winograd sentence, Automatic Background Knowledge Extractor (ABKE) which extracts the background knowledge required to answer the question about the given Winograd sentence, and finally a Logical Reasoning Module (LRM) which uses Logical Programming rules to reason on the semantic representations of the actual Winograd sentence and the Background sentence extracted by other module. As illustrated by the work-flow diagram of the system in Figure 1.3, both the Winograd sentence and question are first passed to SPPE. The output of SPPE i.e. semantic representations of the sentence and Question are then passed as input to ABKE. The background knowledge sentence extracted from ABKE is then passed to SPPE for semantic parsing. Finally the semantic representations of both the given Winograd sentence and its corresponding Background Knowledge sentences are passed to the LRM, which returns the final answer to the Winograd question, or the co-referent of the *pronoun to be resolved.*

After careful consideration of the complexity and semantics, the complete Winograd Schema corpus (containing 282 sentences) is divided into two broad categories namely, *Causal* and *Non Causal.* The categories are defined as follows:

- **Causal**:

  In this category there is a causality relation in the sentence such that a phrase in the sentence contains information caused due the information contained in other phrase present in the sentence. For example:

  Sentence: *The fish ate the worm because it was tasty.*

  Question: *What was tasty ?*

  Answer: *worm*

  In above sentence, the phrase *it was tasty* causes *The fish ate the worm.* This causal relationship is due to the semantics associated with the word *because.*

- **Non Causal**:

  In this category there is a chain of events mentioned in the sentence but there is no causality relation between those events. Events here refer to either actions or an entity having an attribute. For example:

  Sentence: *Mary took out her flute and played one of her favorite pieces. She has had it since she was a child.*

  Question: *What has Mary had since she was a child ?*

Answer: *flute*

In above sentence, the event *She has had it since she was a child.* does not have causal relationship with *Mary took out her flute and played one of her favorite pieces.*

One of category which covers a major part of the Winograd Schema Challenge corpus is *causal* (with more than 200 sentences out of 282). Hence, further analysis of the *Causal* category is performed and two subcategories are identified based on the causality within the sentences in the corpus. In this work, the two subcategories are solved using a novel approach to extract the sentence specific background knowledge on the fly. These subcategories contribute to a significant part of the corpus (100 sentences out of >200). The subcategories are:

- **Direct Causal Events**: In this category, there are two events in the sentence which are connected to each other with a causality relation (defined by a set of relations from KM component library). The pronoun which is required to be resolved participates in one of the events and its candidate co-referent participates in the other. For example,

  Sentence: *The older students were bullying the younger ones , so we rescued them .*

  Question: *Whom did we rescue ?*

  Answer: *younger ones*

  In above sentence the event *rescued* is caused by the event *bullying* and the pronoun, *them* participates in the *rescued* event whereas its, candidate answer *younger ones* participates in the *bullying* event

- **Causal Attributive**: In this category, the *pronoun to be resolved* has a property that plays an important role in occurrence of an event to occur. For example,

Sentence: *Pete envies Martin because he is very successful.*

Question: *Who is successful ?*

Answer: *Martin*

In above sentence the property *successful*, associated with *he*, causes the event *envies* to occur.

In this Chapter the motivation behind this work and a detailed introduction of the problem is provided. Also, the Winograd Schema corpus is categorized based on some characteristics. Furthermore, a brief overview of the novel approach used to solve the Winograd Schema Challenge is discussed. In the following Chapters each of the component of this approach are discussed and explained in detail with appropriate examples.

Chapter 2

SEMANTIC PARSING AND PRONOUN EXTRACTION

In Chapter 1, the motivation behind this thesis was explained and a brief overview of the technique used to solve the Winograd Schema Challenge was provided. It was also mentioned that every Natural Language Understanding problem involves the task of

- Representing the text's syntax and semantics in an expressive formal representation
- Broadening the scope of knowledge present in the text by including information about events and entities in it. This specific kind of information is called world or background knowledge Davis (2013).
- Reasoning on the formal representations of the given text and the Background knowledge about the given text. (Diakidoy *et al.* (2013))

There are three main steps followed to accomplish each of the above illustrated tasks in the system. In this chapter the first step is explained in detail. The step is to define an expressive formal representation language and translate the given Winograd sentence and question into it. Later in this step, both the translations are used along with a set of Answer Set Programming rules and the *pronoun to be resolved* is extracted.

The sections below explain each component of this step in detail.

2.1 Defining Formal Representation

A representation is considered good if it can express the structure of the text, can distinguish between the events and their environment in the text, uses a general set of relations between the events and their participants, and is able to represent the same events or entities in different perspectives. There has been a lot of work in recent years to translate English text into a semantic representation so that it can be used for tasks which involve reasoning on either the syntax or the semantics of the language. Many systems such as Yao and Van Durme (2014) and Berant and Liang (2014), have shown promising results by focusing on a specific domain (eg. Factual Question Answering on Freebase). However, natural languages like English represent a complex interweaving of events and relations between them. The events also have a surrounding environment which includes the entities participating in them. Furthermore, the language is so intricate that sometimes the background about the events and entities is required to actually understand and reason upon it. For example the Winograd Schema Challenge Levesque *et al.* (2011) sentences mentioned below require some background knowledge to answer the question about the sentence.

Sentence: *The drain is clogged with hair. It has to be cleaned.*

Question: *What has to be cleaned?*

Answer: *The drain*

Sentence: *The drain is clogged with hair. It has to be removed.*

Question: *What has to be removed?*

Answer: *The hair*

Most of the current semantic parsers and semantic role labelers lack in capturing the basic semantic relations in the text. However, some recent systems Hermann *et al.* (2014),Das *et al.* (2010) encode the commonsense knowledge from FrameNet, VerbNet, PropBank, etc. while still lacking the semantic relations. Also, each system has its own representation schema, so in most of the cases the representations are not really compatible with each other.

From the above discussion it can be observed that there is a need of a representation which can be used to represent the basic knowledge mentioned in the given text along with the background knowledge required to understand the meaning associated with the sentence.

With this motivation, in this Chapter a formal representation for English text is defined, which would be useful in Co-reference Resolution, Machine Translation, Deep Question Answering, etc. While doing so, a popular ontology, called Knowledge Machine(KM) Clark *et al.* (2004) is used to represent the relations between concepts. The usability of KM ontology has been proved by a number of projects, such as AURA Chaudhri *et al.* (2009) which is developed under Project Halo Gunning *et al.* (2010). The use of KM ontology provides us the ability to

- combine the data provided by several NLP tools,
- merge the information of many sentences into a single representation, and
- make the system's output compatible to a whole range of systems currently using KM.

Such systems include reasoning and factoid question answering Baral and Liang (2012); Chaudhri and Son (2012) for data enrichment (through inheritance and cloning) and Deep Question Answering Baral *et al.* (2012). The concept of proto-classes (from Clark *et al.* (2004)) has also been used to represent the quantification of entities in the text. Furthermore, many off-the-shelf NLP tools are also used for features such as Word Sense Disambiguation and Named Entity Recognition.

In the next sections the related works in this field are discussed and this work is compared with them. Later the techniques behind this work are explained in detail.

2.1.1 Syntactic Dependency Parsing

This component is responsible for parsing the English text into a syntactic dependency parse tree. The Stanford dependency parser De Marneffe *et al.* (2006) is used to parse the text. For example, the sentence "George loves Mary." is parsed into the dependency graph shown in Figure 2.1.

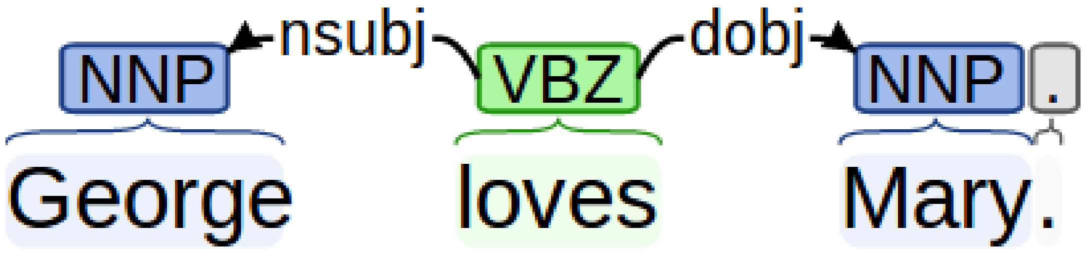


Figure 2.1: Stanford Dependency Parse of *"George loves Mary."*

In the dependency parse, *nsubj* and *dobj* predicates show that *George-1* and *Mary-3* are respectively the nominal subject and object of *loves* event. The meaning of the dependency predicates used by Stanford Dependency parse is analyzed from De Marneffe and Manning (2008).

The predicates in dependency parse of Stanford parser are very fine grained. For example, in the dependency parse of sentence *"George played with the ball in the field."* *"prep_in"* is used to represent the mention of the preposition *"in"*. Similarly, Stanford Dependency Parser has different predicates for all prepositions and even for the conjunctions such as *"and"* and *"but"*. To generalize the predicates, the well defined and popular ontology i.e. Knowledge Machine(KM) Barker *et al.* (2001), Clark *et al.* (2004) is used. KM has been used by various projects such as Project AURA Chaudhri *et al.* (2009), which is developed under Halo Gunning *et al.* (2010). The Stanford dependency predicates are mapped to the set of relations defined in

KM using manual rules. Furthermore, thousands of English sentences are analyzed, and a new *participant* relation is defined as a part of the mappings explained above. It is used to encode a special type of relation between an event and an entity. It can be explained with the help of the example sentence *Tom hit John because he was rude*. The semantic parse of the sentence is shown in Figure 2.2. In this example the edge *participant* between *hit* and *he* is encoded to capture the relation that *he* is a participant in the event *hit* i.e. he is a co-referent of either of its children(*Tom* and *John* in this case).

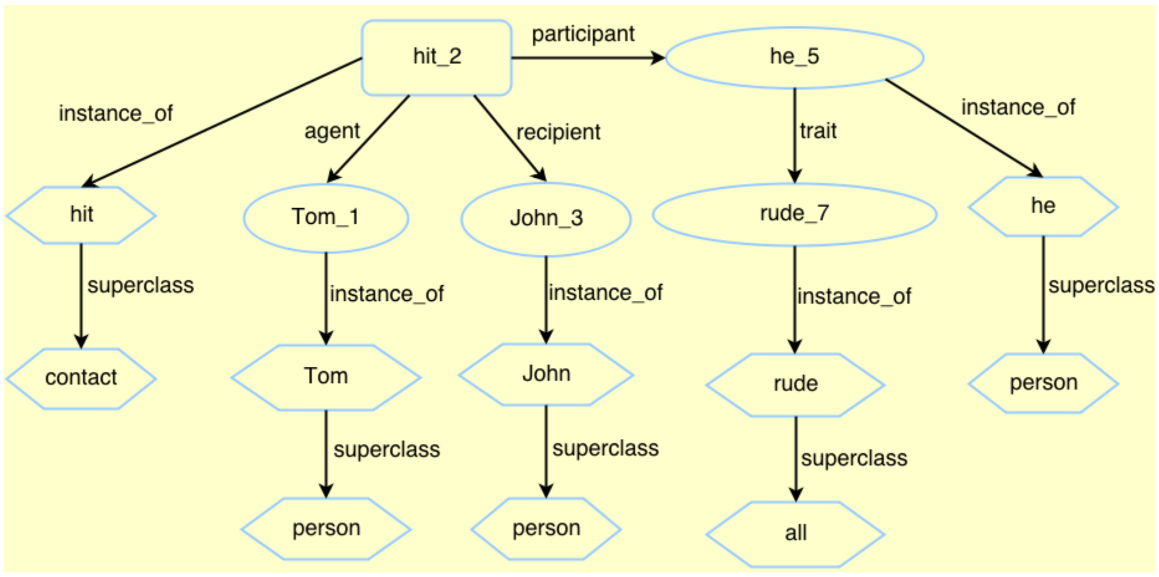


Figure 2.2: Graphical Semantic Representation of *"Tom hit John because he was rude"*

2.1.2 Class Population

This is a two step process. The first step is to encode the relation between two words which are present in the text in different grammatical forms. This is done by adding a common class for both the words in the representation. For example, in the sentence *If John had not loved every cat, he would have loved a dog*, the word *love* occurs in two contexts but their conceptual meaning is same. This information is encoded by adding a directed edge *instance_of* from both the *"love"*s to their base form as illustrated in Figure 2.3.

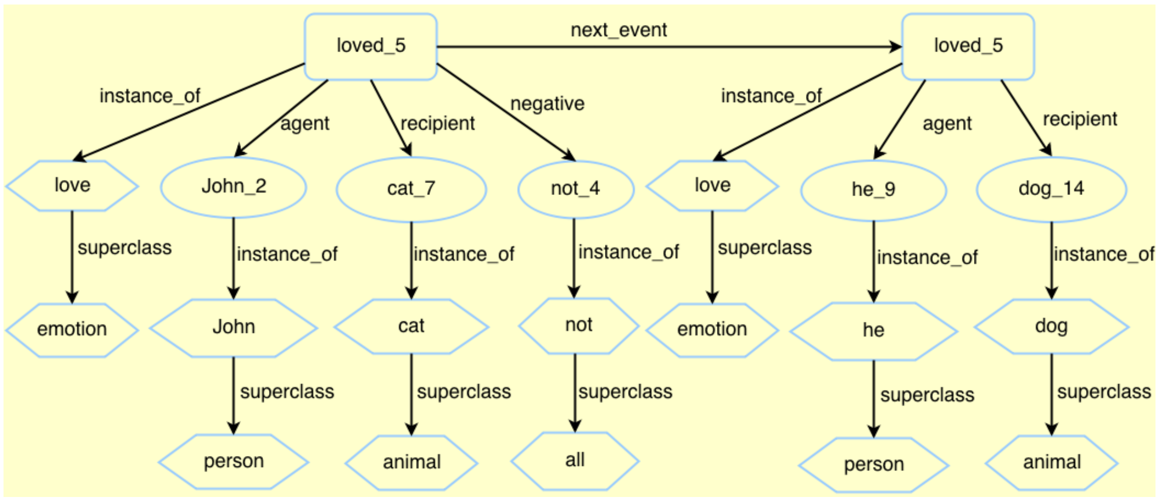


Figure 2.3: Graphical Semantic Representation of *"If John had not loved every cat, he would have loved a dog"*

Another level of generalization is added by adding the superclass information to each word class in the representation. It is helpful in representing the sense associated with each word in the sentence. For example, in the graph in Figure 2.3, the words *John* at position 2 and *cat* at position 7 belong to *John* and *cat* classes respectively, they also belong to same supper classes or sense which are *Person* and *Animal* respectively. This information is encoded by using the lexical information about words from WordNet Miller (1995). Furthermore, Word Sense Disambiguation Basile *et al.* (2007) is used to get lexical information associated with the correct meaning of the word in context of the given sentence.

2.1.3 Other Features

Named Entity Recognition

There are instances in English language where multiple words represent one named entity. For example in the sentence *The boy wants to visit New York City*(from Levesque *et al.* (2011)), the last three words represent a named entity and it is a location. We capture this information by using the Stanford Named Entity Recognizer Finkel *et al.* (2005). The information is used to preprocess the sentence to make *New York City* as a single entity and it is also used while populating the superclass information i.e. its superclass is defined as location. Similar procedure is followed for different named entities with different named tags such as ORGANIZATION, PERSON etc.

Entity Based Co-reference Resolution

There is a possibility that only one entity is present in the given text or some entity specific words are used. For example in the sentence *He carried me so I thanked him.* In this sentence we do not need any Background knowledge about the entities and events in the sentence to resolve *me* to *I* and *He* to *him.* This is because in any sentence *me* and *I* are always co-referents of each other (Its a property of English language). Now, consider the sentence, *She took off because she was in hurry.* In this sentence, it is obvious that there is only one entity present, so both *[she]*s are co-referents of each other. This type of lower level co-reference resolution is done before semantically parsing the sentences. In this preprocessing, all the co-referent words are replaced by $ENT(number)$ where number determines the number of co-referent entities in the sentence. For example the above sentences are preprocessed to *ENT1 carried ENT2 so ENT2 thanked ENT1* and *ENT1 took off because ENT1* *was in hurry* respectively.

2.2 Translating Winograd Sentence and Question into Formal Representation

Continuing with the Winograd example mentioned earlier, the sentence is translated into a graphical representation as shown in Figure 2.4. The sentence is represented in Answer Set Programming language by using the *has* predicate. The first argument of the predicate represents the context of the formal representation graph i.e. *winograd* or *question*. The second and fourth arguments represent the originating and end nodes of the edge in the graph and the label of the edge is represented by the third argument (The detailed explanation of the ASP syntax is present in Chapter4). For example the *lift* example sentence's formal graph is represented in ASP as.

Sentence 1.

```
% Sentence = The man couldn't lift his son because he was so weak
has(winograd,lift_5,instance_of,lift).
has(winograd,lift,superclass,motion).
has(winograd,lift_5,agent,man_2).
has(winograd,lift_5,recipient,son_7).
has(winograd,lift_5,negative,not_4).
has(winograd,lift_5,participant,he_9).
has(winograd,he_9,trait,weak_12).
has(winograd,weak_12,trait,so_11).
has(winograd,man_2,instance_of,man).
has(winograd,man,superclass,person).
has(winograd,son_7,possesed_by,his_6).
has(winograd,son_7,instance_of,son).
has(winograd,son,superclass,person).
has(winograd,his_6,instance_of,his).
has(winograd,his,superclass,person).
has(winograd,not_4,instance_of,not).
has(winograd,not,superclass,all).
has(winograd,he_9,instance_of,he).
has(winograd,he,superclass,person).
has(winograd,weak_12,instance_of,weak).
has(winograd,weak,superclass,all).
has(winograd,so,superclass,all).
```

Similarly the Question's formal graph is represented in *has* format as below.

Question 1.

```
% Question = Who was weak ?
has(question,q_1,instance_of,q).
has(question,q_1,trait,weak_3).
has(question,weak_3,instance_of,weak).
has(question,weak,superclass,all).
```

2.3 Answer Set Programming Rules to Extract Pronoun To Be Resolved

A set of Answer Set Programming rules are proposed to extract the *"pronoun to be resolved"* from the representations of Winograd sentences and the respective questions about them. The rules match a question's formal representation with that of the respective sentence's and extract the *word* which is preceded by the *Wh* string in the question.

An Example of formal representation of a sentence is given in Figure 2.4 and its corresponding question's representation is shown in Figure 2.5.

The ASP rules and predicates used for *pronoun* extraction are mentioned below.

The predicate *toBeResolved* is used to extract all the pronouns needed to be resolved to answer the given question about the Winograd sentence.

Definition 1 (*toBeResolved*(P))**.** *Let there is a node* P *in the Winograd sentence graph. Also, there is an edge* $has(winograd, X2, R, P)$ *or* $has(winograd, P, R, Y2)$

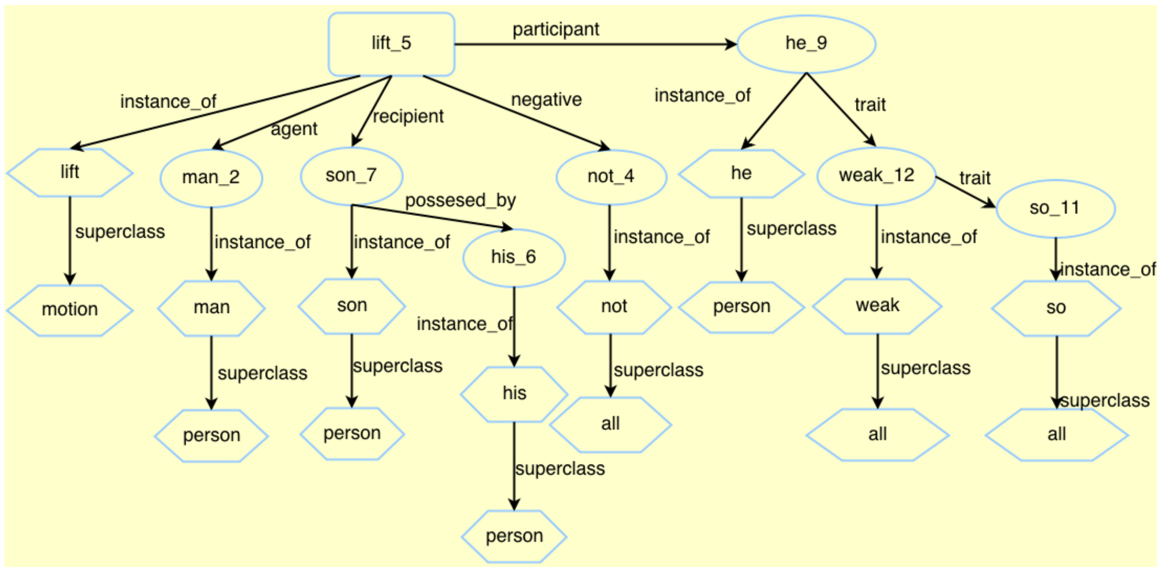


Figure 2.4: Graphical Semantic Representation of the Sentence *"The man couldn't lift his son because he was so weak"*

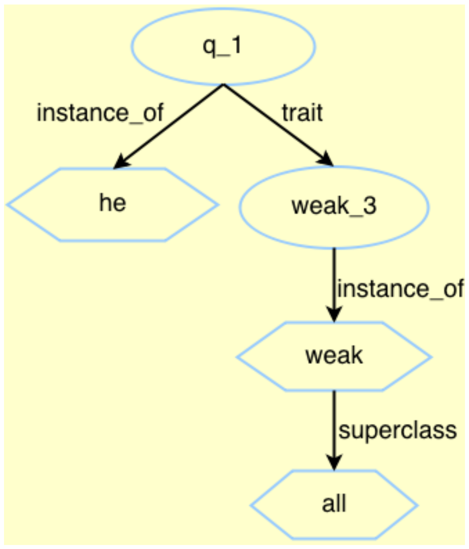


Figure 2.5: Graphical Semantic Representation of the question *"Who was weak ?'*

in the Winograd sentence graph and there is an edge $has(question, X1, R, Y1)$ *in the given question, if*

1. $Y1$ *is an instance of* q *and both* $X1$ *and* $X2$ *have same class node (say* X*) or*

2. $X1$ *is an instance of* q *and both* $Y1$ *and* $Y2$ *have same class node (say* Y*)*

then, P is the pronountoberesolved.

Following ASP rules are defined to extract this predicate.

```
toBeResolved(P) :- has(question,X1,R,Y1),
                   has(question,X1,instance_of,X),
                   has(question,Y1,instance_of,q),
                   has(winograd,X2,R,P),
                   has(winograd,X2,instance_of,X).

toBeResolved(P) :- has(question,X1,R,Y1),
                   has(question,X1,instance_of,q),
                   has(question,Y1,instance_of,Y),
                   has(winograd,P,R,Y2),
                   has(winograd,Y2,instance_of,Y) .
```

After applying the technique to extract the pronoun on the two representations in figure 2.4 and 2.5, we get the pronoun to be resolved as ***it***.

In this Chapter, the formal semantic representation for the given Winograd sentence and its respective question is defined. Also, both the sentence and the question are translated into the representation to extract the *pronoun to be resolved*. In next chapters the remaining two modules of the pronoun resolution system i.e. Automatic Background Knowledge Extraction and Logical Reasoning are explained in detail.

Chapter 3

AUTOMATIC BACKGROUND KNOWLEDGE EXTRACTION

In Chapter 2, a formal representation for English text is defined and both the given Winograd sentence and question are translated into it. In this chapter a technique to automatically extract the Background Knowledge about the given Winograd sentence and question is explained in detail.

3.1 Introduction and Motivation

As mentioned in previous chapters of this work, language is a complex phenomenon. To understand conversations, written or uttered, many a times it is required to have some sort of background information or commonsense knowledge about the entities and incidences which are mentioned in the conversation. For example in the Winograd Schema mentioned below.

Sentence: *The man could not lift his son because he was so weak.*

Question: *Who was weak ?*

Answer: *The man.*

The answer to above question depends on resolution of *he* in the sentence to either *man* or *man's son*. This can be done by using the background knowledge about *lift* incidence that *weak person cannot lift heavy things.* We as humans can observe that we do not need to think or gather any background knowledge before answering this sentence i.e. we do not go search in a Knowledge base to find such a knowledge. This is because we already have it stored in our brain. We just retrieve it at a very high speed.

Now, from the above discussion, the answers to two questions are of utmost importance in simulating human like reasoning. They are

- From where do humans get the background knowledge about things ?
- How do humans extract it ?

One of the main contributor of the answer to the first question is reading performed by humans since the time they started to understand the written language. There are studies[1][2] which suggest that reading does not only make you understand the written language momentarily but it also increases the levels of intelligence in humans. In other words, one of the reason that humans have commonsense knowledge about entities and events is day-to-day reading. Another source of commonsense knowledge for humans are experiences. For example,considering the *lift* example mentioned above, a real scenario might occur that, a human being tries to lift another person and he is not able to. From this scenario he/she knows that the other person is heavy or he/she is weak.

The answer to the second question mentioned above is as important as the other one. When humans answer the *lift* question mentioned above, then they do not search all their knowledge base stored in their brain. However, they filter the knowledge based on the entities and events which participate in the sentence and its corresponding question. And if such knowledge is not present then they relax the filtering criteria by using words which are already stored in their Knowledge Base (i.e. brain) and are conceptually similar to the actual words in the sentence and the question.

[1]A research supported by Spencer Foundation states that "Reading has cognitive consequences that extend beyond its immediate task of lifting meaning from a particular passage"

[2]Another article on http://www.theguardian.com/us states that "There is evidence that reading can increase levels of all three major categories of intelligence."

3.2 An Overview of the Approach

In this work also, it is tried to simulate the human thought process as mentioned above. This includes the extraction of the Winograd sentence and question specific background knowledge from a big source of raw text. It is accomplished by creating string queries using the concepts in the sentence and the question. Later the queries are used to retrieve sentences from a large corpus of raw text. Following subsections explain these steps in detail. The Winograd Schema example mentioned below will be considered to demonstrate the technique in this chapter.

Sentence: *The man could not lift his son because he was so weak .*

Question: *Who was weak?*

3.2.1 Creating String Queries

As mentioned in previous section, humans retrieve the commonsense knowledge pertaining to a particular context. The context is defined by entities and events in the Winograd sentence and question. It is also noticeable that whenever humans search for a context in a sentence they do not consider all the entities, for example in the sentence *John loves Mia because she is a good girl*, *John* and *Mia* are not very important to know the context i.e.*someone loves a good girl*, of the sentence. Whereas the entity *girl* which is a more general term and *good* which identifies a property of *girl* are important. Considering these points, following set of string queries are created to capture the context of the Winograd sentences.

- First set of queries (say $Q1$) is created by using formal representations of both the Winograd sentence and the question. All the nodes from the question's formal representation (except the ones which represent "Wh" words) are traced into the formal representation of the given sentence. If two nodes in different graphs belong to same class and superclass then they are considered as similar. From the traced output, all the words/nodes which do not specify a nominal entity are extracted and their different combinations (with and without prepositions, adverbs and conjunctions in the sentence) are joined together using a wild card (.*) and double quotes (“”). An example query for the lift example mentioned above is, *“.*not.*lift.*because.*weak.*”*. Another query generated for same sentence is, *“.*not.*lift.*because.*so.*weak.*”*

- Second set of queries (say $Q2$) is created by replacing the verbs in the previously created set of queries by their synonyms. For eg. one of the new queries generated for *lift* example is, *“.*not.*pick.*because.*weak.*”*, where *pick* is a synonym of *lift*.

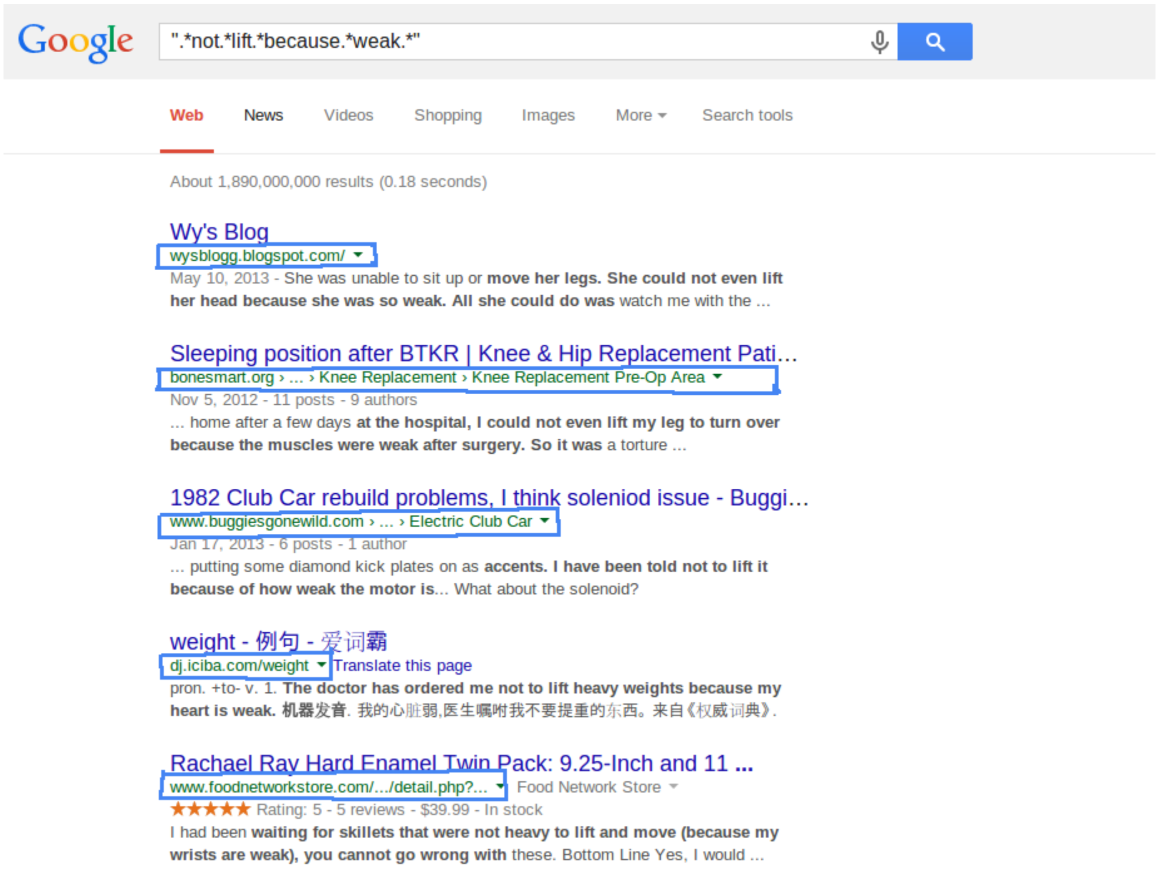
Finally, a combined set of queries is formed by merging the two sets extracted above i.e $Q = Q1 \cup Q2$.

3.2.2 Using Queries to Extract Background Knowledge

As discussed in the previous section, humans store most of the knowledge, that they get by reading and experiences, in their brains. There are a billions of sources of such knowledge which would require billions of terabytes of memory, which is impossible till current day for a personal computer. So, a big source of raw knowledge such as books, magazines, newspapers and websites such as Google, Bing and Yahoo Answers are a possible substitute which can be used.

The second sub-step in background knowledge extraction process is to automatically search such large corpus of raw English text using the queries created in previous subsection and extract the sentences that are retrieved using the queries. Following tools and techniques are used to accomplish this functionality.

- **Automated Google Search:** Currently, the Google search engine is used to extract the background sentences but the searching can be performed on other datasets too (The Google search is called from java program by using honest user agents). The idea here is to extract the sentences which are semantically and structurally similar to the given Winograd sentence. The Google search returns the web search results for a particular search query. These results are the urls of the web pages that has text matching with out search query. For example the web pages extracted for the above mentioned *lift* example while using the query *".*not.*lift.*because.*weak.*"* are highlighted in Figure 3.2.2.



About 1,890,000,000 results (0.18 seconds)

Wy's Blog
wysblogg.blogspot.com/
May 10, 2013 - She was unable to sit up or move her legs. She could not even lift her head because she was so weak. All she could do was watch me with the ...

Sleeping position after BTKR | Knee & Hip Replacement Pati...
bonesmart.org › ... › Knee Replacement › Knee Replacement Pre-Op Area
Nov 5, 2012 - 11 posts - 9 authors
... home after a few days at the hospital, I could not even lift my leg to turn over because the muscles were weak after surgery. So it was a torture ...

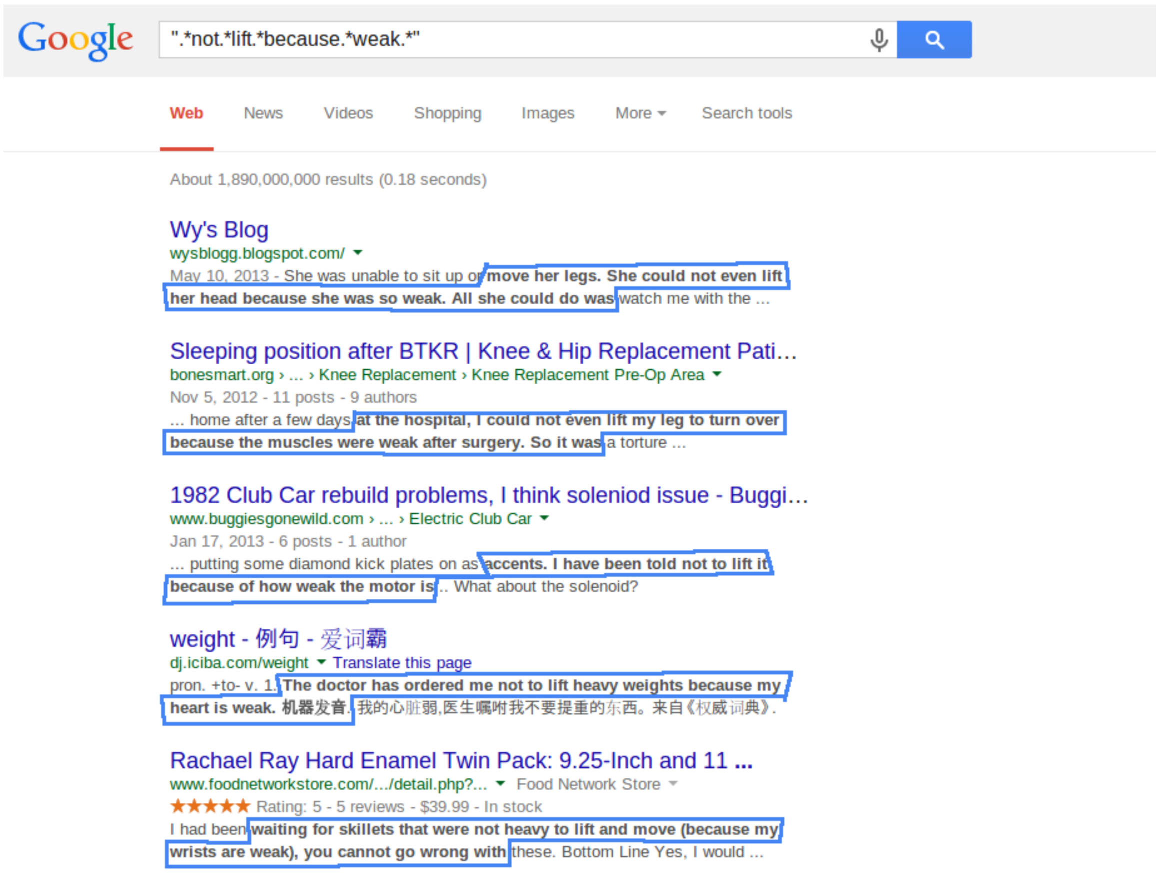
1982 Club Car rebuild problems, I think soleniod issue - Buggi...
www.buggiesgonewild.com › ... › Electric Club Car
Jan 17, 2013 - 6 posts - 1 author
... putting some diamond kick plates on as accents. I have been told not to lift it because of how weak the motor is... What about the solenoid?

weight - 例句 - 爱词霸
dj.iciba.com/weight Translate this page
pron. +to- v. 1. The doctor has ordered me not to lift heavy weights because my heart is weak. 机器发音. 我的心脏弱,医生嘱咐我不要提重的东西。来自《权威词典》.

Rachael Ray Hard Enamel Twin Pack: 9.25-Inch and 11 ...
www.foodnetworkstore.com/.../detail.php?... Food Network Store
★★★★★ Rating: 5 - 5 reviews - $39.99 - In stock
I had been waiting for skillets that were not heavy to lift and move (because my wrists are weak), you cannot go wrong with these. Bottom Line Yes, I would ...

Figure 3.1: Google Search Results for the Query *".*not.*lift.*because.*weak.*"* . The Sentences Are Extracted from the Highlighted URLs.

- **Extracting sentences from Google search result pages:** This is a modification of above mentioned Google search technique. In this technique the text which is shown in bold letters in the Google search results for a given query is extracted directly from the Google search results web page. For example the text extracted for the above mentioned *lift* example while using the query *".*not.*lift.*because.*weak.*"* are highlighted in Figure 3.2.2.



Google ".*not.*lift.*because.*weak.*"

Web News Videos Shopping Images More Search tools

About 1,890,000,000 results (0.18 seconds)

Wy's Blog
wysblogg.blogspot.com/
May 10, 2013 - She was unable to sit up or **move her legs. She could not even lift her head because she was so weak. All she could do was** watch me with the ...

Sleeping position after BTKR | Knee & Hip Replacement Pati...
bonesmart.org › ... › Knee Replacement › Knee Replacement Pre-Op Area
Nov 5, 2012 - 11 posts - 9 authors
... home after a few days **at the hospital, I could not even lift my leg to turn over because the muscles were weak after surgery. So it was** a torture ...

1982 Club Car rebuild problems, I think soleniod issue - Buggi...
www.buggiesgonewild.com › ... › Electric Club Car
Jan 17, 2013 - 6 posts - 1 author
... putting some diamond kick plates on as **accents. I have been told not to lift it because of how weak the motor is**.. What about the solenoid?

weight - 例句 - 爱词霸
dj.iciba.com/weight Translate this page
pron. +to- v. 1. **The doctor has ordered me not to lift heavy weights because my heart is weak. 机器发音.** 我的心脏弱,医生嘱咐我不要提重的东西。来自《权威词典》.

Rachael Ray Hard Enamel Twin Pack: 9.25-Inch and 11 ...
www.foodnetworkstore.com/.../detail.php?... Food Network Store
★★★★★ Rating: 5 - 5 reviews - $39.99 - In stock
I had been **waiting for skillets that were not heavy to lift and move (because my wrists are weak), you cannot go wrong with** these. Bottom Line Yes, I would ...

Figure 3.2: Google Search Results for the Query *".*not.*lift.*because.*weak.*"* . The Highlighted Text Is Extracted from the Search Results Page.

- **Sentence Splitter** Usually, the text extracted by above techniques is a group of many sentences combined together. To divide this text into its constituent sentences a Natural Language Processing tool kit called LingPipe (Alias-i.2008

Table 3.1: Background Sentences Extracted For the Winograd Sentence *"The man could not lift his son because he was so weak ."*

She could not lift it off the floor because she is a weak girl
She could not even lift her head because she was so weak.
I could not even lift my leg to turn over because the muscles were weak after surgery.
The doctor has ordered me not to lift heavy weights because my heart is weak.

(2008)) is used. LingPipe is tool kit for processing text using computational linguistics. LingPipe is used for tasks such as Sentence Splitting, Part of Speech tagging, Sentiment Analysis and Named Entity Recognition.

A few of the sentences extracted from Google by using the above mentioned query for *lift* example are shown in Table 3.1.

3.2.3 Post-processing the Background Knowledge

The techniques and tools mentioned above are used to extract a bunch of supposedly knowledgeable sentences. But the number of sentences extracted from a web page is very large. Only a few of those sentences are actually useful. So, to filter the sentences and extract only the ones which could prove useful, a filtering mechanism is implemented. The sentences are filtered based on below mentioned criteria.

- All the words in the query must be present in the sentence in any form.
- The order of appearance of words in query must match the order in the sentence.

If the above conditions are met, the sentence is kept for further processing otherwise it is discarded as useless.

Furthermore, it is possible that the extracted sentences contain the given Winograd sentence because the words in the query would exactly match the words in the sentence and it would pass through the previous filter. So, another filter to remove

sentences which match with the given Winograd sentence is implemented. On the Internet the Winograd sentences are present as pairs. For example
The man could not lift his son because he was so [weak/heavy] .
Where as one of the given sentence in our system would be
The man could not lift his son because he was so weak .

It can be observed that both of these sentences are similar but not exactly same. To consider this a relaxed sentence matching technique is implemented. In this technique, the sentences are matched based on percentage of words matched. The threshold for the matching to allow a sentence to be rejected in this process is kept as 80%.

3.3 Translating Background Knowledge Sentences Into Formal Representation

After the application of previous sections, a set of sentences is extracted which could have the background knowledge about the given Winograd sentence and could be helpful in answering the given Winograd question. To finally answer the question a logical reasoning is performed on the formal representation of the given sentence, the *pronoun to be resolved* extracted in previous chapter and the formal representation of the background sentences that are extracted using a technique explained in this chapter. Considering this, the last step in this chapter is to translate the extracted background knowledge sentences into the formal representation similar to that of the given Winograd sentence. Let us consider one of the background sentences extracted for the *lift* example mentioned above.

Background sentence: *She could not lift it off the floor because she is a weak girl*

It is translated to the graphical formal representation illustrated in Figure 3.3. Also, in terms of textual logic rules, it is represented as below.

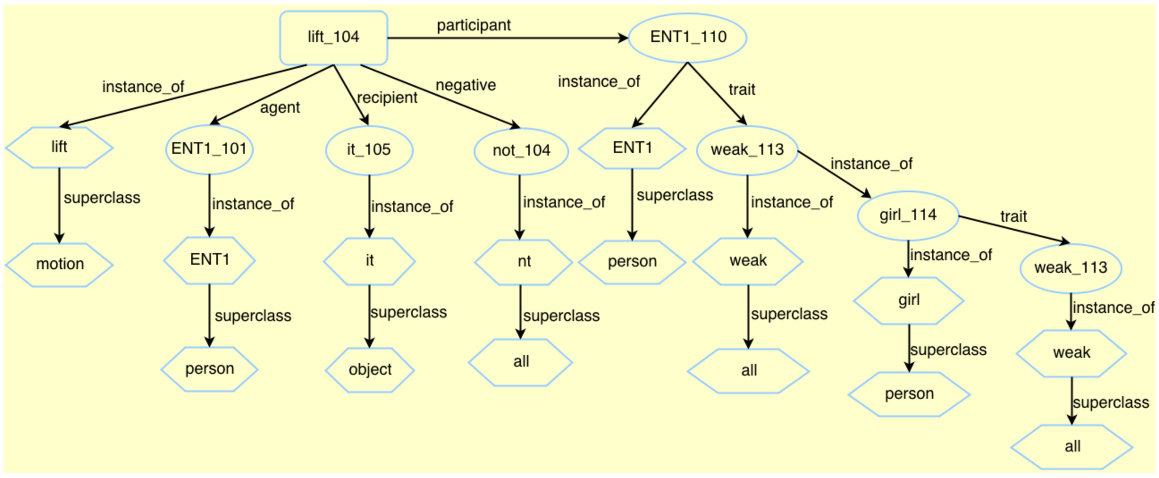


Figure 3.3: Graphical Semantic Representation of the Sentence *"She could not lift it off the floor because she is a weak girl"*

Sentence 2.

```
% She could not lift it off the floor because she is a weak girl
has(background,lift_104,agent,ent1_101).
has(background,lift_104,recipient,it_105).
has(background,lift_104,negative,not_103).
has(background,lift_104,participant,ent1_110).
has(background,lift_104,instance_of,lift).
has(background,lift,superclass,motion).
has(background,it_105,instance_of,it).
has(background,it,superclass,object).
has(background,ent1_101,instance_of,ent1).
has(background,ent1,superclass,person).
has(background,ent1_110,instance_of,ent1).
has(background,not_103,instance_of,nt).
has(background,nt,superclass,all).
has(background,weak_113,instance_of,weak).
has(background,weak,superclass,all).
has(background,ent1_110,instance_of,girl_114).
has(background,girl_114,instance_of,girl).
has(background,girl,superclass,person).
has(background,girl_114,trait,weak_113).
has(background,ent1_110,trait,weak_113).
```

In this chapter a technique is defined which automatically extracts the Background knowledge about a particular Winograd sentence and the question pertaining to the sentence. Also, the Background knowledge is translated into a formal representation similar to that of the given sentence (as explained in previous chapter). In the next chapters, Logical reasoning on formal translations of both the given Winograd sentence and the Background sentence extracted about it, is explained in detail with the help of examples. Also, the evaluation of the technique described in this work is performed and future works are discussed.

Chapter 4

LOGICAL REASONING

In Chapter 3, the process of automatic extraction of Background Knowledge is discussed in detail. How this Knowledge is translated to the formal language is also defined in chapter 3. In this chapter, the reasoning engine to match the formal representations of the given Winograd sentence, the automatically extracted Background Knowledge and the *pronoun to be resolved*, is explained in detail. The details include ASP formulation of the formal representation and a set of logical rules to simulate the reasoning.

4.1 Introduction

Until now it has been explained that when humans tackle a Natural Language Understanding problem like the Winograd Schema Challenge, they first translate the given text into a formal semantic representation. The representation allegedly is graph based because it is fairly easy to represent different concepts and their arguments in a graph based representation. After representing, the humans automatically extract the Background knowledge about the entities and events in the given text. They extract this knowledge from the knowledge that they learned from previous experiences such as reading. After the Background knowledge is extracted, it is required to use that knowledge and merge it with the given text or match both of them to infer something that was not stated in the given text. One way of doing that, which is adapted for most of the cases by humans, is to use logical reasoning on both of them and expand the knowledge.

This chapter explains the simulation of such human behavior in machines. The reasoning task is performed by converting the formal graphical representations of the given text and the automatically extracted Background Knowledge into the logic language syntax. The *pronoun to be resolved* is also translated into the logic language syntax. After that, a set of logic language rules are defined individually for both the types of Winograd sentence which are focused in this work.

4.2 Answer Set Programming

An expressive logic language i.e. Answer Set Programming Gelfond and Lifschitz (1988) (ASP) has been used to simulate the reasoning process in the system.

An ASP program is a collection of rules of the form:

$$a \leftarrow a_1, \dots, a_m, \mathit{not}\ a_{m+1}, \dots, \mathit{not}\ a_{m+n}$$

where a, $a_1, ..., a_{m+n}$ are atoms. The rule reads as "a is true if $a_1 ... a_m$ are all known to be true and $a_{m+1} ... a_{m+n}$ can be assumed to be false". The semantics of answer set programs are defined using answer sets. An entailment relation ($\models$) with respect to answer set programs is defined as follows: A program Π entails an atom p iff p is true in all the answer sets of Π.

Answer Set Programming has been chosen for the reasoning task considering the following features possessed by it

1. ASP has simple syntax but yet is expressive.

2. ASP has a strong theoretical foundation with many building-block results Baral (2003).

4.3 Reasoning Process

As explained in earlier Chapters, we focus on two types of sentences identified from Winograd Schema corpus. The types are *Direct Causal Events* and *Causal*

Attributive. For the type *Direct Causal Events* following example Winograd Schema sentence and question will be focused.

Sentence: *Ann asked Mary what time the library closes , but she had forgotten .*

Question: *Who had forgotten ?*

Answer: *Mary*

And for *Causal Attributive* type following Winograd sentence and question will be demonstrated.

Sentence: *The man could not lift his son because he was so weak.*

Question: *Who was weak ?*

Answer: *The man*

The above example Winograd sentences will be used throughout this chapter and they will be referred as *asked* and *lift* examples respectively.

Similar type of logical reasoning is used for both the types. A subgraph from the given Winograd sentence is matched with a subgraph from the automatically extracted Background sentence by looking for the entity which fills the same slot in Background sentence representation as filled by the *pronoun to be resolved* in the given Winograd sentence.

Reasoning in ASP include various aspects such as, representing the Winograd sentence and the Background sentence in ASP paradigms, defining a set of general ASP predicates and rules which are applicable for both the types of reasoning, and finally defining the predicates and ASP rules specific to a reasoning type. In the subsections below, above mentioned ASP paradigms are explained in detail.

4.3.1 Representing Winograd and Background Sentences

In previous Chapter the formal representation for representing any English text is defined. The representation is a graph of semantics in which each node represents either an entity, an event, an entity class or an event class. To use this graphical representation in ASP paradigms, each edge (including two nodes) is converted into a *has* predicate with four arguments. First argument illustrates the context of the text. For example if the text is a Winograd sentence then the context would be *winograd* whereas if the text is a Background sentence, the context is *background*.

Following ASP tuples demonstrate the ASP representation of formal representation of text.

```
context(winograd;background).
has(C,NODE1,REL,NODE2), context(C).
```

A predicate named, *context*, illustrated above is used to specify the type of sentence (Winograd or Background).

The Example Winograd Sentences mentioned above are translated to ASP predicates as depicted below. For *asked* example,

Example 1.

```
% Ann asked Mary what time the library closes , but she had
    forgotten .
has(winograd,asked_2,recipient,mary_3).
has(winograd,asked_2,agent,ann_1).
has(winograd,asked_2,next_event,forgotten_13).
has(winograd,forgotten_13,agent,she_11).
has(winograd,mary_3,instance_of,mary).
has(winograd,mary,superclass,person).
has(winograd,ann_1,instance_of,ann).
has(winograd,all,superclass,person).
has(winograd,asked_2,instance_of,ask).
has(winograd,ask,superclass,communication).
has(winograd,forgotten_13,instance_of,forget).
has(winograd,forget,superclass,cognition).
has(winograd,she_11,instance_of,she).
has(winograd,she,superclass,person).
```

The graphical representation of the above translation is illustrated in figure 4.3.1

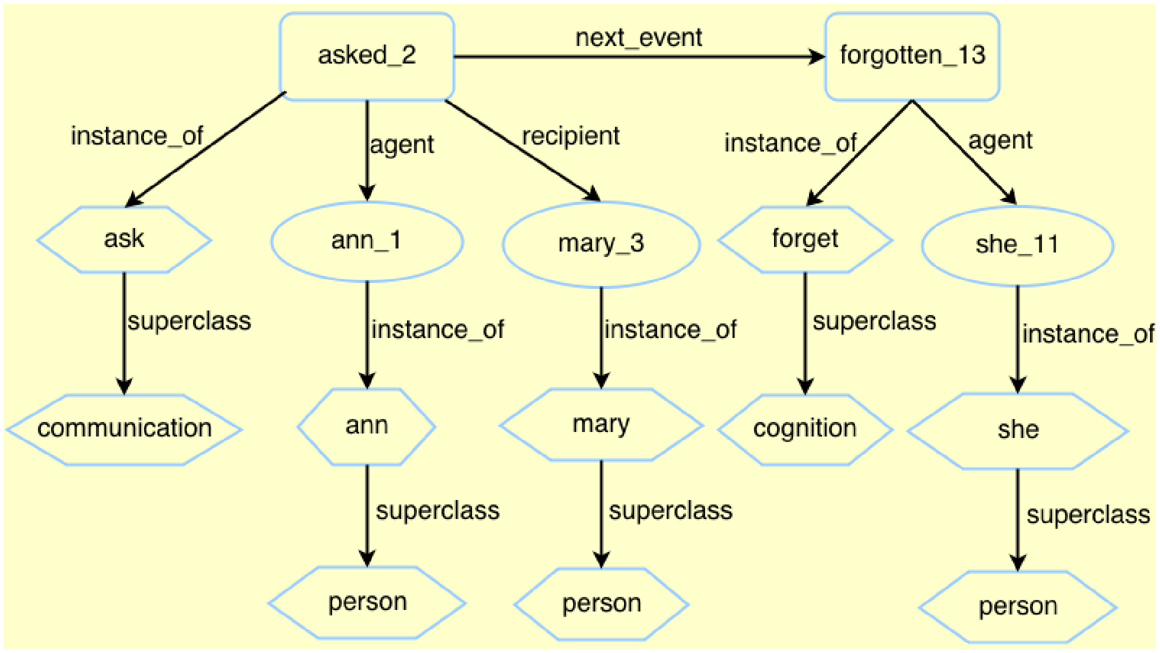


Figure 4.1: Graphical Semantic Representation of *"Ann asked Mary what time the library closes , but she had forgotten ."*

One of the corresponding background knowledge sentences which is used to answer the question related to above mentioned Winograd sentence is represented graphically in Figure 4.3.1 and its ASP formulation is illustrated below.

Background Sentence for Example 1.

```
% But you asked me the security question but I forgotten
has(background,asked_103,agent,you_102).
has(background,asked_103,recipient,ent1_104).
has(background,asked_103,object,question_107).
has(background,question_107,complement_phrase,security_106).
has(background,asked_103,next_event,forgotten_110).
```

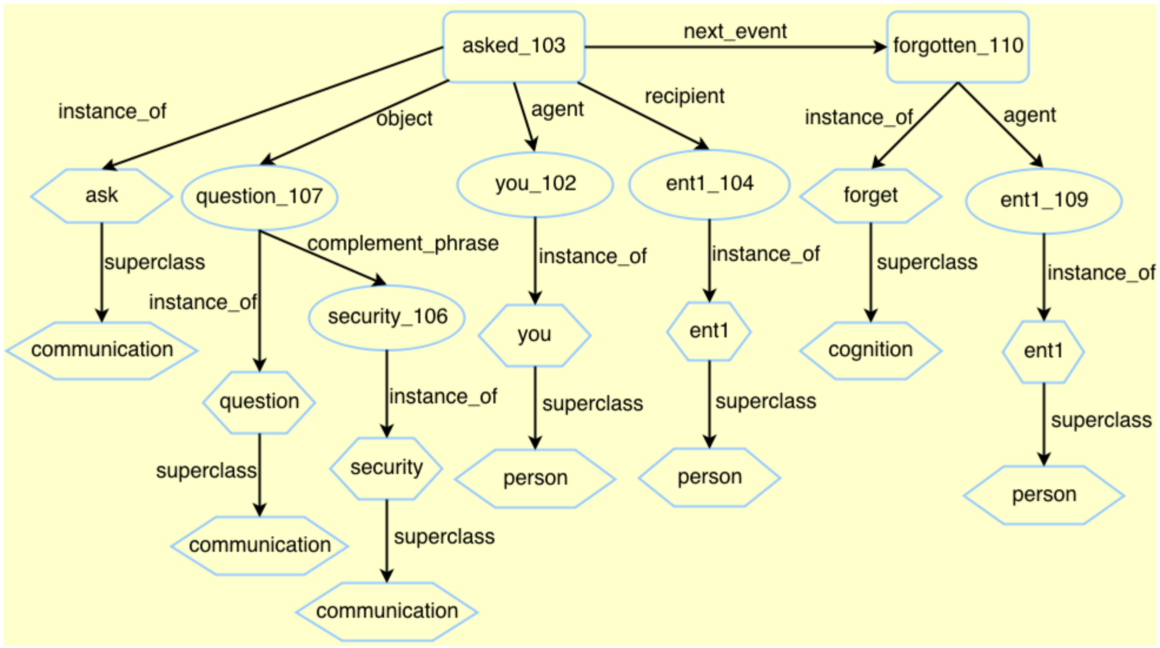


Figure 4.2: Graphical Semantic Representation of *"But you asked me the security question but I forgotten"*

```
has(background,forgotten_110,agent,ent1_109).

has(background,ent1_109,instance_of,ent1).

has(background,asked_103,instance_of,ask).

has(background,ask,superclass,communication).

has(background,you_102,instance_of,you).

has(background,you,superclass,person).

has(background,ent1_104,instance_of,ent1).

has(background,ent1,superclass,person).

has(background,question_107,instance_of,question).

has(background,question,superclass,communication).

has(background,security_106,instance_of,security).

has(background,security,superclass,communication).

has(background,forgotten_110,instance_of,forget).

has(background,forget,superclass,cognition).
```

For *lift* example,

Example 2.

```
% The man could not lift his son because he was so weak.
has(winograd,lift_5,agent,man_2).
has(winograd,lift_5,recipient,son_7).
has(winograd,son_7,possesed_by,his_6).
has(winograd,lift_5,participant,he_9).
has(winograd,he_9,trait,weak_12).
has(winograd,weak_12,trait,so_11).
has(winograd,lift_5,negative,not_4).
has(winograd,lift_5,instance_of,lift).
has(winograd,lift,superclass,motion).
has(winograd,man_2,instance_of,man).
has(winograd,man,superclass,person).
has(winograd,son_7,instance_of,son).
has(winograd,son,superclass,person).
has(winograd,his_6,instance_of,his).
has(winograd,his,superclass,person).
has(winograd,so_11,instance_of,so).
has(winograd,so,superclass,all).
has(winograd,weak_12,instance_of,weak).
has(winograd,weak,superclass,all).
has(winograd,he_9,instance_of,he).
has(winograd,he,superclass,person).
has(winograd,not_4,instance_of,not).
has(winograd,not,superclass,all).
```

The graphical representation of the above translation is illustrated in figure 4.3.1

One of the corresponding background knowledge sentences which is used to answer the question related to above mentioned Winograd sentence is represented graphically in Figure 4.3.1 and its ASP formulation is illustrated below.

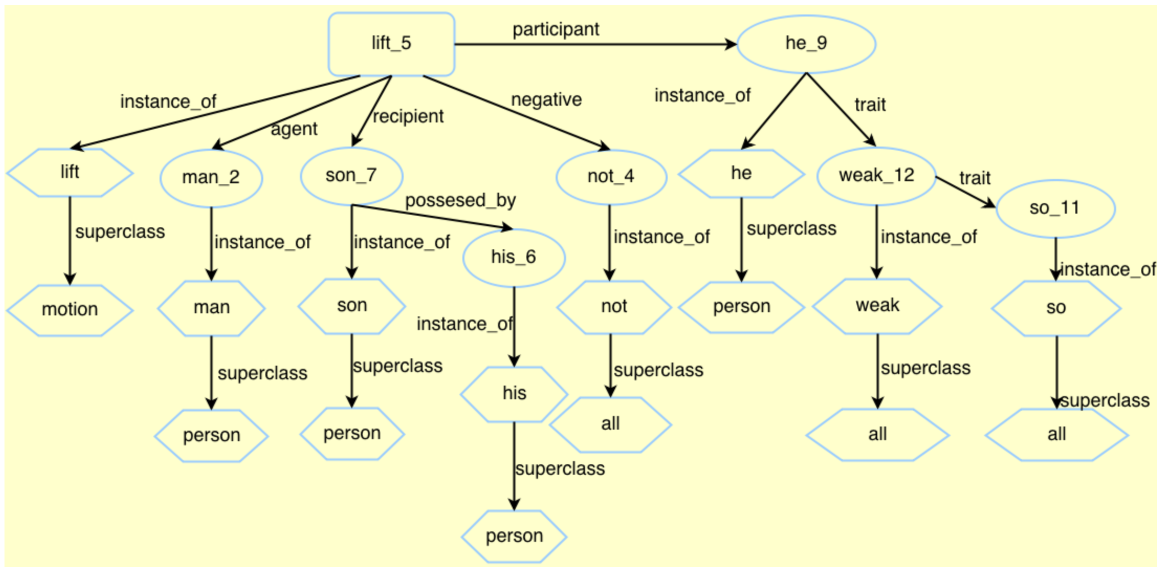


Figure 4.3: Graphical Semantic Representation of *"The man could not lift his son because he was so weak"*

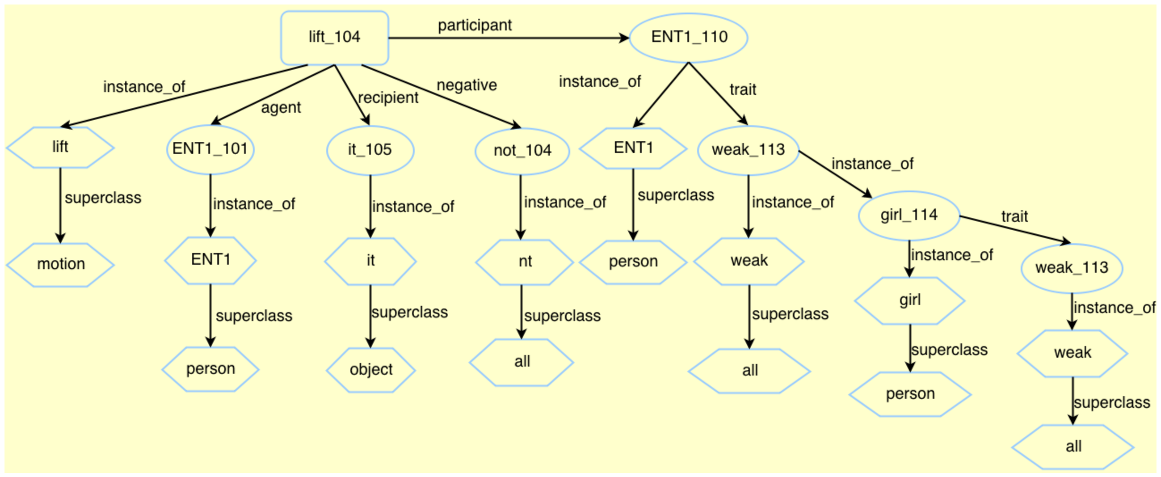


Figure 4.4: Graphical Semantic Representation of *"She could not lift it off the floor because she is a weak girl"*

Background Sentence for Example 2.

```
% She could not lift it off the floor because she is a weak girl.
has(background,lift_104,agent,ent1_101).
has(background,lift_104,recipient,it_105).
has(background,lift_104,negative,not_103).
has(background,lift_104,participant,ent1_110).
has(background,lift_104,instance_of,lift).
has(background,lift,superclass,motion).
has(background,it_105,instance_of,it).
has(background,it,superclass,object).
has(background,ent1_101,instance_of,ent1).
has(background,ent1,superclass,person).
has(background,ent1_110,instance_of,ent1).
has(background,not_103,instance_of,nt).
has(background,nt,superclass,all).
has(background,weak_113,instance_of,weak).
has(background,weak,superclass,all).
has(background,ent1_110,instance_of,girl_114).
has(background,girl_114,instance_of,girl).
has(background,girl,superclass,person).
has(background,girl_114,trait,weak_113).
has(background,ent1_110,trait,weak_113).
```

It is noticeable that each word in the above representation has an underscore at the end followed by a number. The number represents the position of the word in a sentence and helps in differentiating between words in case there are two exact same words used in the sentence in different contexts. The number in the background sentence starts from 101 whereas in the Winograd sentence it starts from 1.

The *pronoun to be resolved* is also translated into the ASP syntax by using a predicate named *toBeResolved*. It is illustrated below.

```
toBeResolved(P).
```

Where P represents the actual *pronoun to be resolved* in the given Winograd sentence. For example *she_11* and *he_9* respectively for the *asked* and *lift* example sentences mentioned above.

4.3.2 General Properties for Winograd and Background Graphs

A set of ASP predicates and rules are explained in subsections below to extract some general properties from the Winograd and the Background sentence's graphical translations.

Reachability

This property represents the basic transitivity relationship between events in a particular type of graph i.e either Winograd or Background. A set of predefined event-event relations is required to find transitivity between events. These event-event relations are taken from the KM component library as mentioned in the Semantic Parsing section of this work. These relations are represented with the help of *eventRelations* predicate. There are 23 such relations mentioned as below.
causes, caused_by, defeats, defeated_by, enables, enabled_by, inhibits, inhibited_by, by_means_of, means_by_which, first_subevent, first_subevent_of, objective, next_event, prev_event, prevents, prevented_by, resulting_state, resulting_from, subevent, subevent_of, supports, supported_by

Formal definition for extracting reachability relation is as follows

Definition 2 ($reachableFrom(C, E1, E2)$)**.** *Let* $E1$ *and* $E2$ *are event nodes in graph with context* C*.* $reachableFrom(C, E1, E2)$ *defines that:* $E2$ *is transitively reachable from* $E1$ *while traversing along any directed edge representing an event-event relation* R*, where* $R \in R_{event-event}$ *(a set of event-event relations).*

ASP rules defining above property are.

```
reachableFrom(C,E1,E2) :- has(C,E1,REL,E2),
                          context(C),
                          eventRelations(REL).

reachableFrom(C,E1,E3) :- reachableFrom(C,E1,E2),
                          has(C,E2,REL,E3),
                          context(C),
                          eventRelations(REL),
                          E1!=E2, E2!=E3.
```

An example of ASP predicates extracted from *asked* example is below.

Reachability Example 1.

reachableFrom(winograd,asked_2,next_event,forgotten_13).

Cross-Context Siblings

This property represents that if two different nodes/words in different sentences(Winograd or Background) are instances of the same conceptual class then they are siblings. The predicate *synonym* is used to represent two synonymous words. As mentioned in Chapter 3 the second set of queries, to extract Background Knowledge sentences, have synonyms of actual words used in the given Winograd sentence. The Background sentences extracted using these queries again have these synonymous sentences. So, to match these words in both the given Winograd sentence and the Background sentences extracted, new facts are added to the representations using *synonym* predicate. For example, let the given Winograd sentence has a word *lifted* and one of the Background sentence extracted for it has a corresponding word *picked* then a fact is added as

Synonym Example 1.

synonym(lift,pick).

Definition 3 ($crossContextSiblings(X, Y)$)**.** *Let X and Y are nodes in Winograd and Background graph such that one of them is in Winograd graph and another is in Background graph. $crossContextSiblings(X, Y)$ defines that: X and Y both are instances of same conceptual class T. In other words there exists an edge in both Winograd and Background graph $has(C, N, instance_of, T)$ where $C \in winograd, background$ and $N \in X, Y$.*

ASP rules defining above property are.

```
crossContextSiblings(X,Y) :- has(background,X,instance_of,T),
                             has(winograd,Y,instance_of,T),
                             X!=Y.

crossContextSiblings(Y,X) :- has(background,X,instance_of,T),
                             has(winograd,Y,instance_of,T),
                             X!=Y.

crossContextSiblings(X,Y) :- has(winograd,X,instance_of,T1),
                             has(background,Y,instance_of,T2),
                             synonym(T1,T2),
                             T1!=T2.
```

An example of ASP predicates extracted from *asked* example and its corresponding Background knowledge sentence, is illustrated below.

Cross Context Siblings Example 1.

```
crossContextSiblings(asked_2,asked_103).
```

Negative Polarity

This property represents the nodes in any context which have negative polarity i.e. preceded by a negation word such as ”not”.

Definition 4 (*negativePolarity*(E))**.** *Let E be a nodes in either Winograd or Background graph such that it has a **negative** edge originating from it. negativePolarity(E) defines that: E has a negative polarity.*

ASP rule defining above property is.

```
negativePolarity(E) :- has(C,E,negative,N1),
                       context(C).
```

An example of ASP predicates extracted from *lift* example, is illustrated below.

Negative Polarity Example 1.

```
negativePolarity(lift_5).
```

4.3.3 Type Specific ASP Rules

As mentioned in previous Chapter, this thesis focuses on solving two kinds of sentences in the Winograd corpus with the help of two kinds of ASP rules. The reasoning criteria implemented for both the types is graph matching but a different set of predicates are defined to extract different relations and properties among them. Following subsections define each of type specific ASP rules and predicates in detail with the help of examples.

Type1: Direct Causal Events

In this section, the different ASP predicates and the rules to extract those predicates are defined.

The predicate *matchingEvents(A,B,A1,B1)* defines that A and B are the reachable event nodes in the sentence graph which has A1 and B1 as *crossContextSibling*, reachable events respectively from Background sentence graph. Formally, it can be defined as

Definition 5 ($matchingEvents(A, B, A1, B1)$)**.** *Let A and B are event nodes in the Winograd sentences graph and $A1$ and $B1$ are event nodes in the Background sentence graph.$matchingEvents(A, B, A1, B1)$ defines that if there exist a relation $reachableFrom(winograd, A, B)$ and a relation $reachableFrom(A1, B1)$ where, A and $A1$ are cross-context siblings and B and $B1$ are cross-context siblings, and any one of the following conditions is satisfied then A and B have matching events $A1$ and $B1$.*

1. *A, $A1$, B and $B1$ are having negative polarity.*

2. *A, $A1$, B and $B1$ are not having negative polarity.*

3. *A and $A1$ are having negative polarity and B and $B1$ are not having negative polarity.*

4. *A and $A1$ are not having negative polarity and B and $B1$ are having negative polarity.*

Following four ASP rules are defined to extract this predicate.

```
matchingEvents(A,B,A1,B1) :- crossContextSiblings(A,A1),
                             reachableFrom(winograd,A,B),
                             crossContextSiblings(B,B1),
                             reachableFrom(background,A1,B1),
                             not negativePloarity(A),
                             not negativePloarity(A),
                             not negativePloarity(A),
                             not negativePloarity(A),
                             A!=B,A1!=B1.

matchingEvents(A,B,A1,B1) :- crossContextSiblings(A,A1),
                             reachableFrom(winograd,A,B),
                                    crossContextSiblings(B,B1),
                                    reachableFrom(background,A1,B1),
                                    not negativePloarity(A),
                                    negativePloarity(A),
                                    not negativePloarity(A),
                                    negativePloarity(A),
                                    A!=B,A1!=B1.

matchingEvents(A,B,A1,B1) :- crossContextSiblings(A,A1),
                                    reachableFrom(winograd,A,B),
                                    crossContextSiblings(B,B1),
                                    reachableFrom(background,A1,B1),
                                    negativePloarity(A),
                                    not negativePloarity(A),
                                    negativePloarity(A),
                                    not negativePloarity(A),
                                    A!=B,A1!=B1.

matchingEvents(A,B,A1,B1) :- crossContextSiblings(A,A1),
                                    reachableFrom(winograd,A,B),
                                    crossContextSiblings(B,B1),
                                    reachableFrom(background,A1,B1),
                                    negativePloarity(A),
                                    negativePloarity(A),
                                    negativePloarity(A),
                                    negativePloarity(A),
                                    A!=B,A1!=B1.
```

An example of ASP predicates extracted from *asked* example, is illustrated below.

Cross Context Matching Events Example 1.

```
matchingEvents(asked_2,forgotten_13,asked_103,forgotten_110)
```

The predicate *eventSubgraph* defines the subgraph from the Winograd sentence which contains the *pronoun to be resolved*, *the event* in which it participates and *their relation*.

Definition 6 ($eventSubgraph(winograd, A, R, X)$)**.** *Let A is an event node in Winograd sentence graph and X is the pronoun which is needed to be resolved to answer the question about the given Winograd sentence. $eventSubgraph(winograd, A, R, X)$ defines the edge from Winograd sentence graph which has A as its origin node and X as its ending node. R is any relation between A and X.*

Following ASP rules are defined to extract this predicate.

```
eventSubgraph(winograd,A,R,X) :- matchingEvents(A,B,C,D),
                                 has(winograd,A,R,X),
                                 toBeResolved(X).

eventSubgraph(winograd,B,R,X) :- matchingEvents(A,B,C,D),
                                 has(winograd,B,R,X),
                                 toBeResolved(X).
```

An example of ASP predicates extracted from *asked* example, is illustrated below.

Event Subgraph Example 1.

```
eventSubgraph(winograd,forgotten_13,agent,she_11)
```

The predicate *eventSubgraph* is also defines the subgraph from Background sentence which contains a *matchingEvent* of the event to which the *pronoun to be resolved* is related in the Winograd sentence.

Definition 7 ($eventSubgraph(background, A1, R, X1)$)**.** *Let $A1$ is an event node in the Background sentence graph. $eventSubgraph(background, A1, R, X1)$ defines the edge from the Background sentence graph which has $A1$ as its origin node and $X1$*

as its ending node. R is any relation between $A1$ and $X1$. Also, there is a relation between the subgraph extracted from the Winograd sentence graph in above step i.e.$eventSubgraph(winograd, A, R, X)$ where $A1$ and A are related with a matching nodes relation i.e. $matchingEvents(A, B, A1, B1)$.

Following ASP rules are defined to extract this predicate.

```
eventSubgraph(background,A1,R,X1) :- eventSubgraph(winograd,A,R,X),
                                     matchingEvents(A,B,A1,B1),
                                     has(background,A1,R,X1).

eventSubgraph(background,B1,R,X1) :- eventSubgraph(winograd,B,R,X),
                                     matchingEvents(A,B,A1,B1),
                                     has(background,B1,R,X1).
```

An example of ASP predicates extracted from *asked* example, is illustrated below.

Event Subgraph Example 2.

eventSubgraph(background,forgotten_110,agent,i_109)

The predicate *eventPronounRelation* is used to extract the event and relation from the Background graph. It is helpful in getting the final answer.

Definition 8 ($eventPronounRelation(background, C, R1)$)**.** *Let $eventsubgraph(background, C, R, X1)$ is the subgraph extracted from the Background sentence graph, $matchingEvents(A, B, C, D)$ is the relation extracted from both the Winograd and the Background sentence graphs, the Background sentence graph contains an edge $has(background, C, R1, X2)$ then if $X1$ and $X2$ are instances of same conceptual class X, then we extract the event-pronoun relation from the Background sentence graph as $eventPronounRelation(background, C, R1)$.*

Following ASP rules are used to extract this predicate in all possible cases.

```
eventPronounRelation(background,C,R1) :-
                                    matchingEvents(A,B,C,D),
                                    eventsubgraph(background,D,R,X1),
                                    has(background,C,R1,X2),
                                    has(background,X1,instance_of,X),
                                    has(background,X2,instance_of,X).

eventPronounRelation(background,D,R1) :-
                                    matchingEvents(A,B,C,D),
                                    eventsubgraph(background,C,R,X1),
                                    has(background,D,R1,X2),
                                    has(background,X1,instance_of,X),
                                    has(background,X2,instance_of,X).
```

An example of ASP predicates extracted from *asked* example, is illustrated below.

Event Pronoun Relation Example 1.

```
eventPronounRelation(background,asked_103,recipient)
```

Finally, the predicate *hasCoreferent* is used to extract the co-referent of the *pronoun to be resolved* from the Winograd sentence graph.

Definition 9 ($hasCoreferent(P, X)$)**.** *Let eventPronounRelation*$(background, C, R)$ *specifies the relation R extracted from the Background sentence graph along with the event node C. The edge with labeled R originates from C. We also have the matching nodes in the Winograd and Background sentence graphs i.e. matchingEvents*(A, B, C, D)*. If the Winograd sentence graph contains an edge has*$(winograd, A, S, X)$ *and P is pronoun to be resolved such that $P! = X$ then co-referent of P in the Winograd sentence is X.*

Following ASP rules are used to extract the co-referent in all possible cases.

```
hasCoreferent(P,X) :- eventPronounRelation(background,C,S),
                      matchingEvents(A,B,C,D),
                      has(winograd,A,S,X),
                      toBeResolved(P), P!=X.

hasCoreferent(P,X) :- eventPronounRelation(background,D,S),
                      matchingEvents(A,B,C,D),
                      has(winograd,B,S,X),
                      toBeResolved(P), P!=X.
```

An example of ASP predicates extracted from *asked* example, is illustrated below.

Has Co-referent Example 1.

```
hasCoreferent(she_11,mary_3)
```

From above example it is shown that the pronoun *she_11* has co-referent *mary_3*

Type2: Causal Attributive

The reasoning for this type also follows a similar graph matching pattern as explained above. The only difference is that in this case the trait/property of entities is matched from background knowledge graph instead of their relations with *matchingEvents*. In this section, the different ASP predicates and the rules to simulate this reasoning are defined.

The predicate *attributeSubgraph(winograd,E,X,A)* is defined to extract a subgraph from the Winograd sentence graph such that it contains the event to which the *pronoun to be resolved* is connected and the node which defines the property of *pronoun to be resolved*. Formally, it can be defined as

Definition 10 ($attributeSubgraph(winograd, E, X, A)$)**.** *Let E is an event node in the Winograd sentence graph, X is the pronoun to be resolved and A is the attribute as-*

sociated with X via the edge labeled trait. Then $attributeSubgraph(winograd, A, X, B)$ defines the partial subgraph from the Winograd sentence graph containing only nodes E, X and A.

Following ASP rules are defined to extract this predicate.

```
attributeSubgraph(winograd,A,X,B) :- has(winograd,A,participant,X),
                                       has(winograd,X,trait,B),
                                       toBeResolved(X).
```

An example of ASP predicates extracted from *lift* example, is illustrated below.

Attribute Subgraph Example 1.
attributeSubgraph(winograd,lift_5,he_9,weak_12)

The predicate *attributeSubgraph* is also defined for the Background sentence graph similar to the Winograd sentence graph, for extracting the corresponding nodes from the Background sentence graph.

Definition 11 ($attributeSubgraph(background, E, X1, A)$)**.** *Let $attributeSubgraph(winograd, E1, X, A1)$ is attribute subgraph extracted from the Winograd sentence graph, E and A nodes in the Background sentence graph are cross context siblings of $E1$ and $A1$ respectively. Also, either both E and $E1$ have positive polarity or both have negative polarity. If there is a subgraph $has(background, E, participant, X1)$ and an edge $has(background, X1, trait, A)$ in the Background sentence graph then $attributeSubgraph(background, E, X1, A)$ is extracted.*

Following ASP rules are defined to extract this predicate.

```
attributeSubgraph(background,E,X1,A) :-
                                 attributeSubgraph(winograd,E1,X,A1),
                                 crossContextSiblings(E1,E),
                                 crossContextSiblings(A1,A),
                                 has(background,E,participant,X1),
                                 has(background,X1,trait,A),
                                 negativePloarity(E1),
                                 negativePloarity(E).

attributeSubgraph(background,E,X1,A) :-
                                 attributeSubgraph(winograd,E1,X,A1),
                                 crossContextSiblings(E1,E),
                                 crossContextSiblings(A1,A),
                                 has(background,E,participant,X1),
                                 has(background,X1,trait,A),
                                 not negativePloarity(E1),
                                 not negativePloarity(E).
```

An example of ASP predicates extracted from *lift* example, is illustrated below.

Attribute Subgraph Example 2.

attributeSubgraph(background,lift_104,ent1_110,weak_113)

The predicate *eventPronounRelation* is defined to extract the relation between an event and possible co-referent of the *pronoun to be resolved* from the Winograd sentence graph.

Definition 12 ($eventPronounRelation(winograd, E, R)$)**.** *Let $attributeSubgraph(background, E1, X1, A)$ be the subgraph extracted from the Background sentence graph in previous step. And there is an edge in Background sentence graph $has(background, E1, R, X2)$ such that E and $E1$ are cross context siblings. If $X1$ and $X2$ are siblings in the Background sentence graph i.e. they are instances of same class, then the relation R is extracted in the predicate $eventPronounRelation(winograd, E, R)$.*

Following ASP rules are defined to extract this predicate.

```
eventPronounRelation(winograd,E,R) :-
                       attributeSubgraph(background,E1,X1,A),
                       has(background,E1,R,X2),
                       crossContextSiblings(E1,E),
                       has(background,X1,instance_of,X),
                       has(background,X2,instance_of,X).
```

An example of ASP predicates extracted from *lift* example, is illustrated below.

Event Pronoun Relation Example 2.

```
eventPronounRelation(winograd,lift_5,participant)
eventPronounRelation(winograd,lift_5,agent)
```

Finally, the predicate *hasCoreferent* is used to extract the co-referent of the *pronoun to be resolved* from the Winograd sentence graph.

Definition 13 ($hasCoreferent(P, X)$)**.** *Let* $eventPronounRelation(winograd, E, R)$ *specifies the relation R extracted from the Background sentence graph along with the event node E from Winograd sentence graph. If the Winograd sentence graph contains an edge* $has(winograd, E, R, X)$ *and P is pronoun to be resolved such that* $P! = X$ *then co-referent of P in the Winograd sentence is X.*

Following ASP rules are defined to extract this predicate.

```
hasCoreferent(P,X) :- eventPronounRelation(winograd,E,R),
                      has(winograd,E,R,X),
                      toBeResolved(P), P!=X.
```

An example of ASP predicates extracted from *lift* example, is illustrated below.

Has Co-referent Example 2.

```
hasCoreferent(he_9,man_2)
```

From above example it is shown that the pronoun *he_9* has co-referent *man_2*

Chapter 5

CONCLUSION AND FUTURE WORKS

In Chapter 4 the final component of the Co-reference resolution system was defined. The component is called the Logical Reasoning Module. It uses a logic language i.e. Answer Set Programming to reason on the given Winograd sentence graph and the automatically extracted Background knowledge sentence graph. In this chapter a conclusion is made about the need and usefulness of the technique described in this work by evaluating it on the Winograd Schema corpus. Also, the possible future works in the same field are proposed.

5.1 Summary

This work aims at solving a very important problem of Co-reference resolution. There are many state of the art Co-reference resolvers which use only statistical techniques like gender agreement between the co-referent entities, number agreement between them and the distance (in terms of words) between them in a sentence or paragraph. The main ingredient that these resolvers lack is human like thinking i.e. considering each and every word in the sentence or paragraph and then make a semantic graph of those words and use background or commonsense knowledge about the entities and events in the text to come up with a real reason of one entity being a co-referent of another. In this work a mechanism is simulated, which would follow human like thinking and reason on a problem like human beings.

Today, Artificial Intelligence has gone beyond human intelligence, for example humans can not predict future whereas AI systems such as Weather Forecast System, Search Query Completion System and other Data Mining systems are able to predict the future. While making AI better than humans we left a big gap in AI and a very less advancement has been made in the direction of making machines think like humans. For example, one of the biggest test of intelligent behavior in machines was suggested by Alan Turing in 1950 and since then there has not been a single system which passed the test with mutual consent among all the AI communities. One of the reason of this test not being passed all these years was the use of only statistical techniques to deceive humans in the test whereas other main reason as proposed by many AI scientists was that this test was not an ideal candidate for testing the intelligent behavior in machines.

Continuing on this a new test called the Winograd Schema Challenge was proposed in 2011 as an alternative to Turing test and is widely accepted. It is a co-reference resolution task and it has been designed in such a way that it is fairly easy for any human being whereas it can not be solved using just the statistical techniques. In this work, a system is proposed to solve this problem for a specific set of Winograd Schema. The mechanism used in the system follows a human like thinking and put a stepping stone in the path to the systems which would think like humans and one day would be available at our disposal for performing chores such as answering customer support questions, giving verbal commands to robots to clean our home and help catching criminals by understanding the surveillance camera feeds..

5.2 Evaluation and Error Analysis

There are 282 total sentence and question pairs in Winograd Schema Challenge corpus. Out of those, we identified a total of 100 sentences from both the categories *Causal Attributive* and *Direct Causal Events* combined. Among the 100 pairs, the system is able to answer 80 and rest 20 are left unanswered. Out of the 80 answered, 70 are correctly answered and 10 are incorrectly answered.

Table 5.1: Evaluation Results Table

Without random selection		With random selection	
Precision	Recall	Precision	Recall
87.5	80	80	100

It must be noted that there are 20 pairs which could not be answered. This is because any type of relevant background knowledge is not found for those pairs. This is an advantageous feature of the system which states that if there is no background knowledge found from one source then another one can be used and this process can be repeated many times or random selection can be done on unanswered pairs[1]. Evaluation statistics are mentioned in Table 5.1.

The reason for 10 pairs being wrongly answered is that inappropriate background knowledge is found for them. This knowledge is inappropriate because it requires a little deeper analysis of this knowledge to filter it and consider it suitable for answering the question. For example following are the Winograd sentence and the Background sentence extracted for it.

Winograd Sentence: *Bob paid for Charlie's college education, he is very grateful.*

Background Sentence: *I paid the price for my stupidity. How grateful I am.*

In the above background sentence, there is only one entity (*I, me*). This entity participates as *agent* and as a part of *recipient* of the event *paid.* It is a condition that violates the knowledge requirement of the given Winograd sentence and forces the system to provide the wrong answer.

[1]It was observed that there are 20 unanswered questions. In another test setting, those 20 sentences are answered randomly with an accuracy of 0.5 making the number of correctly answered as 80 out of 100.

5.3 Future Work

A technique was presented to parse an English sentence into a graphical semantic representation, then automatically extract background knowledge about the given sentence and the question and then reason on the given and background knowledge using Answer Set Programming (ASP) to get the answer to the question pertaining to the sentence. The technique is evaluated on a subset of the Winograd Schema Corpus, which is specifically designed by experts to simulate human like reasoning. The results look promising.

Currently two types of sentences in Winograd Schema are solved. Other types of sentences, as mentioned in previous Chapters, have also been identified. Also, the technique currently implemented represents a general case of reasoning in any Natural Language Understanding problem such as Co-reference Resolution, Machine Reading, Reading Comprehension and Deep Question Answering. In this work the usability of this technique on a subset of The Winograd Schema Challenge has been demonstrated. The problem focused in this work also represents a case of the real world reasoning. It is mentioned that there are many categories in which the Winograd corpus can be divided. Two of which are focused in this work, and it is believed that this technique can be used for others with some modifications or additions. For example a category in the Winograd Schema corpus as observed contain chains of causal events and the solution depends on the events far from each other in the chain. This case can be considered as an extended version of one of the subcategory that currently solved (i.e. Direct Causal Events). The task of solving this category and applying the technique to other problems is one of the future directions of this work.

Furthermore, a mechanism to filter the background knowledge which is responsible for providing wrong answers (as explained in the section above) is another future work of this work.

REFERENCES

ADASHCHIK, A., "Eugene-goostman", URL http://www.princetonai.com/ (2014).

Alias-i.2008, "Lingpipe 4.1.0.", URL http://alias-i.com/lingpipe (2008).

Baral, C., *Knowledge representation, reasoning and declarative problem solving* (Cambridge university press, 2003).

Baral, C. and S. Liang, "From knowledge represented in frame-based languages to declarative representation and reasoning via asp.", in "KR", (2012).

Baral, C., N. H. Vo and S. Liang, "Answering why and how questions with respect to a frame-based knowledge base: a preliminary report.", in "ICLP (Technical Communications)", pp. 26–36 (2012).

Barker, K., B. Porter and P. Clark, "A library of generic concepts for composing knowledge bases", in "Proceedings of the 1st international conference on Knowledge capture", pp. 14–21 (ACM, 2001).

Basile, P., M. Degemmis, A. L. Gentile, P. Lops and G. Semeraro, "The jigsaw algorithm for word sense disambiguation and semantic indexing of documents", in "AI* IA 2007: Artificial Intelligence and Human-Oriented Computing", pp. 314–325 (Springer, 2007).

Bengtson, E. and D. Roth, "Understanding the value of features for coreference resolution", in "Proceedings of the Conference on Empirical Methods in Natural Language Processing", pp. 294–303 (Association for Computational Linguistics, 2008).

Berant, J. and P. Liang, "Semantic parsing via paraphrasing", in "Proceedings of ACL", (2014).

Broscheit, S., M. Poesio, S. P. Ponzetto, K. J. Rodriguez, L. Romano, O. Uryupina, Y. Versley and R. Zanoli, "Bart: A multilingual anaphora resolution system", in "Proceedings of the 5th International Workshop on Semantic Evaluation", pp. 104–107 (Association for Computational Linguistics, 2010).

Chambers, N. and D. Jurafsky, "Unsupervised learning of narrative event chains.", in "ACL", pp. 789–797 (Citeseer, 2008).

Chaudhri, V. K., P. E. Clark, S. Mishra, J. Pacheco and A. Spaulding, "Aura: Capturing knowledge and answering questions on science textbooks", (2009).

Chaudhri, V. K. and T. C. Son, "Specifying and reasoning with underspecified knowledge bases using answer set programming.", in "KR", (2012).

Clark, P., B. Porter and B. P. Works, "Kmthe knowledge machine 2.0: Users manual", Department of Computer Science, University of Texas at Austin (2004).

computer therapist, E., "computer therapist", URL `http://www.manifestation.com/neurotoys/eliza.php3` (1966).

Das, D., N. Schneider, D. Chen and N. A. Smith, "Semafor 1.0: A probabilistic frame-semantic parser", Language Technologies Institute, School of Computer Science, Carnegie Mellon University (2010).

Davis, E., "The singularity and the state of the art in artificial intelligence", (2013).

De Marneffe, M.-C., B. MacCartney, C. D. Manning *et al.*, "Generating typed dependency parses from phrase structure parses", in "Proceedings of LREC", vol. 6, pp. 449–454 (2006).

De Marneffe, M.-C. and C. D. Manning, "Stanford typed dependencies manual", URL http://nlp. stanford. edu/software/dependencies manual. pdf (2008).

Diakidoy, I.-A., A. Kakas, L. Michael and R. Miller, "Narrative text comprehension: From psychology to ai", in "Proc. of 11th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense?13)", (2013).

Finkel, J. R., T. Grenager and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling", in "Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics", pp. 363–370 (Association for Computational Linguistics, 2005).

Ge, N., J. Hale and E. Charniak, "A statistical approach to anaphora resolution", in "Proceedings of the sixth workshop on very large corpora", vol. 71 (1998).

Gelfond, M. and V. Lifschitz, "The stable model semantics for logic programming.", in "ICLP/SLP", vol. 88, pp. 1070–1080 (1988).

Gilbert, N. and E. Riloff, "Domain-specific coreference resolution with lexicalized features.", in "ACL (2)", pp. 81–86 (Citeseer, 2013).

Gunning, D., V. K. Chaudhri, P. E. Clark, K. Barker, S.-Y. Chaw, M. Greaves, B. Grosof, A. Leung, D. D. McDonald, S. Mishra *et al.*, "Project halo updateprogress toward digital aristotle", AI Magazine **31**, 3, 33–58 (2010).

Hermann, K. M., D. Das, J. Weston and K. Ganchev, "Semantic frame identification with distributed word representations", in "Proceedings of ACL", (2014).

Hobbs, J. R., "Pronoun resolution. technical report 76-1", (1976).

Lee, H., Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu and D. Jurafsky, "Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task", in "Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task", pp. 28–34 (Association for Computational Linguistics, 2011).

Levesque, H. J., "On our best behaviour", Artificial Intelligence **212**, 27–35 (2014).

Levesque, H. J., E. Davis and L. Morgenstern, “The winograd schema challenge.”, in “AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning”, (2011).

Mauldin, M. L., “Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition”, in “AAAI”, vol. 94, pp. 16–21 (1994).

Miller, G. A., “Wordnet: a lexical database for english”, Communications of the ACM **38**, 11, 39–41 (1995).

Mitkov, R., “Factors in anaphora resolution: they are not the only things that matter: a case study based on two different approaches”, in “Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts”, pp. 14–21 (Association for Computational Linguistics, 1997).

Ng, V. and C. Cardie, “Improving machine learning approaches to coreference resolution”, in “Proceedings of the 40th Annual Meeting on Association for Computational Linguistics”, pp. 104–111 (Association for Computational Linguistics, 2002).

Raghunathan, K., H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky and C. Manning, “A multi-pass sieve for coreference resolution”, in “Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing”, pp. 492–501 (Association for Computational Linguistics, 2010).

Rahman, A. and V. Ng, “Coreference resolution with world knowledge”, in “Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1”, pp. 814–824 (Association for Computational Linguistics, 2011).

Rahman, A. and V. Ng, “Resolving complex cases of definite pronouns: the winograd schema challenge”, in “Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning”, pp. 777–789 (Association for Computational Linguistics, 2012).

Schuller, P., “Tackling winograd schemas by formalizing relevance theory in knowledge graphs”, (2014).

Yao, X. and B. Van Durme, “Information extraction over structured data: Question answering with freebase”, in “Proceedings of ACL”, (2014).