Towards Understanding Natural Language: Semantic Parsing, Commonsense Knowledge Acquisition, Reasoning Framework and Applications

by

Arpit Sharma

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved June 2019 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Joohyung Lee
Paolo Papotti
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

Reasoning with commonsense knowledge is an integral component of human behavior. It is due to this capability that people know that a weak person may not be able to lift someone. It has been a long standing goal of the Artificial Intelligence community to simulate such commonsense reasoning abilities in machines. Over the years, many advances have been made and various challenges have been proposed to test their abilities. The Winograd Schema Challenge (WSC) is one such Natural Language Understanding (NLU) task which was also proposed as an alternative to the Turing Test. It is made up of textual question answering problems which require resolution of a pronoun to its correct antecedent.

In this thesis, two approaches of developing NLU systems to solve the Winograd Schema Challenge are demonstrated. To this end, a semantic parser is presented, various kinds of commonsense knowledge are identified, techniques to extract commonsense knowledge are developed and two commonsense reasoning algorithms are presented. The usefulness of the developed tools and techniques is shown by applying them to solve the challenge.

DEDICATION

To mom, dad and my wife Megha

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Chitta Baral for providing me the opportunity to work on this research and guiding me throughout.

I would like to thank Dr. Joohyung Lee, Dr. Paolo Papotti and Dr. Yezhou Yang for being a part of my thesis defense committee and providing me their valuable feedback on my work.

I would like to thank my collaborators Dr. Davy Weissenbacher and Dr. Nguyen Vo Ha for their valuable guidance and support in various projects. I would also like to thank Dr. Saadat Anwar for his technical and moral support when I was facing difficult times in my research.

I would like to thank my peers in Dr. Chitta Baral's lab at ASU for their valuable feedback on my works.

Finally, I would like to thank my family and friends for their continuous support throughout the years.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation

Natural Language Understanding (NLU) is an important aspect of Artificial Intelligence. It is because of NLU components in the virtual assistants such as Amazon Alexa [1] that we can switch off/on our bedroom lights by just talking to our phones. In simple words, an NLU based system is able to comprehend natural language text or speech and it is able to act according to the instructions in the comprehended input. To understand natural language text it is often important to have additional knowledge related to that text. It is also important to be able to reason with the knowledge. For example, let us consider the sentence, *"The man could not lift his son because he was so weak."* and the question *"Who was weak?"* about the sentence. Then we can say that an NLU system understands the sentence if it is able to correctly answer the question. To be able to correctly answer the question, an NLU system needs the additional knowledge that *"someone weak may not be able to lift someone else"* and the ability to reason with it.

Recently, various tasks (Rajpurkar *et al.*, 2018; Xu *et al.*, 2016; Chang, 2016; Antol *et al.*, 2015) have been proposed to test the capabilities of NLU systems. Machine reading comprehension is one such task. Several large-scale datasets (Joshi *et al.*, 2017; Trischler *et al.*, 2016; Rajpurkar *et al.*, 2016) have been created for reading comprehension. These datasets have led to a wide variety of model architectures (Huang *et al.*, 2017; Clark and Gardner, 2017). Several systems have even surpassed human-level accuracies on certain datasets (Rajpurkar *et al.*, 2018). Even then these systems are still far from the real nat-

[1] `https://developer.amazon.com/alexa`

ural language understanding because they do not explicitly focus on the use of additional knowledge and reasoning with the knowledge. For example, the recent analysis showed that models which could do well at Stanford Question Answering Dataset (SQuAD) (Rajpurkar *et al.*, 2016) version 1.1 by learning context and type-matching heuristics (Weissenborn *et al.*, 2017), did not perform very well on the SQuAD 2.0 (Rajpurkar *et al.*, 2018) dataset which intentionally contains the questions that can not be answered with respect to a given passage. The systems which do well on SQuAD 2.0 are trained on large amount of examples and do not explicitly focus on additional knowledge and reasoning with the knowledge. This makes them not so useful for the datasets which specially focus on reasoning with respect to additional knowledge and the ones which do not contain large amount of training samples. The Winograd Schema Challenge (WSC) (Levesque *et al.*, 2011) corpus is one such corpus.

Various NLU systems which focus on additional knowledge and reasoning with the knowledge are made up of the following components.

1. **Semantic Parsing Component:** Semantic parsing refers to the task of translating a natural language text into a formal representation. Lately, there has been a new interest in the semantic parsing field with the introduction of Abstract Meaning Representation (AMR) (Banarescu *et al.*, 2012) but the low performance of the AMR parsing systems (Flanigan *et al.*, 2016; Wang *et al.*, 2016) is still a concern in terms of the progress in the field.

2. **Commonsense Knowledge Extractor:** There are several works which extract many different kinds of commonsense knowledge from text (Singh *et al.*, 2002; Allen *et al.*, 2013; Aharon *et al.*, 2010; Chambers and Jurafsky, 2008). Many times, the extraction of knowledge is motivated by its need in the task at hand, which makes the extraction process somewhat task specific. Also, most of the tasks do not require one kind

of knowledge and there must be a way to differentiate those kinds so that another application which requires a particular kind of knowledge can focus only on a subset of knowledge. The knowledge is useful if there is another task that requires it but it becomes useless otherwise. Recent trend of crowd-sourcing knowledge (Sap *et al.*, 2018) is promising, however the process is expensive.

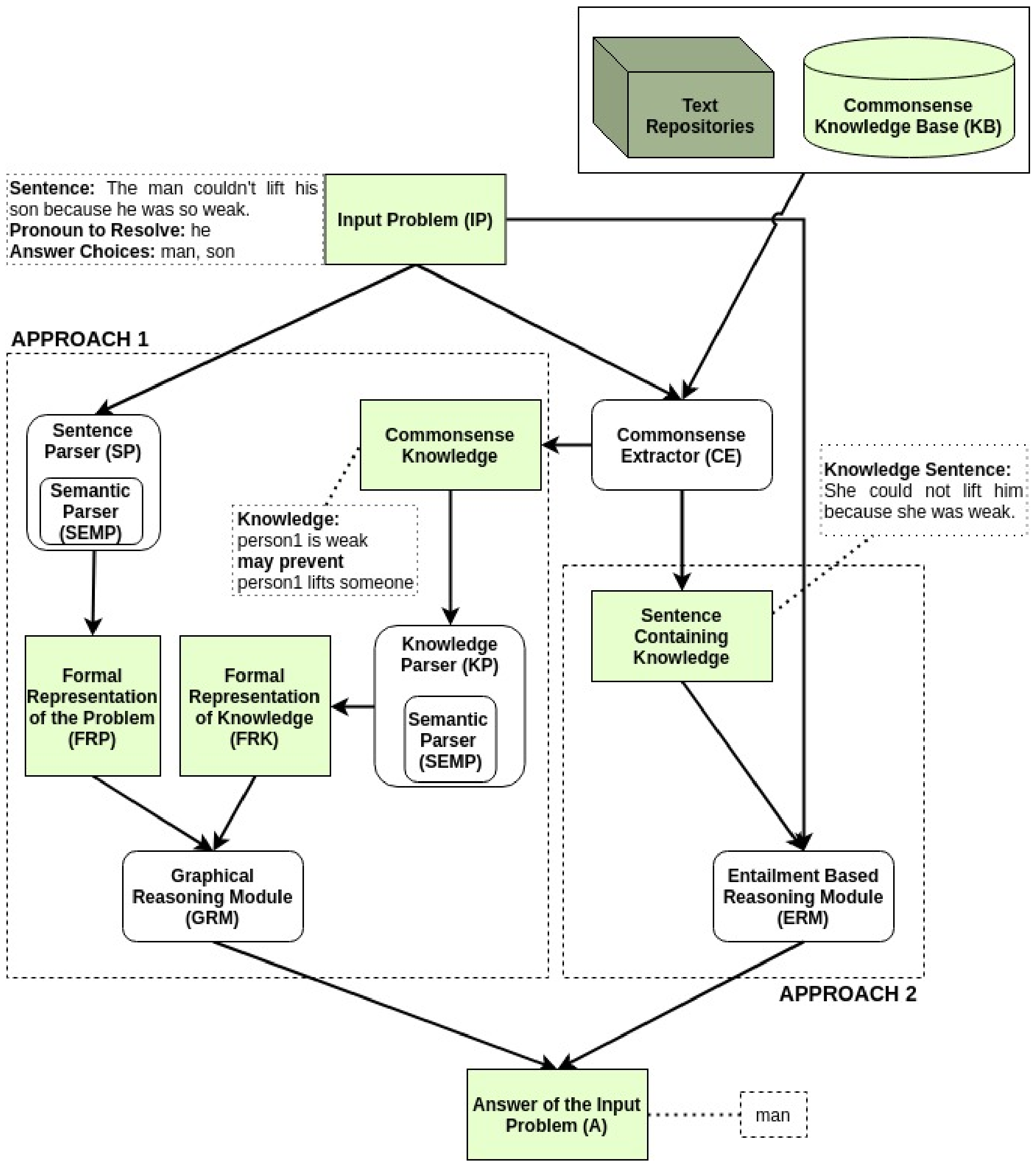
3. **Reasoning with Knowledge:** There are various works which focus on the aspect of reasoning with commonsense knowledge (Bailey *et al.*, 2015; Schüller, 2014). Though these works provide foundations of reasoning frameworks, they rely on a big assumption that the required knowledge is available in their desired format.

Considering the challenges and limitations in the above mentioned components, our goal in this work is to progress towards the development of an automated system to solve a Natural Language Understanding (NLU) problem that requires co-reference resolution. We aim to do that by making progress towards semantic parsing, automatic commonsense knowledge extraction from text, and reasoning with commonsense knowledge.

In this dissertation we present our progress towards the goal. We demonstrate the steps we took and the tools/techniques we developed. We present improvements in a semantic parser, different kinds of newly identified knowledge, two algorithms to reason with commonsense knowledge and techniques/attempts to automatically acquire commonsense knowledge from text. We also show the usefulness of the developed tools and techniques by applying them to solve the Winograd Schema Challenge (WSC) (Levesque *et al.*, 2011). WSC is an NLU task such that the problems in it require reasoning with commonsense knowledge to perform pronoun resolution.

1.2 Contributions of the Research

The Figure 1.1 provides an overview of our two commonsense reasoning approaches. The approaches are designed to solve the Winograd Schema Challenge (WSC) but it can be viewed as a combination of different modules that will be helpful in various NLU tasks. A brief overview of all the modules and their outputs at different stages is provided in the sections below.



Text Repositories
Commonsense Knowledge Base (KB)
Sentence: The man couldn't lift his son because he was so weak.
Pronoun to Resolve: he
Answer Choices: man, son
Input Problem (IP)
APPROACH 1
Sentence Parser (SP)
Semantic Parser (SEMP)
Commonsense Knowledge
Commonsense Extractor (CE)
Knowledge Sentence: She could not lift him because she was weak.
Knowledge: person1 is weak may prevent person1 lifts someone
Formal Representation of the Problem (FRP)
Formal Representation of Knowledge (FRK)
Knowledge Parser (KP)
Semantic Parser (SEMP)
Sentence Containing Knowledge
Graphical Reasoning Module (GRM)
Entailment Based Reasoning Module (ERM)
APPROACH 2
Answer of the Input Problem (A)
man

Figure 1.1: Overall System:

(SEMP) Semantic Parser. (CE) Commonsense Extractor. (GRM) Graphical Reasoning Module. (ERM) Entailment Based Reasoning Module.

- **Input Problem (IP):** This is the input to each of the commonsense reasoning approaches. It consists of a sequence of one or more sentences (from a WSC problem), a pronoun that is needed to be resolved in the sentences and two answer choices. The answer choices are noun phrases in the sentences. An example of an input problem is shown in the Figure 1.1.

- **Semantic Parser (SEMP):** This is a semantic parsing system which translates a piece of text into a graphical semantic representation. The parser is used in Sentence Parser (SP) and Knowledge Parser (KP) modules as shown in the Figure 1.1. The SP and KP modules are respectively used to translate an input problem and a commonsense knowledge into formal graphical representations. The representations generated by SP and KP are used in the reasoning module, as shown in the Figure 1.1, of the first approach to solve the input problem. The semantic parser is a general purpose module and it can be used to translate an English text into its meaning representation. A demonstration of the semantic parser is available online at www.kparser.org. We have three published works
(Sharma *et al.*, 2015d,b,c) explaining the implementation and application details of our semantic parser. More on SEMP is present in the Chapter 6 of this dissertation.

- **Commonsense Extractor (CE):** This module extracts different kinds of commonsense knowledge from large unstructured text repositories. The kinds of knowledge extracted by this module are manually identified by analyzing the WSC corpus. We have one published work (Sharma and Baral, 2016) on extracting commonsense knowledge from the text. Attempts have been made to extract different kinds of commonsense knowledge from various text repositories. The details of the attempts are present in Chapter 5. The CE module has two facets. In the first facet a WSC problem is taken as input and the needed knowledge and a piece of text which may contain the needed knowledge are extracted, from a text repository or a search engine. In the second facet a text repository is taken as input and a knowledge base of predefined knowledge types is produced as output. As shown in the Figure 1.1, both of our approaches towards commonsense reasoning utilize the outputs of the CE module. A detailed description of this module is available in the Chapter 5 of this dissertation.

- **Graphical Reasoning Module (GRM):** This module uses the formal graphical representations of an input WSC problem and a commonsense knowledge to deduce an answer to the input problem. It is the main reasoning component of the first approach towards solving the WSC. We used Answer Set Programming to code the graphical reasoning module (or graphical reasoning algorithm). The reasoning algorithm is general and can be easily implemented in any other high level programming language. A detailed description of this module is available in the Chapter 4 of this dissertation.

- **Entailment Based Reasoning Module (ERM):** This module uses the WSC sentences, pronoun to resolve, two answer choices and a sentence which may contain the needed commonsense knowledge to deduce an answer to the input problem. In this approach, we use a pre-trained Natural Language Inference (NLI) [2] system to predict which answer choice is more likely to be the answer of the input problem. More details about this approach are present in the Chapters 2 and 7.

Below is the list of publications reflecting the contributions of this dissertation:

- IJCAI 2015 (Sharma et al., 2015): *Towards Addressing the Winograd Schema Challenge - Building and Using a Semantic Parser and a Knowledge Hunting Module.*

[2]See `http://nlpprogress.com/english/natural_language_inference.html`

- NAACL 2015 Workshop (Sharma et al., 2015): *Identifying Various Kinds of Event Mentions in K-Parser Output.*

- AAAI 2015 Spring Symposium (Sharma et al., 2015): *An Approach to Solve Winograd Schema Challenge Using Automatically Extracted Commonsense Knowledge.*

- AAAI 2016 Workshop (Sharma et al., 2016): *Automatic Extraction of Events-Based Conditional Commonsense Knowledge.*

- IJCAI 2016 DC (Sharma, 2016): *Towards Understanding Natural Language: Semantic Parsing, Commonsense Knowledge Acquisition and Applications*

- ACL 2019 (Prakash, Sharma, Mitra, Baral 2019): *Combining Knowledge Hunting and Neural Language Models to Solve the Winograd Schema Challenge*

- ICLP 2019 (Sharma, 2019): *Using Answer Set Programming for Commonsense Reasoning in the Winograd Schema Challenge*

Above points presented the brief overview of the overall system. There are various underlying submodules, tools and techniques that are used to develop each module of the system. The intricate details are explained in different chapters of this dissertation. The following section provides an introduction to each chapter in this dissertation.

1.3 Dissertation Organization

Following is the list of chapters in the dissertation:

- Chapter 1 provides an introduction to the Natural Language Understanding, explains the motivation for the research and lists the contributions of the dissertation towards solving an NLU problem called the Winograd Schema Challenge.

- Chapter 2 provides a glimpse of our two commonsense reasoning approaches with the help of worked out examples from the Winograd Schema Challenge corpus.

- Chapter 3 lists and explains the different types of commonsense knowledge identified from the Winograd Schema Challenge corpus that are required to solve the challenge.
- Chapter 4 explains the implementation of a graphical reasoning module which is used to reason with the different kinds of commonsense knowledge identified from the Winograd Schema Challenge corpus.
- Chapter 5 shows the details of the different methods and algorithms used to extract the commonsense knowledge identified in Chapter 3. This chapter lists the attempts towards commonsense knowledge extraction from text.
- Chapter 6 provides the detailed implementation and improvements in a semantic parser called Knowledge Parser (or K-Parser) system which translates English sentences into directed acyclic semantic graphs. The chapter also provides the details of how the K-Parser is used to translate a WSC problem and a needed knowledge into graphical representations that are defined in the Chapter 4.
- Chapter 7 provides the detailed description of the end-end systems based on our two commonsense reasoning approaches.
- Chapter 8 concludes by looking at the contributions of the dissertation and discussing future directions for this work.

Chapter 2

SOLVING THE WINOGRAD SCHEMA CHALLENGE: A GLIMPSE OF OUR TWO APPROACHES

2.1 Introduction

With significant advances and success in many Artificial Intelligence subfields, and instances of acing the Turing test (Turing, 1950), there is a concern about the ability of the Turing test to correctly evaluate if a system exhibits human-like intelligence. There is now a need to more clearly define how to evaluate when a system is replicating humanoid intelligence [1] . The Winograd Schema Challenge (WSC) (Levesque *et al.*, 2011) is one such attempt. WSC was also proposed as an alternative to the Turing Test. It is made up of special types of pronoun resolution problems. Each WSC problem consists of a sequence of one or more sentences which contain a definite pronoun. For example,

The fish ate the worm. $\textbf{\textit{It}}_{pronoun}$ *was tasty*

A WSC problem also contains a binary question about the sentences such that the answer to the question provides the most natural resolution for the pronoun (e.g., *What was tasty?*). Additionally, two answer choices (noun phrases) for the question are also provided (e.g., *fish* and *worm*). The answer choices are present in the sentences. The goal in the WSC challenge is to determine the correct answer choice. Following is an example WSC problem.

[1]See also http://www.newyorker.com/tech/elements/why-cant-my-computer-understand-me

Example 1:

Sentences: The man couldn't lift his son because **he**$_{pronoun}$ was so heavy.

Question: Who was heavy? **Answer Choices:** a) man b) son

Each WSC problem also contains a "special word" in the paragraph, and an "alternate word." Replacing the former by the latter changes the resolution of the pronoun to be resolved. In the example above, the special word is *heavy* and the alternate word is *weak*, which results in the following WSC problem.

Example 2:

Sentences: The man couldn't lift his son because **he**$_{pronoun}$ was so weak.

Question: Who was weak? **Answer Choices:** a) man b) son

The motivation behind this challenge is to evaluate human-like intelligence in machines. A human-like intelligent system that can answer such questions correctly requires to have commonsense knowledge that is not explicitly mentioned in the winograd schema sentences. Very recently, this aspect of human-like intelligence has also been emphasized in several other (besides the Winograd challenge) research initiatives such as the call by Paul G. Allen Family Foundation [2] , and the Big Mechanism call of DARPA. Since the existing knowledge repositories (such as CYC, ConceptNet, and AURA) are not comprehensive enough, and a comprehensive knowledge base may be too unwieldy, we propose two approaches to address this aspect of human-like intelligence. Both the approaches are based on hunting the needed knowledge as plain English text and then reasoning with it. In the following sections, we provide an overview of each of these approaches with the help of worked out examples.

[2]http://www.pgafamilyfoundation.org/programs/investigators-fellows/key-initiative/adi-artificial-intelligence-rfp

2.2 Related Challenges

Similar to the WSC, there are various NLU challenges (Roemmele *et al.*, 2011; Weston *et al.*, 2015; Rajpurkar *et al.*, 2016, 2018) which are used to test the human-like natural language understanding capabilities in machines. For example, Stanford Question Answering Dataset (SQuAD) (Rajpurkar *et al.*, 2016) is a reading comprehension dataset which contains more than 100K questions written by crowdworkers on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding reading passage. Following is a problem from the SQuAD.

Passage:
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

Question 1: *What causes precipitation to fall?* **Answer:** *gravity*
Question 2: *What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?* **Answer:** *graupel*
Question 3: *Where do water droplets collide with ice crystals to form precipitation?* **Answer:** *within a cloud*

There are similarities between the SQuAD and WSC dataset. For example, both of them contain natural language question answering problems, and answering questions in both of them require understanding natural language text. Despite being similar in some aspects, there are differences in the datasets which affect the development of NLU systems to solve these challenges. For instance, SQuAD contains more than 100K problems whereas the latest version of the WSC dataset only contains about 300 problems. The unavailability of large amount of training data makes it difficult to develop machine learning based systems for the WSC.

SQuAD2.0 (Rajpurkar *et al.*, 2018) is the updated version of SQuAD (SQuAD1.1). It contains all the 100K questions which are in SQuAD1.1 and it also contains over 50K unanswerable questions written adversarially by crowdworkers. To perform well on SQuAD2.0, a system must not only answer questions when possible, but also determine when a question can not be answered with respect to the corresponding passage. Following is a problem from the SQuAD2.0.

Passage:
The 8- and 10-county definitions are not used for the greater Southern California Megaregion, one of the 11 megaregions of the United States. The megaregion's area is more expansive, extending east into Las Vegas, Nevada, and south across the Mexican border into Tijuana.

Question 1: *Which border does the megaregion extend over?* **Answer:** *Mexican*
Question 2: *How many megaregions are there in the United States?* **Answer:** *11*
Question 3: *What is Las Vegas one of in the United States?* **Answer:** *<No Answer>*

Various approaches have been proposed to solve the SQuAD1.1 and SQuAD2.0 datasets. A complete list of the approaches is available on the leaderboard [3] for the challenge. Since SQuAD contain reading comprehension problems where an input problem contains a read-

[3] `https://rajpurkar.github.io/SQuAD-explorer/`

ing passage and a list of questions about it, the approaches to solve the datasets can very well be used to solve the WSC dataset. This is because each WSC problem can also be viewed as a reading comprehension problem where a sequence of sentence in a WSC problem corresponds to a reading passage and the question about the sentences corresponds to a question in SQuAD. A limitation of using a SQuAD solver for WSC is that there are very few training examples available in the WSC corpus.

Another challenge which is similar to the WSC is Choice Of Plausible Alternatives (COPA) (Roemmele *et al.*, 2011) challenge. Each problem in the COPA challenge contains a premise and two alternative choices, out of which one is more plausible than the other. The goal is to select the more plausible choice based on the causal relationship (as defined by a question) between the premise and the alternative choices. An example COPA problem is as shown below.

Premise: *The man broke his toe.* **What was the CAUSE of this?**

Alternative 1: *He got a hole in his sock.*

Alternative 2: *He dropped a hammer on his foot.*

Similar to many problems in the WSC dataset, each problem in the COPA challenge requires causal commonsense knowledge. A difference between the COPA challenge and the WSC is that both the choices in the COPA challenge are plausible but only the more plausible one is deemed as the correct choice whereas in WSC only one of the answer choice is plausible. In terms of dataset size, both COPA and WSC contains small number of problems which does not allow data intensive techniques to be easily applied to them. COPA contains 1000 problems and WSC contains about 300 problems.

2.3 Approach 1: Semantic Parsing and Graphical Reasoning Approach

In this approach, a WSC problem and a needed knowledge is formally represented and a reasoning algorithm is defined which outputs the answer to the problem if it is entailed by the formal representations. The formal representation is a graphical semantic representation. In other words, this approach involves semantic parsing of the natural language text, identification and extraction of commonsense knowledge that is needed to solve the problem, and a reasoning framework that uses the semantic representation of the WSC problem, and the commonsense knowledge to deduce the answer to the problem.

The sections below provide a brief overview of the different steps in the approach.

2.3.1 Step 1: Formal Representation of a WSC Problem

The sentences in a WSC problem are formally represented as a graph. The nodes in such a graph represent the concepts in the WSC sentences and the edge labels represent the semantic relations between the concepts. An example of a graphical representation of the sentences in the WSC problem mentioned in the Example 2 above is shown in Figure 2.1 . The detailed description of the graphical representation of English sentences in a WSC problem is provided in the Chapter 4 of this thesis. In this work, we developed a semantic parser called Knowledge Parser (K-Parser). We used K-Parser to translate the sentences in a WSC problem to their graphical representation. The details of the K-Parser algorithm and its implementation are provided in the Chapter 6 of this thesis.

2.3.2 Step 2: Commonsense Knowledge Extraction and Representation

In this work, a commonsense knowledge corresponding to a WSC problem is extracted from text repository by following a sequence of steps detailed in the Chapter 5.

1. Creating search queries by using the input WSC problem. A set of search queries are created by selecting important words from the sentences in an input WSC problem.

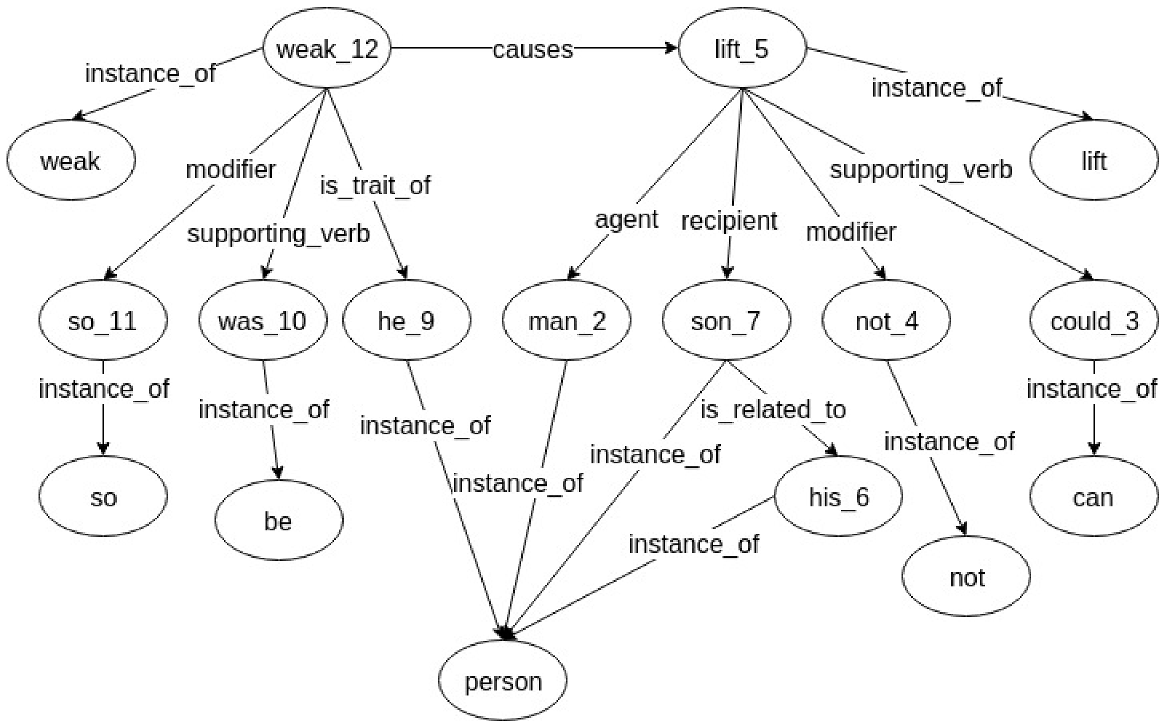


Figure 2.1: K-Parser Output for the Sentence *"The man could not lift his son because he was so weak."*

For example, a query generated from the WSC sentence *"The man could not lift his son because he was so weak."* is *" * could not lift * because * weak * "*

2. A set of text snippets are extracted from Google search engine by using the queries created in the Step 1. For example, a text snippet for the query mentioned in the Step 1 is *"She could not lift him because she was weak."*

3. From each of the text snippets extracted in the Step 3, a knowledge is extracted (if present). For example, the snippet shown in Step 2 is used to extract the knowledge *"person1 is weak may prevent person1 lifts someone"*

The extracted knowledge is also represented as a graph. We used K-Parser to translate a knowledge into the graph. The details of the representation are provided in the Chapter 4

An example of a representation of the knowledge is shown in the Figure 2.2

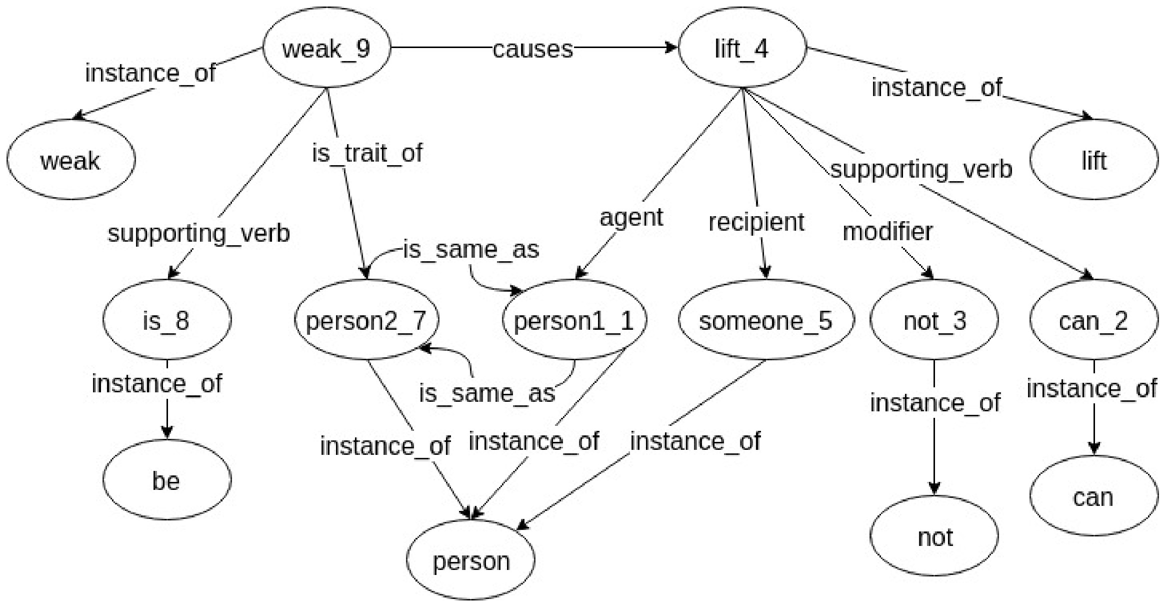


Figure 2.2: Graphical Representation of the Knowledge "person1 is weak prevents person1 lifts someone"

2.3.3 Step 3: Reasoning with Commonsense Knowledge

In this approach, we developed a reasoning algorithm that uses the graphical representations of the sentences in a WSC problem and a needed knowledge, the pronoun to be resolved and the two answer choices as input and outputs the answer choice which provides the most natural resolution to the pronoun in the sentences.

There are four steps in the reasoning algorithm. The steps are briefly mentioned below with the help of examples. The detailed description of the algorithm is provided in the Chapter 6.

1. In this step, a subgraph is extracted from the graphical representation of the WSC sentences such that it contains all the nodes from the original graph except the ones which represent classes. A subgraph extracted from the graph shown in Figure 2.1 is shown in Figure 2.3.

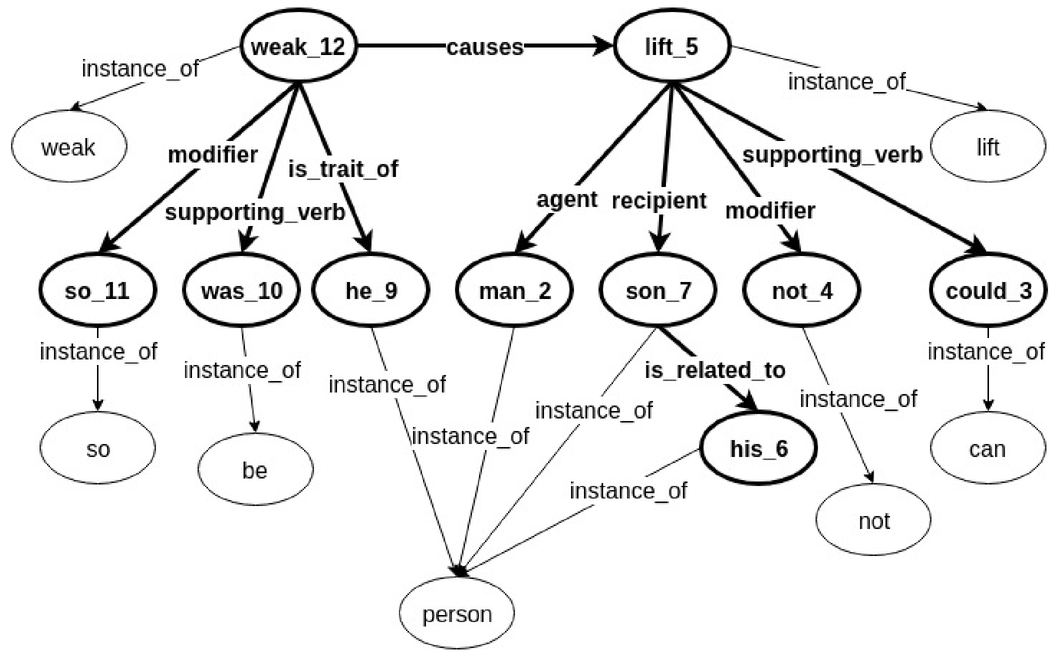


Figure 2.3: An Example of Step 1 Output of the WiSCR Algorithm with Respect to the WSC Sentence *"The man could not lift his son because he was so weak."*

2. In this step, a subgraph is extracted from the graphical representation of the knowledge such that it contains all the nodes from the original graph except the ones which represent classes and it contains all the edges except the ones labeled as *"is_same_as"*. A subgraph extracted from the graph shown in Figure 2.2 is shown in Figure 2.4.

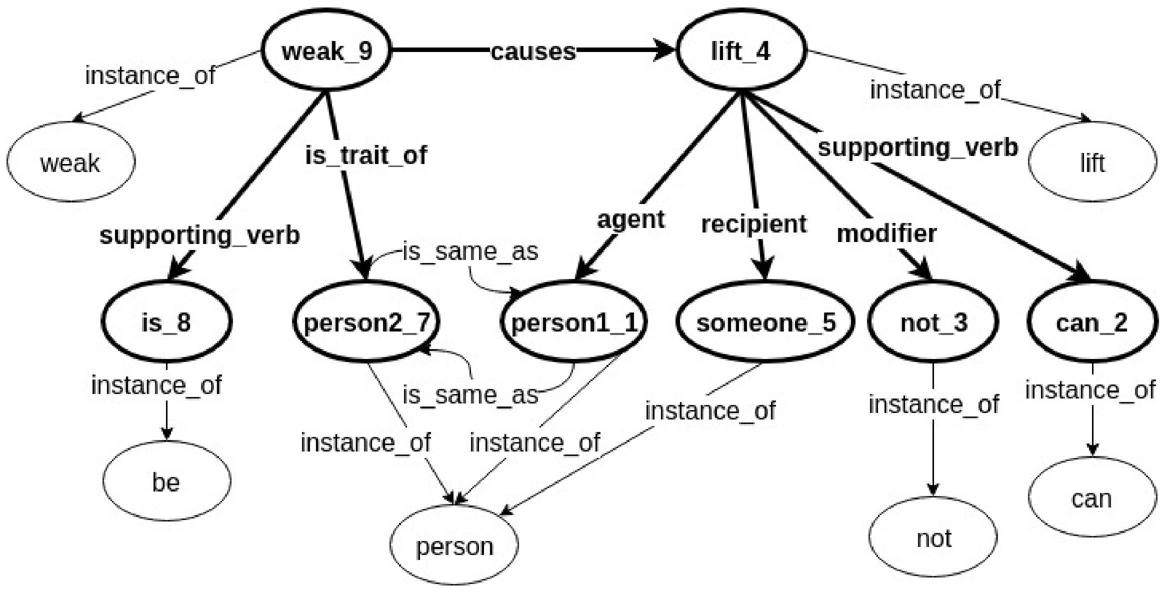


Figure 2.4: An Example of Step 2 Output of the WiSCR Algorithm with Respect to the Knowledge *"person1 is weak prevents person1 lifts someone"*

3. In this step all the possible graph-subgraph isomorphism are detected between the two graphs detected in the previous two steps. We used Answer Set Programming (ASP)(Baral, 2003) to detect the isomorphisms. A graph-subgraph isomorphism is a mapping (say $\mathbb{M}$) between two graphs (say $\mathcal{G}_1$ and $\mathcal{G}_2$) such that $\mathbb{M}$ is a set of pairs of the form (x,y) where x is a node in $\mathcal{G}_1$ and y is a node in $\mathcal{G}_2$, and if for all $(x,y) \in \mathbb{M}$, x is replaced by y then $\mathcal{G}_2$ becomes an induced subgraph of $\mathcal{G}_1$. The main goal of this step is to find all the mappings which make the graph extracted in the step 2 an induced subgraph of the graph extracted in the step 1. In other words, let $\mathbb{M}$ be a graph-subgraph isomorphism such that it makes the subgraph extracted in Step 2 an induced subgraph of the subgraph extracted in the Step 1. If such an isomorphism does not exist then $\mathbb{M} = \emptyset$. An example of the isomorphism detected between the outputs of Step 1 and Step 2 is as shown below.

$$\mathbb{M} = \{(weak\_9, weak\_12),\ (lift\_4, lift\_5),\ (is\_8, was\_10),\ (person2\_7, he\_9),$$

$(person1\_1, man\_2)$, $(someone\_5, son\_7)$, (not_3, not_4), $(can\_2, could\_3)\}$

4. In this step an answer to a WSC problem is deduced from the input representations and the results of the previous steps of this algorithm. Let $\mathbb{M}$ be the set of pairs extracted in the Step 3. In this step, the answer is deduced based on the following conditions which are again based on the concept of 'most natural resolution'. The concept 'most natural resolution' is described in detail in the Chapter 6 of this dissertation. Basically, if x in WSC sentences provides 'most natural resolution' for y in WSC sentences then it means that x and y are suggested to be co-referents of each other by a given knowledge.

   - The answer choice a_1 is the answer of the WSC problem if only a_1 provides the 'most natural resolution' for the pronoun in the WSC sentences.
   - The answer choice a_2 is the answer of the WSC problem if only a_2 provides the 'most natural resolution' for the pronoun in the WSC sentences.
   - No answer otherwise.

   The output of this step based on the inputs of the previous steps, the input representations and the isomorphism mentioned above is the answer choice *'man_2'*. So, the answer to the running WSC problem is ***'man_2'***.

2.4 Approach 2: Natural Language Inference Based Approach

In the approach 1 shown above, to be able to use the knowledge, the reasoning module puts several restrictions on the structure of the extracted knowledge sentence. If the knowledge extraction module could not find any knowledge pertaining to those preferred schema the extracted knowledge would probably of no use. Due to reporting bias, people hardly report the obvious and on top of that if the reasoning system puts hard filtering, the approach of knowledge extraction followed by reasoning will probably face severe difficulties. So, here we take a detour from building better knowledge extraction modules and focus on developing a reasoning system that can better utilize the extracted knowledge. Towards this we manually extract a knowledge sentence for each co-reference resolution problem from the existing Winograd Schema Challenge dataset without paying any attention to the reasoning system. The main steps in this approach are briefly described below with the help of an example. The steps are described in detail in Chapter 7. Let us consider the WSC problem shown below to explain this approach.

Sentence: *The man could not lift his son because* $\textbf{\textit{he}}_{pronoun}$ *was so weak.*

Answer Choices: *a) man b) son*

2.4.1 Step 1: Knowledge Hunting

This step is similar to knowledge sentence extraction step in the approach 1 above. In it a set of search queries are created and a sentence is extracted by using Google search engine which may contain the needed knowledge. For example the knowledge sentence extracted with respect to the above WSC problem is *"She could not lift him because she was weak"*.

2.4.2 Step 2: Entity Alignment

In this step an alignment between the entities (two answer choices and a pronoun to resolve) in the original WSC sentences and the extracted knowledge sentence is identified. For example, in the above WSC example and the knowledge sentence from the Step 1 the alignment is as shown below.

man ***aligns with*** *She*

son ***aligns with*** *him*

he ***aligns with*** *she*

We used semantic role labeling function (Palmer *et al.*, 2010; FitzGerald *et al.*, 2018) and pre-trained NLI systems (Chen *et al.*, 2016; Parikh *et al.*, 2016) from AllenNLP (Gardner *et al.*, 2018) to compute the alignments. The details are shown in the Chapter 7.

2.4.3 Step 3: Answer Retrieval

In this step the alignments extracted in the Step 2 are used to retrieve the answer to the input WSC problem. For example, both *man* (an answer choice) and *he* (the pronoun to resolve) are aligned with a common entity (i.e, *she*) whereas the other answer choice and the pronoun are not, hence the final answer (or coreferent of the pronoun) is ***'man'***.

A detailed description of the entire approach along with various possible alignments is shown in the Chapter 7.

Chapter 3

KNOWLEDGE TYPES IDENTIFICATION

3.1 Introduction

There are various kinds of knowledge that are needed to really understand the natural language. In broader terms, these kinds can be categorized into factual or common knowledge and commonsense knowledge (Cambria and Howard, 2014; Tandon *et al.*, 2011).

The common knowledge include the factual knowledge that humans learn as general knowledge over the years by reading and watching things around them. Examples of common knowledge include *"Donald J Trump became the president of the United State of America in 2017."*, and *"United States of America is a country on Earth."*

The commonsense knowledge is the knowledge about the concepts of the world. Over the years various kinds of commonsense knowledge have been identified. Commonsense knowledge is useful in various applications (Sharma *et al.*, 2015c; Chambers and Jurafsky, 2009). ConceptNet (Liu and Singh, 2004) is a knowledge base which stores one such commonsense knowledge. The knowledge in it is a connected graph of concepts, where nodes represent concepts such as *person*, and the relations between the concepts represent the way they interact with each other. For example *"has"* is the relation between the concepts *"person"* and *"head"*.

Most of the times the extraction of both common and commonsense knowledge is initiated by its need for solving a specific task. Later, when the task is solved, the extracted knowledge is stored in a knowledge base so that it can be used again in a different task which requires similar knowledge. In this work, our main focus is to extract commonsense knowledge which is helpful in solving the Winograd Schema Challenge (WSC). Similar to many of the other approaches (Emami *et al.*, 2018; Liu *et al.*, 2017; Isaak and Michael, 2016; Sharma *et al.*, 2015c; Bailey *et al.*, 2015) towards addressing the challenge, we believe that answering the WSC questions requires knowledge beyond what is given in the text. Let us consider the following WSC example.

Example 1:

Sentences: The fish ate the worm. **It**$_{pronoun}$ was tasty

Question: What was tasty? **Answer Choices:** a) fish b) worm

To answer the given question one needs the knowledge that *"something that is eaten may be tasty"*.

Following up on our belief, in this work we started with an aim to explore this further and answer questions such as, *How to automatically identify the needed knowledge?* and *How to automatically extract such knowledge?* After careful analysis of the challenge problems, we realized that to answer these question, we should first answer a few other questions. Such questions include, *What kind of knowledge is needed?* and *Can we categorize the problems based on the required knowledge?* Hence, in this work we attempted to answer these questions with respect to the WSC corpus [1] .

We found that reasoning with additional knowledge can indeed be helpful in solving the challenge. But, to develop an automated system we need to have a way to (a) obtain such knowledge, and (b) reason with such knowledge. Although automating the process to obtain knowledge is not the focus of this work, from our attempts in this direction, we realized that the first step towards that would be to identify various categories of knowledge. As recently shown in the ATOMIC knowledge base (Sap *et al.*, 2018), such categorization is specially helpful in crowd-sourcing and/or automatically inferring the commonsense

[1]Available at: `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WS.html`

knowledge of particular types. The categorization of knowledge is also useful in developing a reasoning mechanism as different categories may need slightly different reasoning modules. Furthermore, it will also allow the storage of different types of knowledge in separate sections of a Knowledge Base (KB) (or in entirely different KBs) and hence allowing the retrieval to be very efficient in an application that requires only a specific kind of knowledge.

Continuing on the similar lines, as part of this thesis, we identified several new kinds of commonsense knowledge to solve the WSC problems. The knowledge kinds are not directly present in any of the knowledge bases currently available.

In the section below, we present the intuitive meaning and detailed description of all the knowledge types which were identified as part of this work.

3.2 Commonsense Knowledge Categorization

In this section, we present the 12 kinds of commonsense knowledge which were discovered as part of this work. We performed a comprehensive analysis of the problems in the WSC and found that a knowledge required to answer each of them can be categorized as one of the types mentioned in this section. Here we provide the details of the knowledge types with the help of example problems from the WSC corpus. Let us begin by visiting the following WSC example.

Sentence: The man couldn't lift his son because **he**$_{pronoun}$ was so weak.

Question: Who was weak?

Answer Choices: a) man b) son

The above problem can be correctly solved by using the commonsense knowledge that,

someone weak may not be able to lift someone else

OR

person1 is weak ***may prevent*** *person1 lifts someone*

Intuitively, the above knowledge means that *person*1 being weak is a possible reason why *person*1 is unable to lift *someone*. The knowledge needed in the above example has four main components, i.e., an action (*'lifts'*), a property (*'weak'*), two entities (*'person1'* and *'someone'*) and the relationship between the action and the property (*'may prevent'*). Another important aspect of the knowledge mentioned above is that there are two mentions of *person*1.

Similar to the above example knowledge, a knowledge in each of the first 10 categories defined in this work is based on the relation between two actions or a property and an action. Also, there is atleast one entity with two mentions in the knowledge. In the following section we explain each of the first 10 categories with the help of examples from the WSC corpus. In a later section we describe the remaining two categories of knowledge.

3.2.1 Knowledge Type 1: A Property May Prevent an Action

In this category, if a property prevents an action from executing then an entity associated with the property is also a participant in the action. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: The man couldn't lift his son because **he**$_{pronoun}$ was so weak.

Question: Who was weak?

Answer Choices: a) man b) son

Required Knowledge:

person1 is weak ***may prevent*** *person1 lifts someone*

3.2.2 Knowledge Type 2: An Action May Cause an Action

In this category of commonsense knowledge if an action (say A_1) causes another action (say A_2) then an entity that participates in A_1 also participates in A_2. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: The city councilmen refused the demonstrators a permit because **they**$_{pronoun}$ feared violence.

Question: Who feared violence?

Answers Choices: a) councilmen b) demonstrators

Required Knowledge:

group1 fears violence ***may cause*** *group1 refuses permit*

3.2.3 Knowledge Type 3: A Property May Cause an Action

In this category, if a property causes an action then an entity that participates in the action is also associated with the property. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: The sculpture rolled off the shelf because **it**$_{pronoun}$ was not anchored.

Question: What was not anchored?

Answer Choices: a) sculpture b) shelf

Knowledge Needed:

object1 is not anchored ***may cause*** *object1 is rolled off*

3.2.4 Knowledge Type 4: An Action May Cause a Property

In this category, if an action causes a property then an entity that participates in the action is also associated with the property. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: I took the water bottle out of the backpack so that **it**$_{pronoun}$ would be handy.

Question: What would be handy?

Answer Choices: a) bottle b) backpack

Required Knowledge:

*object1 is taken out of something **may cause** object1 is handy*

3.2.5 Knowledge Type 5: An Action May Prevent an Action

In this category of commonsense knowledge if an action (say A_1) prevents another action (say A_2) then an entity that participates in A_1 also participates in A_2. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Beth didn't get angry with Sally, who had cut her off, because **she**$_{pronoun}$ stopped and counted to ten.

Question: Who counted to ten?

Answers: a) Beth b) Sally

Required Knowledge:

*person1 counts to ten **may prevent** person1 gets angry*

3.2.6 Knowledge Type 6: An Action May be Followed By an Action

In this category of commonsense knowledge if an action (say A_1) is followed by another action (say A_2) then an entity that participates in A_1 also participates in A_2. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: The customer walked into the bank and stabbed one of the tellers. He was immediately taken to the hospital.

Question: Who was taken to the hospital?

Answers: a) teller b) customer

Required Knowledge:

*person1 is stabbed **may be followed by** person1 is taken to hospital*

3.2.7 Knowledge Type 7: An Action May be Followed by a Property

In this category, if an action is followed by a property then an entity that participates in the action is also associated with the property. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Sam broke both his ankles and he is walking with crutches. But a month or so from now $\textbf{they}_{pronoun}$ should be unnecessary.

Question: What should be unnecessary?

Answer Choices: a) ankles b) crutches

Required Knowledge:

*person1's ankles are broken and person1 walks with crutches **may be followed by** crutches are unnecessary*

3.2.8 Knowledge Type 8: A Property May be followed by an Action

In this category, if a property is followed by an action then an entity that participates in the action is also associated with the property. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Thomson visited Cooper's grave in 1765. At that date **he**$_{pronoun}$ had been dead for five years.
Question: Who had been dead for five years?
Answer Choices: a) Cooper b) Thomson

Knowledge Needed:
person1 is dead ***may be followed by*** *person1's grave is visited*

3.2.9 Knowledge Type 9: A Property May Cause a Property

In this category, if a property (say P_1) causes another property (say P_2) then an entity associated with P_1 is also associated with P_2. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Sam and Amy are passionately in love, but Amy's parents are unhappy about it, because **they**$_{pronoun}$ are fifteen.
Question: Who are fifteen?
Answer Choices: a) Sam and Amy b) Amy's parents

Knowledge Needed:
person1 is in love and person1 is fifteen years old ***may cause*** *person1's parents are unhappy*

3.2.10 Knowledge Type 10: A Co-occurring Set of Actions and Properties

This category accounts for the following two cases.

Case 1 - Two Actions: In this case there are two co-occuring actions (say A_1 and A_2) such that a common entity participates in both the actions. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Steve follows Fred's example in everything. He influences **him**$_{pronoun}$ hugely.

Question: Who is influenced?

Answer Choices: a) Steve b) Fred

Knowledge Needed:

person1 follows person2's example in everything ***may co-occur with*** *person1 is influenced by person2*

Case 2 - An Action and a Property: In this case an action and a property co-occur, such that a common entity participates in both the action and the property. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: The fish ate the worm. **It**$_{pronoun}$ was hungry.

Question: What was hungry?

Answer Choices: a) fish b) worm

Knowledge Needed:

animal1 eats something ***may co-occur with*** *animal1 is hungry*

3.2.11 Additional Knowledge Categories

Along with the above mentioned 10 knowledge categories, in this work we identified two additional knowledge types. In this section we describe those types.

Knowledge Type 11: Statement 1 is more likely than Statement 2 This type consists of a likelihood comparison between two statements or propositions. The likelihood is based on the acceptability of a statement in a normal scenario. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Sam tried to paint a picture of shepherds with sheep, but **they**$_{pronoun}$ ended up looking more like dogs.

Question: What looked like dogs?

Answer Choices: a) sheep b) shepherds

Knowledge Needed:

sheep look like dogs ***is more likely than*** *shepherds look like dogs*

The above knowledge represents that the statement *'sheep look like dogs'* is more likely than the statement *'shepherds look like dogs'*. The validity of the likelihood relies on the normal scenario because for example if a shepherd is sitting then he may look like a dog but that would not be a normal scenario.

A generalized representation for this kind of knowledge is as shown below.

Statement1 ***is more likely than*** *Statement2*

Knowledge Type 12: Multiple Pieces of Knowledge

In this category, a list of more than one piece of knowledge is called one knowledge. Each knowledge piece in the list may be of a type from Knowledge Type 1 to Knowledge Type 10. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Mary tucked her daughter Anne into bed, so that **she**$_{pronoun}$ could work.

Question: Who is going to work?

Answer Choices: a) Mary b) daughter

Knowledge Needed:

1. *person1 tucks person2 into bed* ***may cause*** *person2 sleeps*
2. *daughter of person1 sleeps* ***may prevent*** *person1 is disturbed*
3. *person1 is not disturbed* ***may cause*** *person1 can work*

Table 3.1 below summarizes all the knowledge types and their corresponding examples from the Winograd Schema Challenge corpus.

Knowledge Type	Example Knowledge	WSC Example
1. PROPERTY ***may prevent*** ACTION	*person1 is weak* ***may prevent*** *person1 lifts someone*	**Sentence:** The man couldn't lift his son because **he**$_{pronoun}$ was so weak. **Question:** Who was weak? **Answer Choices:** a) man b) son
2. ACTION1 ***may cause*** ACTION2	*group1 fears violence* ***may cause*** *group1 refuses permit*	**Sentence:** The city councilmen refused the demonstrators a permit because **they**$_{pronoun}$ feared violence. **Question:** Who feared violence? **Answers Choices:** a) councilmen b)
3. PROPERTY ***may cause*** ACTION	*object1 is not anchored* ***may cause*** *object1 rolls off*	**Sentence:** The sculpture rolled off the shelf because **it**$_{pronoun}$ was not anchored. **Question:** What was not anchored? **Answer Choices:** a) sculpture b) shelf
4. ACTION ***may cause*** PROPERTY	*object1 is taken out of something* ***may cause*** *object1 is handy*	**Sentence:** I took the water bottle out of the backpack so that **it**$_{pronoun}$ would be handy. **Question:** What would be handy? **Answer Choices:** a) Paul b) George
5. ACTION1 ***may prevent*** ACTION2	*person1 counts to ten* ***may prevent*** *person1 gets angry*	**Sentence:** Beth didn't get angry with Sally, who had cut her off, because **she**$_{pronoun}$ stopped and counted to ten. **Question:** Who counted to ten? **Answers:** a) Beth b) Sally
6. ACTION1 ***may be followed by*** ACTION2	*person1 is stabbed* ***may be followed by*** *person1 is taken to hospital*	**Sentence:** The customer walked into the bank and stabbed one of the tellers. He was immediately taken to the hospital. **Question:** Who was taken to the hospital? **Answers:** a) teller b) customer
7. ACTION ***may be followed by*** PROPERTY	*person1's ankles are broken and person1 walks with crutches* ***may be followed by*** *crutches are unnecessary*	**Sentence:** Sam broke both his ankles and he is walking with crutches. But a month or so from now **they**$_{pronoun}$ should be unnecessary. **Question:** What should be unnecessary? **Answer Choices:** a) ankles b) crutches
8. PROPERTY ***may be followed by*** ACTION	*person1 is dead* ***may be followed by*** *person2 visits person1's grave*	**Sentence:** Thomson visited Cooper's grave in 1765. At that date **he**$_{pronoun}$ had been dead for five years. **Question:** Who had been dead for five years? **Answer Choices:** a) Cooper b) Thomson
9. PROPERTY1 ***may cause*** PROPERTY2	*person1 is in love and person1 is fifteen years old* ***may cause*** *person1's parents are unhappy*	**Sentence:** Sam and Amy are passionately in love, but Amy's parents are unhappy about it, because **they**$_{pronoun}$ are fifteen. **Question:** Who are fifteen? **Answer Choices:** a) Sam and Amy b) Amy's parents
10. Co-occurring ACTION(s) and PROPERTY(s)	*animal1 eats something* ***may co-occur with*** *animal1 is hungry*	**Sentence:** The fish ate the worm. **It**$_{pronoun}$ was hungry. **Question:** What was hungry? **Answer Choices:** a) fish b) worm
11. Statement 1 is more likely than Statement 2	*sheep look like dog* ***is more likely than*** *shepherds look like dogs*	**Sentence:** Sam tried to paint a picture of shepherds with sheep, but **they**$_{pronoun}$ ended up looking more like dogs. **Question:** What looked like dogs? **Answer Choices:** a) sheep b) shepherds
12. Multiple Pieces of Knowledge	1). *person1 tucks person2 into bed* ***may cause*** *person2 sleeps* 2). *daughter of person1 sleeps* ***may prevent*** *person1 is disturbed* 3). *person1 is not disturbed* ***may cause*** *person1 can work*	**Sentence:** Mary tucked her daughter Anne into bed, so that **she**$_{pronoun}$ could work. **Question:** Who is going to work? **Answer Choices:** a) Mary b) daughter

Table 3.1: Table of Knowledge Examples in Different Formats

Chapter 4

REASONING WITH THE COMMONSENSE KNOWLEDGE

In this chapter we present the details of our reasoning algorithm for the Winograd Schema Challenge (WSC) problems and a logical implementation of the algorithm. We provide the details of the reasoning paradigms used and the various formalisms introduced to perform the reasoning.

One of the reasoning approaches used in this work is based on translating a WSC problem and a needed commonsense knowledge into graphical representations. The representations are then used as input in the reasoning algorithm to produce an answer to the WSC problem. In the following sections, we first provide the detailed description of the graphical representations of a WSC problem and a needed knowledge. Then, we provide the details of the reasoning algorithm that uses those representations as input.

4.1 Formal Representation of a WSC Problem

In this work, we use a commonsense reasoning approach to tackle the problems in the WSC corpus. A requirement of using such an approach is to formally represent an input WSC problem. A WSC problem mainly consists of a sequence of one or more sentences. We use a graph based representation schema for the sentences in a WSC problem. In this section, we provide a brief overview of some of the popular graphical semantic representations which are used for representing text and inspired us to use a similar schema.

4.1.1 Abstract Meaning Representation (AMR)

AMR (Banarescu *et al.*, 2013) is a logical semantic representation which was developed to represent a natural language text. Its development was motivated by the goal to generate semantic banks similar to the syntactic treebanks. This would ultimately be helpful in initiating new work in the development of semantic parsers which are as ubiquitous as their syntactic counterparts. An AMR of a text is a rooted, edge labeled, leaf labeled and directed graph. An AMR can be represented in various forms such as conjunctions of logical triples, a directed graph, and PENMAN [1] inputs (Matthiessen and Bateman, 1991). An example of an AMR of a sentence is shown in Figure 4.1.

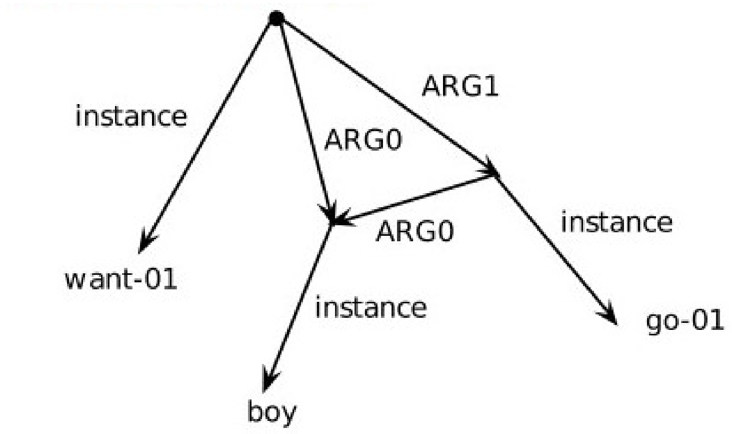


Figure 4.1: An Abstract Meaning Representation (AMR) of the Sentence *"The boy wants to go."*

4.1.2 Knowledge Parser (K-Parser) Representation

K-Parser (Sharma *et al.*, 2015c) is a semantic parser which generates a graphical semantic representation of an English sentence. Similar to AMR, a representation generated by K-Parser can be formatted as a conjunction of logical triples or a rooted, edge-labeled and node-labeled, directed acyclic graph. An example of a K-Parser representation of a sentence is as shown in Figure 4.2.

[1] https://github.com/goodmami/penman

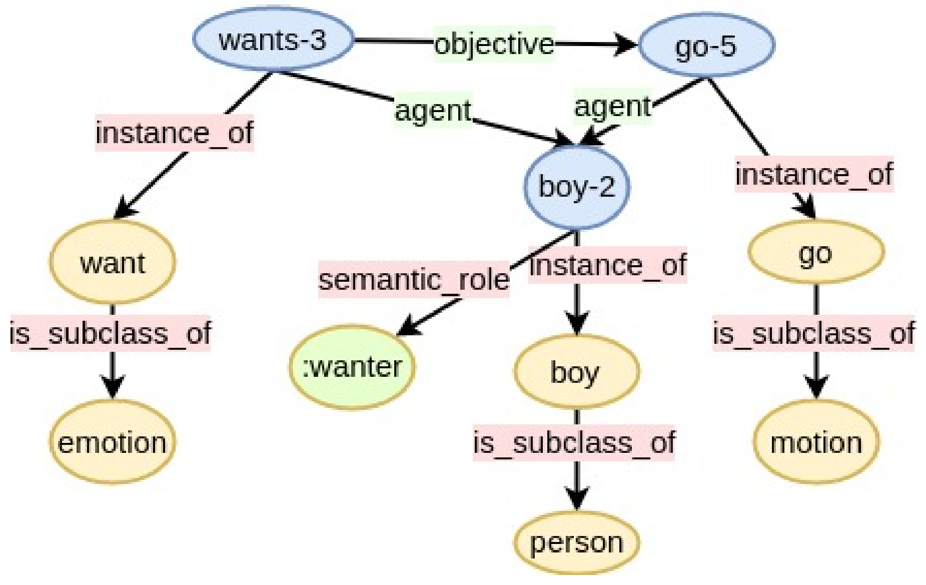


Figure 4.2: K-Parser Output for the Sentence *"The boy wants to go."*

4.1.3 TRIPS Parser Representation

TRIPS (Allen *et al.*, 2008) is another semantic parser which translates an English text snippet into its logical semantic representation. The representation generated can be formatted as a graph or as an AMR like structure. An example of a TRIPS representation of a sentence is as shown Figure 4.3.

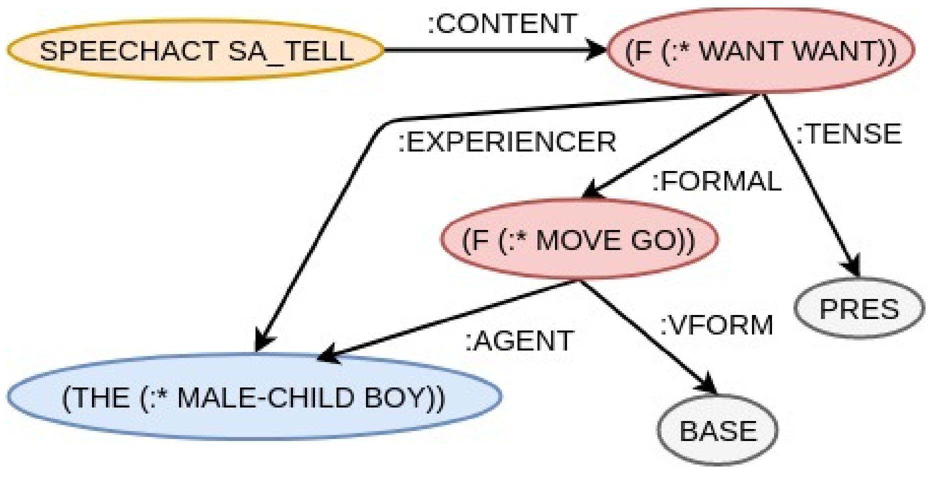


Figure 4.3: TRIPS output for the Sentence *"The boy wants to go."*

In our approach, we formally represent a WSC problem, i.e., a sequence of sentences, a pronoun and two answer choices, in a graphical representation that is inspired from the representations mentioned above. In the following section of this chapter, we define a graphical representation of a sequence of English sentences in a WSC problem. For that reason we define a set of tokens in a sequence of sentences, a POS tagging function which maps each token in a sequence of sentences to a POS tag, a class mapping function which maps each token in a sequence of sentences to its class (or type) and finally we define a graphical representation of a sequence of sentences by using a POS tagging function and a class mapping function. The nodes in the graphical representation are made up of the tokens in the sentences and the classes of the tokens. The edge labels in the graphical representation are from a set of binary relations between two nodes in the representation.

Definition 1 (Set of Tokens in a Sequence of Sentences) *Let* $\mathcal{S}$ = ($\mathcal{S}_1$, $\mathcal{S}_2$, ..., $\mathcal{S}_n$), $n \geq 1$, *be a sequence of English sentences,* $\mathcal{W}_i$ *be the sequence of words in the sentence* $\mathcal{S}_i$ *and* $\mathcal{W}_{\mathcal{S}} = \mathcal{W}_1^\frown \mathcal{W}_2^\frown ...^\frown \mathcal{W}_n$ *be the concatenation of the word sequences. Then the set of tokens* $\mathbb{T}(\mathcal{S})$ *is defined as follows:*

$$\mathbb{T}(\mathcal{S}) = \{w\_i \mid w \text{ is the ith word in } \mathcal{W}_{\mathcal{S}}\}$$

Example 1 *Let us consider the sequence of English sentences 'John ate the pizza. It was very tasty.' Then the set of tokens in the sequence is,*

$$\mathbb{T}(\mathcal{S}) = \{John\_1,\ ate\_2,\ the\_3,\ pizza\_4,\ It\_5,\ was\_6,\ very\_7,\ tasty\_8\}.$$

Definition 2 (A POS Tagging Function) *Let* $\mathcal{S}$ *be a sequence of one or more English sentences,* $\mathbb{T}(\mathcal{S})$ *be the set of tokens in* $\mathcal{S}$. *Then, the POS tagging function* $f_{\mathcal{S}}^{pos}$ *maps an element in* $\mathbb{T}(\mathcal{S})$ *to an element in the set* $\{verb, noun, pronoun, adverb, adjective, other\}$, *i.e.,*

$$f_{\mathcal{S}}^{pos} : \mathbb{T}(\mathcal{S}) \rightarrow \{verb, noun, pronoun, adverb, adjective, other\}$$

Example 2 *Let us consider the sequence of English sentences 'John ate the pizza. It was very tasty.' The set of tokens in the sequence is,* $\mathbb{T}(S)$ = $\{John\_1,\ ate\_2,\ the\_3,\ pizza\_4,\ It\_5,\ was\_6,\ very\_7,\ tasty\_8\}$. *Then an example of a mapping produced by a POS tagging function is as shown below,*

$$f^{pos}_{\mathcal{S}}(John\_1) = noun$$
$$f^{pos}_{\mathcal{S}}(ate\_2) = verb$$
$$f^{pos}_{\mathcal{S}}(the\_3) = other$$
$$f^{pos}_{\mathcal{S}}(pizza\_4) = noun$$
$$f^{pos}_{\mathcal{S}}(It\_5) = pronoun$$
$$f^{pos}_{\mathcal{S}}(was\_6) = verb$$
$$f^{pos}_{\mathcal{S}}(very\_7) = adverb$$
$$f^{pos}_{\mathcal{S}}(tasty\_8) = adjective$$

Definition 3 (A Class Mapping Function) *Let* $\mathcal{S}$ *be a sequence of one or more English sentences,* $\mathbb{T}(\mathcal{S})$ *be the set of tokens in* $\mathcal{S}$. *Then, the class mapping function* $f^{class}_{\mathcal{S}}$ *maps an element of* $\mathbb{T}(\mathcal{S})$ *to an element in a set* $\mathbb{C}$, *i.e.,* $f^{class}_{\mathcal{S}} : \mathbb{T}(\mathcal{S}) \rightarrow \mathbb{C}$ *where the set* $\mathbb{C}$ *is a union of three sets* $\mathbb{C}_1$, $\mathbb{C}_2$ *and* $\{\phi\}$ *such that,*

- $\mathbb{C}_1 = \{c \mid c$ *is the lemmatized* [2] *form of* w *where* $w\_i \in \mathbb{T}(\mathcal{S})$ *and* $f^{pos}_{\mathcal{S}}(w\_i) \in \{verb, adverb, adjective\}\}$

- $\mathbb{C}_2 = \{object,\ person,\ group,\ location,\ quantity,\ shape,\ animal,\ plant,\ cognition,\ communication,\ event,\ feeling,\ act,\ motive,\ phenomenon,\ possession,\ process,\ relation,\ state,\ time\}$ [3]

[2]https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html, https://www.thoughtco.com/what-is-base-word-forms-1689161

[3]Inspired from WordNet (Miller, 1995) lexicographer files `https://wordnet.princeton.edu/documentation/lexnames5wn`

and,

$$f_{\mathcal{S}}^{class}(x) = \begin{cases} c_1 \in \mathbb{C}_1 & \textit{if } f_{\mathcal{S}}^{pos}(x) \in \{verb, adjective, adverb\} \\ c_2 \in \mathbb{C}_2 & \textit{if } f_{\mathcal{S}}^{pos}(x) \in \{noun, pronoun\} \\ \phi & \textit{otherwise} \end{cases}$$

Example 3 *Let us consider the sequence of English sentences 'John ate the pizza. It was very tasty.' The set of tokens in the sequence is,* $\mathbb{T}(\mathcal{S}) = \{John\_1,\ ate\_2,\ the\_3,\ pizza\_4,\ It\_5,\ was\_6,\ very\_7,\ tasty\_8\}$. *Also a mapping produced by a POS tagging function is,*

$$f_{\mathcal{S}}^{pos}(John\_1) = noun$$
$$f_{\mathcal{S}}^{pos}(ate\_2) = verb$$
$$f_{\mathcal{S}}^{pos}(the\_3) = other$$
$$f_{\mathcal{S}}^{pos}(pizza\_4) = noun$$
$$f_{\mathcal{S}}^{pos}(It\_5) = pronoun$$
$$f_{\mathcal{S}}^{pos}(was\_6) = verb$$
$$f_{\mathcal{S}}^{pos}(very\_7) = adverb$$
$$f_{\mathcal{S}}^{pos}(tasty\_8) = adjective$$

Then an example of a mapping produced by a class mapping function is as shown below,

$$f_{\mathcal{S}}^{class}(John\_1) = person$$
$$f_{\mathcal{S}}^{class}(ate\_2) = eat$$
$$f_{\mathcal{S}}^{class}(the\_3) = \phi$$
$$f_{\mathcal{S}}^{class}(pizza\_4) = object$$
$$f_{\mathcal{S}}^{class}(It\_5) = object$$
$$f_{\mathcal{S}}^{class}(was\_6) = be$$
$$f_{\mathcal{S}}^{class}(very\_7) = very$$
$$f_{\mathcal{S}}^{class}(tasty\_8) = tasty$$

Definition 4 (A Formal Representation of a Sequence of English Sentences) *Let $\mathcal{S}$ be a sequence of one or more English sentences, $\mathbb{T}(\mathcal{S})$ be a set of tokens in $\mathcal{S}$, $f_{\mathcal{S}}^{pos}$ be a POS tagging function and $f_{\mathcal{S}}^{class}$ be a class mapping function. Then, a formal representation of $\mathcal{S}$ is an edge labeled directed acyclic graph, $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}, \mathbb{E}, f)$. The set of vertices $\mathbb{V}$, is a union of two disjoint sets $\mathbb{V}_1$ and $\mathbb{V}_2$, such that,*

- $\mathbb{V}_1$ = $\{w\_i \mid w\_i \in \mathbb{T}(\mathcal{S})$ *and* $f_{\mathcal{S}}^{pos}(w\_i) \in \{verb,\ adverb,\ adjective,\ noun,\ pronoun\}\}$
- $\mathbb{V}_2$ = $\{c \mid f_{\mathcal{S}}^{class}(w\_i) = c$ *where* $w\_i \in \mathbb{V}_1\}$

$\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$, *has following properties,*

- $\mathbb{E}$ *is a union of the two disjoint sets* $\mathbb{E}_1$ *and* $\mathbb{E}_2$,
- $(v_1, v_2) \in \mathbb{E}_1$ *if* $v_1 \in \mathbb{V}_1$ *and* $v_2 \in \mathbb{V}_1$,
- $(v_1, v_2) \in \mathbb{E}_2$ *if* $v_1 \in \mathbb{V}_1$ *and* $v_2 \in \mathbb{V}_2$,
- *if* $(v_1, v_2) \in \mathbb{E}_2$ *then there does not exist* $v \in \mathbb{V}_2$ *such that* $(v_1, v) \in \mathbb{E}_2$ *where* $v \neq v_2$

$f : \mathbb{E} \rightarrow \mathbb{L} \cup \{instance\_of\}$, *is an edge labelling function where* $\mathbb{L}$ *is a set of binary relations between two nodes in* $\mathbb{V}_1$ *and 'instance_of' is a binary relation between a node in* $\mathbb{V}_1$ *and a node in* $\mathbb{V}_2$, *i.e.,*

$$f((v_1, v_2)) = \begin{cases} l \in \mathbb{L} & \text{if } (v_1, v_2) \in \mathbb{E}_1 \\ \text{``}instance\_of\text{''} & \text{if } (v_1, v_2) \in \mathbb{E}_2 \end{cases}$$

Example 4 *Let us consider the sequence of sentences 'John ate the pizza. It was very tasty.'. Also, let us consider a POS tagging for the sequence as shown in Example 2 and a class mapping for the sequence as shown in Example 3. Then, according to the above definition, a representation of the sequence of sentences is as shown in Figure 4.4.*

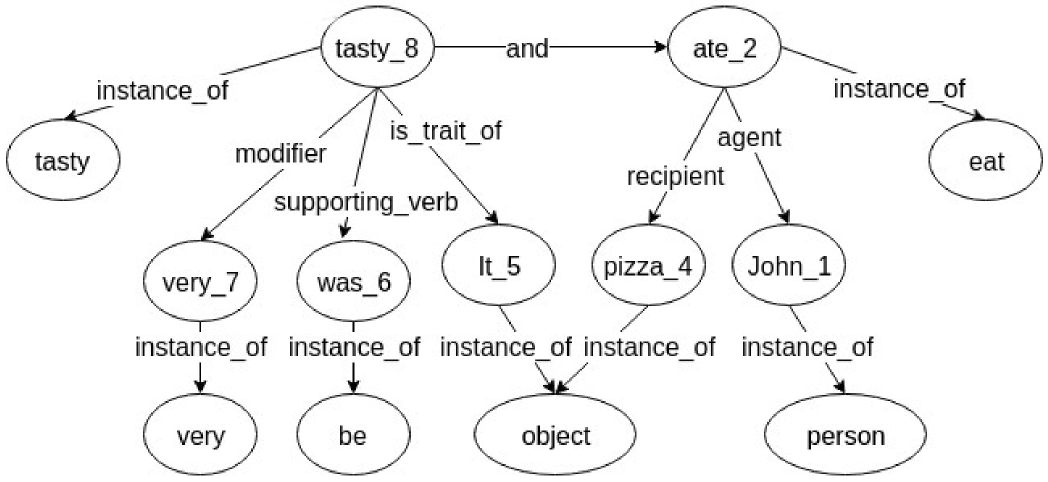


Figure 4.4: A Representation of a Sequence of English Sentences, *"John ate the pizza. It was very tasty."*

Example 5 *Figure 4.5 shows another example of a graphical representation of a sequence of sentences. The sentences are taken from a WSC problem.*

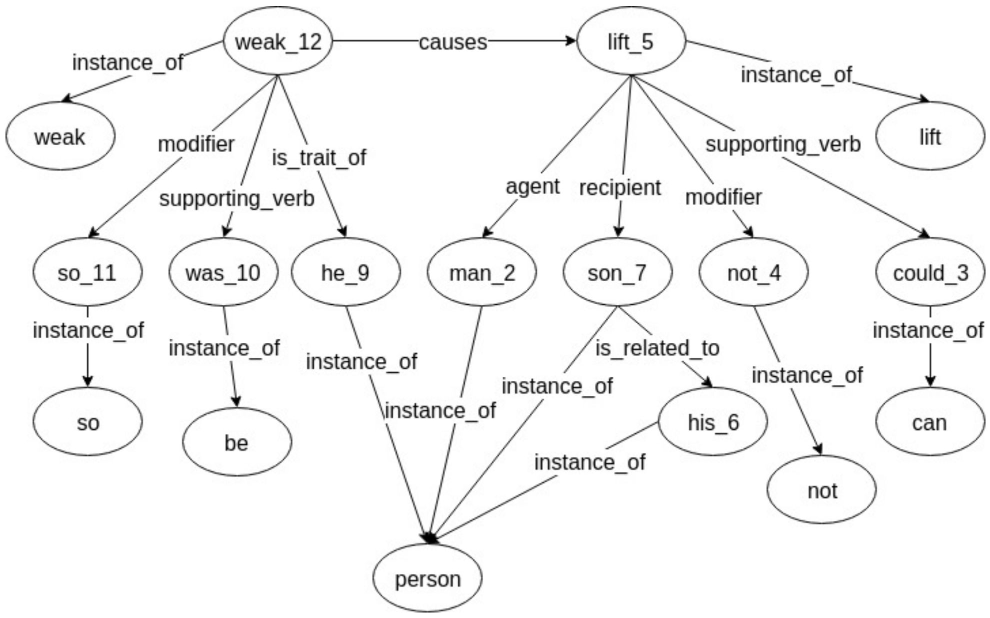


Figure 4.5: A Representation of the Sequence of Sentences, *"The man could not lift his son because he was so weak"*

Semantics of a Representation of a Sequence of Sentences: The semantics or meaning of a graphical representation of a sequence of sentences are defined with respect to each of the components in the representation. There are following two components of a graphical representation presented in the Definition 4.

1. **Nodes:** There are two types of nodes. The first are the token nodes. A token node represents a particular occurrence of a concept in the concerned sequence of sentences. Only verb, adverb, adjective, noun and pronoun words are converted into token nodes. For example the node *'weak_12'* is a token node in the example representation shown in the Figure 4.5. The second type of nodes are the class nodes. A class node represents a 'type' of a token node. For example *'person'* in Figure 4.5 is a class node where *'man_2'* is of 'type' *'person'*. A type or class node may be grounded in an ontology (for example schema.org ontology) of classes but to keep things simple we do not use such a grounding in this work. Following is an example of grounding the class *'person'* in a hierarchy, *'person'* is a subclass of *'human'* and *'human'* is a subclass of *'living things'*.

2. **Edges and Edge Relations:** There are two types of directed edges. The first type accounts for the edges from a token node to another token node. Since token nodes represent particular occurrences of concepts in a sequence of sentences, this type of edges represent a semantic dependency between two such concept occurrences. The labels of these edges define the semantics of the dependency. For example, the edge labeled *'agent'* from the node *'lift_5'* to the node *'man_2'* represents that *'man_2'* is an *agent* or *doer* of the action represented by *'lift_5'*. The second types of edges are from a token node to a class node. All such edges are labeled as *'instance_of'*. Let such an edge exists between a token node t and a class node c. Then we say that t is of type c. For example *'man_2'* in the Figure 4.5 is of type (or *instance_of*) *'person'*.

A token node is always connected to exactly one class node in the representation.

4.2 Representation of the Knowledge Types

In this section we define the formal representations of the knowledge types defined in the Chapter 3. In Chapter 3 we identified 12 different kinds of commonsense knowledge. The first ten types follow a similar pattern and hence a common reasoning algorithm is defined to handle them. In the following sub-section (4.2.1) we present the formal representation of the knowledge types 1 through 10 and then in Section 4.3 we present the reasoning algorithm with respect to those ten knowledge types.

Later in the Sections 4.5 and 4.6 we present the representation and reasoning with respect to the remaining two knowledge types.

4.2.1 Representation of Knowledge Types 1 through 10

In this work we write a knowledge of types 1 through 10 (See Chapter 3) in a English like format which allows co-reference resolution. The format is formally defined in the Definition 5 below. The definition uses the Definitions 1 and 2. An example knowledge is also shown below.

Definition 5 (A Knowledge) *A knowledge $\mathcal{K}$ is a statement of the form '**IF** $\mathcal{S}$ **THEN** x **is same as** y' where $\mathcal{S}$ is an English sentence, $\mathbb{T}(\mathcal{S})$ is a set of tokens in $\mathcal{S}$, $x, y \in \mathbb{T}(\mathcal{S})$, $f^{pos}_{\mathcal{S}}(x) = noun$ and $f^{pos}_{\mathcal{S}}(y) = noun$, where $f^{pos}_{\mathcal{S}}$ is a POS tagging function.*

Example 6 *An example of a knowledge is,*

*"**IF** person1 can not lift someone because person2 is weak **THEN** person1_1 **is same as** person2_7".*

Here, $\mathcal{S}$='person1 can not lift someone because person2 is weak' is an English sentence and person1_1 and person2_7 are tokens in $\mathcal{S}$.

Table 4.1 shows examples of the knowledge (Types 1 through 10) in the format shown in the Chapter 3 and the format defined in the Definition 5.

Knowledge Type	**Example Knowledge**	**Example Knowledge in Co-reference Enabled Format**
1. PROPERTY ***may prevent*** ACTION	*Person1 is weak* ***may prevent*** *Person1 lifts someone*	***IF*** *Person1 can not lift someone because Person2 is weak* ***THEN*** *Person1_1 is same as Person2_7*
2. ACTION1 ***may cause*** ACTION2	*Group1 fears violence* ***may cause*** *Group1 refuses permit*	***IF*** *Group1 refuses permit because Group2 fears violence* ***THEN*** *Group1_1 is same as Group2_5*
3. PROPERTY ***may cause*** ACTION	*Object1 is not anchored* ***may cause*** *Object1 rolls off*	***IF*** *Object1 rolls off because Object2 is not anchored* ***THEN*** *Object1_1 is same as Object2_5*
4. ACTION ***may cause*** PROPERTY	*Object1 is taken out of something* ***may cause*** *Object1 is handy*	***IF*** *Object1 is handy because Object2 is taken out of something* ***THEN*** *Object1_1 is same as Object2_5*
5. ACTION1 ***may prevent*** ACTION2	*Person1 counts to ten* ***may prevent*** *Person1 gets angry*	***IF*** *Person1 does not get angry because Person2 counts to ten* ***THEN*** *Person1_1 is same as Person2_7*
6. ACTION1 ***may be followed by*** ACTION2	*Person1 is stabbed* ***may be followed by*** *Person1 is taken to hospital*	***IF*** *Person1 is taken to hospital after Person2 is stabbed* ***THEN*** *Person1_1 is same as Person2_7*
7. ACTION ***may be followed by*** PROPERTY	*Person1's ankles are broken and Person1 walks with crutches* ***may be followed by*** *crutches are unnecessary*	***IF*** *crutches are unnecessary after Person1's ankles are broken and Person1 walks with crutches* ***THEN*** *crutches_1 is same as crutches_14*
8. PROPERTY ***may be followed by*** ACTION	*Person1 is dead* ***may be followed by*** *Person2 visits Person1's grave*	***IF*** *Person1 visits Person2's grave after Person3 is dead* ***THEN*** *Person2_3 is same as Person3_7*
9. PROPERTY1 ***may cause*** PROPERTY2	*Person1's army is larger* ***may cause*** *Person1 is victorious*	***IF*** *Person1 is victorious because Person2's army is larger* ***THEN*** *Person1_1 is same as Person2_5*
10. Co-occurring ACTION(s) and PROPERTY(s)	*Animal1 eats something* ***may co-occur with*** *Animal1 is hungry*	***IF*** *Animal1 eats something and Animal2 is hungry* ***THEN*** *Animal1_1 is same as Animal2_5*

Table 4.1: Table of Knowledge Examples in Different Formats

Similar to the sentences in a WSC problem, the knowledge of types 1 through 10 in co-reference enabled format (in Definition 5) is formally represented as a graph. The graph is formally defined in the Definition 6.

Definition 6 (A Graphical Representation of a Knowledge) *Let $\mathcal{K}$ = '**IF** $\mathcal{S}$ **THEN** x **is same as** y' be a knowledge where $\mathcal{S}$ is an English sentence, x and y are tokens in $\mathcal{S}$ and $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}}, \mathbb{E}_{\mathcal{S}}, f_{\mathcal{S}})$ be a graphical representation of $\mathcal{S}$. Then, a graphical representation of $\mathcal{K}$ is an edge labeled directed graph $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}}, \mathbb{E}_{\mathcal{K}}, f_{\mathcal{K}})$, such that,*

- $\mathbb{V}_{\mathcal{K}} = \mathbb{V}_{\mathcal{S}}$,
- $\mathbb{E}_{\mathcal{K}} = \mathbb{E}_{\mathcal{S}} \bigcup \{(x,y),(y,x)\}$, *and*
- $$f_{\mathcal{K}}((v_1,v_2)) = \begin{cases} f_{\mathcal{S}}((v_1,v_2)) & \textit{if } (v_1,v_2) \in \mathbb{E}_{\mathcal{S}} \\ \text{``}\textit{is\_same\_as}\text{''} & \textit{Otherwise} \end{cases}$$

Here, we say that $f_{\mathcal{K}}$ is defined using $f_{\mathcal{S}}$.

Example 7 *An example of a representation of a knowledge (Type 1) is shown in Figure 4.6.*

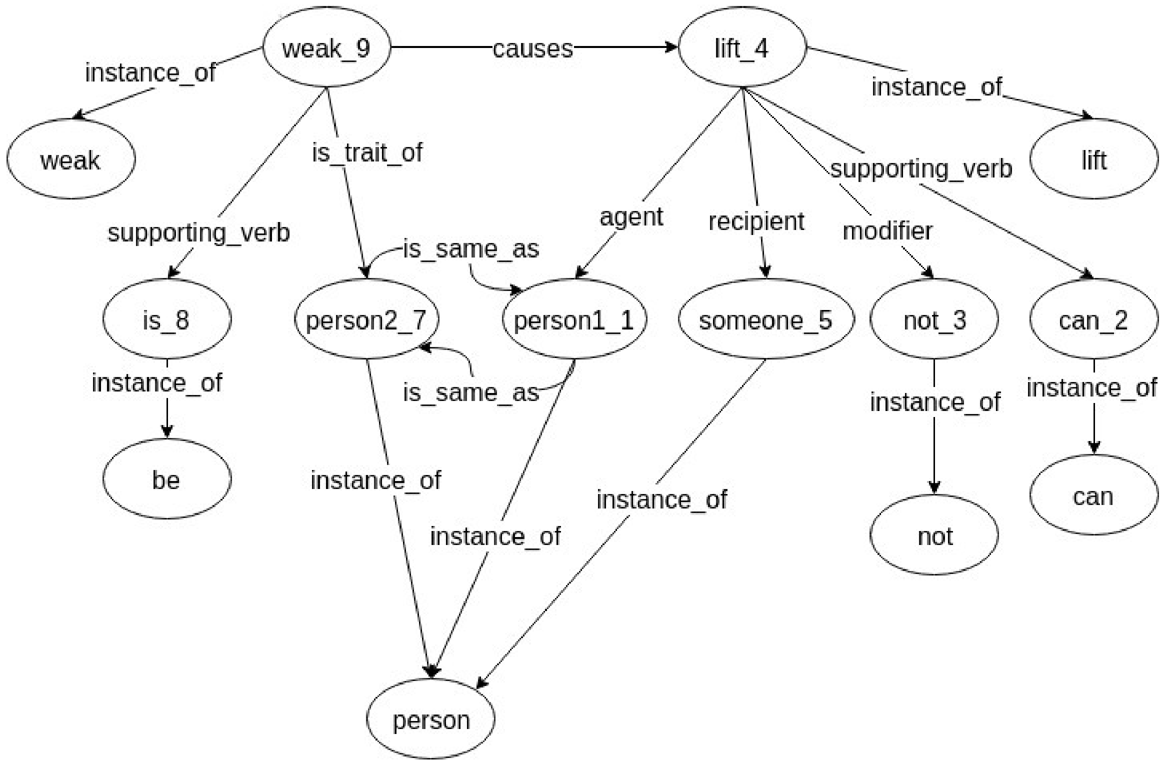


Figure 4.6: Graphical Representation of the Knowledge, "**IF** *person1 can not lift someone because person2 is weak* **THEN** *person1_1 is same as person2_7*"

4.3 Reasoning Algorithm for Knowledge Types 1-10

In this work we defined a reasoning algorithm for solving the WSC problems. The algorithm takes graphical representations of a WSC problem and a knowledge (Types 1-10) as input and outputs the answer of the WSC problem if it is inferred from the inputs. As per the problem definition the correct answer provides the 'most natural resolution' for the pronoun in the WSC sentences. In the following two definitions we formally define the 'most natural resolution' and the answer of a WSC problem with respect to the graphical representations of a WSC problem and a commonsense knowledge that is needed to answer it.

Definition 7 (Most Natural Resolution) *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem, $\mathcal{G}_{\mathcal{S}}$ = ($\mathbb{V}_{\mathcal{S}}$,$\mathbb{E}_{\mathcal{S}}$,$f_{\mathcal{S}}$) be a graphical representation of $\mathcal{S}$, $\mathcal{G}'_{\mathcal{S}}$ = ($\mathbb{V}'_{\mathcal{S}}$,$\mathbb{E}'_{\mathcal{S}}$,$f'_{\mathcal{S}}$) be a subgraph of $\mathcal{G}_{\mathcal{S}}$ such that $\mathbb{V}'_{\mathcal{S}} = \mathbb{V}_{\mathcal{S}} - \mathbb{V}^c_{\mathcal{S}}$ where $\mathbb{V}^c_{\mathcal{S}}$ is the set of all the class nodes in $\mathcal{G}_{\mathcal{S}}$, $f'_{\mathcal{S}} = f_{\mathcal{S}}$ and $\mathbb{E}'_{\mathcal{S}} = \mathbb{E}_{\mathcal{S}} - \mathbb{E}^c_{\mathcal{S}}$ where $e \in \mathbb{E}^c_{\mathcal{S}}$ iff $f_{\mathcal{S}}(e) =$ "$instance\_of$". Let $\mathcal{G}_{\mathcal{K}}$ = ($\mathbb{V}_{\mathcal{K}}$,$\mathbb{E}_{\mathcal{K}}$,$f_{\mathcal{K}}$) be a graphical representation of a knowledge where $f_{\mathcal{K}}$ is defined using $f_{\mathcal{S}}$, $\mathcal{G}'_{\mathcal{K}}$ = ($\mathbb{V}'_{\mathcal{K}}$,$\mathbb{E}'_{\mathcal{K}}$,$f'_{\mathcal{K}}$) be a subgraph of $\mathcal{G}_{\mathcal{K}}$ such that $\mathbb{V}'_{\mathcal{K}} = \mathbb{V}_{\mathcal{K}} - \mathbb{V}^c_{\mathcal{K}}$ where $\mathbb{V}^c_{\mathcal{K}}$ is the set of all the class nodes in $\mathcal{G}_{\mathcal{K}}$, $f'_{\mathcal{K}} = f_{\mathcal{K}}$ and $\mathbb{E}'_{\mathcal{K}} = \mathbb{E}_{\mathcal{K}} - \mathbb{E}^c_{\mathcal{K}}$ where $e \in \mathbb{E}^c_{\mathcal{K}}$ iff $f_{\mathcal{K}}(e) \in \{is\_same\_as, instance\_of\}$. Also, let $\mathbb{M}$ be a set of pairs of the form (a,b) such that either all of the below conditions are satisfied or $\mathbb{M} = \emptyset$.*

- *$a \in \mathbb{V}'_{\mathcal{S}}$ and $b \in \mathbb{V}'_{\mathcal{K}}$,*
- *a and b are instances of same class, i.e., $(a,i) \in \mathbb{E}_{\mathcal{S}}$, $(b,i) \in \mathbb{E}_{\mathcal{K}}$, $f_{\mathcal{S}}((a,i))$ = $instanc$-e_of and $f_{\mathcal{K}}((b,i))$ = $instance\_of$*
- *if for every pair (a,b)$\in \mathbb{M}$, a is replaced by b in $\mathbb{V}'_{\mathcal{S}}$ then $\mathcal{G}'_{\mathcal{K}}$ becomes a subgraph of the node replaced $\mathcal{G}'_{\mathcal{S}}$*

*Then we say that $x \in \mathbb{V}'_{\mathcal{S}}$ provides the **'most natural resolution'** for $y \in \mathbb{V}'_{\mathcal{S}}$ if (x,n_1)$\in \mathbb{M}$, (y,n_2)$\in \mathbb{M}$ and either one of the following is true*

- *$(n_1, n_2) \in \mathbb{E}_{\mathcal{K}}$ and $f_{\mathcal{K}}((n_1, n_2)) = is\_same\_as$*
- *$(n_2, n_1) \in \mathbb{E}_{\mathcal{K}}$ and $f_{\mathcal{K}}((n_2, n_1)) = is\_same\_as$*

Example 8 *Let us consider the representation of a knowledge shown in the Figure 4.6, the representation of the sentences in a WSC problem as shown in the Figure 4.5 and the set of node pairs $\mathbb{M}$ = {(weak_12, weak_9), (lift_5, lifts_4), (he_9, person2_7), (son_7, someone_5), (was_10, is_8), (not_4, not_3), (could_3, can_2), (man_2, person1_1)}. Then, the **'most natural resolution'** for **he_9** is **man_2**.*

Definition 8 (Answer of a WSC Problem) *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem $\mathcal{P}$, $\mathbb{T}(S)$ be the set of tokens in $\mathcal{S}$, $p \in \mathbb{T}(S)$ be the token which represents the pronoun to be resolved, $a_1, a_2 \in \mathbb{T}(S)$ be two tokens which represent the two answer choices, $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}}, \mathbb{E}_{\mathcal{S}}, f_{\mathcal{S}})$ be a graphical representation of $\mathcal{S}$, and $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}}, \mathbb{E}_{\mathcal{K}}, f_{\mathcal{K}})$ be a graphical representation of a knowledge such that $f_{\mathcal{K}}$ is defined using $f_{\mathcal{S}}$. Then,*

- *a_1 is the answer of $\mathcal{P}$, if only a_1 provides the 'most natural resolution' for p,*
- *a_2 is the answer of $\mathcal{P}$, if only a_2 provides the 'most natural resolution' for p,*
- *no answer otherwise*

Example 9 *Let us consider the representation of a knowledge from Figure 4.6, the representation of WSC sentences from Figure 4.5, the token for pronoun to resolve is 'he_9', the tokens for answer choices are 'man_2' and 'son_7'. Then according to the Definition 7, only 'man_2' provides the 'most natural resolution' for 'he_9'. Hence, according to the Definition 8* ***'man_2'*** *is the answer of the WSC problem.*

4.3.1 Winograd Schema Challenge Reasoning (WiSCR) Algorithm

Input to the WiSCR Algorithm: a graphical representation, $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}}, \mathbb{E}_{\mathcal{S}})$, of the sentences in a WSC problem (By Definition 4), a node p in $\mathcal{G}_{\mathcal{S}}$ which represents the pronoun to be resolved, two nodes a_1 and a_2 in $\mathcal{G}_{\mathcal{S}}$ which represent the two answer choices for the WSC problem, and a graphical representation, $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}}, \mathbb{E}_{\mathcal{K}})$, of a commonsense knowledge (By Definition 6).

Output of the WiSCR Algorithm: The algorithm outputs a_1, a_2 or it does not output any answer.

Behavior of the WiSCR Algorithm:

STEP 1: In this step a subgraph of $\mathcal{G}_S$ is extracted. Let the extracted subgraph is named $\mathcal{G}_S{}'$. $\mathcal{G}_S{}'$ contains all the nodes which are not class nodes in $\mathcal{G}_S$. All the edges which connect such nodes are also extracted. An example of the output of the Step 1 is shown in the Figure 4.7. The entire graph is the representation of the sentences in a WSC problem, and the highlighted part of the graph represents the subgraph extracted in this step.

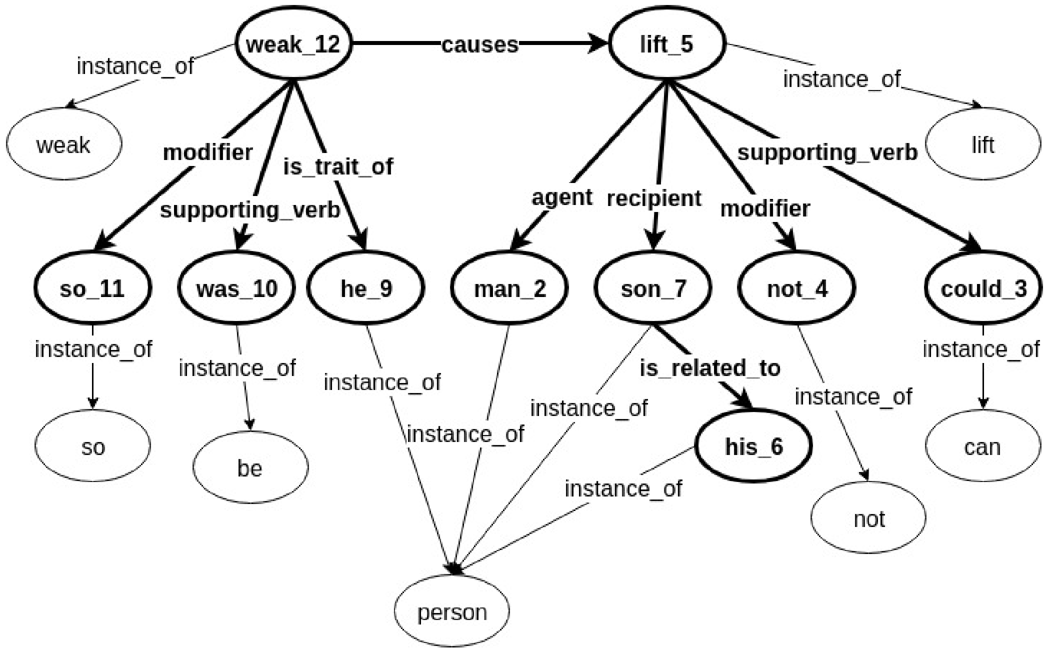


Figure 4.7: An Example of Step 1 Output of the WiSCR Algorithm with Respect to the WSC Sentence *"The man could not lift his son because he was so weak."*

STEP 2: In this step a subgraph of $\mathcal{G}_\mathcal{K}$ is extracted. Let the extracted subgraph is named $\mathcal{G}_\mathcal{K}{}'$. $\mathcal{G}_\mathcal{K}{}'$ contains all the nodes from $\mathcal{G}_\mathcal{K}$ which are not class nodes and it contains all the edges which connect such nodes, except the edges which are labeled as *'as_same_as'*. An example of the output of the Step 2 is shown in the Figure 4.8. The entire graph in the figure is the representation of a knowledge (as shown in Figure 4.6) and the highlighted part of the graph is the subgraph extracted in this step.

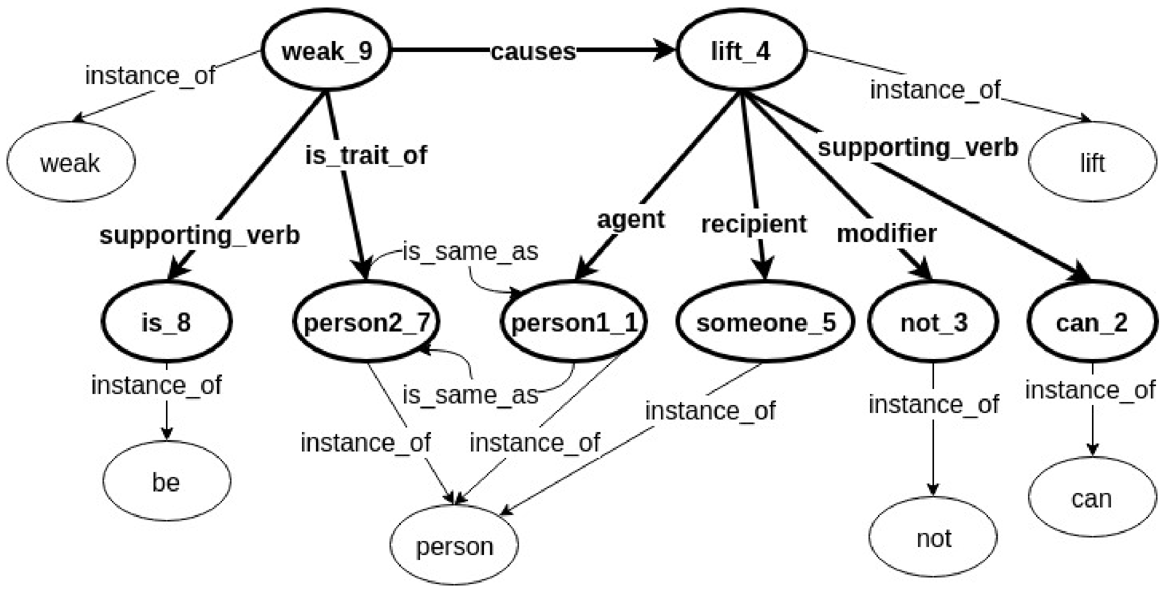


Figure 4.8: An Example of Step 2 Output of the WiSCR Algorithm with Respect to the Knowledge *"IF person1 can not lift someone because person2 is weak* ***THEN*** *person1_1 is same as person2_7"*

STEP 3: In this step, all the possible graph-subgraph isomorphisms (Cordella *et al.*, 2004) are detected between $\mathcal{G}_{\mathcal{S}}'$ and $\mathcal{G}_{\mathcal{K}}'$ (the subgraphs from the previous two steps respectively). A graph-subgraph isomorphism is a mapping (say $\mathbb{M}$) between two graphs ($\mathcal{G}_{\mathcal{S}}'$ and $\mathcal{G}_{\mathcal{K}}'$) such that $\mathbb{M}$ is a set of pairs of the form (x,y) where x is a node in $\mathcal{G}_{\mathcal{S}}'$, y is a node in $\mathcal{G}_{\mathcal{K}}'$, and if for every $(x,y) \in \mathbb{M}$, x is replaced by y then $\mathcal{G}_{\mathcal{K}}'$ becomes a subgraph of the node replaced $\mathcal{G}_{\mathcal{S}}'$. If such a mapping does not exist then $\mathbb{M} = \emptyset$. An important constraint that we put on the mapping set is that for each $(x,y) \in \mathbb{M}$, both x and y must be instances of same class. This is because our assumption for a correct knowledge is that it represents a scenario which is similar to the sentences in the concerned WSC problem. For example if a WSC sentence mentions about *'lift'* action with the help of the word *'lifting'* then a suitable knowledge must also mention about *'lift'* action. It does not matter which form of a word (e.g., *'lifting'* or *'lifts'*) is used in the knowledge or the WSC sentences. This information is captured by the class nodes in the graphical representations.

STEP 4: In this step an answer to a WSC problem is deduced from the input representations and the results of the previous steps of this algorithm. For each of the graph-isomorphism detected in Step 3, an answer to the input WSC problem is extracted by using the following rules.

- The answer choice a_1 is an answer with respect to the set $\mathbb{M}$ if $(p,n_1) \in \mathbb{M}$, $(a_1,n_2) \in \mathbb{M}$, either (n_1,n_2) or (n_2,n_1) is an edge in $\mathcal{G}_\mathcal{K}$ and it is labeled as *'is_same_as'*, and there does not exist an n and an x such that $(x,n) \in \mathbb{M}$ and either (n_1,n) or (n,n_1) is an edge in $\mathcal{G}_\mathcal{K}$ labeled as *'is_same_as'*
- The answer choice a_2 is an answer with respect to the set $\mathbb{M}$ if $(p,n_1) \in \mathbb{M}$, $(a_2,n_2) \in \mathbb{M}$, either (n_1,n_2) or (n_2,n_1) is an edge in $\mathcal{G}_\mathcal{K}$ and it is labeled as *'is_same_as'*, and there does not exist an n and an x such that $(x,n) \in \mathbb{M}$ and either (n_1,n) or (n,n_1) is an edge in $\mathcal{G}_\mathcal{K}$ labeled as *'is_same_as'*
- Otherwise the input WSC problem does not have an answer with respect to the set $\mathbb{M}$

Finally, after processing all the isomorphisms, if a_1 is the only answer retrieved then a_1 is the final answer. If a_2 is the only answer retrieved then a_2 is the final answer. Otherwise the algorithm does not ouput an answer.

Theorem 1 *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem $\mathcal{P}$, $\mathcal{G}_\mathcal{S} = (\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$ be a graphical representation of $\mathcal{S}$, p be a node in $\mathcal{G}_\mathcal{S}$ such that it represents the pronoun to be resolved in $\mathcal{P}$, a_1 and a_2 be two nodes in $\mathcal{G}_\mathcal{S}$ such that they represent the two answer choices for $\mathcal{P}$, and $\mathcal{G}_\mathcal{K} = (\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$ be a graphical representation of a knowledge such that $f_\mathcal{K}$ is defined using $f_\mathcal{S}$. Then, the Winograd Schema Challenge Reasoning (WiSCR) algorithm outputs,*

- *a_1 as the answer of $\mathcal{P}$, if only a_1 provides the 'most natural resolution' (By Definition 7) for p in $\mathcal{G}_\mathcal{S}$,*
- *a_2 as the answer of $\mathcal{P}$, if only a_2 provides the 'most natural resolution' for p in $\mathcal{G}_\mathcal{S}$,*
- *no answer otherwise*

Proof 1 *Proof of the theorem is in the Appendix.*

4.3.2 Implementation of the WiSCR Algorithm

There are various constraints imposed on the two input graphs in the WiSCR algorithm to retrieve the final answer. For example, in Step 3 a constraint that both the nodes in a pair belonging to an isomorphism set must be instances of the same class node. Considering that, our main motivation while implementing the WiSCR algorithm was to make the process of adding new constraints easier. Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988; Baral, 2003) provides this facility. Furthermore, 1) ASP has simple syntax yet is expressive and it is non-monotonic, 2) ASP has a strong theoretical foundation with many building-block results, allowing us to prove the correctness of our implementation in this work, and 3) ASP has several efficient solvers (Gebser *et al.*, 2007; Niemelä and Simons, 1997; Leone *et al.*, 2006).

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

$$a \leftarrow a_1, \ldots, a_m, \textit{not } a_{m+1}, \ldots, \textit{not } a_{m+n}$$

where a, $a_1, \ldots, a_{m+n}$ are atoms. The rule reads as "a is true if $a_1 \ldots a_m$ are all known to be true and $a_{m+1} \ldots a_{m+n}$ 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.

In this section, first we present the details of the ASP encoding of the inputs to WiSCR algorithm and an ASP implementation of the WiSCR algorithm. Then we show, with the help of examples, how the current implementation can be easily updated to include new constraints.

ASP encoding of Inputs

There are four inputs to the algorithm, a sequence of sentences in a WSC problem, a pronoun to be resolved, two answer choices and a knowledge. The WSC sentences are represented as a graph. Each edge in the graph is encoded in the ASP format by using a ternary predicate `has_s(h,l,t)`, where `h` and `t` are two nodes and `l` is an edge label of the directed edge from `h` to `t`. Similarly, a knowledge is represented as a graph and encoded in ASP by using a ternary predicate `has_k`. The pronoun is encoded in ASP by using a unary predicate `pronoun(p)` where `p` is the pronoun. Similarly, the two answer choices are encoded by using the unary predicates `ans_ch1(a1)` and `ans_ch2(a2)`, respectively. Based on the above, following are the ASP encoding of the WSC problem and the knowledge shown in the Figures 4.5 and 4.6 respectively.

```
% ASP encoding of the graph in Figure 4.5
has_s("he_9","instance_of","person").
has_s("weak_12","is_trait_of","he_9").
has_s("weak_12","modifier","so_11").
has_s("so_11","instance_of","so").
has_s("weak_12","supporting_verb","was_10").
has_s("was_10","instance_of","be").
has_s("weak_12","instance_of","weak").
has_s("lift_5","instance_of","lift").
has_s("lift_5","agent","man_2").
has_s("lift_5","recipient","son_7").
has_s("son_7","instance_of","person").
has_s("man_2","instance_of","person").
has_s("son_7","is_related_to","his_6").
has_s("his_6","instance_of","person").
has_s("weak_12","causes","lift_5").
has_s("lift_5","modifier","not_4").
has_s("not_4","instance_of","not").
has_s("lift_5","supporting_verb","could_3").
has_s("could_3","instance_of","can").

% ASP encoding of pronoun from the graph in Figure 4.5
pronoun("he_9").

% ASP encoding of answer choices from the graph in Figure 4.5
ans_ch1("man_2").
ans_ch2("son_7").

% ASP encoding of the graph in Figure 4.6
has_k("weak_9","instance_of","weak").
has_k("weak_9","is_trait_of","person2_7").
has_k("person2_7","instance_of","person").
has_k("weak_9","supporting_verb","is_8").
has_k("is_8","instance_of","be").
has_k("lift_4","instance_of","lift").
has_k("lift_4","agent","person1_1").
has_k("person1_1","instance_of","person").
has_k("lift_4","recipient","someone_5").
has_k("someone_5","instance_of","person").
has_k("weak_9","causes","lift_4").
has_k("lift_4","modifier","not_3").
has_k("not_3","instance_of","not").
has_k("lift_4","supporting_verb","can_2").
has_k("can_2","instance_of","can").
has_k("person1_1","is_same_as","person2_7").
has_k("person2_7","is_same_as","person1_1").
```

ASP implementation of the Step 1 of WiSCR Algorithm

In Step 1 of the WiSCR algorithm a subgraph of the graphical representation of WSC sentences is extracted such that the subgraph contains only the non-class nodes and the edges which are not labeled as *instance_of*. Following ASP rules encode the first step of the WiSCR algorithm.

```
s11: node_G_s(X) :- has_s(X,R,Y), R!="instance_of".
s12: node_G_s(Y) :- has_s(X,R,Y), R!="instance_of".
s13: edge_G_s(X,R,Y) :- has_s(X,R,Y), R!="instance_of".
```

`node_G_s(X)` represents a node `X` in the extracted subgraph, `edge_G_s(X,R,Y)` represents an edge, labeled `R`, between the nodes `X` and `Y` in the extracted subgraph. The output of the above rules, based on the encoding of inputs, as mentioned above, is,

```
node_G_s("weak_12")
node_G_s("lift_5")
node_G_s("son_7")
node_G_s("he_9")
node_G_s("so_11")
node_G_s("was_10")
node_G_s("man_2")
node_G_s("his_6")
node_G_s("not_4")
```

```
 node_G_s("could_3")
 edge_G_s("weak_12","is_trait_of","he_9")
 edge_G_s("weak_12","modifier","so_11")
 edge_G_s("weak_12","supporting_verb","was_10")
 edge_G_s("lift_5","agent","man_2")
 edge_G_s("lift_5","recipient","son_7")
 edge_G_s("son_7","is_related_to","his_6")
 edge_G_s("weak_12","causes","lift_5")
 edge_G_s("lift_5","modifier","not_4")
 edge_G_s("lift_5","supporting_verb","could_3")
```

ASP implementation of the Step 2 of WiSCR Algorithm

In Step 2 of the WiSCR algorithm a subgraph of the graphical representation of a knowledge is extracted such that the subgraph contains only the non-class nodes and the edges which are not labeled as *instance_of* or *is_same_as*. Following ASP rules encode the second step of the WiSCR algorithm.

```
s21: node_G_k(X) :- has_k(X,R,Y), R!="instance_of".
s22: node_G_k(Y) :- has_k(X,R,Y), R!="instance_of".
s23: edge_G_k(X,R,Y) :- has_k(X,R,Y), R!="instance_of",
                        R!="is_same_as".
```

`node_G_k(X)` represents a node `X` in the extracted subgraph and `edge_G_k(X,R,Y)` represents an edge, labeled `R`, between the nodes `X` and `Y` in the extracted subgraph. The output of the above rules, based on the encoding of inputs, as mentioned above, is,

```
node_G_k("weak_9")
node_G_k("lift_4")
node_G_k("person1_1")
node_G_k("person2_7")
node_G_k("is_8")

node_G_k("someone_5")

node_G_k("not_3")

node_G_k("can_2")

edge_G_k("weak_9","is_trait_of","person2_7")

edge_G_k("weak_9","supporting_verb","is_8")

edge_G_k("lift_4","agent","person1_1")

edge_G_k("lift_4","recipient","someone_5")

edge_G_k("weak_9","causes","lift_4")

edge_G_k("lift_4","modifier","not_3")

edge_G_k("lift_4","supporting_verb","can_2")
```

ASP implementation of the Step 3 of WiSCR Algorithm

Let $\mathcal{G}'_{\mathcal{S}}$ and $\mathcal{G}'_{\mathcal{K}}$ be the graphs extracted in step 1 and 2 of the WiSCR algorithm respectively. Then, in this step, all possible sets of pairs (say $\mathbb{M}_i$) of the form (x,y) are extracted from $\mathcal{G}'_{\mathcal{S}}$ and $\mathcal{G}'_{\mathcal{K}}$ such that x is a node in $\mathcal{G}'_{\mathcal{S}}$, y is a node in $\mathcal{G}'_{\mathcal{K}}$, both x and y are instances of the same class and if for every $(x,y) \in \mathbb{M}_i$, x is replaced by y then $\mathcal{G}'_{\mathcal{K}}$ becomes a subgraph of the node replaced $\mathcal{G}'_{\mathcal{S}}$. Following ASP rules encode the third step of the WiSCR algorithm.

```
s31: { matches(X,Y) : node_G_s(X), node_G_k(Y) }.

s32: :- matches(X,Y), matches(X1,Y), X!=X1.

s33: :- matches(X,Y), matches(X,Y1), Y!=Y1.

s34: k_node_matches(Y) :- matches(X,Y).

s35: :- not k_node_matches(Y), node_G_k(Y).

s36: :- matches(X,Y), has_s(X,"instance_of",C),

        not has_k(Y,"instance_of",C).

s37: :- edge_G_k(X1,R,Y1), matches(X,X1), matches(Y,Y1),

        not edge_G_s(X,R,Y).
```

Here, matches(X,Y) represents a pair in a $\mathbb{M}_i$. The rule s31 above generates all possible groundings of the form matches(X,Y) such that X is a node in the graph extracted in Step 1 and Y is a node in the graph extracted in Step 2. The rules s32 and s33 only keep the answer sets in which each X in the groundings of matches(X,Y) contains exactly one corresponding Y and vice-versa. The remaining answer sets are removed by the rules s32 and s33. The rules s34 and s35 removes all the answer sets in which there does not exist a grounding of matches(X,Y) corresponding to each node in the graph extracted in Step 2. The rule s36 removes all the answer sets in which at least one grounding of matched(X,Y) exists such that both X and Y are not instances of the same node in the knowledge graph. Finally, the rule s37 ensures that if two node X and Y in the graph extracted in the Step 2 match with two nodes X1 and Y1 respectively in the graph extracted in the Step 1, and (X1,R,Y1) is an edge in the graph from Step 2 then *(X,R,Y)* is an edge in the graph from Step 1.

The only set of pairs generated by the above rules, based on the encoding of inputs presented above, and the outputs of previous two steps is,

```
matches("weak_12","weak_9")
matches("lift_5","lift_4")
matches("man_2","person1_1")
matches("he_9","person2_7")
matches("was_10","is_8")
matches("son_7","someone_5")
matches("not_4","not_3")
matches("could_3","can_2")
```

As shown above, the rule s31 generates all possible candidate sets for graph-subgraph isomorphism between $\mathcal{G}'_S$ and $\mathcal{G}'_K$. Then the rules/constraints s32-37 remove the candidates which do not satisfy the required criteria. For example, the rule s36 removes the candidates which contain at least one grounding of `matches(X,Y)` where both `X` and `Y` are not instances of same class. The ease of adding such constraints motivated us to use ASP in this work. While adding such constraints in high level implementations of isomorphism algorithms would require one to delve deep into the implementations.

Implementation of the Step 4 of WiSCR Algorithm

In this step an answer to the input WSC problem is retrieved from the inputs of the WiSCR algorithm and the outputs of the steps 1 through 3. There are two parts of this the implementation in this step. The first part uses ASP rules to extract an answer from each set of pairs generated by the ASP implementation of Step 3 of the algorithm. Separate rules are used for each answer choice. Following ASP rules encode this part of Step 4 for the first answer choice.

```
s41: invalid_1 :- matches(P,N1), matches(X,N2),
                  ans_ch1(A), pronoun(P),
                  A!=X, N1!=N2,
                  has_k(N1,"is_same_as",N2).


s42: invalid_2 :- matches(P,N1), matches(X,N2),
                  ans_ch2(A), pronoun(P),
                  A!=X, N1!=N2,
                  has_k(N1,"is_same_as",N2).


s43: ans(A) :- matches(P,N1), matches(A,N2),
               ans_ch1(A), not invalid_1,
               pronoun(P), has_k(N1,"is_same_as",N2).


s44: ans(A) :- matches(P,N1), matches(A,N2),
               ans_ch2(A), not invalid_2,
pronoun(P), has_k(N1,"is_same_as",N2).
```

Here, `ans(A1)` represents that `A1` is an answer of the input WSC problem given a set of `matches`. Similar rules are written for the second answer choice (assume rules `s45`, `s46`, `s47`, `s48`). Finally the following rule makes sure that there is one answer generated with respect to one set of `matches(X,Y)` facts.

```
s49:  :- ans(A1), ans(A2), A1!=A2.
```

The above AnsProlog program produces zero or more answer sets. Zero answer sets mean that none of the sets of `matches` were able to produce an answer. The second part assembles all the answers and produces the final answer of the input WSC problem. To perform this aspect of the WiSCR algorithm, we defined a sub-algorithm named ANSWERFINDER. The ANSWERFINDER sub-algorithm takes as input the answers generated by the ASP code and outputs the final answer based on the following conditions.

1. if all the answers correspond to one common answer then the algorithm outputs it as final answer,
2. otherwise the algorithm does not ouput anything.

4.3.3 Adding New Constraints

In this section we present examples of how new constraints can be easily added to the above ASP implementation.

Suppose we would like to add a constraint that a pair of nodes are valid in a graph-subgraph isomorphism if the two nodes in it are synonyms of each other or they are instances of the same class node. Then we can encode such constraint by replacing the rule `s36` with the following three rules.

```
valid_pair(X,Y) :- has_s(X,"instance_of",C),
                   has_k(Y,"instance_of",C).
valid_pair(X,Y) :- synonyms(X,Y).
:- matches(X,Y), not valid_pair(X,Y).
```

Here, `synonyms(X,Y)` represents that a node `X` in the WSC sentences' graph is synonymous to a node `Y` in the knowledge graph. We assume that a set of `synonymous(X,Y)` facts are provided as input.

Let us consider the following WSC problem as an example to understand the significance of the above rules,

Sentence: The man could not lift his son because **he**$_{pronoun}$ was so weak.

Question: Who was weak? **Answer Choices:** a) man b) son

Now, let us consider the knowledge that,

IF *person1 could not lift someone because person2 was frail*

THEN *person1_1* **is same as** *person2_7*

The basic implementation of the WiSCR algorithm will not be able to utilize the above knowledge because the knowledge has the word *frail* instead of *weak*. However since *weak* is a synonym of *frail*, if we provide `synonyms(weak_12,frail_9)` as an input to the code which is updated by replacing the rule `s36` with the above mentioned three rules then the ASP implementation can handle the knowledge and the algorithm outputs the correct answer, i.e., *man_2*.

Replacing an existing rule with only three new ones allows the algorithm to be more flexible with respect to the needed knowledge. This also shows how additional constraints and generalizations can be easily expressed as new ASP rules.

Another generalization could be done by using similarity along with synonymy to add node pairs in an isomorphism. We say that if the similarity between two nodes is above a certain threshold then allow them to be added to the isomorphism set. An additional rule to encode that would be,

```
valid_pair(X,Y) :- similar(X,Y).
```

Here, `similar(X,Y)` represents that a node `X` in the WSC sentences' graph is similar to a node `Y` in the knowledge graph. We assume that a set of `similar(X,Y)` facts are provided as input.

Definition 9 (AnsProlog Program for WiSCR Algorithm) *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem $\mathcal{P}$, $\mathbb{T}(S)$ be the set of tokens in $\mathcal{S}$, $p \in \mathbb{T}(S)$ be the token which represents the pronoun to be resolved, $a_1, a_2 \in \mathbb{T}(S)$ be two tokens which represent the two answer choices, $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}}, \mathbb{E}_{\mathcal{S}}, f_{\mathcal{S}})$ be a graphical representation of $\mathcal{S}$, and $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}}, \mathbb{E}_{\mathcal{K}}, f_{\mathcal{K}})$ be a representation of a knowledge such that $f_{\mathcal{K}}$ is defined using $f_{\mathcal{S}}$. Then, we say that the AnsProlog program $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ is the answer set program consisting of a) the facts of the form $has\_s(h_1, l_1, t_1)$ and $has\_k(h_2, l_2, t_2)$,b) a fact of the form $pronoun(p)$,c) two facts of the form $ans\_ch1(a_1)$ and $ans\_ch2(a_2)$,d) the rules `s11` to `s49`*

Theorem 2 *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem $\mathcal{P}$, $\mathbb{T}(S)$ be the set of tokens in $\mathcal{S}$, $p \in \mathbb{T}(S)$ be the token which represents the pronoun to be resolved, $a_1, a_2 \in \mathbb{T}(S)$ be two tokens which represent the two answer choices, $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}}, \mathbb{E}_{\mathcal{S}}, f_{\mathcal{S}})$ be a graphical representation of $\mathcal{S}$, and $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}}, \mathbb{E}_{\mathcal{K}}, f_{\mathcal{K}})$ be a representation of a knowledge such that $f_{\mathcal{K}}$ is defined using $f_{\mathcal{S}}$. Also, $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ be the AnsProlog program for WiSCR algorithm and* ANSWERFINDER *be the sub-algorithm defined in Section 4.3.2. Then, the*

WiSCR algorithm produces an answer x to the input WSC problem iff $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ *and* ANSWERFINDER *together output the answer x.*

Proof 2 *Proof of the theorem is in the Appendix.*

4.4 Empirical Evaluation of the WiSCR Algorithm

The reasoning framework defined in this work relies on the following two assumptions,

1. A WSC problem can be automatically translated into the desired formal representation, and
2. The required knowledge instances can be automatically extracted.

In this section, we show the results of an empirical evaluation of the reasoning framework by automating the above two assumptions to a great extent with the help of a semantic parsing system (K-Parser) and an in-house knowledge extraction system.

We used K-Parser (Sharma *et al.*, 2015b) for translating the input problems into the desired formal representation. K-Parser (or Knowledge Parser) is a semantic parsing and knowledge augmentation system which was developed as a part of this work. The detailed explanation of the implementation of K-Parser and how it is used to translate WSC problems into graphs is available in the Chapter 6 of this thesis.

We also developed a rule based knowledge extractor for this work. There are three aspects of our system,

1. creating search queries from the WSC problems,
2. using a search engine to collect the sentences which satisfy the queries, and
3. extract knowledge from the collected sentences, if they contain any.

Because of the limited availability of search engine access, the first two aspects were carried out manually. The collected sentences are then passed one by one to an in-house rule based knowledge extraction module [4] . The module uses semantics (from K-Parser) of the input sentences to find the patterns which satisfy the knowledge types (Type 1 to 10 in Chapter 3) which are currently handled by our reasoning framework. The details of the knowledge extraction module are available in the Chapter 5 of this thesis.

Based on the above mentioned setup, we were able to automatically extract knowledge corresponding to 120 WSC problems and the reasoning algorithm answered all of the 120 problems correctly. The details about the number of problems solved for each knowledge type is as shown in the Table 4.2 below.

Knowledge Type	**Total WSC Prolems**	**Solved Correctly**
Knowledge Type 1	17	9
Knowledge Type 2	58	17
Knowledge Type 3	38	14
Knowledge Type 4	21	3
Knowledge Type 5	8	3
Knowledge Type 6	25	15
Knowledge Type 7	1	0
Knowledge Type 8	1	0
Knowledge Type 9	68	59
Knowledge Type 10	3	0

Table 4.2: Detailed Results of Empirical Evaluation of the Reasoning Framework

In another setting, to evaluate the performance of the reasoning algorithm and the semantic parser, we provided a WSC problem and a knowledge (in *“IF S **THEN** x is same*

[4]preliminary version of the system is mentioned in detail in Chapter 5

as y" format) as input to the system. There were 240 problems which required a knowledge of types 1 through 10 (See Chapter 3 for knowledge types). The system was able to answer 200 out of 240 WSC problems. All of the 200 problems were correctly answered by the system. The remaining 40 problems were not translated into the needed graphical representation by the semantic parser. The main reasons for such errors were incorrect part-of-speech tagging and syntactic parsing.

The high precision of the reasoning algorithm empirically supports the correctness of the algorithm. However the low recall of the knowledge extraction aspect reflects on the difficult nature of the commonsense knowledge extraction.

4.5 Representation of the Knowledge Types: Revisited

The representation and reasoning with respect to the first 10 types of knowledge (As shown in Chapter 3) are covered in the previous sections. So, in this section we will discuss the remaining two types of knowledge (Type 11 and Type 12 from Chapter 3).

4.5.1 Representation of the Knowledge Type 11: Statement 1 is more likely than Statement 2

This type consists of a comparison between two statements or propositions. The likelihood is based on the acceptability of a statement in a normal scenario. Following is a WSC problem and a knowledge of this type that is required to solve the problem. This kind of knowledge is represented as a statement of the form,

Statement1 ***is more likely than*** *Statement2*

where *Statement1* and *Statement2* are two English sentences.

Following is a WSC example which requires a knowledge of type 11. The needed knowledge is also mentioned below.

Sentence: Sam tried to paint a picture of shepherds with sheep, but **they**$_{pronoun}$ ended up looking more like dogs.

Question: What looked like dogs?

Answer Choices: a) sheep b) shepherds

Knowledge Needed:

sheep look like dog ***is more likely than*** *shepherds look like dogs*

4.5.2 Representation of the Knowledge Type 12: Multiple Knowledge Pieces

In this category, a list of more than one piece of knowledge of types 1 through 10 is called a knowledge. Following is a WSC problem and a knowledge of this type that is required to solve the problem.

Sentence: Mary tucked her daughter Anne into bed, so that **she**$_{pronoun}$ could work.

Question: Who is going to work?

Answer Choices: a) Mary b) daughter

Knowledge Needed:

1. *Person1 tucks Person2 into bed* ***normally causes*** *Person2 sleeps*
2. *daughter of Person1 sleeps* ***normally prevents*** *Person1 is disturbed*
3. *Person1 is not disturbed* ***normally causes*** *Person1 can work*

Similar to the representation of the knowledge from types 1 through 10, each piece of knowledge in this category is represented as a separate graph (By Definition 6).

4.6 Reasoning : Revisited

In this section we revisit the reasoning approach and define the reasoning process to handle the knowledge types 11 and 12 from the Chapter 3.

4.6.1 Reasoning Algorithm to Handle "is more likely than" Knowledge

The knowledge type 11 that is defined in Chapter 3 of this work is based on the likelihood of a proposition with respect to another proposition under normal situations. The reasoning algorithm defined in the sections above is not capable of handling such knowledge. This is because it relies on matching a representation of a knowledge with respect to that of an input sentence but in this case such a matching does not help. So, we developed an algorithm to handle the "is more likely than" kind of knowledge. In this section we present the details of the algorithm.

Let W be a Winograd Schema Challenge problem, S be the sentence in W, Q be the question in W, P be the pronoun in S such that to answer Q correctly one must resolve P to its correct co-referent in S, $C1$ and $C2$ be the two possible answers of Q and both of them are present in S, and A be the correct answer of Q i.e., $A = C1$ or $A = C2$. Also, let K be a knowledge of "is more likely than" kind. K is of the form *PROP1* ***is more likely than*** *PROP2*. Then, following are the main components of the algorithm.

- **Input:** The inputs to the algorithm are S, Q, P, $C1$, $C2$, A and K.
- **Output:** The output of the algorithm is an answer to Q. It is either $C1$ or $C2$.
- **Behavior:** Following are the steps that depict the functionality of the algorithm.

  1. **Premises Production:** In this step two premises are generated from S. The first premise ($Premise1$) is generated by replacing P in S by $C1$. The second premise ($Premise2$) is generated by replacing P in S by $C2$.

2. **Hypothesis Identification:** In this step, a proposition/statement which is deemed to be more likely in the input knowledge is identified as a hypothesis H.

3. **Final Answer Retrieval:** In this step the entailment scores between the pairs $(Premise1, H)$ and $(Premise2, H)$ are calculated. If the entailment score of $(Premise1, H)$ is greater than the entailment score of $(Premise2, H)$ then $C1$ is the final answer, otherwise $C2$ is the answer. We used ESIM with ELMO encodings (Parikh *et al.*, 2016; Chen *et al.*, 2016) to calculate the entailment scores.

Let us consider a couple of examples from the Winograd Schema Challenge corpus to better understand the above algorithm.

Example 1

S = Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like dogs.

Q = What looked like dogs?

P = they

$C1$ = sheep

$C2$ = shepherds

A = sheep

K = sheep look like dogs **is more likely than** shepherds look like dogs

Following are the outputs of the different modules of the behavior of the reasoning algorithm as explained above.

1. Premise Production: Following premises are generated in this step.

Premise 1: Sam tried to paint a picture of shepherds with sheep, but sheep ended up looking more like dogs.

Premise 2: Sam tried to paint a picture of shepherds with sheep, but shepherds ended up looking more like dogs.

2. Hypothesis Identification: Following hypothesis is identified in this step.

Hypothesis (H): sheep look like dogs

3. Final Answer Retrieval: Following are the final answers and intermediate results retrieved in this step.

Entailment_score(Premise1,H) = 75.4%

Entailment_score(Premise2,H) = 58.3%

Final Answer = sheep

Example 2

S = Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like golfers.

Q = What looked like golfers?

P = they

$C1$ = sheep

$C2$ = shepherds

A = shepherds

K = shepherds look like golfers **is more likely than** sheep look like golfers

Following are the outputs of the different modules of the behavior of the reasoning algorithm as explained above.

1. Premise Production: Following premises are generated in this step.

Premise 1: Sam tried to paint a picture of shepherds with sheep, but sheep ended up looking more like golfers.

Premise 2: Sam tried to paint a picture of shepherds with sheep, but shepherds ended up looking more like golfers.

2. Hypothesis Identification: Following hypothesis is identified in this step.

Hypothesis (H): shepherds look like golfers

3. Final Answer Retrieval: Following are the final answers and intermediate results retrieved in this step.

Entailment_score(Premise1,H) = 90.1%

Entailment_score(Premise2,H) = 94.9%

Final Answer = shepherds

Evaluation

We found 26 winograd schema challenge problems which required the "is more likely" knowledge. To evaluate our algorithm with respect to the problems, we manually wrote down the needed knowledge and used it in the algorithm. out of 26 problems, 17 were correctly answered by the algorithm. The remaining 9 were incorrectly answered.

4.6.2 Reasoning Algorithm to Handle Multiple Knowledge Pieces

The reasoning algorithms defined above handle the knowledge types 1 through 11 as identified in Chapter 3. Each of the knowledge in those types consists of only one piece of knowledge. However, each knowledge instance of the type 12 knowledge defined in Chapter 3 consists of more than one piece of knowledge. Also, each of the piece in those pieces is of either one of the types 1 through 10. Let us consider an example from the Winograd Schema Challenge corpus which requires multiple pieces of knowledge.

Example

Sentence: *Tom threw his schoolbag down to Ray after* ***he***$_{pronoun}$ *reached the bottom of the stairs.*

Question: Who reached the bottom of the stairs?

Answer Choices: *a) Ray b) Tom*

Needed Knowledge:

1. *x reach bottom* ***may cause*** *x at bottom*
2. *y is at bottom* ***may co-occur with*** *z throws down to y*

Reasoning with multiple knowledge pieces is an interesting line of work. In this work, we propose it as a possible future work. Though reasoning with multiple knowledge pieces is not the focus of this work, in Chapter 8 we present a proof of concept reasoning algorithm for handling multiple knowledge pieces.

Chapter 5

KNOWLEDGE HUNTING AND KNOWLEDGE BASE CREATION

In this chapter, we present our attempts in the direction of extracting various kinds of commonsense knowledge from text. Here we focus on the commonsense kinds which were defined in the Chapter 3.

5.1 Introduction & Motivation

Commonsense knowledge plays an important part in Natural Language Understanding. It is because of commonsense that people, when perceive an action, are able to predict its plausible causes, effects, preconditions and the conditions which prevents the action from executing. For example, people can easily predict that *"someone attends to a person"* may be an effect of *"the person is injured"*. Reasoning with such knowledge plays an important role in NLU tasks such as the Winograd Schema Challenge (Levesque *et al.*, 2011). But, the current NLU systems lack such commonsense knowledge and often the knowledge is needed to be extracted separately.

In this chapter we provide a detailed description of our attempts to automatically extract such knowledge from text. Although the commonsense knowledge acquisition community have discussed about reporting bias (Gordon and Van Durme, 2013) and why that makes it difficult to extract commonsense knowledge from text, in this work we overcome the bias by performing the human validation of the automatically extracted knowledge.

In recent years, various challenges have been proposed that require various kinds of commonsense knowledge. Two of the famous challenges include the Winograd Schema Challenge (Levesque *et al.*, 2011) and the COPA challenge (Roemmele *et al.*, 2011). Much of the knowledge needed in these challenges revolves around actions and their plausible

The Winograd Schema Challenge **Sentence:** The man could not lift his son because he was weak. **Question:** Who was weak? **Answer:** man **Needed Knowledge:** person1 is weak may prevent person1 lifts something
Choice of Plausible Alternatives (COPA) **Premise:** I knocked on my neighbor's door. **Question:** What happened as an effect? **Alternative 1:** My neighbor invited me in. **Alternative 2:** My neighbor left the house. **Correct Alternative:** Alternative 1 **Needed Knowledge:** person X knocks on person Y's door may cause person Y invites person X in.

Table 5.1: Example Problems That Require Commonsense Knowledge

causes, effects, preconditions and the conditions which prevent the actions from executing. See examples in the Table 5.1.

The popular NLU tasks such as hard co-reference resolution, and deep QA require some sort of world knowledge about the problem. Both factual knowledge such as *Gravity is a kind of force*, and commonsense knowledge such as *if there was no gravity then we would not stay on the ground* are important in completely understanding the meaning of text. A way in which such knowledge is attained by humans by experiencing different scenarios that include entities (concrete or abstract), their interaction with other entities, and their participation in events (actions or state of being). There are systems such as IBM Watson for Deep Question Answering which make use of various Knowledge Bases (Singhal, 2012;

Leacock and Chodorow, 1998) that contain such knowledge.

There has been various works for extraction of both factual and commonsense knowledge in the past such as WordNet (Miller, 1995), ConceptNet (Liu and Singh, 2004) and Cyc (Lenat, 1995). Most of these are created by hand by a small number of people and hence it is not possible for them to have every kind of commonsense knowledge that all the humans have. Recently, there has also been some work for automatic extraction of knowledge from natural language text. One such attempt is NELL (Betteridge *et al.*, 2009). It also does not focus on the kind of knowledge that we have extracted in this work. These knowledge repositories are a great source but as mentioned before, the knowledge contained in them is not sufficient to solve many hard NLU problems. For example the knowledge used in (Sharma *et al.*, 2015a) is not present in any of them.

In this chapter we present our attempts towards extracting commonsense knowledge about actions. We present an algorithm to automatically extract such knowledge from text. We explain the technical details of our algorithm and present the detailed results of the extraction experiments. We discuss how this knowledge is different from the already existing ones. We validate the extracted knowledge with the help of human workers and make the valid knowledge publicly available for download as well as through an on-line interface to view the knowledge. We also present an approach to use the extracted knowledge in generating a set of pronoun disambiguation problems which require commonsense knowledge for disambiguation.

5.2 Events-Based Conditional Commonsense Knowledge Extraction

In our preliminary attempt, we defined an algorithm to automatically extract a special kind of commonsense knowledge about events. The knowledge is proved helpful in solving a subset of the Winograd Schema Challenge (WSC) (Sharma *et al.*, 2015a), which is a pronoun resolution challenge. A problem in the challenge consists of a sentence and a question about the sentence such that to answer the question, one must resolve a pronoun to its correct co-referent in the sentence. Let us take an example inspired from the WSC to understand the extracted knowledge through an example.

Sentence: *John was bullying Tom so we rescued him.* **Question:** *Who did we rescue ?*

Required commonsense knowledge:

IF A *bullying* B *causes* T *rescued* Z ***THEN*** (possibly) Z = B

In other words, if someone bullies someone/something and due to that someone is rescued then the one who is rescued is likely to be the one who was bullied. The knowledge is based on two events, for example *someone bullying someone/something* and *someone being rescued*, so in this work we call such knowledge as an Event-Based Conditional Commonsense (ECC).

The sections below explain details of the knowledge extraction algorithm.

5.2.1 Background

Corpus

One of the ways in which people acquire commonsense knowledge is by reading. They acquire the knowledge about actions and the relationships between them by reading about them in a story or another form of free text. In this work, we use this intuition to extract the above mentioned kind of knowledge from raw English sentences. We selected the freely available Open American National Corpus (OANC) [1] for this project. OANC is a massive collection American English text, both written and spoken (for this work we chose only the written part of it). It is a collection of 15 million words, and it is designed to represent a wide cross-section of American English collected on or after 1990. The OAN corpus comes with different annotations but we have used only the unannotated form of the written texts.

[1]The OAN corpus is available for free at *http://www.anc.org/data/oanc/download/*.

Semantic Parsing

The knowledge type focused in this work requires understanding of semantics of natural language sentences. So, we need some sort of semantic representation of the text for knowledge extraction. We do not believe that such a knowledge extraction can be easily done by using the bag-of-words or word-vector representations as it often ignores the context. To perform the extraction of entities, events and their inter-relations, we used our in house semantic parsing and knowledge augmentation system called K-Parser (available at *www.kparser.org*) (Sharma *et al.*, 2015a). The K-Parser system translates a sentence into a semantic graph that captures the semantics in the sentence. The parser is developed as a part of the project (Sharma *et al.*, 2015a) to solve the Winograd Schema Challenge (WSC) (Levesque *et al.*, 2011) but it has evolved into a general purpose semantic parser since then. A detailed explanation of the parsing system is available in the Chapter 6 of this dissertation.

The parser uses the Stanford Dependency Parser (De Marneffe and Manning, 2008) as a base to retrieve the dependencies between the events and entities in the input text. Due to the lack of generality in relations of Stanford Dependency parse, the parser uses a more general relation set from Knowledge Machine Component Library (Clark *et al.*, 2004), and many other new relations inspired from the need. The parser maps the Stanford dependencies to the semantic relations. The output of K-Parser also has two level of conceptual class information about the events and entities in the input text. More information about K-Parser is available at *www.kparser.org*

5.2.2 Knowledge Extraction Algorithm

The knowledge extraction algorithm defined in this work takes an English sentence as input and outputs a piece of knowledge that belongs to the type of knowledge defined above, provided the input sentence contains the knowledge.

The goal of the knowledge extraction algorithm is to identify and extract the relevant knowledge in a sentence. We took a logic-based approach for the extraction, where we see the knowledge extractor as an intelligent agent which uses a specific language to represent its knowledge and a reasoning algorithm to extract commonsense from text. We used Answer Set Programming(ASP) (Gelfond and Lifschitz, 1988; Baral, 2003) to perform the logical reasoning with respect to the semantic representation generated by the K-Parser system. Following are the main steps in the algorithm which are used to extract a knowledge from an input sentence.

STEP 1: Representing an English Sentence

We used the K-Parser system to translate an input English sentence into its semantic representation. The output of the K-Parser system is a directed acyclic graph. Let *S* be an input sentence and *G* be the semantic graph of *S* as generated by the K-Parser system. We used an RDF triples style representation of *G* to use it in our logical knowledge extraction system. In this step we make the semantic graph of a sentence to comply with the syntax requirements of our logic programming module. Each edge in *G* is translated into a *has*-predicate. Each *has*-predicate is of arity three and has the following form $has(X, rel, Y)$, where X, Y are the nodes in the graph and *rel* is the edge label between X and Y. For example an edge labeled *"causes"* between *"bullying_3"* and *"rescued_7"* is represented as

```
has(bullying_3,causes,rescued_7).
```

Similarly all the edges are translated into *has*-predicates. The label *"bullying_3"* in the graph refers to the word *"bullying"* appearing at the third position in the input sentence.

STEP 2: Logical Knowledge Extraction

In this step a logical knowledge extraction engine is used to extract the previously defined kind of knowledge from the semantic representation of the input sentence. The goal of the logical engine is to find relevant knowledge in the set of *has*-predicates corresponding to the representation of the input sentence. To achieve this goal, we first encoded the domain knowledge in the agent's brain. The following block of code describes all possible relations between any two events in the K-Parser output.

```
eventRelation(causes).
eventRelation(caused_by).
eventRelation(enables).
eventRelation(enabled_by).
eventRelation(objective).
eventRelation(next_event).
eventRelation(previous).
eventRelation(event).
eventRelation(resulting_state).
eventRelation(subevent).
eventRelation(inibits).
eventRelation(inhibited_by).
```

Knowing "all" possible relations between two events, an intelligent agent should be able to tell whether any given relation is an event relation. The ASP rule below encodes that information by using the `nonEventRelation` predicate for all the non event relations in the input graph.

```
nonEventRelation(R) :- has(X,R,Y), not eventRelation(R).
```

Similarly the following block of ASP code describes domain knowledge encoded in the agent's mind that two event nodes are connected via an event relation and a node is negative or positive.

```
relatedEvents(V1,R,V2) :- has(V1,R,V2), eventRelation(R).
           negative(V1) :- has(V1,negative,N).
```

```
positive(V1) :- not negative(V1),
                relatedEvents(V1,R,V2).
positive(V2) :- not negative(V2),
                relatedEvents(V1,R,V2).
```

Having this knowledge, the following block of code shows how the agent can extract relevant commonsense knowledge from the *has*-predicates.

```
answerEvents(positive,V1,VV1,R1,X1,R,positive,V2,VV2,R2,X2
   ):-
                            relatedEvents(V1,R,V2),
                            has(V1,R1,X1),
                            has(V2,R2,X2),
                            has(X1,instance_of,X),
                            has(X2,instance_of,X),
                            has(V1,instance_of,VV1),
                            has(V2,instance_of,VV2),
                            positive(V1),positive(V2),
                            nonEventRelation(R1),
                            nonEventRelation(R2).
```

When the predicate *answerEvents*(.....) evaluates to $true$ for some assignment of the input variables according to the rule specified above, it describes that

"if event V1 is related to event V2 by an event relation R and the polarity of both the events are positive, then the entity X1 related to V1 with relation R1 is identical to the entity X2 related to V2 by the relation R2."

The values inside the predicate *answerEvents*(.....) are (in order of occurrence) the polarity of the event V1, the actual value of event V1, base form of V1 i.e. VV1, relation between V1 and X1 i.e. R1, the actual value of X1, the relation between V1 and V2, the polarity of the event V2, the actual value of event V2, base form of V2 i.e. VV2, relation between V2 and X2 i.e. R2 and the actual value of X2.

The above block of code shows one ASP rule that is used to extract the commonsense in the case where both the related events are of positive polarity. The other three cases with different combinations of polarities also work in similar fashion.

5.2.3 Storage and Retrieval of the Knowledge

Storage

Hence, the goal here was to make the extracted knowledge available to the NLU research community so that it can be used in a variety of applications. To accomplish this we have used MongoDB database to save the extracted knowledge. MongoDB was chosen because of its speed, accessibility and usefulness.

Knowledge Retrieval

The knowledge database consists of three sets of elements, namely, the set of events ($\mathcal{E}$), the set of relation among events ($\mathcal{R}$) and the set of slot-relations connecting events and entities ($\mathcal{S}$). Each query in the knowledge retrieval language posed to the database is a tuple consisting of elements from all these sets. Currently, we have defined the following seven queries,

1. *Event1=*E_1*,Rel=R,Event2=*E_2*,Slot1=*S_1*,Slot2=*S_2

2. *Event1=*,Rel=R,Event2=*E_2*,Slot1=*S_1*,Slot2=*S_2

3. *Event1=*E_1*,Rel=R,Event2=*,Slot1=*S_1*,Slot2=*S_2

4. *Event1=*E_1*,Rel=R,Event2=*E_2*,Slot1=*,Slot2=*S_2

5. *Event1=*E_1*, Rel=R, Event2=*E_2*, Slot1=*S_1*, Slot2=**

6. *Event1=*E_1*,Rel=*$*$*,Event2=*E_2*,Slot1=*S_1*,Slot2=*S_2

7. *Event1*=E_1,*Rel*=$*$,*Event2*=E_2,*Slot1*=$*$,*Slot2*=$*$ The star ("*") in above queries means any legal value. For example, query 5 is used to extract all the knowledge instances where *"event1"* is E_1, *"event2"* is E_2, *"relation"* is R, *"slot1"* is S_1 and any legal value for *"slot2"*.

We have also developed a web interface for querying the knowledge base. The web application accepts the input query in both form-based and free-form (natural language). A demo version of the system is available at **`http://bioai8core.fulton.asu.edu/knet/`**

5.2.4 Evaluation

The kind of knowledge extracted here is already proved useful in solving a hard problem (Sharma *et al.*, 2015a) such as WSC. Now the questions arise, if the knowledge base created in this experiment has sufficient instances of such knowledge and are those instances are of any use. Both quantitative and qualitative analysis were performed to address these questions. The details are mentioned below.

Quantitative & Qualitative Evaluation

We performed the knowledge extraction experiment on the Open American National (OAN) corpus. The written part of the corpus contains a total of 6405 documents from six genre. On an average each document contains 106 sentences. Out of all the sentences our system was able to extract 19336 instances of knowledge. In other words, about 2.85% (19336 out of 678930) of the sentences in OAN corpus were found to contain the required commonsense knowledge.

For the qualitative analysis, a set of 886 instances of the commonsense knowledge were randomly sampled from the 19336 total instances. Two human evaluators [2] were employed to test the quality of the instances. Each evaluator was provided with a natural language

[2]One undergraduate Computer Science student and one graduate Computer Science student.

translation of 543 commonsense knowledge instances (with an overlap of 200) along with the instructions to rank the quality of the knowledge. This evaluation instructions were inspired from (Gordon *et al.*, 2010). The table 5.2 shows the evaluation results. Here, rank 1 means the evaluator agrees that the knowledge instance is clear and entirely plausible. Rank 6 means the evaluator disagrees on the good quality of the knowledge. Rank 2 to 5 are in the order of decreasing evaluator agreement.

Table 5.2: Evaluation Results for 886 Randomly Selected Knowledge Instances

Rank	**Evaluator 1 (E_1)**	**Evaluator 2 (E_2)**	**Both E_1 and E_2**
1	326	359	91
2	54	44	15
3	33	16	6
4	19	30	3
5	12	21	3
6	99	73	38
Miss-Matches	-	-	44
Total	543	543	200

We also analyzed the agreement among the evaluators by counting the difference in the rankings of both the evaluators. The results chart is shown in Fig. 5.1. From this experiment, we found that the evaluators were in agreement 156 of the times out of 200 and 25 times there was only a difference of 1 between their rankings (cased on our evaluation schema). This shows that most (90.5%) of the times the evaluators were either in complete agreement with each other or they ranked next to each other.

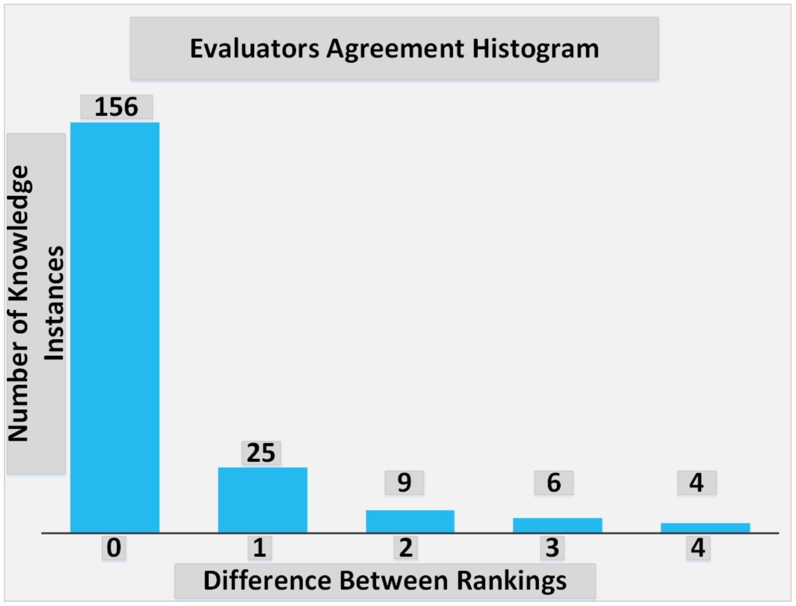


Figure 5.1: Evaluator Agreement Chart

5.3 Extraction of Commonsense Knowledge about Actions

As a second attempt towards extracting commonsense knowledge, we extracted the knowledge about actions. The knowledge types 1 through 10 mentioned in the Chapter 3 can be viewed as the causes, effects preconditions of actions. Below are the details of our extraction attempt and the results.

5.3.1 Background

In this work, we attempt to automatically extract commonsense knowledge about actions. We extract the knowledge in terms of plausible causes, effects, preconditions of actions and conditions which prevent actions from executing. A few examples of the knowledge extracted in this work are shown in Table 5.3. The extracted knowledge is characterized based on the entities which play various roles in the concerned action. For example, *"person1 is injured may cause someone attends to person1"*. Where, *"attends"* is the main action and a plausible cause of its execution is *"person1 is injured"*. Here, we

Knowledge Type	**Example Knowledge**
Cause of an Action	*person1 is injured may cause* ***people attend to person1***
Effect of an Action	***everybody hates person1*** *may cause person1 is nervous*
Precondition of an Action	*someone hit person1 may be followed by* ***person1 faints***
Condition Preventing an Action from Executing	*person1 is jet-lagged may prevent* ***person1 could sleep at night***

Table 5.3: Knowledge Types and Their Examples

extract the knowledge with respect to the entities that participate in actions. The extracted knowledge is general because the entities are represented by their types appended by a number at the end for discrimination among them (e.g. *person1*).

In this work, every cause, effect, precondition and condition which prevents an action from executing is categorized into either an action or a property. An action refers to a verb phrase and a property refers to an adjective phrase. For example, let *"person1 attends to person2"*, then two possible causes of the action *"attends"* are, (1) a property such as *"ill"* in *"person2 is ill"* and (2) an action such as *"faints"* in *"person2 faints"*.

5.3.2 The Extraction Algorithm

The extraction Algorithm developed in this work takes an English sentence as input and outputs a set K of knowledge instances. Each instance in K belongs to one of the first ten knowledge types mentioned in the Chapter 3. The complete algorithm is shown in the Algorithm 1. The basic idea is to first divide an input sentence into two parts such that one of the parts contains an action and the other part contains its cause, effect, precondition or condition which prevents the action. Then the two parts are used to generate a knowledge instance. If the input sentence can not be divided in such a way then no knowledge is extracted from it.

```
Data: An English sentence
Result: A set of knowledge (K) about actions
K ← []
splits ← SPLIT(S)
SG ← GETSEMANTICPARSE(S)
for each triple <S1,S2,S3> in splits do
    if $S1 \neq \varepsilon$ & $S2 \neq \varepsilon$ & $S3 \neq \varepsilon$ then
        SGS1 ← GETSEMGRAPH(SG,S1)
        SGS2 ← GETSEMGRAPH(SG,S3)
        raw_knowledge ← GETKNOWLEDGE(SGS1,SGS2,S2)
        knowledge = POSTPROCESS(raw_knowledge)
        Add knowledge to K
    end
end
```

Algorithm 1: Knowledge Extraction Algorithm

There are following main procedures in the extraction algorithm.

1. **SPLIT:** There are different phrases in English which act as the cues for the knowledge types mentioned in the Table 5.3. For example *"because"* and *"leads to"* correspond to causality based knowledge. Following up on the idea of dividing a sentence into two parts, this procedure uses the knowledge identification cue

phrases [3] and divides a sentence into triples of the form $< S1, S2, S3 >$. The first part of a triple (i.e., $S1$) corresponds to the part of the input sentence which may contain an action, the second part in the triple (i.e., $S2$) corresponds to a knowledge type identification cue phrase, and the third part in the triple (i.e., $S3$) corresponds to the part of the input sentence which may contain a plausible cause, effect, precondition or a condition which prevents the action from executing. If $S2$ is not found in the input sentence then $S1 = \varepsilon$ and $S3 = \varepsilon$. For example let us consider the input sentence *"She could not lift him because she is weak"*. Then, the only split generated by this procedure is <*She could not lift him,because,she is weak*>. Here $S1$ = *She could not lift him*, $S2$ = *because* and $S3$ = *she is weak.*

2. **GETSEMANTICPARSE:** This procedure takes an English sentence as input and returns its graphical meaning representation that is produced by the Knowledge Parser (K-Parser) (Sharma *et al.*, 2015b). The representation is a directed, acyclic, edge-labeled graph. The nodes in the graph represent actual words (or concepts including actions, entities and properties) in the input sentence and the types of the concepts (e.g. *"She"* is of type *"person"*). The edges on the other hand represent the semantic relationships between the nodes. More details about K-Parser output are present in Chapter 6. For example the semantic parse of the sentence *"She could not lift him because she is weak"* is as shown in the Figure 5.2.

[3] the complete list of cues is available at *https://github.com/arpit7123/CommonsenseKnowledgeExtraction*

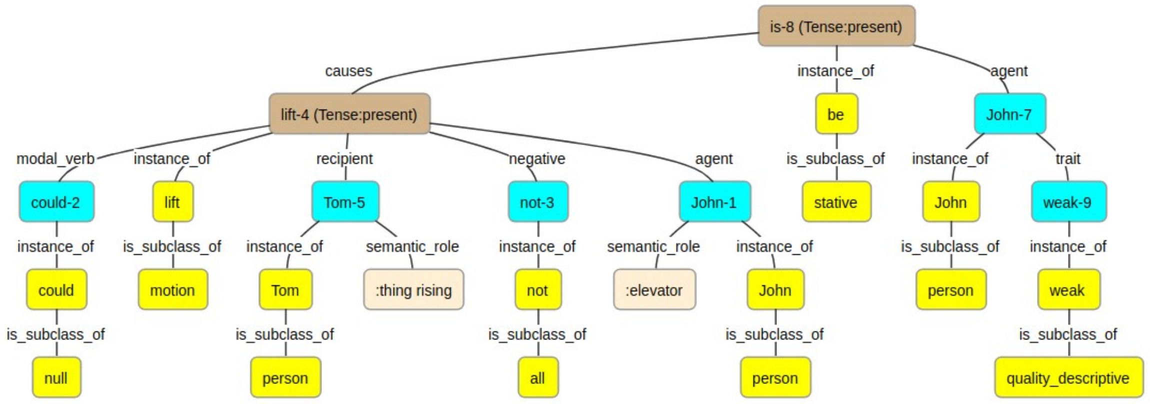


Figure 5.2: Semantic Parse of the Sentence, *"John could not lift Tom because John is weak"*.

3. **GETSEMGRAPH:** This procedure takes a semantic graph ($\mathcal{G}_S$) and a sequence of words (S') as input and outputs a subgraph ($\mathcal{G}_E$) of $\mathcal{G}_S$ such that $\mathcal{G}_E$ is an intersection of $\mathcal{G}_S$ and the semantic representation graph of S' (say $\mathcal{G}_{S'}$). To do that, $\mathcal{G}_E$ is initialized as an empty graph and each node N from $\mathcal{G}_S$ is added to $\mathcal{G}_E$ if N is same as a node in $\mathcal{G}_{S'}$. An edge E is added from $\mathcal{G}_S$ to $\mathcal{G}_E$ if the two nodes connected to E in $\mathcal{G}_S$ are already present in $\mathcal{G}_E$. For example, let $\mathcal{G}_S$ be the graph shown in the Figure 5.2 and a sequence of words be *"John could not lift Tom"*, then $\mathcal{G}_E$ is as shown in the Figure 5.3.

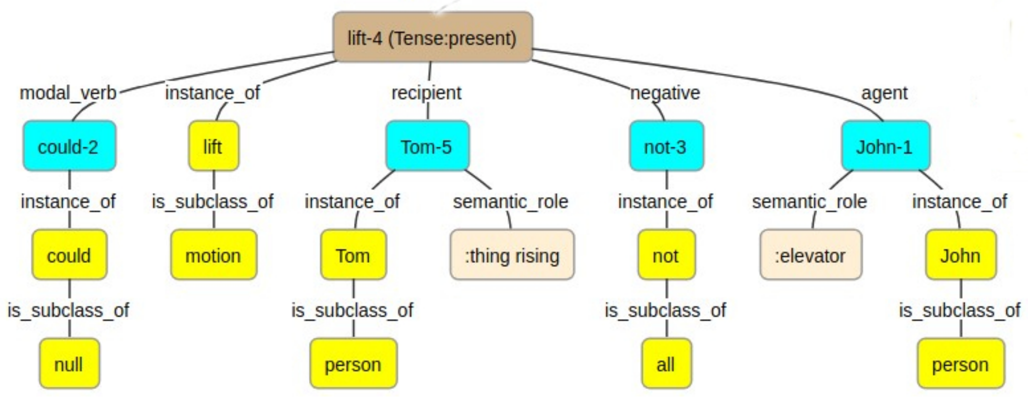


Figure 5.3: An Example of a Semantic Graph Extracted by the Procedure GETSEMGRAPH

Similarly, if $\mathcal{G}_S$ be the graph shown in the Figure 5.2 and a sequence of words be *"John is weak"*, then $\mathcal{G}_E$ is as shown in the Figure 5.4.

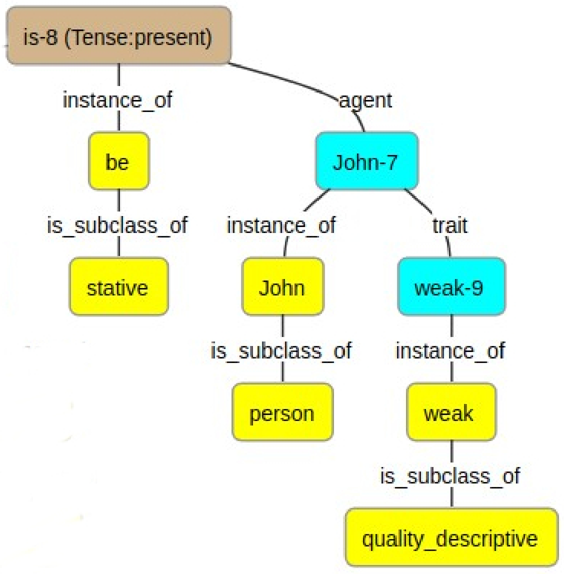


Figure 5.4: An Example of a Semantic Graph Extracted by the Procedure GETSEMGRAPH

4. **GETKNOWLEDGE:** This procedure takes two semantic graphs ($\mathcal{G}_{S1}$ and $\mathcal{G}_{S2}$, which are sub-graphs of a K-Parser output graph) and a knowledge type identification cue

phrase C as inputs and outputs a set of knowledge instances. Here, $\mathcal{G}_{S1}$ and $\mathcal{G}_{S2}$ respectively refer to the semantic representations of the first and the second parts of the input sentence as mentioned above in the definition of the SPLIT procedure. This procedure uses a set of if-then rules to extract a knowledge from $\mathcal{G}_{S1}$, $\mathcal{G}_{S2}$ and C. A sample rule is mentioned below.

IF $\mathcal{G}_{S1}$ contains a action A which is not executed, $\mathcal{G}_{S2}$ contains a property P, an entity X of type T is associated with A in $\mathcal{G}_{S1}$ with a semantic relation R, X is also associated with P in $\mathcal{G}_{S2}$ and C is *because*
THEN $T^\frown 1$ is P may prevent A R $T^\frown 1$

The symbol $\frown$ stands for concatenation.

For example, the knowledge extracted in this step by using the graph in Figure 5.3 as $\mathcal{G}_{S1}$, the graph in Figure 5.4 as $\mathcal{G}_{S2}$ and *because* as the causal clue C, is as shown below.

person1 is weak may prevent lift agent person1

5. **POSTPROCESS:** This procedure takes the output raw knowledge from the GET-KNOWLEDGE procedure along with the graphical representation of the input sentence and the sentence itself and outputs a processed knowledge in a more English like format. For example, if the input sentence is *John could not lift Tom because John is weak*, its semantic graph is as shown in the Figure 5.2 and the raw knowledge generated by the above procedure is *person1 is weak may prevent lift agent person1*, then the post processed knowledge is, *person1 is weak may prevent person1 lifts person2*.

5.3.3 Evaluation of the Extracted Knowledge

Since the automatic evaluation of generated language is an open research question (Liu *et al.*, 2016a), we also validate the extracted knowledge through human evaluation. The details of the validation and its result are mentioned in the section below.

Human Evaluation for Validity

We randomly selected a collection of 5000 knowledge instances from the extracted knowledge base of over 50k instances. The examples were selected in such a way that they are uniformly distributed with respect to the knowledge types mentioned in the Table 5.3. The collection of the selected knowledge is evaluated by two human workers on a binary scale of zero and one. One signifies that the knowledge is valid and the zero signifies invalid. An overlap of 100% was kept between the workers. Overall, 2002 out of 5000 knowledge were found meaningful. The filtered knowledge is available at `https://github.com/arpit7123/CommonsenseKnowledgeExtraction`

Pronoun Disambiguation Problems

The Winograd Schema challenge (Levesque *et al.*, 2011) is an NLU challenge which contains pronoun resolution problems. Due to the need for commonsense knowledge, it was proposed as an alternative to the Turing test in 2011. In 2016, first annual contest on solving the challenge was organized. In the first step of the contest, each participating system was tested on a set of Pronoun Disambiguation Problems (PDPs). Unfortunately, none of the systems surpassed a predefined accuracy threshold and could not proceed to the next step. The best performing system (Liu *et al.*, 2016b) was based on learning knowledge embedding in a neural network framework. The unavailability of a large number of problems to learn from is to an extent responsible for the low performance of such a system. And due to the same reason not many of the other learning based approaches (Wang *et al.*, 2018) can be applied on the dataset. So, in this work we explore the avenue of automatically generating pronoun disambiguation problems by using the extracted commonsense knowledge. Following are the steps in the automated problem generation process.

Step 1: Retrieve a knowledge that is of one of the types in the Table 5.3. Also retrieve an English sentence from which the knowledge was extracted by using the Algorithm 1.

Step 2: Each knowledge type extracted in this work can be abstractly written as *"A R B"*. Here *R* is either *"may cause"*, *"may prevent"* or *"has plausible precondition"*, and *A* and *B* are the two parts of knowledge which are on the left and right hand sides of *R* respectively. An important aspect of the knowledge, the relation between *A* and *B* is via a common entity. That entity is identified and its two occurrences in the sentence from which the knowledge is extracted and its first occurrence is replaced by *"Tom"* and the second occurrence by *"he"*. The sentence generated after the replacement now contains an ambiguous pronoun *"he"*.

Human Evaluation of PDPs

We generated a total of 1100 pronoun disambiguation problems. The problems were generated by using the knowledge type *"effects of an action"*. Each problem is accompanied with three answer choices. Two of the choices refer to two distinct entities in the input sentence such that one of them is the correct co-referent of the ambiguous pronoun. The third choice is *"Do not understand"*. It refers to the case when a human worker does not understand the problem because of any reason, including the case when the problem sentence does not make any sense to the worker. We used one human worker to answer all of the problems. A problem is deemed valid if it is answered correctly by the human worker. We found 777 (70.6%) out 1100 such problems. The rest were answered *"Do not understand"*. The valid problems are available at `https://github.com/arpit7123/CommonsenseKnowledgeExtraction`.

5.4 Automatic Extraction of Knowledge For the Winograd Schema Challenge Problems

There are various kinds of commonsense knowledge that are required to solve the problems in the Winograd Schema Challenge. The input text does not explicitly contain such knowledge. To overcome the lack of the required commonsense knowledge, in this work we have devised a technique to automatically extract commonsense knowledge from text repositories by using an input WSC problem.

An example of a WSC problem and the needed knowledge is as shown below.

Sentence: *The man could not lift his son because **he**$_{pronoun}$ was so weak.*

Question: *Who was weak?* **Answer Choices:** *a) man b) son*

Knowledge Needed:

*person1 is weak **may prevent** person1 lifts person2*

It should be noticed here that the knowledge contains a similar scenario to that of the associated WSC problem. In other words the knowledge contains similar verbs and adjectives. In this vein, we aim to extract such knowledge from plain text by first extracting the text which is similar to the text in a WSC problem and it may contain the needed knowledge. The following steps explain the extraction of such sentences which may contain commonsense knowledge.

1. The verbs, adjectives, adverbs and discourse connectives in a given WSC problem are identified. For example *"could"*, *"not"*, *"lift"*, *"weak"* and *'because'* from the above example are identified.

2. A set of search queries are created by using the words extracted in the previous step. The order of words is preserved in the queries. Also, an asterisk symbol ('*') is added between two words if they are not adjacent to each other in the original WSC

sentences. For example a query *"* could not lift * because * weak * "* is created for the problem mentioned above.

3. the queries generated in the previous step are used to search and extract sentences from a search engine. In this work, we used the Google search engine. An example sentence extracted by using the query mentioned above is, *"She could not lift him because she is weak."*. Not all the results from the search engine will be useful. Some sentences might not contain enough information about the scenario (e.g. "she lifts"). Some sentences might be similar to the original scenario but contain the co-reference ambiguity that is present in the original scenario. Thus in this step, the sentences which were similar to the original sentences and the ones which contained obvious ambiguities are filtered out. Additionally, more queries are generated by using the variants of the words used in the previous query. The variants include synonyms, similar words (*lifted*, *raise*), and base forms of verbs in the verb phrases of the original query.

After several iterations of step 2 and step 3 we obtain sentences which may contain the needed knowledge. We call such sentences as knowledge sentences. The knowledge sentences that are extracted often contain pronouns. For example the extracted sentence "She could not lift him because she was weak." also contains two pronoun occurrences of (*"she"*,). However, we make sure that the pronouns in the extracted knowledge sentences can be easily resolved using the following procedure:

1. Two pronouns refer to each other if they have the same string description. For e.g. all the occurrences of the pronoun "she" always refer to the same entity.

2. Two pronouns (p_1, p_2) refer to each other if they belong to a special list containing the following: $\{(he, him), (she, her), (i, me), (they, them), (he, his), (his, him)\}$. We

also ignore knowledge sentences where any of these special pair of pronouns appears as an argument to a common verb (e.g. “she could not lift her because ...”).

The knowledge sentences extracted by using the above steps are passed to the knowledge extraction system described in the Section 5.3. An example knowledge extracted from the above knowledge sentence by using the algorithm is *“person1 is weak may prevent person1 lifts person2”*.

5.5 Related Works

WordNet (Miller, 1995) is one of the most popular knowledge base used by Natural Language Processing community. However, it is a lexical database consisting words, their senses, synonyms, hyponyms and hypernyms, it does not contain the commonsense knowledge that we are extracting in this work.

ConceptNet (Liu and Singh, 2004) is another big source of commonsense knowledge. It is a semantic network containing more than 1.6 million edges connecting more than 300000 nodes where nodes represent concepts(words, small phrases) and edges represent the relation between nodes. However, the knowledge in ConceptNet is very high level and it does not have the kind that we are extracting in this work. Furthermore, the relations in ConceptNet are very coarse grained and also the participants of both the concepts are not specifically related.

Narrative Chains (Chambers and Jurafsky, 2008), is another automatically extracted commonsense knowledge base. It contains a list partially ordered set of events that are centered around a common protagonist. The ordering of the events is temporal. Because of this, other relationships such as causality are not captured properly in narrative chains. An example of this is given in the *“bullying”* example mentioned in the sections above. In it there exists a causal relation between the events i.e. *“bullying”* causes *“rescue”*. In this example it seems obvious to say that the recipient of bullying is also the recipient of rescue but if we do not consider the causal relationship between events then the knowledge becomes less obvious.

Other popular knowledge bases such as Cyc (Lenat, 1995) tend to compile complex assertions such as *every human has exactly one mother*. WebChild (Tandon *et al.*, 2014) is a knowledge base created by extracting information from web. It contains properties of objects. For example *"Orange"* is *"round"* and its color is *"Orange"*. It lacks the constraint defined property knowledge such as the one mentioned in this work.

In addition to the above, there has been a lot of research towards building Knowledge bases with deeper knowledge instead of basic facts. One of the most interesting works are carried out bye Dr. Oren Entzioni's group on Open Information Extraction. In (Lin *et al.*, 2009), they focus on more interesting assertions such as *"the FDA banned Ephedra"* ignoring less useful statements such as *"the FDA banned products"*. In (Lin *et al.*, 2010), they have focused on commonsense knowledge inference based on the properties of relations such as functionality and transitivity. They have also proposed how to detect such properties, for example the functionality [4] of particular relations such as *bornIn*. Though different from our primary goals, their work on event extraction from twitter (Ritter *et al.*, 2012) and entity linking in (Lin *et al.*, 2012) has inspiring thoughts in building higher-order knowledge bases in comparison to factoid ones.

Accumulation of commonsense knowledge is a topic of interest since a long time. The earlier attempts of accumulating commonsense knowledge focused mainly on the factual or taxonomic knowledge. Such attempts included manual knowledge collection (Lenat, 1995; Miller, 1995) and automatic knowledge extraction from text (Bollacker *et al.*, 2008; Carlson *et al.*, 2010; Tandon *et al.*, 2017). Such knowledge bases, although useful where taxonomic knowledge is needed, do not contain the commonsense knowledge about actions

[4]Functionality of a relation such as *bornIn* indicates that if (A,bornIn,B) and (A,bornIn,C) then B is same as C if they are either both locations or both time-range.

which is the focus of this work.

Recently, various knowledge bases have been collected that focus on more involved commonsense knowledge about actions and events. These knowledge bases are created by crowd sourcing. In many cases a set of seed actions/events are automatically identified from a corpus and then used to develop a crowd sourcing task to generate knowledge about those actions/events. For example in Atomic (Sap *et al.*, 2018) a set of events are automatically extracted from a corpus and then specific questions about those events are asked in a crowd-sourcing task to get the cause and effect knowledge about those events. A similar approach is used in VerbPhysics (Forbes and Choi, 2017) and Event2Mind (Rashkin *et al.*, 2018). Though the idea of collecting knowledge from people is meaningful, the process is expensive. This is because the generation of information such as effects and causes of actions is a complicated task and hence require more focus from crowd workers. Whereas validating a given information is relatively easier. Here we use this intuition to develop an algorithm which automatically extracts knowledge from text and later the extracted knowledge is validated by crowd workers. This makes the overall extraction process relatively less expensive.

5.6 Conclusion

In this chapter we presented our two attempts towards extraction different kinds of commonsense knowledge about events and actions. The first attempt uses a semantic parsing and logical reasoning technique to extract a new kind of commonsense knowledge from text repository. The knowledge extracted has already been proved useful in solving an NLU task. Also, this knowledge is not present in currently available knowledge bases. The quantity of knowledge extracted is noticeably less than some other similar works such as KNEXT (Van Durme and Schubert, 2008), where there are about 1.78 unique knowledge instances extracted from each sentence in the corpus. This is because the kind of knowledge that we are extracting in this work is of very different nature and significance. Also, the knowledge we have extracted is based on the experience earned by people over the years and it is very difficult for a small group of people to list such knowledge by hand. Furthermore, the quality of the knowledge is determined in a fashion similar to KNEXT and found to be better ($\approx 67\%$ as compared to 54% in KNEXT) .

In the second knowledge extraction attempt we presented our algorithm to automatically extract a knowledge base about actions and their plausible causes, effects, preconditions and the conditions which prevent the actions from executing. We validated the extracted knowledge by using human workers. We made the valid knowledge base of 2002 knowledge pieces. We also contributed to increase the size of problems in the Winograd Schema Challenge by using the extracted knowledge to generate a set of 777 pronoun disambiguation problems.

Chapter 6

PARSING TEXT TO SEMANTIC GRAPHS: THE KNOWLEDGE PARSER

In this chapter, improvements in a semantic parsing system are presented. The parser translates English sentences into a graphical semantic representation. It is used to translate the sentences in WSC problems and the knowledge that is needed to solve the problems into the graphical representations defined in the Chapter 4. This chapter is organized as follows.

- The Section 6.1 presents the details of a semantic parsing and knowledge augmentation system called Knowledge Parser or K-Parser.
- A basic version of the K-Parser system was implemented as part of my Masters thesis (Sharma, 2014). In this chapter we present the improvements made to the K-Parser system as part of this thesis. The Section 6.2 presents the details of those improvements.
- The Section 6.3 presents a wrapper around the K-Parser system to translate the sentences in a WSC problem into a graphical representation as mentioned in the Chapter 4.
- The Section 6.4 presents a wrapper around the K-Parser system to translate knowledge into a graphical representation as mentioned in the Chapter 4.

6.1 Knowledge Parser (K-Parser): An Events Centered Semantic Representation System

Events and the relationship between them are important aspects of natural languages such as English. They play an important role for people to understand the meaning of a language. For example, it is because of the *arrest* event in the sentence *“Jim arrested Tom”* that people understand that *"Jim"* took *"Tom"* into his custody. People can also infer that *"Jim"* may be a police officer. So, the identification and representation of event mentions is important not just for semantically representing a sentence but it also helps in inferring additional information about it. Many natural language understanding applications such as question answering on narratives (Hajishirzi *et al.*, 2011) rely on the identification of events in the text. The narratives are the basic parts of many other applications such as reading comprehension of stories and sports commentary (Hajishirzi *et al.*, 2012b), (Hajishirzi *et al.*, 2012a). These applications are primarily concerned about the questions whose answers are not directly mentioned in a given text. To answer such questions, one needs to identify and formally represent the phrases in a given text which depict events. For example let us consider the text *"John shot Jim and a few hours later he was arrested."*, and the question *"Who was arrested?"*. To answer the question, one must identify and formally represent the events expressed by the phrases *"John shot Jim"* and *"he was arrested"*. There are various other works which further stress the importance of identifying events in the given text. Some such works include (Ouyang and McKeown, 2014; Glavaš *et al.*, 2014; Glavaš and Šnajder, 2014; Swanson *et al.*, 2014; Chambers and Jurafsky, 2009) and even a separate workshop [1] on events' definition, detection and representation.

Broadly speaking, an event is a phrase which contains an action and the entities that participate in the action in various capacities. Attributes or properties of the entities are also important because most of the times altering the attributes changes the meaning of the entire event. Considering this, in this work we defined a semantic representation which focuses on formalizing events. We also defined and implemented an algorithm to automatically parse a given English sentence and translate it into the events centered semantic representation. The algorithm is a combination of both rule based and statistical approaches towards semantic parsing. It uses the syntactic dependency parse of a given sentence along

[1] *https://sites.google.com/site/wsevents2015/home*

with various other NLP tools and techniques such as preposition and word sense disambiguation, discourse parsing and WordNet.

In this work we first describe the details of our events centered graphical semantic representation. Then we explain the semantic parsing algorithm. The algorithm is implemented as a parsing tool called K-Parser which is available for download from `https://github.com/arpit7123/K-Parser-JAR`. In a later section we present the evaluation of the K-Parser. The evaluation was performed via three separate experiments. First an in-vitro quality evaluation of the parser was performed by examining the results of K-Parser on STEP 2008 (Bos, 2008a) "shared task" texts. The results were compared with various systems that participated in the task. Secondly, the parser was evaluated with respect to the sentences in the Winograd Schema Challenge corpus. The parser outputs were compared with the gold standard representations and precision and recall were calculated with respect to the main components of the representation. Thirdly, an in-vivo evaluation of the parser outputs was performed by solving a subset of the Winograd Schema Challenge corpus with the help of a logical reasoning framework and comparing the results with the existing work on the subset.

The sections below explain the semantic representation, the parsing algorithm and the detailed evaluation of the parser.

6.1.1 Semantic Representation

In this work we defined a graphical semantic representation for English sentences. A graphical representation is useful because (a) it is easier for people to comprehend as compared to other representation (e.g., a logic rule). (b) There exists a rich literature on graph algorithms which can be leveraged for processing a graphical representation. (c) It can easily be used in both the logical (Sharma *et al.*, 2015d) and the probabilistic (Rao *et al.*, 2017) reasoning frameworks. Inspired by the usefulness of a graphical representation based formalisms such as AMR (Banarescu *et al.*, 2013), Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013), Semantic Dependency Parsing (Oepen *et al.*, 2015) and Alexa Meaning Representation language (Kollar *et al.*, 2018), we adopted it in this this work. In this section we mention the basic components of the representation and how the components combine into the final representation. An example of a representation of an English sentence is as shown in the Figure 6.1. The representation shown in the Figure 6.1 is displayed as a tree instead of a graph for making it easier to read.

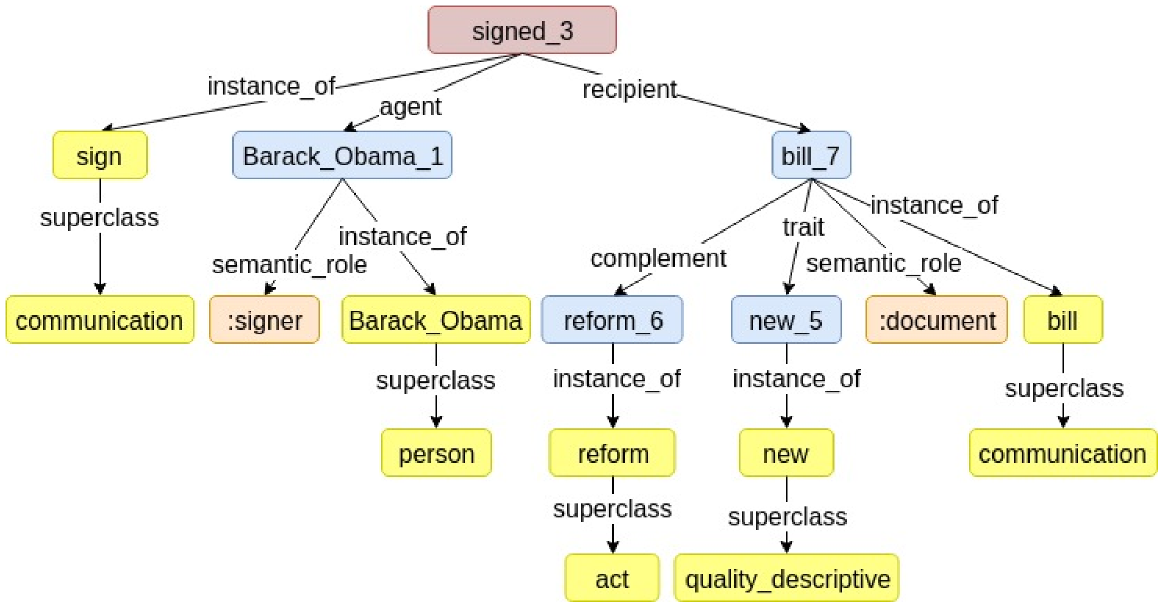


Figure 6.1: A Semantic Representation of the Sentence *"Barack Obama signed the new reform bill."*

Basic Components

There are following basic components of the representation.

1. **Action Nodes:** The verbs in an input sentence are called actions. Actions includes both auxiliary and the non-auxiliary verbs. The actions are represented as nodes called action nodes in our semantic representation. For example the node *"signed_3"*

is an action node in the Figure 6.1. It is generated from the word *"signed"* in the sentence. The *"_3"* in the node represents the index of the word *"signed"* in the sentence.

2. **Entity Nodes:** The content words in an input sentence, except the verbs, are called entities. The entities are represented as entity nodes in our semantic representation. The named entities in the sentence (an n-gram where $n \geq 1$) are also called entities and hence translated into entity nodes in the representation. For example the nodes *"Barack_Obama_1"*, *"bill_7"* and *"new_5"* are the entity nodes in the Figure 6.1.

3. **Conceptual Class Nodes:** These depict the 'type' of an action or an entity. In other words, a conceptual class performs the grounding of an entity or an action. The conceptual classes of entities and actions in a sentence are shown with the help of conceptual class nodes in the semantic representation. For example, the conceptual class of the entity *"Barack_Obama"* is represented by the node *"person"* in the Figure 6.1. There are two levels of class information. The first level refers to the base form of the word which is transformed into an event or entity node and the second level refers to an abstract class such as *"person"*.

4. **Semantic Relations:** The action, entity and conceptual class nodes which are semantically dependent on each other are connected to each other through a set of semantic relations. Such relations define a meaningful connection between the nodes. There are three main types of semantic relations namely *action-entity*, *entity-entity* and *action/entity-class*. A few examples of the semantic relations are as shown in the Figure 6.1. The relations used in this work are inspired from the Knowledge Machine (Clark *et al.*, 2004) library.

The Representation of English Sentences

A representation of English sentences (as defined in this work) is an edge labeled and directed acyclic graph. The graph is made up of action, entity and conceptual class nodes and the semantic relations between the nodes that are semantically dependent on each other. The representation is centered around events in the sentences. An event is made up of an action and a set of entities which participate in the action. An action is a concrete or an auxiliary verb such that it is not a modifier of a concrete verb. For example in the event *"Jason kicks Michael"* the entities *"Jason"* and *"Michael"* participate in the action *"kicks"*. The representation of sentences is made up of the representations of the events present in them and connected to each other via *event-event* semantic relations. A representation of a unique event is called an event mention in this work.

In other words, an event mention is a graph that is rooted at an action node and a set of entity nodes connected to the action node. For example, the graph in Fig. 6.1 represents an event mention, rooted at the action node *"signed_3"*.

Representing Various kinds of Events

There are three broad kinds of event mentions defined in the existing literature. All of these can be easily represented using our schema. Following are the details of the types.

- **Aspectual Category:** There are four categories of aspectual event mentions, namely achievements, accomplishments, process or activity and states. Pustejovsky (Pustejovsky, 1991) demonstrates how same verbs can be used in different kinds of event mentions (see example sentences 1(a) and 2(a) in Table 6.1). The difference between these kinds is determined by the arguments of the action in the event mention. For example in 1(a), the action is an unbounded *'process'* whereas in 2(a) it is an *accomplishment* because of the bounding done by the phrase *"to the store"*.

Table 6.1: Event Categories and Example Sentences

Category	**Sub-category**	**Example Sentences**
Aspectual	Process or Activity	*1(a) Mary walked.* *1(b) John ran.*
	Accomplishment	*2(a) Mary walked to the store.*
	Achievement	*3(a) Tim ran two miles.* *3(b) John arrived at his destination.*
	State	*4(a) John loves Mia.* *4(b) I knew about the incident.* *4(c) He fell asleep during the meeting.*
Complexity	Complex Events	*5(a) The knife was used to kill the dog.* *5(b) George was bullying Tim so we rescued him.*
	Simple Events	*6(a) John loves Mia, and Mia hates John.* *6(b) Tom killed John before Tom and Jane ran away.*
Temporal	-	*7(a) Tom killed John before Tom and Jane ran away.* *7(b) She sat opposite him and looked into his eyes.*

- **Complexity Category:** Another criteria for categorizing event mentions is based on the complexity. An event mention M is defined as complex if an argument of the action in M is the main action of another event mention M1. For example, 5(a) in Table 6.1 has a complex event consisting of two actions *"used"* and *"killing"*. The *"killing"* action is an argument to the *"used"* action. On the other hand a simple event mention has only one action. See Table 6.1 for examples.

- **Temporal Category:** Temporal ordering is other criteria for categorization of events mentions. It is used to specify the order of occurrence of unique events in a chain of events. See examples in Table 6.1.

6.1.2 Semantic Parsing

In this work we defined a semantic parsing algorithm and implemented it as a semantic parser that can be downloaded for free. The algorithm takes English sentences as an input and produces a graphical semantic representation as defined in the sections above. Following are the details of the algorithm and the parser.

The Semantic Parsing Algorithm

The Figure 6.2 provides an overview of the semantic parsing algorithm. The algorithm takes an English sentence as input and produces a graphical semantic representation as defined in the sections above. The algorithm uses the below mentioned six modules to transform the input into the output.

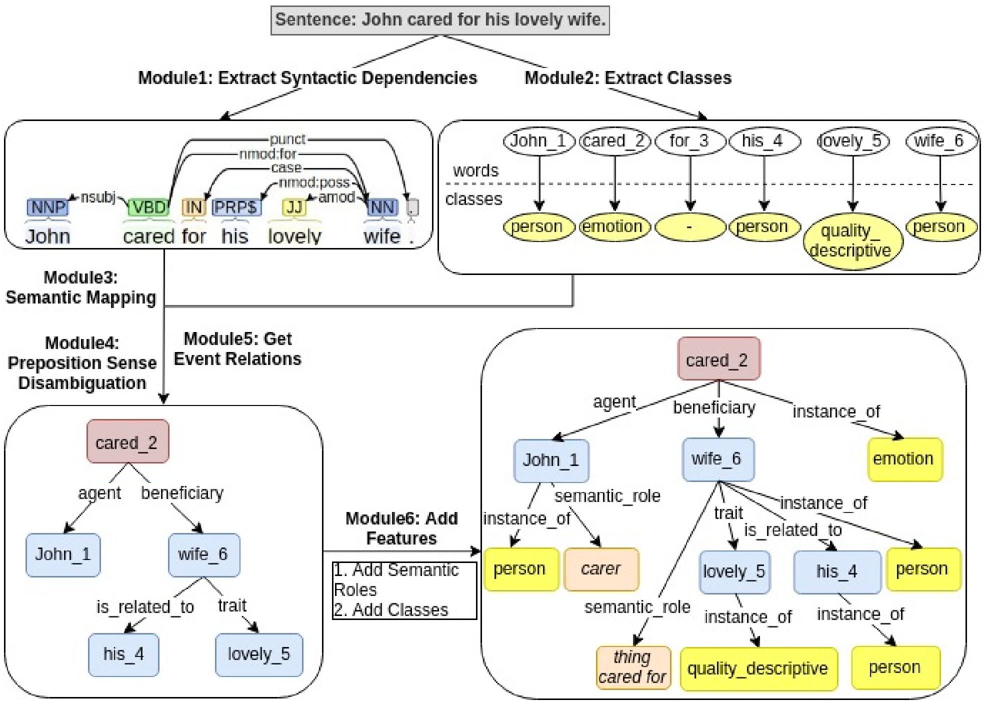

Sentence: John cared for his lovely wife.
Module1: Extract Syntactic Dependencies
Module2: Extract Classes
NNP nsubj VBD IN PRP$ JJ amod NN
punct
nmod:for
case
nmod:poss
John cared for his lovely wife .
John_1 cared_2 for_3 his_4 lovely_5 wife_6
words
classes
person emotion - person quality_descriptive person
Module3: Semantic Mapping
Module4: Preposition Sense Disambiguation
Module5: Get Event Relations
cared_2
agent beneficiary
John_1 wife_6
is_related_to trait
his_4 lovely_5
Module6: Add Features
1. Add Semantic Roles
2. Add Classes
cared_2
agent beneficiary instance_of
John_1 wife_6 emotion
semantic_role instance_of
person carer
trait is_related_to instance_of
lovely_5 his_4 person
semantic_role instance_of instance_of
thing cared for quality_descriptive person

Figure 6.2: K-Parser System Algorithm Explained with the Help of the Example Sentence *"John cared for his lovely wife."*

Module 1 - EXTRACT SYNTACTIC DEPENDENCIES: This module transforms the given sentences into their syntactic dependency graph. Such a graph establishes the dependencies usually between the content words in the sentences by representing the function words as dependency relations from a predefined set of relations. An example of a dependency graph is shown in Figure 6.2. In this work we used the Stanford Dependency Parser (De Marneffe and Manning, 2008) to extract the syntactic dependencies between words in sentences. The parser uses a probabilistic context free grammar to produce a syntactic tree of the input sentences and then extracts the syntactic dependencies for the input sentences. Stanford Dependency Parser is one of the most popular syntactic dependency parser for English.

Module 2 - EXTRACT CLASSES: This module extracts two levels of conceptual classes with respect to the words in the input sentences. The first level of classes are extracted by finding the base form of the respective words. For example the base form of the verb *signed* is *sign*. It is a normalization step that illustrates that if two or more words with the same base form are used in the text then all those words belong to class defined by the based form. The second level of classes represent the type of a word. For example *John* is of type *person*. Both part-of-speech and correct sense of a word are important in extracting the type information. So, we used the Stanford POS tagger and the Word Sense Disambiguation algorithm mentioned in (Basile *et al.*, 2007) on WordNet (Miller, 1995) knowledge base to get the correct sense of the words and then extracted their corresponding lexical senses from the WordNet knowledge base.

Module 3 - SEMANTIC MAPPING: This module transforms a syntactic dependency graph into a preliminary version of a semantic graph by translating the syntactic relations into semantic relations. For example if a syntactic relation *nsubj* exists between *John* and *cared* then it is translated into a semantically rich *agent* relation. The module consists of a set of if-then rules. An example rule is shown below.

IF

$nsubj(N1,N2)$ **AND** $pos(N1) == verb$ **AND** $pos(N2) == noun$ **AND** $type(N2) == person$

THEN

$has(N1,agent,N2)$

In the above rule, $N1$ and $N2$ are two nodes in a syntactic dependency graph. $nsubj(N1,N2)$ represents a *nominal subject* syntactic relation as produced by the Stanford parser. $pos(N1)$ and $pos(N2)$ represent the part-of-speech of the nodes $N1$ and $N2$ respectively. $type(N2)$ represents the type of the node $N2$.

There are a set of 92 such if-then rules in the current implementation of the parser. These rules cover all the syntactic dependency relations in the Stanford Dependency Parser version 3.4.1 except the ones which correspond to prepositions i.e. relations of the form *prep_**. The rules were manually created by analyzing the example sentences provided in the Standford Dependency Manual with respect to each syntactic dependency.

Module 4 - PREPOSITION SENSE DISAMBIGUATION: This module performs preposition sense disambiguation to find the correct senses of prepositions in the given sentences. The senses are then used to allocate semantic relations between concepts in the sentences. The semantic mapping module presented above (i.e., Module 3) does not cover the syntactic dependencies generated due to prepositions in the text. This is because a unique preposition may be interpreted in more than one way. In other words it may be responsible for two or more distinct meanings. For example in the sentences *John came by his car* and *John came by his office*, a unique preposition (i.e., *by*) have different meanings. In K-Parser implementation we used a multilayer perceptron classifier to classify the preposition senses into a set of senses (semantic relations) from KM library. We used *The Preposition Project* (Litkowski, 2013) corpus as the training corpus and manually annotated the training sentences with appropriate semantic labels from KM library.

Module 5 - GET EVENT RELATIONS: This module allocates semantic relations between two semantically dependent events in the given sentences. Various events in sequential sentences are many times dependent on each other. Such dependencies are of various forms such as causal, sequential and temporal (see Table 6.1 for examples). This module extracts such dependencies and allocates suitable semantic relations to those dependencies. We used a discourse parsing based approach to allocate such relations. There are three main steps in the approach. In the first step, the input sentence is parsed based on the explicit and implicit discourse connectives by using a Penn Discourse Treebank style discourse parser (Lin *et al.*, 2014). The discourse parser extracts the discourse connectives from the sentence and their two argument phrases which are related to the connectives. The second step translates the discourse connectives into semantic relations between the arguments of the connectives. There are only small number of predefined connectives that are annotated with their corresponding semantic relation. For example the connective *"because"* is mapped to *"caused_by"* relation. The third step extracts the action words from the arguments of the connective. Afterwards the semantic relation which was extracted in the second step connects an action node generated by an action in one argument of connective to an action node generated by an action in the second argument of the connective.

Module 6 - ADD FEATURE: This module adds two new features to the semantic representation. First, the Propbank frame sets (Palmer *et al.*, 2005) are used to add the semantic roles to the entities in the sentence. Secondly, the two levels of classes are added with respect to each of the event and entity node in the representation. The classes are extracted in the module 2 explained above.

6.1.3 *Evaluation*

In this section we present the evaluation results of our semantic parsing algorithm. We performed three separate experiments. In the first experiment, we followed the existing trend and manually analyzed the outputs of our parser with respect to the texts in STEP 2008 "shared task" (Bos, 2008a). In the second experiment, we compared the outputs of K-Parser with the gold representations with respect to the sentences in the Winograd Schema Challenge (Levesque *et al.*, 2011). In the third experiment we used the K-Parser in a logical reasoning based framework to solve a subset of the Winograd Schema Challenge corpus. The details of each experiment are mentioend in the sections below.

Experiment 1 - STEP 2008 "shared task"

We evaluated the performance of our semantic parser with respect to the text pieces in STEP 2008 "shared task" (Bos, 2008a). There are total seven text pieces in the corpus. Each piece of text is made up of one or more English sentences. We followed the evaluation criteria used in various other works (see Table 6.2) and manually analyzed the representation produced by K-Parser to verify if it satisfies the properties of a good semantic representation (As mentioned in the existing works shown in Table 6.2). Table 6.2 shows the comparison of K-Parser with various existing works in terms of representing the important aspects of a semantic representation.

We also evaluated the performance of K-Parser by comparing the K-Parser outputs for the text pieces in the shared task with their gold representations as per the representation defined in this work. The evaluation focused on determining the percentage of the nodes and the edges which were correctly or incorrectly identified in the K-Parser output. The evaluation results are shown in the Tables 6.3 and 6.4.

Table 6.2: Evaluation Results for Comparison with Other Systems

Feature	**Example Sentenc**	**Systems**							
		K-Parser	**TRIPS**	**BLUE**	**Boxer**	**GETARUNS**	**LXGram**	**TextCap**	**OntoSem**
Coordination	*The atmosphere was warm **and** friendly*	✔	✔	✔	✔	✘	✔	✔	✔
Entities' Attributes	*the gun crew was killed, they were crouching **unnaturally***	✔	✔	✔	✘	✔	✘	✘	✘
Uncertainty	*Researchers have been looking for other cancers that **may be** caused by viruses*	✔	✔	✘	✘	✘	✔	✔	✔
Named Entities	*In Mortagua, **Marta Gomes** coordinates the project that seven people develop in this school*	✔	✘	✔	✔	✘	✔	✔	✔
Complex Time	*In the **mid-'80s**, wind turbines had a typical maximum power rating of 150 kW*	✔	✔	✘	✔	✘	✔	✘	✘
Conceptual Classes	*The waiter took the order*	✔	✘	✘	✘	✘	✘	✔	✔
Easily Readable	*He began to read his book*	✔	✘	✔	✘	✘	✘	✔	✘
Parsing Questions	*What is the duration of the fall?*	✔	✔	✔	✔	✘	✘	✘	✘

Experiment 2 - Winograd Schema Challenge

Gold Standard Evaluation

There are various versions of WSC corpus available on its official website. In this experiment we selected the version [2] which consists of 285 individual problems. We manually defined the gold representations for the sentences in the WSC corpus. Then we compared the K-Parser outputs for the WSC sentences with the gold representations. We identified five important categories to assess the accuracy of K-Parser. The categories are Number of Events, Number of Entities, Number of Classes, Number of Event-Event Relations and Number of Event-Entity Relations. Each of the categories are compared with the gold standard based on measures (a) t_1 = identified and relevant and the label is correct, (b) t_2 =

[2]https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.xml

	Correct Labels (%)		**Incorrect Labels (%)**	
Text	**Edges**	**Nodes**	**Edges**	**Nodes**
1	52.1	68.5	47.9	31.5
2	72.2	76.2	27.8	23.8
3	92.3	83.3	07.7	15.7
4	67.3	79.7	32.7	20.3
5	68.1	80.8	31.9	19.2
6	66.7	80.0	33.3	20.0
7	67.2	78.2	32.8	21.8
AVG.	69.4	78.1	30.6	21.9

Table 6.3: Evaluation Results Based on Gold Standard for *"Shared Task"* Texts.

Criteria	**Precision**	**Recall**
Events	0.94	0.92
Entities	0.97	0.96
Classes	0.86	0.79
Event-Event Relations	0.91	0.79
Event-Entity Relations	0.94	0.89

Table 6.4: Evaluation Results Based on the Events, Classes and Semantic Relations in the Output of K-Parser

identified and relevant and the label is wrong, (c) t_3 = identified, but not relevant and (d) t_4 = not identified, but relevant. We then defined precision and recall of our system based on the above terms.

$$Precision = t_1/(t_1 + t_2 + t_3) \tag{6.1}$$

$$Recall = t_1/(t_1 + t_2 + t_4) \tag{6.2}$$

Table 6.4 shows the evaluation results for K-Parser based on the above measures.

6.1.4 Related Works

There are two types of popular semantic or meaning representations of text. The first is a logic based representation in which a text is represented in a logical formalism. Such a formalism includes very basic first order representation such as the classic first order logic (Liang, 2016) or more complex and expressive ones such as first order logic with lambda calculus (Carpenter, 1997) and lambda DCS (Liang, 2013). These representations have been used in various applications including to query databases (Zettlemoyer and Collins, 2012), conversational agents (Artzi and Zettlemoyer, 2011), providing instructions to robots (Artzi and Zettlemoyer, 2013) and various other applications (Berant *et al.*, 2013; Pasupat and Liang, 2015; Krishnamurthy *et al.*, 2017). The second is a graphical representation in which nodes represent actions/events/entities and the edges represent the semantic relations between the nodes. A few examples of such a graph based formalism are the Abstract Meaning Representation (AMR) (Banarescu *et al.*, 2013), Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013), Semantic Dependency Parsing (Oepen *et al.*, 2015) and the Alexa Meaning Representation language (Kollar *et al.*, 2018). There are various advantages of using a graphical representation over others which inspired us to use such a representation in this work. Such advantages are, (1) they are easily readable by people, and (2) there exists a rich literature on graph processing algorithms which can be used for such a representation.

As far as semantic parsing goes, based on the parsing algorithm, such systems can be divided into two types. The first includes the systems which use a rule based algorithm to transform a given NL text into a semantic representation. For example domain specific systems which are based on pattern matching (Johnson, 1984) or the systems which utilize syntactic structure of text (Woods, 1973). Other rule based systems use predefined semantic grammars (Templeton and Burger, 1983; Hendrix *et al.*, 1978). The second type of semantic parsing algorithms are based on statistical techniques and they are more popular these days. Such algorithms include supervised and unsupervised learning algorithms such as the ones used in (Lyu and Titov, 2018; Guo and Lu, 2018) and (Poon, 2013) respectively. In this work, we utilized the benefits of both, the rule based and statistical learning, approaches by developing an algorithm which is a merge of both.

Though the available parsing systems produce representations which have many important features, they lack the other important ones. For example SEMAFOR parser (Das *et al.*, 2010) assigns semantic roles to entities and verbs in a text but lacks in defining events and relations between them. It also does not correctly process the implications, quantification and conceptual class information about a text. Similarly, the Boxer system (Bos, 2008b) translates English sentences into first order logic. It fails to represent the event-event and event-entity relations in the text. Other parsing systems such as (Carbonell and Smith, 2014), TRIPS (Allen *et al.*, 2007), TEXTCAP semantic interpreter (Callaway, 2008) also capture some important features but lack in representing the others. In our parser, we covered the main aspects of the existing systems and also the aspects which were not considered in the existing systems. A comparison between our work and some of the important existing systems is as shown in the Table 6.2.

6.2 Improvements in the K-Parser System

In this work, improvements were made on a basic version of the K-Parser system. The implementation details of the current version of K-Parser are provided in the previous section. A basic version of K-Parser was implemented as part of my masters thesis (Sharma, 2014). Here we started with the basic version of K-Parser and performed various improvements on it to make it more robust and efficient. In this section we provide the details about those improvements.

Following are the improvements made in the K-Parser implementation.

1. **Use of Preposition Sense Disambiguation:** In the basic version of the parser the prepositions in the input sentences were translated into the semantic relationships by using a set of rules similar to the ones described in the module 3 of Section 6.1.2 above. For example a rule for the preposition *'to'* translates into a semantic relation *'destination'*. Such a relation is correct with respect to the sentence *'I went from Italy to Rome'* where *'Rome'* is the destination of *'went'* action but it is not correct with respect to the sentence *'John went to eat.'* because here *'eat'* is an objective of *'went'* action. Such issues are handled in the current version of the parser because the preposition relations are retrieved after performing preposition sense disambiguation (as mentioned in the module 4 of Section 6.1.2). The current approach relies on large amount of data to distinguish different senses of a preposition which was not possible by simple semantic mapping rules.

2. **Use of Discourse Parsing:** In the basic version of the parser, the semantic relationships between actions was determined based on the existence of explicit connectives, such as *'because'* and *'so'*, between them in the sentences. Each action on one side of the connective was then connected with each action on the other side of the connective by a semantic relation corresponding to the connective, event if they are not

semantically related. To overcome this, in the current version of the parser we use discourse parsing (see Module 5 of Section 6.1.2) where the arguments of a connective (which may not span over all the actions on either side) are also predicted along with its type.

3. **Ability to Parse Questions:** The basic version of the parser was designed to only parse sentences. Whereas the current version has the ability to parse questions as well. There are following two types of questions which are currently parsed by the K-Parser system.

   (a) **Yes/No Questions:** The answer to these questions is either *'yes'* or *'no'*. For example, *'Did she like it?'*. An example K-Parser parse for this types of question is shown in the Figure 6.3.

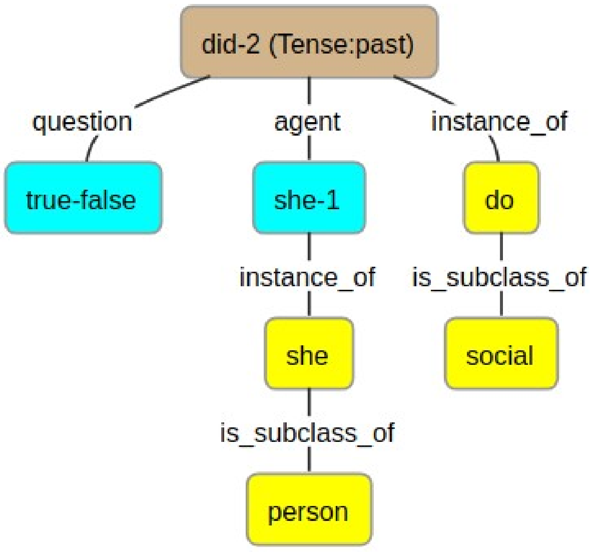


Figure 6.3: K-Parser Output for the Question *'Did she like it?'*

   (b) **Wh-Questions:** These questions contain a *Wh*-word which usually represents an unknown. For example, *'Who killed the dog?'*. An example K-Parser parse for this types of question is shown in the Figure 6.4.

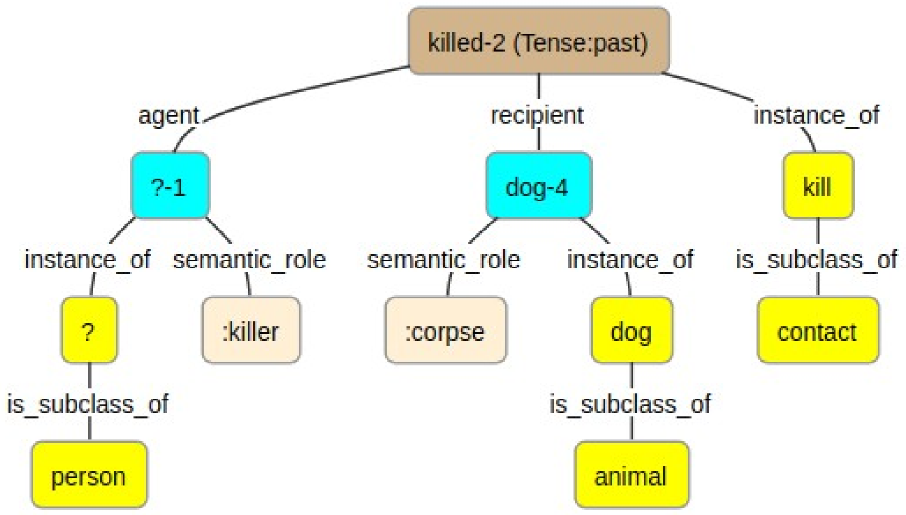


Figure 6.4: K-Parser Output for the Question *'Who killed the dog?'*

4. **Classes for Adjectives:** The basic version of the K-Parser system categorized every adjective as type *'all'* because in the WordNet knowledge base each adjective belongs to the lexical file named *'all'*. Whereas in the current version a type for each adjective is explicitly defined. These types are defined based on their types, such as size, color and description. A complete list of adjectives and their types is available in the K-Parser implementation at `https://github.com/arpit7123/Kparser`.

6.3 K-Parser Wrapper to Automatically Represent WSC Sentences

Each WSC problem is made up of a sequence of one or more sentences. The WiSCR algorithm as defined in the Chapter 4 of this thesis takes a graphical formal representation of the sentences as input. We use K-Parser to automatically generate a graphical representation for the sentences in the WSC sentences. We used the definition of the formal representation (Definition 4) as a reference to write a wrapper around the K-Parser system which transforms the output of K-Parser according to the definition.

K-Parser produces a graph for an input English text. Following modifications were made to the K-Parser output of a sequence of WSC sentences to translate it according to the Definition 4.

Similar to the Definition 4, there are two types of nodes in the K-Parser output. One is the words in the input text appended with their occurrence index and the other contains the conceptual classes of the first types of nodes. There are two levels of conceptual classes in the K-Parser output. The first level refers to the lemmatized form of words and the second level refers to the conceptual class from the WordNet KB. But the Definition 4 has only one level of classes. So, in this wrapper, the two levels of classes are transformed into one level by performing the following.

1. The two levels are reduced to one level by keeping the first level (lemmatized form) only if the concerned word is not a noun or a pronoun. For example, let Figure 6.5 below represents a part of K-Parser output such that *'signed_3'* is a verb then its transformed output is shown in Figure 6.6 below.

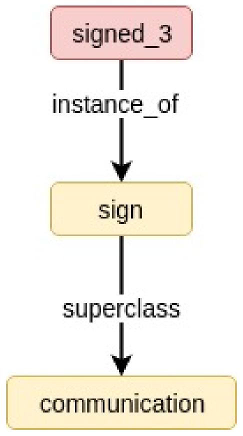


Figure 6.5: A Part of a K-Parser Output

2. The two levels are reduced to one level by keeping the second level (lemmatized form) only if the concerned word is a noun or a pronoun. For example, let Figure

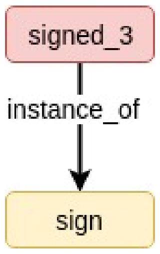


Figure 6.6: A Part of a K-Parser Output Transformed

6.7 below represents a part of K-Parser output such that *'John_3'* is a noun then its transformed output is shown in Figure 6.8 below.

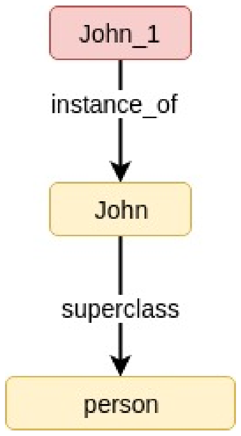


Figure 6.7: A Part of a K-Parser Output

6.4 K-Parser Wrapper to Automatically Represent Commonsense Knowledge

We also used the K-Parser system to automatically transform the knowledge of types 1 through 10 (See Chapters 3 and 4 for details) into graphical representations. A knowledge, as defined in the Definition 6, is of the type,

IF *S* ***THEN*** *x* ***is same as*** *y*,

where *S* is an English sentence, *x* and *y* are tokens in *S*.

We use the wrapper defined in the Section 6.3 above to transform *S* into a graph and

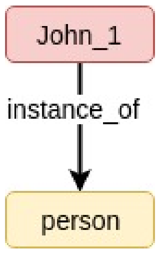


Figure 6.8: A Part of a K-Parser Output Transformed

then add the two edges labeled *'is_same_as'* between the nodes in the graph of *S* which represent the tokens *x* and *y*. The directions of the edges are opposite.

An example of a representation of a knowledge is as shown in the Figure 6.9.

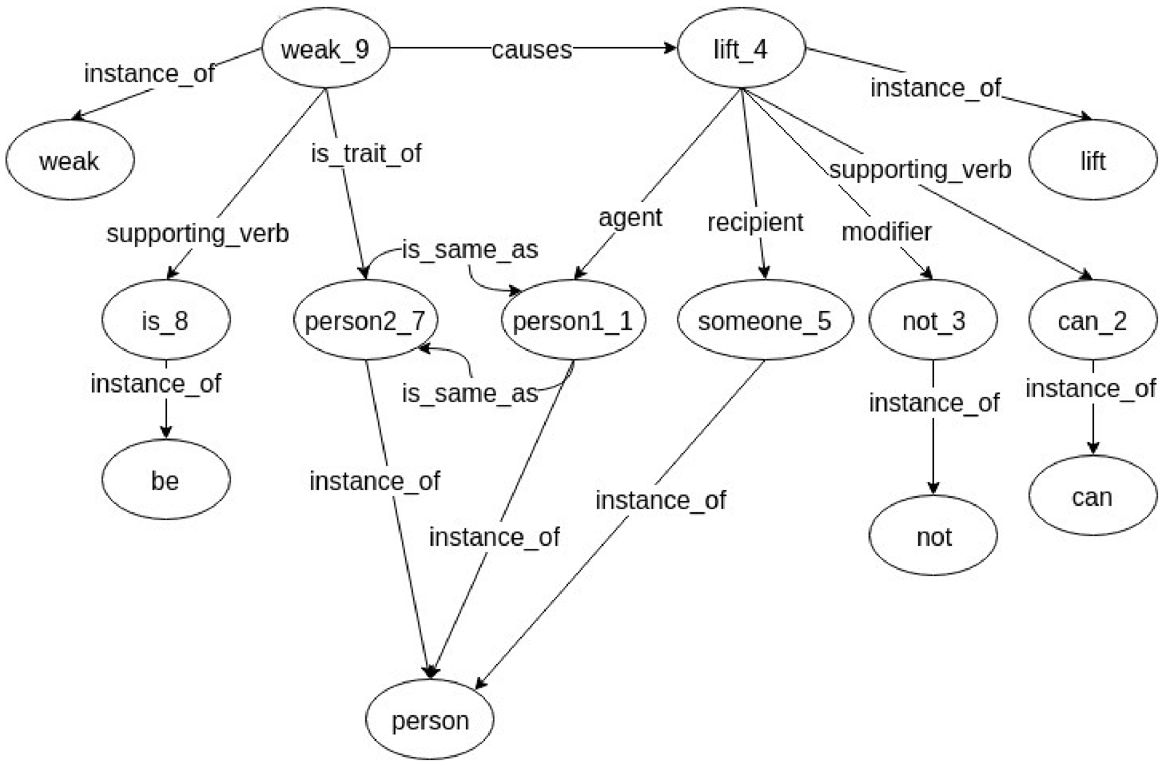


Figure 6.9: Graphical Representation of the Knowledge, "**IF** *person1 can not lift someone because person2 is weak* **THEN** *person1_1 is same as person2_7*"

Chapter 7

END-TO-END SYSTEMS

In this chapter we present two end-to-end systems to solve the Winograd Schema Challenge (WSC). The systems were developed as part of this work. The first system is based on formally representing an input WSC problem and the needed commonsense knowledge into a graphical reasoning algorithm. The second system and its updated version on the other hand takes WSC problem and the needed commonsense knowledge as plain English text and uses a Natural Language Inference based approach to address it. The details of each of the systems and their evaluation results are provided in the sections below.

7.1 System 1: Graphical Reasoning Based System

In this section we explain how various components, which are described throughout this thesis, are used to develop an end-to-end graphical reasoning based system to solve the Winograd Schema Challenge.

7.1.1 Overview of the System

Figure 7.1 provides an overview of the graphical reasoning based system. There are various components of the system including, a semantic parser to translate an English text into a graphical representation, a knowledge extraction module which takes a WSC problem as input and outputs zero or more pieces of commonsense knowledge, each of which belongs to a type defined in Chapter 3, and a reasoning algorithm which takes the graphical representations of a WSC problem and the needed knowledge as input and outputs the answer to the problem.

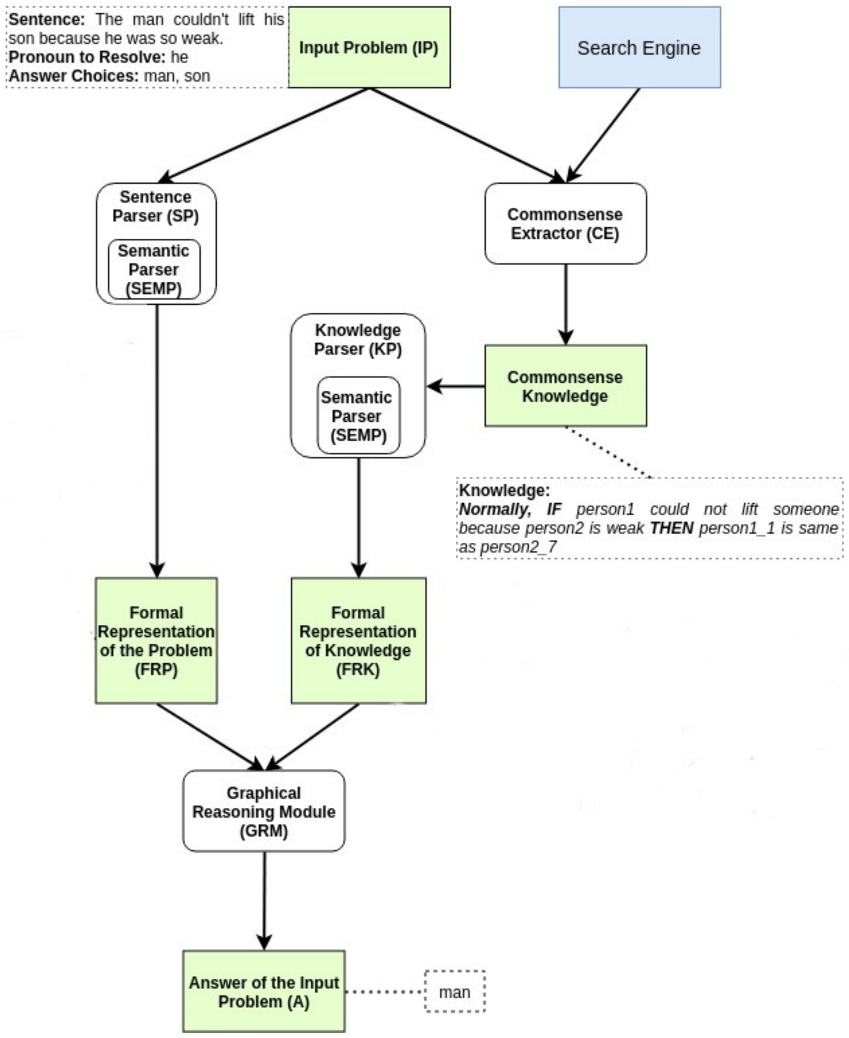

Sentence: The man couldn't lift his son because he was so weak.
Pronoun to Resolve: he
Answer Choices: man, son
Input Problem (IP)
Search Engine
Sentence Parser (SP)
Semantic Parser (SEMP)
Commonsense Extractor (CE)
Knowledge Parser (KP)
Semantic Parser (SEMP)
Commonsense Knowledge
Knowledge:
Normally, IF person1 could not lift someone because person2 is weak THEN person1_1 is same as person2_7
Formal Representation of the Problem (FRP)
Formal Representation of Knowledge (FRK)
Graphical Reasoning Module (GRM)
Answer of the Input Problem (A)
man

Figure 7.1: An Overview of the Graphical Reasoning Based System

7.1.2 Representing WSC Problems

Each WSC problem is represented as a graph. The representation is as defined in the Definition 4 of Chapter 4. A wrapper around the K-Parser system, as shown in the Chapter 6, is used to generate the graphical representations for the sentences in each WSC problem. An example is shown in Figure 7.2.

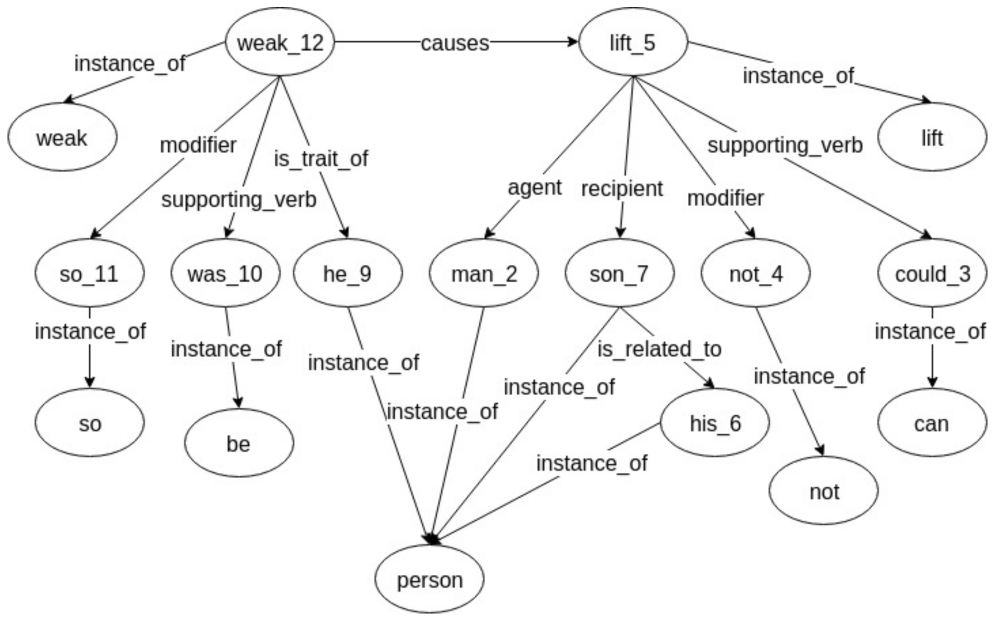


Figure 7.2: A Representation of the Sentence, *"The man could not lift his son because he was so weak"*

7.1.3 Extraction and Representation of Commonsense Knowledge

In this work, a commonsense knowledge corresponding to a WSC problem is automatically extracted by following a sequence of steps. The detailed description of the extraction process is provided in the Chapter 5. The extraction process involves the following main steps.

1. Creating search queries from the sentences in an input WSC problem. The queries are then used to extract text snippets which may contain the needed knowledge.

2. Extracting commonsense knowledge from the text snippets retrieved in the previous step as shown in the Chapter 5.

Based on the above steps a knowledge extracted for the WSC sentences shown in Figure 7.2 is, *"**IF** person1 could not lift someone because person2 is weak **THEN** person1_1 is same as person2_7"*.

The above knowledge is then translated into a graph by using a wrapper around the K-Parser system as explained in the Chapter 6. Figure 7.3 below shows the graphical representation of the knowledge.

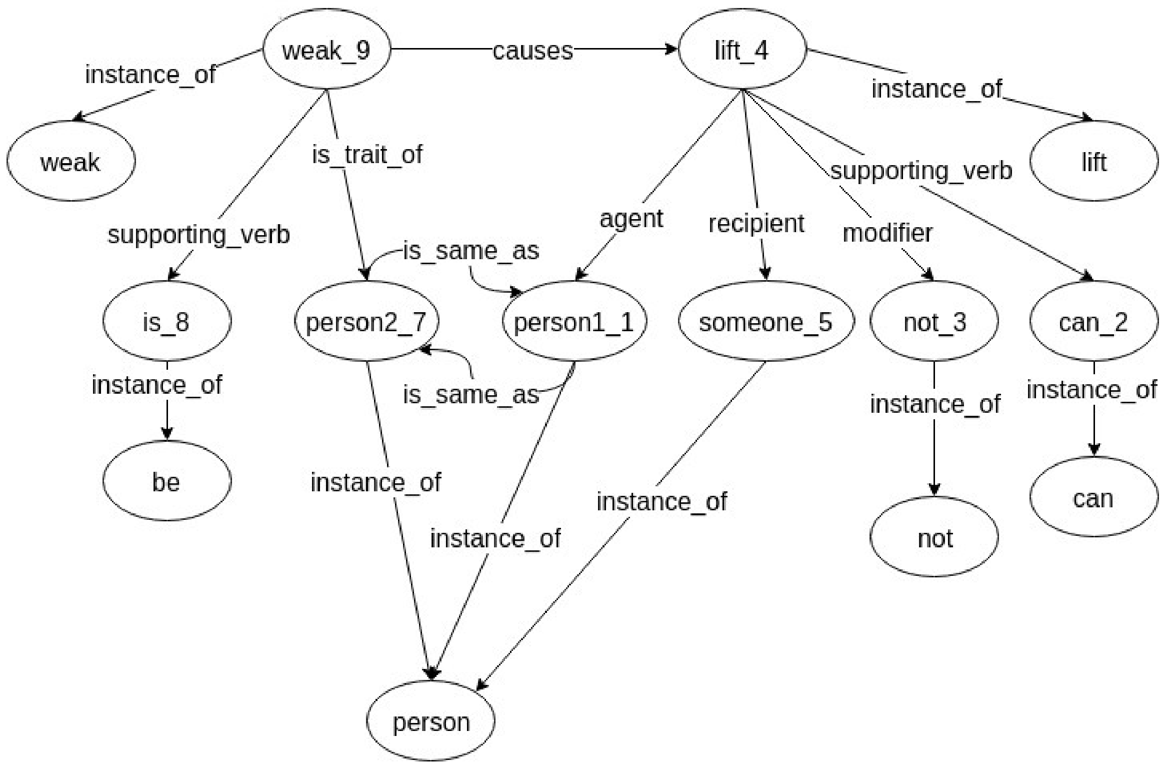


Figure 7.3: Graphical Representation of the Knowledge, "**IF** *person1 can not lift someone because person2 is weak* **THEN** *person1_1 is same as person2_7*"

7.1.4 Reasoning Algorithm

The reasoning algorithm is explained in the Chapter 4.

7.1.5 Evaluation

The goal of the evaluation process is two-fold. First, we validate if the WiSCR algorithm is able to correctly answer the WSC problems if the problem and a relevant knowledge is provided as inputs to it in the graphical as well as plain English formats. Secondly, we evaluate the automatic knowledge extraction along with the reasoning algorithm by providing only WSC problem as input to the system. We evaluated a corpus [1] of 291 WSC problems. . In this section we present the experiments to validate the WiSCR algorithm and our findings with respect to those experiments.

Experiment 1: First, we manually created the input graphical representations of the WSC sentences and the needed knowledge. We found that the WSC problems require different kinds of knowledge. The knowledge defined in this work (See Definition 5) is helpful in tackling 240 out of 291 WSC problems (82.47%). So we wrote the representations for those 240 problems by hand. The ASP implementation answered all of those problems correctly. The reasoning algorithm defined in this work relies on the fact that the provided knowledge contains the same or similar scenarios as that of the original WSC sentences. A scenario is basically defined by the actions, properties and the 'types' of entities present. By performing a comprehensive analysis of the WSC problems, we found that 240 out of 291 WSC problems can be answered using such knowledge. The remaining problems require two different kinds of knowledge. 26 problems require multiple pieces of knowledge. For example, **WSC Sentence:** *Mary tucked her daughter Anne into bed, so that she could work.* **Question:** Who is going to work? **Knowledge 1:** *someone who is tucked into bed, may sleep* **Knowledge 2:** *someone who's daughter is sleeping may be able to work.* It was observed that such knowledge has a partial overlap with the scenarios in a WSC problem. For example see the WSC sentence and knowledge 1 shown above. Due to this, such knowledge is not handled by the current algorithm. If one tries to format such knowledge according to the Definition 5 then the reasoning algorithm will not answer anything because it will not be able to find a graph-subgraph isomorphism between the subgraphs of WSC sentences' representation and knowledge's representation. The remaining 25 problems require the knowledge that one statement is more like to be true than the other. For example,

[1] Avaiable at `https://tinyurl.com/y22ykz5p`

WSC Sentence: *Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like dogs.* **Question:** *What looked like dogs?* **Knowledge:** *Sheep looks like a dog* **is more likely to be true than** *Shepherd looks like a dog*. Such knowledge is also not handled by the current reasoning algorithm because it does not satisfy the definition (Def 5) of knowledge reasoned with in this work. A list of the WSC problems which are not handled by the WiSCR algorithm because of the reasons mentioned above is also present at `https://tinyurl.com/y22ykz5p`.

Experiment 2: Secondly, we provided both the knowledge and the WSC sentences to the system in plain English format. We used the K-Parser wrappers to translate both WSC sentences and knowledge into graphical representations. The representations were then used by the reasoning algorithm to predict the answers to the problems. We found that out of the 240 problem which satisfied the reasoning criteria according to the previous evaluation, 200 were automatically translated into graphs and all of them were correctly solved by the reasoning algorithm. The main reasons for the incorrect representation generations for the remaining 40 problems were incorrect part-of-speech tagging and incorrect syntactic dependency parsing.

Experiment 3: Thirdly, we automatically extracted the knowledge by providing only a WSC problem as input to the system. The knowledge was found and automatically extracted for 120 problems. The ASP implementation was able to correctly answer all of the 120 problems. The automated extraction of knowledge is inspired from the work done in (Sharma *et al.*, 2015c).

7.1.6 Conclusion

In this work, we attempted to solve the Winograd Schema Challenge by creating a system which reasons with additional knowledge. To that end we defined a graphical representation of the English sentences in the input problems and a graphical representation of the relevant knowledge. We also defined a commonsense reasoning algorithm for WSC (WiSCR algorithm). We showed with the help of three experiments how an approach built on top of graph-subgraph isomorphism encoded in ASP is able to tackle 240 out of 291 WSC problems.

7.2 System 2: Entailment Based Reasoning System

One of the most difficult challenges in Artificial Intelligence (AI) is to develop systems that exhibit commonsense. It is a general consensus in the AI community that building machine commonsense is much more difficult than building a machine for tasks that require "expert adult thinking". The reason for which is nicely described in (Minsky, 1988):

> To be considered an "expert", one needs a large amount of knowledge of only a relatively few varieties. Each type of knowledge needs some form of "representation" and a body of skills adapted to using that style of representation. In contrast, an ordinary person's "common sense" involves a much larger variety of different types of knowledge and this requires more complicated management systems.

This challenging nature of commonsense has drawn the attention of a large body of researchers. Several benchmarks have been proposed to track the progress towards the direction. Two popular benchmarks among these are the task of science question answering (Clark *et al.*, 2018; Clark, 2015; Clark and Etzioni, 2016; Mihaylov *et al.*, 2018; Mishra *et al.*, 2018) and Winograd Schema challenge (Levesque *et al.*, 2011). The benchmark of science question answering, with the aim of building machines that are smarter than a 8^{th} grade science students could accommodate a large test-bed that evaluates a wide variety of commonsense behaviours such as understanding day-to-day context, reasoning about what led to a situation or what would be true in a situation or how the situation will evolve.

The Winograd Schema challenge, on the other hand, is more focused and concentrates solely on the task of coreference resolution. The main part of a Winograd Schema is a sentence(s) containing a pronoun, for instance:

> The city councilmen refused the demonstrators a permit because they feared violence.

In addition, two definite noun phrases, called "answer choices" are given; in the example above, the answer choices are *the city councilmen* and *the demonstrators*. The objective is to select the answer choice which provides the correct resolution for the pronoun. For instance, the correct answer to the question *Who feared violence?* is *the city councilmen*. A Winograd Schema also specifies an "alternate word" for a "special word" in the sentence. Replacing the "special word" by the "alternate word" changes the resolution of the pronoun. In the example above, the special word is *feared* and the alternate word is *advocated*. Thus every schema represents a pair of coreference resolution problems that are almost identical but have different answers. The presence of the "special word" and "alternate word" suggests that to solve these coreference resolution problems one need commonsense knowledge.

The restricted nature of the Winograd Schema challenge makes it an interesting benchmark. Also developing better coreference resolution system has broader impact on a variety of natural language applications. In this work we focus on Winograd Schema challenge.

Many existing methods on Winograd Schema Challenge use knowledge extraction and reasoning. Given a problem, the knowledge extraction module first extracts relevant knowledge. The reasoning module then takes those knowledge and make a prediction. To be able to use the knowledge, the reasoning module puts several restrictions on the structure of the extracted knowledge sentence. If the knowledge extraction module could not find any knowledge pertaining to those preferred schema the extracted knowledge would probably of no use. Due to reporting bias, people hardly report the obvious and on top of that if the reasoning system put hard filtering, the approach of knowledge extraction followed by reasoning will probably face severe difficulties. Several knowledge extraction methods have been proposed to get relevant knowledge however the state-of-the-art accuracy of the knowledge based systems is still at 58.3%. We take a detour from building better knowledge extraction modules in this work and focus on developing reasoning systems that can better utilize the extracted knowledge. Towards this we manually extract a knowledge sentence for each coreference resolution problem from the existing Winograd Schema Challenge dataset without paying any attention to the reasoning system. Through experiments we observe that existing knowledge based methods is unable to utilize the gold knowledge sentences. We then propose a simple reasoning method that uses Natural Language Inference (MacCartney and Manning, 2009; Bowman *et al.*, 2015) to deal with a broad varieties of knowledge sentences. Given a Winograd Schema problem and a knowledge sentence, our method uses a semantic role labelling function (Palmer *et al.*, 2010; FitzGerald *et al.*, 2018) to convert a Winograd Schema problem into a Natural Language Inference (NLI) problem (Bowman *et al.*, 2015). It then uses pre-trained NLI systems (Chen *et al.*, 2016; Parikh *et al.*, 2016) from AllenNLP (Gardner *et al.*, 2018) to compute the answer.

Our contributions in this work are two-fold: (a) we manually pair each coreference resolution problem from the Winograd Schema Challenge dataset with a knowledge sentence to create a knowledge augmented Winograd Schema Challenge dataset. The dataset is publicly available at `https://goo.gl/Q5khUC`; (b) we propose a novel algorithm that takes a coreference resolution problem from a Winograd Schema, a knowledge sentence and outputs confidence for each answer choice by using question answer based semantic role labelling (FitzGerald *et al.*, 2018) and Natural Langugae Inference.

7.2.1 Problem Formulation

In this section we define the task of knowledge based coreference resolution from Winograd Schema as follows:

Definition 10 ***Knowledge Based Winograd Schema Coreference Resolution*** *A knowledge based Winograd Schema coreference resolution problem is a quintuple* $C_{KB}^{WSCR} = \langle D, A_1, A_2, P_1, K \rangle$*, where*

1. *D is a sequence of sentences, that describes a specific scenario.*
2. *P refers to the pronoun in the D which needs to be resolved.*
3. A_1 *and* A_2 *are sub-strings of D and are called answer choices. The pronoun refers to one of these two answer choices.*
4. *K is a sentence, called the knowledge sentence which justifies if P is more likely to refer to* A_1 *or* A_2 *in the situation described by D.*

A solution to a C_{KB}^{WSCR} *problem is* A_1 *if P is more aligned towards* A_1 *based on D and K. Otherwise the solution is* A_2*. Table 7.1 shows an example of a* C_{KB}^{WSCR} *problem.*

D	⟨Fish ate the worm., It was tasty.⟩
P	It
A_1	Fish
A_2	the worm
K	I can eat it because it is tasty.

Table 7.1: An Example of a Knowledge Based Winograd Schema Co-reference Resolution Problem.

7.2.2 Knowledge Extraction

The goal of the knowledge extraction process is to extract a knowledge sentence for each Winograd Schema coreference resolution (WSCR) problem. The knowledge sentence should be able to justify the answer of the associated WSCR problem. In this vein, we aim to extract knowledge sentences that depict a similar scenario to that of the associated WSCR problem. We roughly characterize a WSCR *scenario* in terms of the events (verb phrases) and the properties of the participants that are associated with the *scenario*. The characterization of a scenario optionally includes the discourse connectives between the events of the *scenario*. For example, in the sentence *"The fish ate the worm. It was tasty."*, the scenario is mainly characterized by the verb phrase *"ate"*, the property *"tasty"*.

In this work, we use this abstract notion of a scenario to extract knowledge sentences which depict similar scenarios. The following steps summarize the extraction process:

1. First, the verb phrases, properties and discourse connectives in a given WSCR *scenario* are identified. For example the verb phrase *"ate"* and the property *"tasty"* in the example mentioned above.

2. Secondly, a set of search queries are created by using the items extracted in the previous step. For example a query *"* ate * tasty * "* is created for the problem mentioned above.

3. Thirdly, the above created query is used to search and extract sentences from a search engine. In this work, we have used the search engine of Google. An example sentence extracted by using the query mentioned above is, *"I can eat it because it is tasty"*. Not all the results from the search engine will be useful. Some sentences might not contain enough information about the scenario (e.g. "she ate"). Some sentences might be similar to the original scenario but contain the coreference ambiguity

that is present in the original scenario. Thus in this step, we manually classify results to be useful or not. If we are not able to find a useful sentence with the initial query then we go back to the second step and generate a new query by using a variants of the words used in the previous query. The variants include synonyms, similar words (*looser*, *lost*), and base forms of verbs in the verb phrases of the original query.

After several iterations of step 2 and step 3 we obtain sentences which can justify the answer of the given WSCR and do not contain the coreference ambiguity which was present in the original WSCR sentence. The knowledge sentences that are extracted often contain pronouns. For example the extracted sentence "I can eat it because it is tasty" also contains two pronoun occurrences (*"it"*,). However, we make sure that the pronouns in the extracted knowledge sentences can be easily resolved using the following procedure:

1. Two pronouns refer to each other if they have the same string description. For e.g. all the occurrences of the pronoun "it" always refer to the same entity.

2. Two pronouns (p_1, p_2) refer to each other if they belong to a special list containing the following: $\{(he,him),(she,her),(i,me),(they,them),(he,his),(his,him)\}$. We also ignore knowledge sentences where any of these special pair of pronouns appears as an argument to a common verb (e.g. "it ate it because ...").

We call such sentences as *"easily resolvable"* sentences. The manually curated knowledge base contains only *"easily resolvable"* and no-pronoun sentences. Table 7.2 shows some of the sample knowledge sentences and the corresponding WSCR problem.

Scenario	**Knowledge Sentence**
The *city councilmen* refused the *demonstrators* a permit because **they** feared violence.	He also refused to give his full name because he feared for his safety.

The *city councilmen* refused the *demonstrators* a permit because **they** advocated violence.	He has been refused travel to the West because he has openly advocated for terror against Israeli citizens as well as Americans in Iraq and Afghanistan.
Joan made sure to thank *Susan* for all the help **she** had received.	I just wanted to thank you because you have helped me the best when I received the deny letter from immigration some years ago.
Frank felt crushed when his longtime rival *Bill* revealed that **he** was the winner of the competition.	We lose property, and feel crushed and poor. We lose our ideals, our hopes.
Although they ran at about the same speed, *Sue* beat *Sally* because **she** had such a good start.	If you get off to a good start , you are successful in the early stages of doing something.
Although they ran at about the same speed, *Sue* beat *Sally* because **she** had such a bad start.	England got off to a bad start in the Five Nations Championship, losing 35-10 to France.
Anna did a lot better than her good friend *Lucy* on the test because **she** had studied so hard.	He succeeded because he studied hard.
The *firemen* arrived before the *police* because **they** were coming from so far away.	My teachers know that I arrive late sometimes. They do not punish me because they know I live far away, he said.

The sack of potatoes had been placed above the bag of flour, so it had to be moved first.	Coal seams are extracted from a mountain by removing the land , or overburden, above the seams.
Susan knew that Ann's son had been in a car accident , so she told her about it.	he got out of the car, and I told him to get back in because I knew he would do it all over again the next day.
Susan knew that Ann's son had been in a car accident, because she told her about it.	You knew about this pain, because we told you.
Bob was playing cards with Adam and was way ahead. If Adam had not had a sudden run of good luck, he would have won.	And if you're not lucky, you loose your money like my friend and cry!

Table 7.2: Examples of Manually Extracted Knowledge Sentences for C_{KB}^{WSCR} Problems. Each Row Shows a Winograd Schema Coreference Resolution Problem and a Knowledge Sentence That Depicts a Similar Scenario.

7.2.3 Alignment Algorithm

Existing Winograd solvers that use explicit commonsense knowledge to solve a Winograd Schema problem ($C_{KB}^{WSCR} = \langle D, A_1, A_2, P_1, K \rangle$) first convert the knowledge sentence K and the Winograd scenario D into a logical form and then use a set of axioms to compute the answer. However, it is a daunting task to convert free form sentences into causal logical representation. Thus these methods often produce low recall. In this work, we take a detour from this approach and aim to build an "alignment" function. Informally, the job of the alignment function is to decide which replacement $D[P/A_1]$ or $D[P/A_2]$ closely mimics K. Here, $D[P/A_i]$ represents the sentence(s) that is obtained by replacing the pronoun P by

A_i in D.

Intuition behind the Alignment Algorithm By definition, every replacement of a Winograd scenario $D[P/A_i]$ contains three special entities A_1,A_2 and A_3 where A_3 is A_1 if $i = 1$ or A_2 if $i = 2$. Each occurrences of these entities may have a semantic role with respect to some of the events in D. From the choice of extraction, the knowledge sentence K also contains similar events. The goal is to replace the special entities in $D[P/A_i]$ by the "similar" entities (noun phrases) from K. Let θ denotes such a substitution and $D[P/A_i, \theta]$ denote the string that is obtained by performing such a substitution. A textual entailment function is then used to compute the entailment score between K (premise) and $D[P/A_i, \theta]$ (hypothesis), which we treat as the alignment score between $D[P/A_i]$ and K.

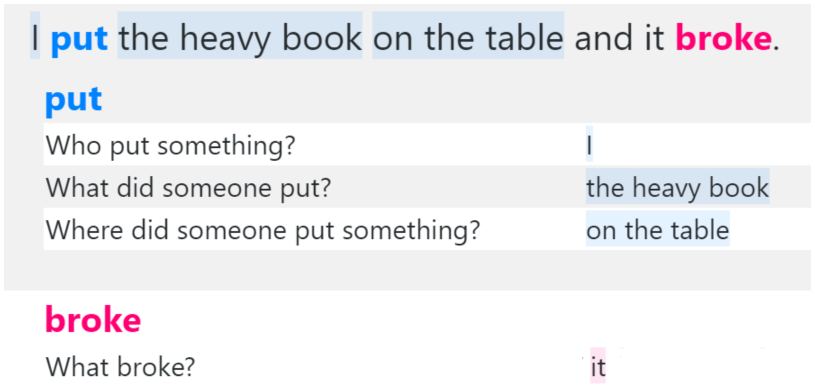
For example, for the Winograd scenario $D =$, "I put *the heavy book* on *the table* and **it** broke ." and the knowledge sentence $K =$ "A broken (fractured) toe is an injury normally caused by impact , usually from dropping *a heavy object* on the *toe* or stubbing the toe hard.", a possible substitution is θ={*"the heavy book"/"a heavy object"*,*"the table"/"toe"*}. We then compute, $D[P/A_1, \theta] =$"I put *the heavy object* on *toe* and **the heavy object** broke ." and $D[P/A_2, \theta] =$"I put *the heavy object* on *toe* and **toe** broke ." and the textual entailment score for (*premise*=K and *hypothesis*=$D[P/A_1, \theta]$) and (*premise*=K and *hypothesis*=$D[P/A_2, \theta]$) to find out that K is more aligned towards $D[P/A_2]$.

Computing Similar Entities We define an entity (noun phrase) E_j from K to be *similar* to an entity A_j from a replacement $D[P/A_i], i \in \{1, 2\}$ if the following holds:

1. There exists a verb v in D and v' in K such that either $v = v'$ or v is a synonym of v'.

2. The "semantic role" of A_j with respect to v is same as the "semantic role" of E_j with respect to v'.

To compute the semantic role of each entity we use the semantic role labelling function, called QASRL(He *et al.*, 2015). QASRL represents the semantic roles of an entity, in terms of question-answer pairs. Figure 7.4 shows the QASRL representation of the scenario "'I put *the heavy book* on *the table* and **it** broke .". The scenario involves two events "put" and "broke". The questions represent the roles of the participating entities.


I put the heavy book on the table and it broke.

put

Who put something?	I
What did someone put?	the heavy book
Where did someone put something?	on the table

broke

What broke?	it

Figure 7.4: QASRL Output for the Sentence *"I put the heavy book on the table and it broke."*.

Textual Entailment: Recognizing textual entailment is an important aspect of the approach described in this section. It refers to the task of determining whether the meaning of one text fragment can be inferred from the other (Dagan *et al.*, 2005). More formally [2] , "*textual entailment is a directional relation between two text fragments Text (t) and Hypothesis (h) such that t entails h ($t \Rightarrow h$) if, typically, a human reading t would infer that h is most likely true*". For example, if t = *'seafood is tasty'* and h = *'crab is tasty'* then *t entails h.*

Textual entailment is different than the logical [3] entailment. According to the logical entailment, *a entails b* ($a \vDash b$) if each model of a is a model of b. In other words $a \vDash b$ if every interpretation that satisfies a also satisfies b. Also, if there does not exist an interpretation which satisfies a then anything is entailed by a. For example, according to the definition of

[2]http://www.lsi.upc.edu/~ageno/anlp/textualEntailment.pdf

[3]https://en.wikipedia.org/wiki/Logical_consequence

logical entailment *‘1=2’* $\models$ *’unicorns are real’* because there does not exists an interpretation which satisfies *‘1=2’*. But according to the definition of textual entailment *‘1=2’* does not entail that *‘unicorns are real’* because a human reading *‘1=2’* would not infer from it that *‘unicorns are real’*.

Formal Specification of the Alignment Algorithm Each of the A_1, A_2 and A_3 may or may not have a similar entity in K. Thus there are $2^3 = 8$ cases ranging from the situation where all of A_1,A_2 and A_3 has a matching pair to the situation where none of them have a matching pair. The following list shows the behavior of the alignment function in all these 8 scenarios. Through this section, we use E_i to denote the similar entity for A_i.

Case TTT: This case occurs when each of A_1, A_2 and A_3 has a similar entity in K. In this case, P refers to A_1 if E_1 and E_3 refers to the same entity. Otherwise P refers to A_2. Since E_1, E_2, E_3 belongs to K, by the property of K, we know if E_1 and E_3 refers to the same entity or not. The following is an example of this case:

D: *Lily spoke to Donna, breaking* ***her*** *concentration.*

K: *If someone (E_1) speaks to you (E_2) and breaks your (E_3) concentration, or you realise a harness strap is twisted and have to untwist it, start checks again at the beginning.*

Case TTF: This case occurs when A_1 and A_2 have a similar entity but A_3 does not. In this case, a score is computed for each of $D[P/A_1]$ and $D[P/A_2]$ using textual entailment. Here, the θ for $D[P/A_i]$ is $\{A_1/E_1, A_2/E_2, A_3/E_i\}$. An example of this situation is the following:

D: *I put* ***the heavy book*** *on* ***the table*** *and* ***it*** *broke .*

K: *A broken (fractured) toe is an injury normally caused by impact , usually from dropping a heavy object (E_1) on the toe (E_2) or stubbing the toe hard .*

Case TFT: This case occurs when only A_1 and A_3 have a similar entity. In this case, P refers to A_1 if E_1 and E_3 refers to the same entity. Otherwise P refers to A_2. We do not have any example of this case in the dataset.

Case FTT: This case occurs when only A_2 and A_3 have a similar entity. In this case, P refers to A_2 if E_2 and E_3 refers to the same entity. Otherwise P refers to A_1. The following is an example of this scenario:

D: ***Mary*** *tucked* ***her daughter Anne*** *into bed, so that* ***she*** *could sleep.*

K: *My dog (E_2,E_3) is very serious about being tucked in to sleep in the morning and at night.*

Case TFF: This case occurs when, only A_1 has a similar entity. In this case, the substitution for $D[P/A_1]$ is $\{A_1/E_1, A_3/E_1\}$ but the substitution for $D[P/A_2]$ is $\{A_1/E_1\}$. The case of **FTF** (only A_2 has similar entity) and **FFT** (only A_3 has similar entity) similar to the case of **TFF**. The following is an example of the case **TFF**:

D: ***Frank*** *felt crushed when his longtime rival* ***Bill*** *revealed that* ***he*** *was the winner of the competition.*

K: *We (E_1) lose property, and feel crushed and poor. We lose our ideals, our hopes .*

Case FFF: This case occurs when none of A_1,A_2,A_3 have a similar entity. In this case, the alignment score for $D[P/A_i]$ is computed by taking the entailment score between K and $D[P/A_i]$. We do not have any example of this case in the dataset.

7.2.4 Experiments

Each problem in the Winograd Schema Challenge (WSC) requires additional knowledge to solve it. Consequently, every system which attempts to solve the challenge can be thought of having two main components. The first component is the knowledge identification component and the other is a reasoning component. The objective of the knowledge identification component is to identify the knowledge needed to solve a problem. Whereas, the objective of the reasoning component is to use the knowledge identified by the knowledge identification component to get the final answer to the problem. Both, the knowledge identification and the reasoning, components can be parts of a joint neural network architecture (Trinh and Le, 2018; Liu *et al.*, 2017) or they can be parts of a knowledge hunting and inference based reasoning framework (Sharma *et al.*, 2015c; Emami *et al.*, 2018).

In this work our focus is on developing a reasoning component hence we generate a new WSC dataset which contains the knowledge needed to solve each of the problems in the dataset. Therefore, we perform various experiments to highlight the capabilities of our reasoning algorithm. First we compare the existing systems with respect to their precision and recall on the challenge. Then we compared different versions of our system among themselves. The experiments show that the reasoning algorithm defined in this work is promising.

In this section, we start by introducing the datasets used in the experiments. Afterwards, we present the experimental setup, results and finally the error analysis.

Dataset

The Winograd Schema Challenge corpus [4] consists of pronoun resolution problems where a sentence is given along with a pronoun in the sentence and two possible answer choices. As mentioned in Section 4, a modified version of the Winograd Schema Challenge dataset is created as part if this work. The dataset contains the WSC problems and a piece of knowledge corresponding to each of the problems. In total, there are 285 problems in the WSC dataset [5] . The problems are made up of 141 pairs (as mentioned in the Introduction section) and one triple (three variants of a problem with a couple of words difference). From this point onward, we will call this dataset as WSC_{285}. The generation of the original WSC dataset (without the knowledge) itself is an ongoing work. Hence the dataset keeps getting updated. This is why the systems earlier than this paper used a smaller dataset of 273 problems (135 pairs and one triple). All the problems in it are also present in WSC_{285}. From this point onward, we will call this subset of WSC_{285} as WSC_{273}. The WSC_{273} also contains knowledge corresponding to each WSC problem in it. For a more fair comparison between our work and others', we performed our experiments with respect to both WSC_{285} and WSC_{273}.

Experimental Setup and Results

First, we compared the results of our system with the previous works in terms of the number of correct predictions, the precision, recall and F1 scores. We compared our system with four other systems. The comparison results are as shown in the Table 7.3. The first two, (Sharma *et al.*, 2015c) and (Liu *et al.*, 2017) hereafter called S2015 and L2017 respectively, address a subset of WSC problems (71 problems). Both of them are able to

[4]Available at `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.xml`

[5]Available at `https://drive.google.com/file/d/1TGnzoLuiRHQ8c04-c211_66qmvZ2eIKT/view?usp=sharing`

exploit only causal knowledge. This explains their low coverage over the entire corpus. We overcome this issue by using any form of knowledge sentence. More recently, two approaches on solving the WSC_{273} dataset have been proposed. The first work (Emami *et al.*, 2018) (hereafter called E2018) extract knowledge in form of sentences to find evidences to support each of the possible answer choices. We planned to execute their reasoning system with respect to our dataset but it could not be done as the complete code is not publicly available yet. A comparison between their results and our is however present in the Table 7.3. Another work (Trinh and Le, 2018) (hereafter called T2018) uses a neural network architecture to learn language models from huge data sources to predict the probability of choosing one answer over the other is also compared as shown in the Table 7.3.

	#correct	**% Correct**	**P**	**R**	**F1**
S2015	49	18.0	**0.92**	0.18	0.30
L2017	43	15.0	0.61	0.15	0.25
E2018	119	44.0	0.60	0.44	0.51
T2018	174	63.7	0.637	0.637	0.637
Our Method (WSC_{273})	176	**64.4**	0.644	0.644	**0.644**
Our Method (WSC_{285})	184	**64.5**	0.645	0.645	**0.645**

Table 7.3: Evaluation Results of Our System for WSC_{273} and WSC_{285} Along with Comparison with Other Systems

As mentioned in the previous section, our algorithm uses textual entailment to identify the more plausible answer. As part of second set of experiments we evaluate our reasoning algorithm with respect to two state of the art textual entailment systems , namely ESIM (Chen *et al.*, 2016) and Decoposibale Attention Model (DecAtt) (Parikh *et al.*, 2016). The results of using these two entailment systems with respect to the WSC_{285} and WSC_{273} are shown in the Table 7.4.

Dataset	Criteria	Textual Entailment System	
		DecAtt Model	**ESIM Model**
WSC_{285}	Correctly Answered	180	184
	Incorrectly Answered	105	101
	Total	285	285
WSC_{273}	Correctly Answered	172	176
	Incorrectly Answered	101	97
	Total	273	273

Table 7.4: Experimental Results for Different Textual Entailment (or Natural Language Inference) Systems

We performed a third set of experiments to further investigate the robustness of our reasoning algorithm as compared to the state of the art system (T2018). As mentioned earlier, there are 141 pairs of problems in the entire WSC dataset. The sentences in the problems which make up a pair differ only by a word or two. The two answer choices for both the problems in the pair are same but the answers to each of the problems are different. For example, consider the following pair of problems.

Problem1:

Sentence: *The firemen arrived* ***after*** *the police because they were coming from so far away.*; **Pronoun:** *they*; **Answer Choice 1:** *The firemen*; **Answer Choice 2:** *The police*; **Correct Answer:** *The firemen*

Problem2:

Sentence: *The firemen arrived* ***before*** *the police because they were coming from so far away .*;**Pronoun:** *they*; **Answer Choice 1:** *The firemen*; **Answer Choice 2:** *The police*; **Correct Answer:** *The Police*

In the above problems, only changing one word (*before/after*) in the sentence changes the answer to the problem. Due to this property of the dataset, a system can achieve an accuracy of 50% by just answering choice 1 as the correct answer for every problem. To make sure that this is not the case in our system, we performe the following two experiments.

1. **Experiment to Evaluate Pairwise Accuracy:** In this experiment we evaluate our system and the state of the art system (T2018) to find out how many of the problem pairs were correctly solved. The table 7.5 shows the results of the experiment.

Method	**Total Pairs**	**Correctly Answered**	**Incorrectly Answered**
T2018	131	42	89
Our Method (WSC_{273})	131	60	71
Our Method (WSC_{285})	141	61	80

Table 7.5: Experimental Results for Pairwise Accuracy

   It can be seen from the results that our system solves 61 pairs correctly, which is significantly more than the state of the art system.

2. **Experiment to Evaluate System Bias:** In this experiment we evaluate our system and the state of the art system (T2018) to find out if the system is biased to chose the answer choice which is closer to the pronoun in the WSC sentence. We found that usually the answer choice 2 in the problem is closer to the pronoun to be resolved.

Hence the experiments were performed to figure out how many times a system answers choice 2 as the final answer. The results of the experiments are as shown in the Table 7.6 below.

Method	**#Times Choice2 is Chosen**
T2018	142 out of 273
Our Method (WSC_{273})	143 out of 273
Our Method (WSC_{285})	148 out of 285

Table 7.6: Experimental Results for System Bias

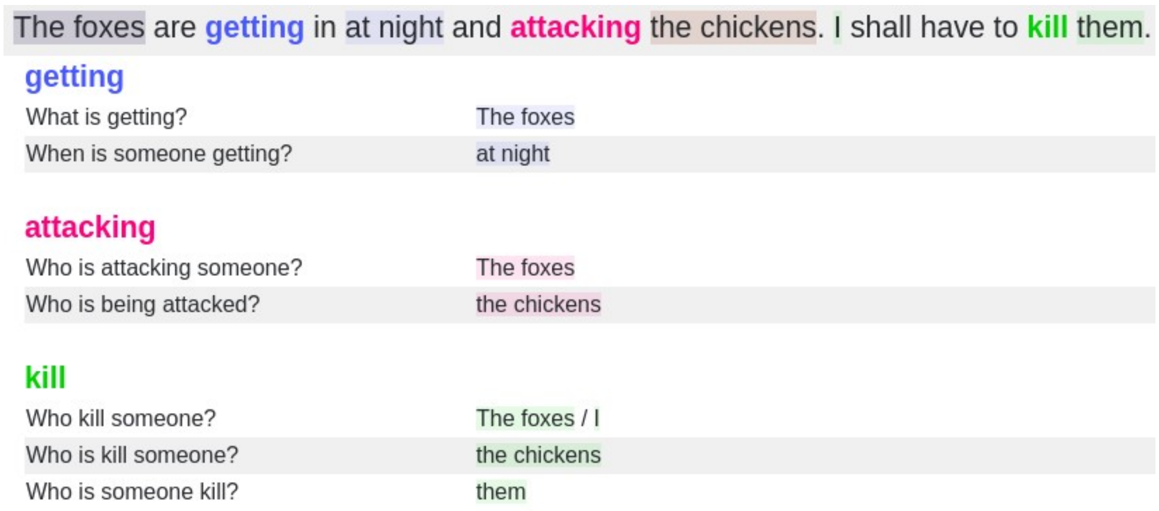
As seen from the results in Table 7.6 above, both, the language model based approach and our approach passed the test by not having a bias towards one of the answer choices.

Error Analysis

We categorize the errors of our reasoning algorithm by based on their cause. Following are the different types of errors observed.

Errors Caused by the QASRL System

We use the QASRL (FitzGerald *et al.*, 2018) system to translate the sentences into questions and answers. Sometimes the QASRL system failed to identify an important verb. Sometimes it assigned incorrect roles to the participants. These errors led to further errors in the downstream reasoning. An example of an incorrect role assignment is shown in Figure 7.5. Here, both *"I"* and *"The foxes"* are found to be the answers of the question *"Who kill someone?"*, which is incorrect.



The foxes are getting in at night and attacking the chickens. I shall have to kill them.

getting

What is getting?	The foxes
When is someone getting?	at night

attacking

Who is attacking someone?	The foxes
Who is being attacked?	the chickens

kill

Who kill someone?	The foxes / I
Who is kill someone?	the chickens
Who is someone kill?	them

Figure 7.5: QASRL Output for the Sentences, *"The foxes are getting in at night and attacking the chickens. I shall have to kill them."*.

Errors in the Reasoning Process

These errors are categorized into four 'buckets'. Each 'bucket' corresponds to a collection of cases in the alignment algorithm (Section 5). Bucket 1 contains case **TTT**, bucket 2 contains case **TTF, TFT** and **FTT** , bucket 3 contains case **TFF, FTF** and **FFT** and finally the bucket 4 contains the case **FFF**. Few examples of the errors with respect to these different buckets are shown below.

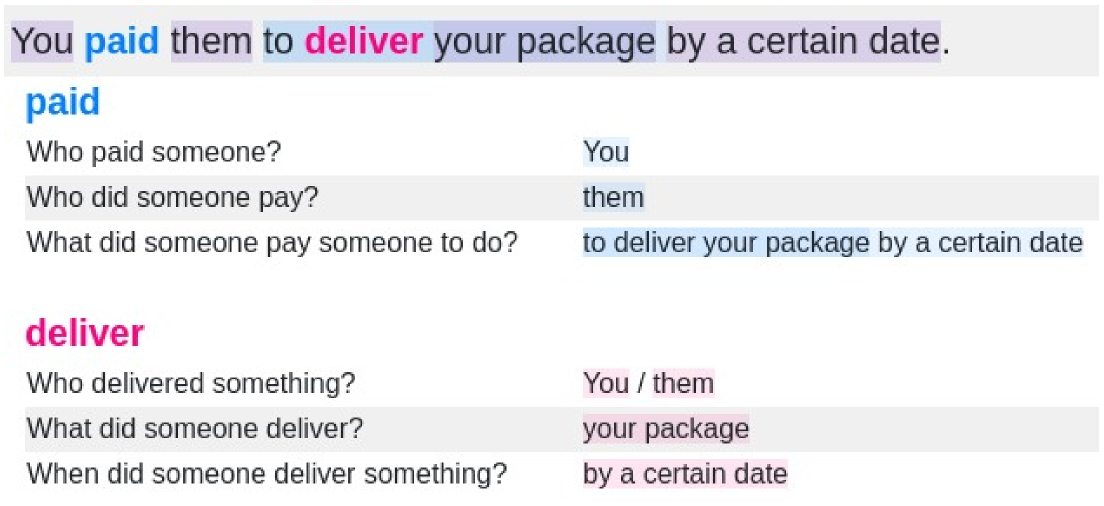
Example from Bucket 1

Scenario: *Joe paid the detective after he delivered the final report on the case.*

Pronoun: *he*

Knowledge Sentence: *You paid them to deliver your package by a certain date.*

Cause of Error: The cause of error for the above problem was found to be an error semantic role labeling by the QASRL system. The output of the QASRL system for the knowledge sentence in the above problem is shown in Figure 7.6. Here, both *You* and *them* are found to be the answers to the question *Who delivered something?* Because of this error both the answer choices are found to be the correct answers, which contradicts the initial assumption.



You paid them to deliver your package by a certain date.

paid

Who paid someone?	You
Who did someone pay?	them
What did someone pay someone to do?	to deliver your package by a certain date

deliver

Who delivered something?	You / them
What did someone deliver?	your package
When did someone deliver something?	by a certain date

Figure 7.6: QASRL Output for the Sentence *"You paid them to deliver your package by a certain date."*.

Example from Bucket 2

Sentence: *Thomson visited Cooper's grave in 1765 . At that date he had been dead for five years.*

Pronoun: *he*

Knowledge Sentence: *After Her Dad Died , She Wanted To Visit His Grave On Prom Night , But Noticed Something Odd.*

Cause of Error: The cause of error for the above problem was found to be an error made by the entailment function (ESIM).

Example from Bucket 3

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

Pronoun: *it*

Knowledge Sentence: *I loved the musical piece and the analogy!*

Cause of Error: Here, the number of similar entities between the Winograd sentence and the knowledge sentence is only one. And particularly in this case the two generated hypotheses are quite different from the premise (knowledge sentence). As a result, the textual entailment scores were incorrect.

Example from Bucket 4 None of the problem from the current dataset fell into this bucket. Due to this reason, we did not encounter any error example from this bucket. However, due to issues in "similar" entity computation, sometimes a problem from different case falls into Bucket 4. Consider the following two phrases for example: "he was the winner" and "John won". Ideally, the reasoning system should match the phrase *"was the winner"* with *"win"* and conclude that "he" and "John" are *similar*. But the existing reasoning algorithm can only match verb pairs with string equality or synonymy. Thus it fails to detect that "he" and "John" are *similar* which sometimes results in an error.

7.2.5 Conclusion

One of the major obstacle towards solving the Winograd Schema Challenge is knowledge extraction. However, that obstacle becomes more prominent if the reasoning system is naive. This is due to the old consensus that commonsense knowledge does not follow simple representation schemes and thus require complex knowledge management (Minsky, 1988). Thus it is important to produce better commonsense reasoning modules. In this work, we created a dataset to make progress towards this cause. We observe that the existing knowledge based reasoning systems for the Winograd Schema Challenge cannot utilize the knowledge sentences well. However, significant improvements can be made with a sophisticated general purpose reasoning system. Extraction of the required knowledge took significant amount of manual effort. So we could not automate the knowledge extraction at this point and compute the accuracy in an end-to-end fashion. However, our work provides a proof of concept that with knowledge extraction and reasoning, it is possible to achieve noticeable performance in the Winograd Schema Challenge.

7.3 Updated System 2: Combining Automatic Knowledge Extraction and Neural Language Models

System 2 mentioned above relies on the manual extraction of knowledge sentences corresponding to the WSC problems. Furthermore, it relies only on the extracted knowledge to predict the correct answer of a WSC sentences. But sometimes the needed knowledge are embedded in the pre-trained language models. Let us consider the WSC example mentioned below.

Sentences: The painting in Mark's living room shows an oak tree. It is to the right of a house.

Pronoun to resolve: It

Answer Choices: a) painting b) tree

Here, the knowledge that *'a tree is to the right of a house'* is more likely than *'a painting is to the right of a house'* is needed. With recent developments in neural network architectures for language modeling, it is evident that they are able to capture such knowledge by predicting that *'a tree is to the right of a house'* is a more probable phrase than *'a painting* *is to the right of a house'*. This is because language models are trained on huge amounts of text and they are able to learn the frequently co-occurring concepts from that text. Although the knowledge from language models is helpful in many examples, it is not suitable for several others. For example, with respect to the WSC sentences *"The fish ate the worm. It was tasty."*, the language models in (Trinh and Le, 2018) predict that *'fish is tasty'* is a more probable than *'worm is tasty'*. This is because the words *'fish'* and *'tasty'* occur in the same context more often than the words *'worm'* and *'tasty'*.

So, considering the benefits and limitations of the above mentioned approaches, in this work, we combine the knowledge hunting and neural language models to solve the Winograd Schema Challenge (WSC). The main contribution of this work is to tackle the WSC by:

- developing and utilizing an automated knowledge hunting approach to extract the needed knowledge and reason with it without relying on a strict formal representation,
- utilizing the knowledge that is embedded in the language models, and
- combining the knowledge extracted from knowledge hunting and the knowledge in language models.

As a result, our approach improves on the existing state-of-the-art accuracy by 7.36% and solves 71.06% of the WSC problems correctly.

The sections below provide the details of our approach. The parts which were common with the system 2 mentioned above are omitted to prevent repetition.

The knowledge hunting approach described in this work is similar to the knowledge extraction approach defined in the Chapter 5 for the WSC problems. The language modeling and combining the results of knowledge hunting and language modeling were performed in collaboration with a graduate student at Arizona State University. The details of the implementations are present in the ACL 2019 paper (Prakash *et al.*, 2019). The sections below provide an overview of the entire approach.

7.3.1 Knowledge Extraction

The goal of the knowledge extraction module is to automatically extract a set of knowledge texts for a given WSC problem. Ideally, a *knowledge text* should be able to justify the answer of the associated WSC problem. In this vein, we aim to extract the texts that depict a scenario that is similar to that of the associated WSC problem. We roughly characterize a WSC scenario in terms of the events (verb phrases) and the properties of the entities that are associated with the scenario. The characterization of a scenario optionally includes the discourse connectives between the events and properties of the scenario. For example, in the WSC sentence *"The city councilmen refused the demonstrators a permit because they feared violence ."*, the scenario is mainly characterized by the verb phrases *"refused"* and *"feared"*, and the discourse connective *"because"*.

In this work, we use this abstract notion of a scenario to extract *knowledge texts* which depict similar scenarios. The following are the steps in the extraction module.

1. First, the module identifies the verb phrases, properties and discourse connectives in a given WSC *scenario*. For example the one-word verb phrases *"refused"* and *"feared"*, and the discourse connective *"because"* in the example mentioned above.

2. Secondly, the module automatically generates a set of search queries by using the keywords extracted in the previous step. The first query in the set is an ordered combination (as per the WSC sentence) of the keywords extracted in the previous step. For example the query *"* refused * because * feared * "* is the first query for the problem mentioned above. Afterwards the following set of modifications are performed with respect to the first query and the results are added to the set of queries.

- The verb phrases are converted to their base form. For example, *" * refuse * because * fear * "*.
- The discourse connectives are omitted. For example, *"* refuse * fear * "*.
- The verbs in verb phrases and the adjectives are replaced with their synonyms from the WordNet KB (Miller, 1995). The top five synonyms from the top synset of the same part of speech are considered. An example query generated after this step is *"* decline * because * fear * "*.

3. Thirdly, the module uses the generated queries to search and extract text snippets, of length up to 30 words, from a search engine. The top 10 results (urls) from the search engine are retrieved for each query and text snippets from those results are scraped. Out of the extracted texts, the 10 text snippets which are most similar to the WSC text are filtered and passed to the alignment module. We used a natural language inference model (Parikh *et al.*, 2016) to find the most similar sentences. Since we also do not want to extract the snippets which contain the corresponding WSC sentences (because of ambiguity), this module removes the results with WSC sentences in them. We filtered out the *knowledge texts* which contained 80% or more words from the sentences in any of the WSC problems.

An example *knowledge text* extracted by using the query *" * refused * because * feared * "* via the steps mentioned above is, *"He also refused to give his full name because he feared for his safety."*

7.3.2 Entity Alignment

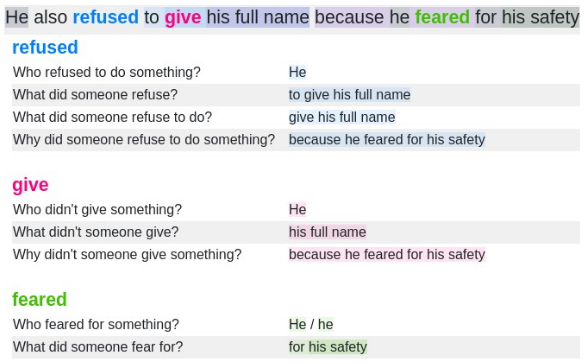
A total of up to 10 *knowledge texts* are extracted with respect to each WSC problem. Each of them is processed individually along with the WSC problem to produce a corresponding intermediate result from the knowledge hunting module.

Let $W = \langle S, A_1, A_2, P, K \rangle$ be a modified WSC problem such that S be a set of WSC sentences, A_1 and A_2 be the answer choices one and two respectively, P be the pronoun to be resolved, and K be a *knowledge text*. The existing solvers (Sharma *et al.*, 2015c) that use explicit knowledge to solve a WSC problem of the form W first convert K and S into a logical form and then use a set of axioms to compute the answer. However, it is a daunting task to convert free form text into a logical representation. Thus these methods often produce low recall. In this work, we take a detour from this approach and aim to build an "alignment" function. Informally, the task of the alignment function is to align the answer choices (A_1 and A_2) and the pronoun to be resolved (P) in S with the corresponding entities (noun/pronoun phrases) in K. These alignments are the intermediate results of the knowledge hunting module.

By the choice of knowledge extraction approach, the knowledge texts are similar to the WSC sentences in terms of events, i.e., they contain similar verb phrases, properties and discourse connectives. So, in an ideal situation we will have entities in K corresponding to each one of the concerned entities (A_1, A_2 and P) in W respectively. The goal of the alignment algorithm is to find that mapping. The mapping result is generated in the form of a *aligned_with* predicate of arity three. The first argument represents an entity (an answer choice or the pronoun) from S, the second argument represents an entity from K and the third argument is an identifier of the knowledge text used. We define an entity (noun phrase) E_j from a *knowledge text* K to be *aligned_with* to an entity A_j from a WSC text S if the following holds:

1. There exists a verb v in S and v' in K such that either $v = v'$ or v is a synonym of v'.
2. The "semantic role" of A_j with respect to v is same as the "semantic role" of E_j with respect to v'.

We use the semantic role labelling function, called QASRL (He *et al.*, 2015) to compute the semantic roles of each entity. QASRL represents the semantic roles of an entity, in terms of question-answer pairs. Figure 7.7 shows the QASRL representation of the *knowledge text* "*He* also refused to give his full name because *he* feared for his safety." It involves three verbs "refused", "feared" and "give". The questions represent the roles of the participating entities.



He also **refused** to **give** his full name because he **feared** for his safety

refused

Who refused to do something?	He
What did someone refuse?	to give his full name
What did someone refuse to do?	give his full name
Why did someone refuse to do something?	because he feared for his safety

give

Who didn't give something?	He
What didn't someone give?	his full name
Why didn't someone give something?	because he feared for his safety

feared

Who feared for something?	He / he
What did someone fear for?	for his safety

Figure 7.7: QASRL Output for the Sentence *"He also refused to give his full name because he feared for his safety."*

An example alignment generated for the WSC sentence,

S = *"The city councilmen refused the demonstrators a permit because they feared violence."*

and the *knowledge text*,

K = *"He also refused to give his full name because he feared for his safety."*

is,

aligned_with(city councilmen,He,K)

aligned_with(they,he,K)

There are three relevant entities in an input WSC problem, i.e., A_1, A_2 and P. Based on the existence of the entities corresponding to the entities in the WSC problem there are 2^8 possible cases. For example, the case $\{True\ True\ True\}$, abbreviated as $\{TTT\}$, represents that each of the entities A_1, A_2 and P are aligned with corresponding entities in a *knowledge text*.

Case	**Details**	**Example**
TTT	Each entity (among A_1, A_2 and P) in the WSC sentences W have corresponding entities in the corresponding *knowledge text K*	**WSC Sentence:** ***Jim*** *comforted* ***Kevin*** *because* ***he*** *was so upset .* **Knowledge Text (K):** *She says* ***I*** *comforted* ***her****, because* ***she*** *was so upset* **Alignments:** *aligns_with(Jim,I,K), aligns_with(Kevin,her,K), aligns_with(he,she,K)*
TFT	Only the entity representing the answer choice one (A_1) and the pronoun to be resolved (P) have corresponding entities in the *knowledge text K*	**WSC Sentence:** *The* ***trophy*** *does not fit into the brown* ***suitcase*** *because* ***it*** *is too large .* **Knowledge Text (K):** *installed CPU and* ***fan*** *would not fit in because the* ***fan*** *was too large* **Alignments:** *aligns_with(trophy,fan,K), aligns_with(it,fan,K)*
FTT	Only the entity representing the answer choice 2 (A_2) and the pronoun to be resolved (P) have corresponding entities in the *knowledge text K*	**WSC Sentence:** ***James*** *asked* ***Robert*** *for a favor but* ***he*** *refused .* **Knowledge Text (K):** *He asked the* ***LORD*** *what he should do, but the* ***LORD*** *refused to answer him, either by dreams or by sacred lots or by the prophets.* **Alignments:** *aligns_with(Robert,LORD,K) and aligns_with(he,LORD,K)*

Table 7.7: Alignment Cases in the Knowledge Hunting Approach. A_1 and A_2 Are Answer Choices One and Two, and P Is Pronoun to Resolve

The intuition behind the alignment approach is to find a common entity in a *knowledge text* such that it aligns with one of the answer choices (say A_i) and also with the pronoun to be resolved (P). Then we can say that both A_i and P refer to same entity and hence they refer to each other. An important aspect of such a scenario is the existence of the entities in a *knowledge text* which align with at least one of the answer choices and the pronoun to be resolved. In other words the cases $\{TTT\}$, $\{TFT\}$ and $\{FTT\}$. So we consider the alignments generated only with respect to these three cases as an output of the alignment module. The three cases and their details are shown in the Table 7.7 along with examples from the dataset.

7.3.3 Using the Knowledge from Language Models

In recent years, deep neural networks have achieved great success in the field of natural language processing (Liu *et al.*, 2019; Chen *et al.*, 2018). With the recent advancements in the neural network architectures and availability of powerful machine it is possible to train unsupervised language models and use them in various tasks (Devlin *et al.*, 2018; Trinh and Le, 2018). Such language models are able to capture the knowledge which is helpful in solving many WSC problems. Let us consider the WSC problem shown below.

S3: I put the heavy book on the table and it broke.

Pronoun to resolve: it

Answer Choices: a) table b) book

A knowledge that, "*table broke* is more likely than *book broke*" is sufficient to solve the above WSC problem. Such a knowledge is easily learned by the language models because they are trained on huge amounts of text snippets which are transcribed by people. Furthermore, these models are good at learning the frequently occurring patterns from data.

In this work, we aim to utilize such knowledge that is embedded in the neural language models. We replace the pronoun to be resolved in the WSC text with the two answer choices, one at a time, generating two possible texts. For example the two texts generated in the above WSC example are, S3(a) = *I put the heavy book on the table and table broke.*, S3(b) = *I put the heavy book on the table and book broke.* Then a pre-trained language model is used to predict the probability of each of the generated texts. Let P_a be the probability of S3(a) and P_b be the probability of S3(b). To be able to use the result of language models in Probabilistic Soft Logic (PSL) (Kimmig *et al.*, 2012), the output of this step contains ***coref***(P,A_1)*:PROB*1 and ***coref***(P,A_2)*:PROB*2, where P is the pronoun to be resolved, A_1 and A_2 are answer choices one and two respectively, and *PROB*1 and *PROB*2 are the probabilities of the texts generated by replacing P with A_1 and A_2 in the WSC text respectively, i.e., P_a and P_b in the example above.

7.3.4 Combining Knowledge Hunting and Language Models

In this step, the alignment results generated from the knowledge hunting module and the co-reference probabilities generated from the language models are combined in a Probabilistic Soft Logic (PSL) (Kimmig *et al.*, 2012) framework to infer the confidence for each of the answer choices in a WSC problem. The details of the PSL framework are present in Prakash *et al.* (2019).

7.3.5 Experiments

The Winograd Schema Challenge corpus [6] consists of pronoun resolution problems where a set of sentences is given along with a pronoun in the sentences and two possible answer choices such that only one choice is correct. There are 285 problems in the WSC dataset. From this point onward, we will call this dataset as WSC_{285}. The generation of the original WSC dataset itself is an ongoing work. Hence the dataset keeps getting updated. This is why the works earlier than ours, used a smaller dataset containing 273 problems. All the problems in it are also present in WSC_{285}. From this point onward, we will call this subset of WSC_{285} as WSC_{273}. For a fair comparison between our work and others', we performed our experiments with respect to both WSC_{285} and WSC_{273}. The core to reproduce the results of this paper is available at `https://github.com/Ashprakash/CKLM`.

7.3.6 Results

First, we compared the results of our system with the previous works in terms of the number of correct predictions. The language models based component of our approach

[6] Available at `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.xml`

relies on pre-trained language models. Here we compared two different language models. First we used the ensemble of 14 pre-trained language models which are used in (Trinh and Le, 2018). Secondly, we used BERT (Devlin *et al.*, 2018) pre-trained model. Based on the language model used, in the following experiments we use OUR_METHOD$_{T2018}$ to represent our approach which uses models from (Trinh and Le, 2018) and OUR_METHOD$_{BERT}$ to represent our approach which uses the BERT language model. We compared our method with five other methods (two language models based and three others). The comparison results are as shown in the Table 7.8. The first two, (Sharma *et al.*, 2015c) and (Liu *et al.*, 2017) hereafter called S2015 and L2017 respectively, address a subset of WSC problems (71 problems). Both of them are able to exploit only causal knowledge. This explains their low coverage over the entire corpus. We overcome this issue by using any form of knowledge text making predictions for each of the problems in the dataset. More recently, two approaches on solving the WSC_{273} dataset have been proposed. The first work (Emami *et al.*, 2018) (hereafter called E2018) extract knowledge in form of sentences to find evidences to support each of the possible answer choices. A comparison between their results and our is present in the Table 7.8. Another work (Trinh and Le, 2018) (hereafter called T2018) uses a neural network architecture to learn language models from huge data sources to predict the probability of choosing one answer over the other is also compared as shown in the Table 7.8.

7.4 An Entailment Based Reasoning Approach for Natural Language Question Answering

In this section we provide an overview of how the entailment based approaches mentioned above can be used in developing a general reasoning approach for any natural language question answering problem that requires the use of additional knowledge.

Let P be a natural language question answering problem such that P contains,

	#correct	**% Correct**
S2015	49	18.0
L2017	43	15.0
E2018	119	44.0
T2018 (WSC_{273})	174	63.70
T2018 (WSC_{285})	180	63.15
BERT Only (WSC_{273})	173	63.36
BERT Only (WSC_{285})	179	62.80
OUR_METHOD$_{T2018}$ (WSC_{273})	**189**	**69.23**
OUR_METHOD$_{T2018}$ (WSC_{285})	**195**	**68.42**
OUR_METHOD$_{BERT}$ (WSC_{273})	**194**	**71.06**
OUR_METHOD$_{BERT}$ (WSC_{285})	**200**	**70.17**

Table 7.8: Evaluation Results of Our Method with Respect to WSC_{273} and WSC_{285}, Compared with Other Systems.

- a sequence of one or more sentences (or a reading passage), say S,
- a question about the sentences, say Q,
- a list of answer choices, say $A = A_1, A_2, ..., A_n$

Also, let K be a list of additional knowledge about P, written in English. Then, a general approach for solving P, which uses entailment based reasoning, has the three steps in it. Let us consider the following example problem to better understand the steps in the reasoning approach.

Passage: *In some countries, formal education can take place through home schooling. Informal learning may be assisted by a teacher occupying a transient or ongoing role, such as a family member.*
Question: *Who is most likely to teach a child at home?*
Answer Choices: *a) mom b) friend c) stranger*
Knowledge: *mom is a family member*

- **Step 1: Combine S and K.** In this step the English text which represents the additional knowledge is appended to the reading passage in P. Let the combined passage be C. For example, following is the combined passage with respect to the above example.

  C = *In some countries, formal education can take place through home schooling. Informal learning may be assisted by a teacher occupying a transient or ongoing role, such as a family member. mom is a family member.*

- **Step 2: Generate sentences by using the input question and the answer choices.** In this step, each answer choice is used along with the input question to generate an English sentence. For example, following are the sentences generated with respect to the question and the answer choices in the above mentioned problem.

  A_1 = *mom is most likely to teach a child at home.*
  A_2 = *friend is most likely to teach a child at home.*
  A_3 = *stranger is most likely to teach a child at home.*

- **Step 3: Use a Natural Language Inference (NLI) system to select the correct answer choice.** In this step an NLI system is used to identify the textual entailment score between C and each of the sentences generated in the Step 2. Then, the answer choice which corresponds to the generated sentence that resulted in the highest entailment score is deemed as the correct answer of the input problem P. For example, entailment scores for (C,A_1), (C,A_2) and (C,A_3) will be retrieved from an NLI system for the example problem mentioned above. The answer choice *‘a) mom‘* will be selected as the final answer because (C,A_1) gets the highest textual entailment score.

Chapter 8

CONCLUSION AND FUTURE WORK

In this dissertation, progress towards Natural Language Understanding (NLU) is made by presenting commonsense reasoning algorithms, identifying different types of commonsense knowledge, developing and implementing commonsense extraction techniques and improving a semantic parser. The tools and techniques developed as part of this work are used to solve a popular NLU task called the Winograd Schema Challenge (WSC). WSC is a textual question-answering challenge such that the answer to a question requires resolution of a pronoun to its correct antecedent in a sentence.

Although while developing various tools and techniques our main focus was to solve a particular application (i.e., WSC), the instruments and procedures developed as part of this work are general purpose and they are useful for solving the other NLU applications which require additional commonsense knowledge and reasoning with the knowledge.

The following section presents a review of all the modules developed in this work and a summary of experimental evaluation of those modules.

8.1 Research Contributions

- **Knowledge Types Identification:** As part of this dissertation we identified several new kinds of commonsense knowledge. We used the problems in the Winograd Schema Challenge (WSC) corpus to identify these kinds because these types of commonsense knowledge are found to be necessary to solve the WSC (Sharma *et al.*, 2015c). Furthermore, we found that such commonsense is also helpful in other NLU problems such as the COPA challenge (Roemmele *et al.*, 2011). Furthermore, these kinds are not present in the currently available knowledge bases.

In Chapter 3, the various new kinds of commonsense knowledge which are identified in this work are explained. There are 12 knowledge kinds identified in this work and they are useful in solving all of the problems in the WSC corpus.

- **Graphical Reasoning Algorithm:** We defined and implemented a graphical reasoning algorithm to solve the WSC problems. The framework takes graphical representations of a problem and a piece of commonsense knowledge as inputs and produces the answer to the problem if it is supported by the inputs. The reasoning module has the ability to reason with the different kinds of commonsense knowledge identified from the WSC corpus. The framework takes the input and produces the answer to the question if the commonsense knowledge is suitable for answering the problem.

  In Chapter 4, the graphical reasoning algorithm is explained in detail.

- **Knowledge Extraction:** There are various kinds of commonsense knowledge identified as part of this work. Various techniques to automatically extract those types of knowledge are implemented. A knowledge base to store the extracted knowledge is also implemented as part of this work.

  In Chapter 5, the details of different methods that are used to extract commonsense knowledge are showed. The details of the database which is used to store the extracted knowledge are also provided in the chapter.

- **Semantic Parsing:** In this thesis, improvements in a general domain semantic parser and knowledge augmentation system are made. The system is called Knowledge Parser or K-Parser. K-Parser takes an English text as input and produces a directed acyclic semantic graph. The nodes in the graph represent the concepts in the text such as actions, entities and traits. The edges in the graph represent the relationships between concepts (or nodes).

In Chapter 6, the detailed implementation of the K-Parser system and improvements made to it in this work are provided.

8.1.1 Summary of Experimental Evaluations

Various experiments were performed in this work to evaluate the contributions defined above. Below is a summary of those experiments and results with respect to them.

1. **Evaluation of Knowledge Types Identification:** A total of 291 WSC problems were considered for knowledge type identification. All 291 problems were categorized in the 12 categories mentioned in the Chapter 3. The split of WSC problems according to the knowledge types is as shown in the Table 8.1 below.

   Table 8.1: Table of Knowledge Types and Number of WSC Problems in Them

Knowledge Type	Number of WSC Problems
1: A PROPERTY may prevent an ACTION	17
2: An ACTION may cause an ACTION	58
3: A PROPERTY may causes an ACTION	38
4: An ACTION may cause a PROPERTY	21
5: An ACTION may prevent an ACTION	8
6: An ACTION may be followed by an ACTION	25
7: An ACTION may be followed by a PROPERTY	1
8: A PROPERTY may be followed by an ACTION	1
9: A Co-occurring ACTION(s) and PROPERTY(s)	68
10: A PROPERTY may cause a PROPERTY	3
11: Statement1 is more likely than Statement2	25
12. Multiple Pieces of Knowledge	26

2. **Evaluation of the Graph Based Reasoning Approach:** We performed the following three experiments to evaluate the graph based reasoning approach for commonsense reasoning.

   (a) **Experiment 1:** There are 240 problems in the knowledge types 1-10 (See Table 8.1). All the 240 WSC problems were automatically translated into graphs by using K-Parser wrapper mentioned in the Chapter 6. The needed knowledge for each problem was manually written in graph format. All the 240 problems were correctly answered by the WiSCR Algorithm mentioned in the Chapter 4.

(b) **Experiment 2:** Similar to the experiment 1 above, in this experiment we considered the 240 WSC problems. The needed knowledge for all the 240 problems was manually written in the ***'IF** S **THEN** x is same as y'* format defined in the Chapter 4. Both, the WSC problems and the needed knowledge were automatically converted into graphs by using the K-Parser wrappers explained in the Chapter 6. 200 out of 240 problems were correctly answered in this experiment by the graphical reasoning algorithm. The remaining 40 problems were not answered because of syntactic dependency parsing errors and part-of-speech errors while generating the graphical representations of the input WSC problems and the needed knowledge.

(c) **Experiment 3:** In this experiment, the knowledge needed to solve the 240 WSC problems which require a knowledge of type 1-10 was automatically extracted by using the procedure described in the Chapter 5. Both, the WSC problems and the automatically extracted knowledge were translated into graphs by using the K-Parser wrappers described in the Chapter 6. 120 out of 240 problems were correctly answered. The remaining 120 were not answered because of the inability of the knowledge extraction module to extract a suitable knowledge. This happened mainly because of the limited access to the search engine.

3. **Evaluation of the Entailment Based Reasoning Approach:** The evaluation results for the entailment based reasoning approach as as shown in the Table 8.2 below. Our approach beats the previous state-of-the-art approach on two versions of WSC corpus (WSC_{273} and WSC_{285}).

4. **Evaluation of the Knowledge Hunting and Language Models Based Approach:**

	#correct	**% Correct**
S2015	49	18.0
L2017	43	15.0
E2018	119	44.0
T2018	174	63.7
Our Method (WSC_{273})	176	**64.4**
Our Method (WSC_{285})	184	**64.5**

Table 8.2: Evaluation Results of Our Entailment Based Reasoning Approach for WSC_{273} and WSC_{285} Along with Comparison with Other Systems

The evaluation results for the knowledge hunting and language models based reasoning approach as as shown in the Table 8.3 below. This approach also beats the previous state-of-the-art approach on two versions of WSC corpus (WSC_{273} and WSC_{285}).

8.2 Future Work: Multi-Hop Reasoning

Many problems in the Winograd Schema Challenge (WSC) corpus require more than one pieces of knowledge. We call the process of reasoning with more than one piece of knowledge as multi-hop reasoning.

Let us consider the following Winograd Schema Challenge problem.

Sentence: The dog chased the cat , the cat ran up a tree . **It**$_{pronoun}$ waited at the top .

Question: What waited at the top?

Answer Choices: *a) dog, b) cat*

Knowledge Needed:

1. *entity1 runs up a tree* ***may cause*** *entity1 is at top*
2. *entity1 is at top* ***may co-occur with*** *entity1 waits at top*

	#correct	**% Correct**
S2015	49	18.0
L2017	43	15.0
E2018	119	44.0
T2018 (WSC_{273})	174	63.70
T2018 (WSC_{285})	180	63.15
BERT Only (WSC_{273})	173	63.36
BERT Only (WSC_{285})	179	62.80
OUR_METHOD$_{T2018}$ (WSC_{273})	**189**	**69.23**
OUR_METHOD$_{T2018}$ (WSC_{285})	**195**	**68.42**
OUR_METHOD$_{BERT}$ (WSC_{273})	**194**	**71.06**
OUR_METHOD$_{BERT}$ (WSC_{285})	**200**	**70.17**

Table 8.3: Evaluation Results of Knowledge Hunting and Language Models Based Approach with Respect to WSC_{273} and WSC_{285}, Compared with Other Systems.

The above example shows that two knowledge pieces are needed to solve the problem. Similarly, any number (≥ 2) of such simple knowledge pieces can be required for a WSC problem.

Although multi-hop reasoning is not the main contribution of this work, as a stepping stone for future work, we developed a multi-hop reasoning procedure which takes a WSC problem and a list of required commonsense knowledge and produces the answer to the WSC problem. Below are the details of the procedure and a worked out example.

Multi-Hop Reasoning Procedure For WSC

Let us start by revisiting the above WSC example.

Sentence: The dog chased the cat , the cat ran up a tree . **It**$_{pronoun}$ waited at the top .

Question: What waited at the top?

Answer Choices: *a) dog, b) cat*

Knowledge Needed:

1. *entity1 runs up a tree* ***may cause*** *entity1 is at top*

2. *entity1 is at top* ***may co-occur with*** *entity1 waits at top*

As mentioned in the Chapter 4 the above two pieces of knowledge can also be written as,

1. ***IF*** *entity1 is at top because entity2 runs up a tree* ***THEN*** *entity1_1 is same as entity2_6*
2. ***IF*** *entity1 is at top and entity2 waits at the top* ***THEN*** *entity1_1 is same as entity2_6*

There are following main steps in the reasoning procedure.

1. The input WSC sentences and each of the knowledge in the list of knowledge are represented as graphs by using the same process as explained in the Chapter 4. The graphical representations of the WSC sentences and the knowledge pieces shown above are depicted below in the Figures 8.1, 8.2 and 8.3.

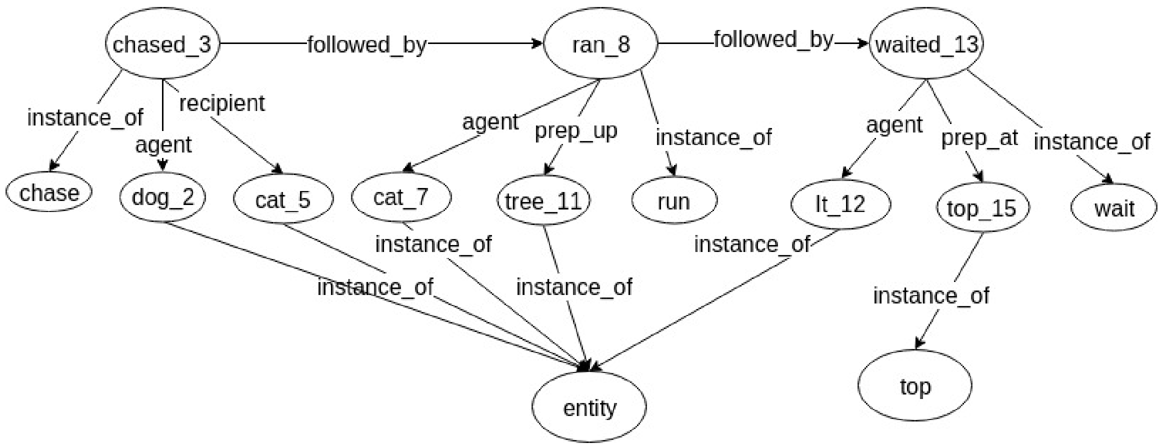


Figure 8.1: Graphical Representation of the WSC Sentences, *"The dog chased the cat , the cat ran up a tree . It waited at the top ."*

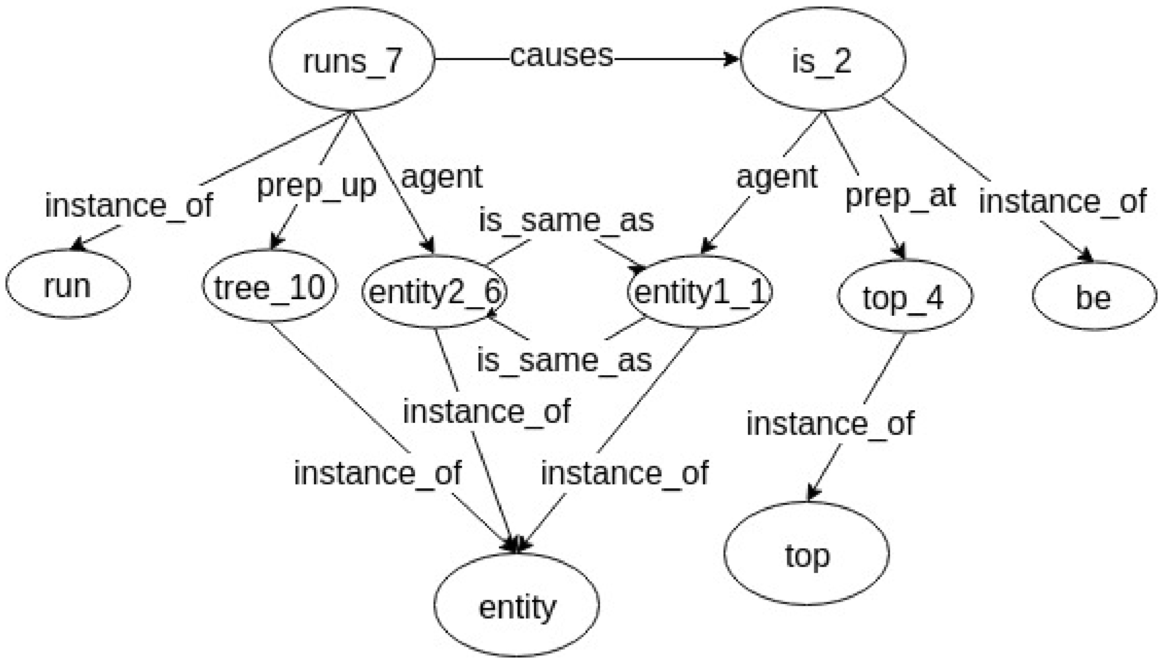


Figure 8.2: Graphical Representation of the Knowledge, *"IF entity1 is at top because entity2 runs up a tree THEN entity1_1 is same as entity2_6"*

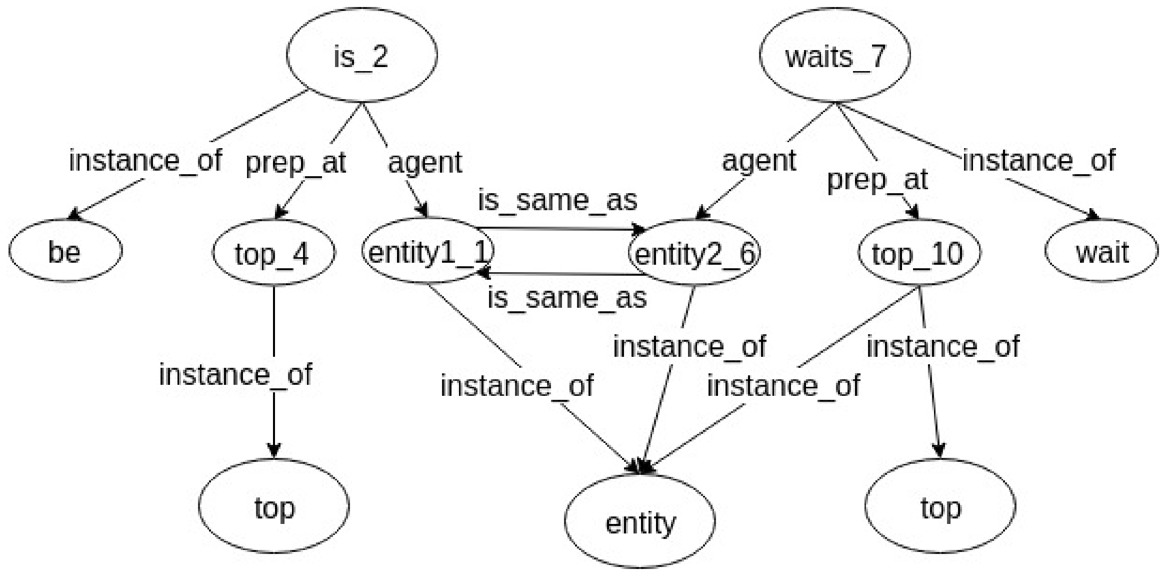


Figure 8.3: Graphical Representation of the Knowledge, *“**IF** entity1 is at top and entity2 waits at the top **THEN** entity1_1 is same as entity2_6”*

2. The graphical representation of the first knowledge in the list is merged with the representation of the input WSC sentences to obtain a merged representation graph. There are two steps in the merging operation.

   - **Step 1:** In this step all the nodes in the knowledge graph are identified which have a corresponding node in the graph of the WSC sentences. The correspondence between two nodes is determined if two nodes are instances of the same conceptual class and if they have same outgoing and incoming edges to and from already corresponding nodes. We used Answer Set Programming (ASP) to extract corresponding nodes from the two graphs. The corresponding nodes are extracted by using a binary predicate called *k_s_crossdomain_clone*. *k_s_crossdomain_clone(x,y)* represents that a node *x* in the representation of a knowledge corresponds to a node *y* in the representation of input WSC sentences. Following ASP rules are used to extract such nodes. Here, `has_s(X,R,Y)` represents an edge in WSC sentences' graph and `has_k(X,R,Y)` represents an edge in a knowledge graph.

```
% General rules to extract usful information from
% input graphs
k_val(X) :- has_k(X,R,Y).
k_val(Y) :- has_k(X,R,Y).

s_val(X) :- has_s(X,R,Y).
s_val(Y) :- has_s(X,R,Y).

s_const(X) :- has_s(X,"instance_of",I).
k_class(X) :- has_k(A,"instance_of",X).
k_const(X) :- not k_class(X), k_val(X).

s_has_par(X) :- has_s(P,R,X), s_const(X), s_const(P).
k_has_par(X) :- has_k(P,R,X), k_const(X), k_const(P).

s_has_child(X) :- has_s(X,R,C), s_const(X), s_const(C).
k_has_child(X) :- has_k(X,R,C), k_const(X), k_const(C).

% Rules to determine the cross representation nodes
% which belong to same conceptual class
not_k_s_crossdom_sib(X,Y) :- has_k(X,"instance_of",I1),
     has_k(X,"instance_of",I2),
     has_s(Y,"instance_of",I1),
     not has_s(Y,"instance_of",I2),
     I1!=I2.

not_k_s_crossdom_sib(X,Y) :- has_k(X,"instance_of",I1),
     has_k(X,"instance_of",I2),
     not has_s(Y,"instance_of",I1),
     has_s(Y,"instance_of",I2),
     I1!=I2.

not_k_s_crossdom_sib(X,Y) :- has_k(X,"instance_of",I1),
     has_k(X,"instance_of",I2),
     not has_s(Y,"instance_of",I1),
     not has_s(Y,"instance_of",I2),
     s_const(Y),
     I1!=I2.

k_s_crossdom_sib(X,Y) :- has_s(Y,"instance_of",I),
 has_k(X,"instance_of",I),
 not not_k_s_crossdom_sib(X,Y),
 s_const(Y), k_const(X).

% Rules to determine the nodes, across knowledge
% and WSC sentences' graphs, which correspond to
% each other
k_sibling_exists(X) :- k_s_crossdom_sib(X,Y),
   k_val(X), s_val(Y).
```

```
entity_class(person;object;entity).
k_entity(X) :- has_k(X,"instance_of",E), entity_class(E).
k_not_entity(X) :- k_const(X), not k_entity(X).

s_entity(X) :- has_s(X,"instance_of",E), entity_class(E).
s_not_entity(X) :- s_const(X), not s_entity(X).

k_not_have_sib(X) :- not k_sibling_exists(X), k_const(X).
k_child_not_have_sib(X) :- has_k(X,R1,C1),
       has_k(X,R2,C2),
       k_not_have_sib(C1),
       k_const(C1),
       k_const(C2).
all_child_have_sib(X) :- not k_child_not_have_sib(X),
   k_const(X).

k_s_crossdom_clone(X,Y) :- has_k(Pj,Rj,X),
       k_not_entity(X),
       has_s(Pj_prime,Rj_prime,Y),
       s_not_entity(Y),
       k_s_crossdom_sib(X,Y),
       not k_sibling_exists(Pj),
       k_const(Pj),k_const(X),
       s_const(Pj_prime),s_const(Y),
       Rj!="instance_of",
       Rj_prime!="instance_of".

k_s_crossdom_clone(X,Y) :- has_k(X,Rk,Cj),
       k_not_entity(X),
       has_s(Y,Rk_prime,Cj_prime),
       s_not_entity(Y),
       k_s_crossdom_sib(X,Y),
       not k_sibling_exists(Cj),
       k_const(Cj),k_const(X),
       s_const(Cj_prime),s_const(Y),
       Rk!="instance_of",
       Rk_prime!="instance_of".

k_s_crossdom_clone(X,Y) :- has_k(Pj,Rj,X),
   has_s(Pj_prime,Rj,Y),
   k_s_crossdom_sib(X,Y),
   k_s_crossdom_clone(Pj,Pj_prime),
   k_const(Pj),
   has_k(X,Rk,Cj),
   has_s(Y,Rk,Cj_prime),
   k_s_crossdom_sib(Cj,Cj_prime),
   k_const(Cj).

k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
   not k_has_par(X),
   has_k(X,Rk,Cj),
   has_s(Y,Rk,Cj_prime),
   k_s_crossdom_sib(Cj,Cj_prime),
   k_const(Cj).

k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
   has_k(Pj,Rj,X),
   has_s(Pj_prime,Rj,Y),
   k_s_crossdom_clone(Pj,Pj_prime),
   k_const(Pj),
   not k_has_child(X).

k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
   not k_has_par(X),
   not k_has_child(X).
```

- **Step 2:** In this step the corresponding nodes which were extracted in the previous step are used to merge the graphs of input WSC sentences and the input knowledge. The basic motivation behind merging is to add the knowledge graph nodes which do not have corresponding nodes in the WSC sentences' graph. Following set of ASP rules are used in this step. Here *has_m(X,R,Y)* represents an edge in the merged representation.

```
has_m(X,R,Y) :- has_s(X,R,Y).

k_covered(X) :- k_const(X),
    s_const(Y),
    k_s_crossdom_clone(X,Y).
```

```
k_not_covered(X) :- not k_covered(X), k_const(X).

k_not_all_covered :- not k_covered(X), k_const(X).

k_all_covered :- not k_not_all_covered.


k_covered_edge(X,R,Y) :- has_k(X,R,Y),
 R!="instance_of",
 k_s_crossdom_clone(X,X1),
 k_s_crossdom_clone(Y,Y1),
 has_s(X1,R,Y1),
 s_const(X1),
 s_const(Y1).
k_covered_edge(X,"instance_of",Y) :- has_k(X,"instance_of",Y),
k_s_crossdom_clone(X,X1),
has_s(X1,"instance_of",Y),
s_const(X1).
k_not_all_edges_covered :- has_k(X,R,Y),
not k_covered_edge(X,R,Y).
k_all_edges_covered :- not k_not_all_edges_covered.


has_m(X,R,Y) :- has_s(X,R,Y1),
    has_s(X2,R2,Y),
    Y1!=Y,
    k_s_crossdom_clone(Y_k,Y1),
    k_s_crossdom_clone(Y_k,Y),
    k_all_covered,
    k_all_edges_covered.
```

```
has_m(X,R,Y) :- has_s(X1,R,Y),

    has_s(X,R2,Y2),

    X1!=X,

    k_s_crossdom_clone(X_k,X1),

    k_s_crossdom_clone(X_k,X),

    k_all_covered,

    k_all_edges_covered.


has_m_candidate_right(N1S,R,N2,N2S) :- has_k(N1,R,N2),

k_covered(N1),

k_not_covered(N2),

k_s_crossdom_clone(N1,N1S),

coref(N2,N2K),

k_s_crossdom_clone(N2K,N2S).

has_m_candidate_right(N1S,R,N2,N2S) :- has_k(N1,R,N2),

k_covered(N1),

k_not_covered(N2),

k_s_crossdom_clone(N1,N1S),

coref(N2K,N2),

k_s_crossdom_clone(N2K,N2S).

has_m_candidate1_right(N1S,R,N2) :- has_k(N1,R,N2),

k_covered(N1),

k_not_covered(N2),

k_s_crossdom_clone(N1,N1S).
```

```
has_m(X,R,Y) :- has_m_candidate1_right(X,R,Y),
not has_m_candidate_right(X,R,Y,Y1),
s_const(Y1).
has_m(X,R,Y1) :- has_m_candidate_right(X,R,Y,Y1).

has_m(Y,"instance_of",YI) :- has_m_candidate1_right(X,R,Y),
not has_m_candidate_right(X,R,Y,Y1),
has_k(Y,"instance_of",YI),
s_const(Y1).

has_m_candidate_left(N1S,N1,R,N2S) :- has_k(N1,R,N2),
k_not_covered(N1),
k_covered(N2),
k_s_crossdom_clone(N2,N2S),
coref(N1,N1K),
k_s_crossdom_clone(N1K,N1S).
has_m_candidate_left(N1S,N1,R,N2S) :- has_k(N1,R,N2),
k_not_covered(N1),
k_covered(N2),
k_s_crossdom_clone(N2,N2S),
coref(N1K,N1),
k_s_crossdom_clone(N1K,N1S).
has_m_candidate1_left(N1,R,N2S) :- has_k(N1,R,N2),
k_not_covered(N1),
k_covered(N2),
k_s_crossdom_clone(N2,N2S).
```

```
has_m(X,R,Y) :- has_m_candidate1_left(X,R,Y),
    not has_m_candidate_left(X1,X,R,Y),
    s_const(X1).
has_m(X1,R,Y) :- has_m_candidate_left(X1,X,R,Y).

has_m(X,"instance_of",XI) :- has_m_candidate1_left(X,R,Y),
not has_m_candidate_left(X1,X,R,Y),
s_const(X1),
has_k(X,"instance_of",XI).

has_m_both(N1S,NI1,R,N2S,NI2) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
coref(N1,N1K),
coref(N2,N2K),
k_s_crossdom_clone(N2K,N2S),
k_s_crossdom_clone(N1K,N1S),
has_k(N1,"instance_of",NI1),
has_k(N2,"instance_of",NI2).

has_coref(N1) :- coref(N1,N2).
n_coref(N1) :- not has_coref(N1), k_const(N1).

has_m_both1(N1S,R,N2,NI2) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
coref(N1,N1K),
n_coref(N2),
k_s_crossdom_clone(N1K,N1S),
has_k(N2,"instance_of",NI2).

has_m_both2(N1,NI1,R,N2S) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
coref(N2,N2K),
n_coref(N1),
k_s_crossdom_clone(N2K,N2S),
has_k(N1,"instance_of",NI1).

has_m(N1,R,N2) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
n_coref(N1),
n_coref(N2).
has_m(N1,"instance_of",NI1) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
n_coref(N1),
n_coref(N2),
has_k(N1,"instance_of",NI1).
has_m(N2,"instance_of",NI2) :- has_k(N1,R,N2),
R!="instance_of",
k_not_covered(N1),
k_not_covered(N2),
n_coref(N1),
n_coref(N2),
has_k(N2,"instance_of",NI2).

has_m(N1S,R,N2S) :- has_m_both(N1S,NI1,R,N2S,NI2).
has_m(N1S,"instance_of",NI1) :-
has_m_both(N1S,NI1,R,N2S,NI2).
has_m(N2S,"instance_of",NI2) :-
has_m_both(N1S,NI1,R,N2S,NI2).

has_m(N1S,R,N2) :- has_m_both1(N1S,R,N2,NI2).
has_m(N2,"instance_of",NI2) :- has_m_both1(N1S,R,N2,NI2).

has_m(N1,R,N2S) :- has_m_both2(N1,NI1,R,N2S).
has_m(N1,"instance_of",NI1) :- has_m_both2(N1,NI1,R,N2S).
```

An Example of a merged representation generated by using the graphical representation of WSC sentences shown in the Figure 8.1 and the representation of a knowledge piece shown in is as shown in the Figure 8.2 is as shown in the Figure 8.4 below. The highlighted part in the Figure 8.4 represents the information that is added to the WSC sentences by using a piece of knowledge.

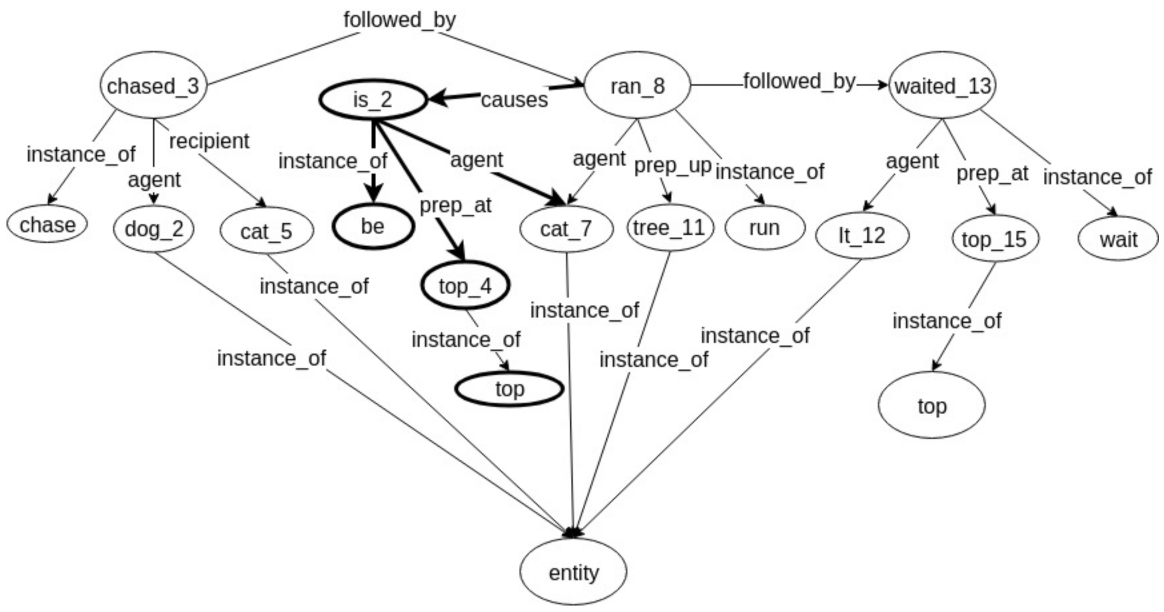


Figure 8.4: An Example of a Merged Representation

3. The output merged graph from the previous step is merged with the graph of the next knowledge in the list to generate another merged representation graph using the same merging procedure as explained above. This step is continued until only one knowledge in the list of knowledge pieces remains.

4. Finally, the *k_s_crossdomain_clones(X,Y)* with respect to the last knowledge in the list and the merged representation generated in the previous step are extracted and the following ASP rule is used to extract the final answer to the input WSC problem.

```
ans(A) :- k_s_crossdom_clone(X1,P),
          k_s_crossdom_clone(X2,A),
          pronoun(P),
          has_k(X1,"is_same_as",X2).
```

Here, `ans(A)` represents that `A` is the answer to the WSC problem and `pronoun(P)` represents that `P` is the pronoun to be resolved.

For example, the *k_s_crossdomain_clones(X,Y)* groundings generated from the representation in the Figure 8.4 and the knowledge shown in the Figure 8.3 are,

```
k_s_crossdom_clone("is_2","is_2")
k_s_crossdom_clone("top_4","top_4")
k_s_crossdom_clone("entity1_1","cat_7")
k_s_crossdom_clone("entity2_6","It_12")
k_s_crossdom_clone("waits_7","waited_13")
k_s_crossdom_clone("top_10","top_15")
```

As per the input question, the pronoun to resolve is *'It_2'*, i.e., `pronoun(It_2)`. So, the answer to the input problem is ***cat_7***.

REFERENCES

Abend, O. and A. Rappoport, “Universal conceptual cognitive annotation (ucca)”, in “Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, vol. 1, pp. 228–238 (2013).

Aharon, R. B., I. Szpektor and I. Dagan, “Generating entailment rules from framenet”, in “Proceedings of the ACL 2010 Conference Short Papers”, pp. 241–246 (Association for Computational Linguistics, 2010).

Allen, J., W. De Beaumont, L. Galescu, J. Orfan, M. Swift and C. M. Teng, “Automatically deriving event ontologies for a commonsense knowledge base”, in “Proceedings of the International Conference for Computational Semantics”, (2013).

Allen, J., M. Manshadi, M. Dzikovska and M. Swift, “Deep linguistic processing for spoken dialogue systems”, in “Proceedings of the Workshop on Deep Linguistic Processing”, pp. 49–56 (ACL, 2007).

Allen, J. F., M. Swift and W. de Beaumont, “Deep semantic analysis of text”, in “Proceedings of the 2008 Conference on Semantics in Text Processing”, pp. 343–354 (Association for Computational Linguistics, 2008).

Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick and D. Parikh, “Vqa: Visual question answering”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 2425–2433 (2015).

Artzi, Y. and L. Zettlemoyer, “Bootstrapping semantic parsers from conversations”, in “Proceedings of the conference on empirical methods in natural language processing”, pp. 421–432 (Association for Computational Linguistics, 2011).

Artzi, Y. and L. Zettlemoyer, “Weakly supervised learning of semantic parsers for mapping instructions to actions”, Transactions of the Association of Computational Linguistics **1**, 49–62 (2013).

Bailey, D., A. Harrison, Y. Lierler, V. Lifschitz and J. Michael, “The winograd schema challenge and reasoning about correlation”, in “In Working Notes of the Symposium on Logical Formalizations of Commonsense Reasoning”, (2015).

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer and N. Schneider, “Abstract meaning representation (amr) 1.0 specification”, in “Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on EMNLP. Seattle: ACL”, pp. 1533–1544 (2012).

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer and N. Schneider, “Abstract meaning representation for sembanking”, (2013).

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

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 (2007).

Berant, J., A. Chou, R. Frostig and P. Liang, “Semantic parsing on freebase from question-answer pairs.”, in “EMNLP”, pp. 1533–1544 (2013).

Betteridge, J., A. Carlson, S. A. Hong, E. R. Hruschka Jr, E. L. Law, T. M. Mitchell and S. H. Wang, “Toward never ending language learning.”, in “AAAI Spring Symposium: Learning by Reading and Learning to Read”, pp. 1–2 (2009).

Bollacker, K., C. Evans, P. Paritosh, T. Sturge and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge”, in “Proceedings of the 2008 ACM SIGMOD international conference on Management of data”, pp. 1247–1250 (AcM, 2008).

Bos, J., “Introduction to the shared task on comparing semantic representations”, in “Proceedings of the 2008 Conference on Semantics in Text Processing”, pp. 257–261 (ACL, 2008a).

Bos, J., “Wide-coverage semantic analysis with boxer”, in “Proceedings of the 2008 Conference on Semantics in Text Processing”, pp. 277–286 (Association for Computational Linguistics, 2008b).

Bowman, S. R., G. Angeli, C. Potts and C. D. Manning, “A large annotated corpus for learning natural language inference”, arXiv preprint arXiv:1508.05326 (2015).

Callaway, C. B., “The textcap semantic interpreter”, in “Proceedings of the 2008 Conference on Semantics in Text Processing”, pp. 327–342 (ACL, 2008).

Cambria, E. and N. Howard, “Common and common-sense knowledge integration for concept-level sentiment analysis.”, in “FLAIRS Conference”, (2014).

Carbonell, J. F. S. T. J. and C. D. N. A. Smith, “A discriminative graph-based parser for the abstract meaning representation”, (2014).

Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr and T. M. Mitchell, “Toward an architecture for never-ending language learning.”, in “AAAI”, vol. 5, p. 3 (Atlanta, 2010).

Carpenter, B., *Type-logical semantics* (MIT press, 1997).

Chambers, N. and D. Jurafsky, “Unsupervised learning of narrative event chains.”, in “ACL”, vol. 94305, pp. 789–797 (Citeseer, 2008).

Chambers, N. and D. Jurafsky, “Unsupervised learning of narrative schemas and their participants”, in “Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2”, pp. 602–610 (ACL, 2009).

Chang, M.-W., “From entity linking to question answering recent progress on semantic grounding tasks”, WNUT 2016 p. 2 (2016).

Chen, Q., X. Zhu, Z. Ling, S. Wei, H. Jiang and D. Inkpen, “Enhanced lstm for natural language inference”, arXiv preprint arXiv:1609.06038 (2016).

Chen, Y., H. Li and Z. Xu, “Convolutional neural network-based question answering over knowledge base with type constraint”, in “China Conference on Knowledge Graph and Semantic Computing”, pp. 28–39 (Springer, 2018).

Clark, C. and M. Gardner, “Simple and effective multi-paragraph reading comprehension”, arXiv preprint arXiv:1710.10723 (2017).

Clark, P., “Elementary school science and math tests as a driver for ai: take the aristo challenge!”, in “AAAI”, pp. 4019–4021 (2015).

Clark, P., I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick and O. Tafjord, “Think you have solved question answering? try arc, the ai2 reasoning challenge”, arXiv preprint arXiv:1803.05457 (2018).

Clark, P. and O. Etzioni, “My computer is an honor studentbut how intelligent is it? standardized tests as a measure of ai”, AI Magazine **37**, 1, 5–12 (2016).

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).

Cordella, L. P., P. Foggia, C. Sansone and M. Vento, “A (sub) graph isomorphism algorithm for matching large graphs”, IEEE transactions on pattern analysis and machine intelligence **26**, 10, 1367–1372 (2004).

Dagan, I., O. Glickman and B. Magnini, “The pascal recognising textual entailment challenge”, in “Machine Learning Challenges Workshop”, pp. 177–190 (Springer, 2005).

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).

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

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, arXiv preprint arXiv:1810.04805 (2018).

Emami, A., N. De La Cruz, A. Trischler, K. Suleman and J. C. K. Cheung, “A knowledge hunting framework for common sense reasoning”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 1949–1958 (2018).

FitzGerald, N., J. Michael, L. He and L. Zettlemoyer, “Large-scale qa-srl parsing”, arXiv preprint arXiv:1805.05377 (2018).

Flanigan, J., C. Dyer, N. A. Smith and J. Carbonell, "Generation from abstract meaning representation using tree transducers", in "Proceedings of NAACL-HLT", pp. 731–739 (2016).

Forbes, M. and Y. Choi, "Verb physics: Relative physical knowledge of actions and objects", arXiv preprint arXiv:1706.03799 (2017).

Gardner, M., J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform", arXiv preprint arXiv:1803.07640 (2018).

Gebser, M., B. Kaufmann, A. Neumann and T. Schaub, "clasp : A Conflict-Driven Answer Set Solver.", in "LPNMR", edited by C. Baral, G. Brewka and J. S. Schlipf, vol. 4483 of *Lecture Notes in Computer Science*, pp. 260–265 (Springer, 2007), URL `http://dblp.uni-trier.de/db/conf/lpnmr/lpnmr2007.html#GebserKNS07a`.

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

Glavaš, G. and J. Šnajder, "Constructing coherent event hierarchies from news stories", TextGraphs-9 p. 34 (2014).

Glavaš, G., J. Šnajder, P. Kordjamshidi and M.-F. Moens, "Hieve: A corpus for extracting event hierarchies from news stories", in "Proceedings of 9th Language Resources and Evaluation Conference (LREC 2014)", pp. 3678–3683 (2014).

Gordon, J. and B. Van Durme, "Reporting bias and knowledge acquisition", in "Proceedings of the 2013 workshop on Automated knowledge base construction", pp. 25–30 (ACM, 2013).

Gordon, J., B. Van Durme and L. K. Schubert, "Evaluation of commonsense knowledge with mechanical turk", in "Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk", pp. 159–162 (Association for Computational Linguistics, 2010).

Guo, Z. and W. Lu, "Better transition-based amr parsing with a refined search space", in "Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing", pp. 1712–1722 (2018).

Hajishirzi, H., J. Hockenmaier, E. T. Mueller and E. Amir, "Reasoning about robocup soccer narratives", arXiv preprint arXiv:1202.3728 (2012a).

Hajishirzi, H., E. T. Mueller and E. Amir, "Symbolic probabilistic reasoning for narratives.", in "AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning", (2011).

Hajishirzi, H., M. Rastegari, A. Farhadi and J. K. Hodgins, "Semantic understanding of professional soccer commentaries", arXiv preprint arXiv:1210.4854 (2012b).

He, L., M. Lewis and L. Zettlemoyer, “Question-answer driven semantic role labeling: Using natural language to annotate natural language”, in “Proceedings of the 2015 conference on empirical methods in natural language processing”, pp. 643–653 (2015).

Hendrix, G. G., E. D. Sacerdoti, D. Sagalowicz and J. Slocum, “Developing a natural language interface to complex data”, ACM Transactions on Database Systems (TODS) **3**, 2, 105–147 (1978).

Huang, H.-Y., C. Zhu, Y. Shen and W. Chen, “Fusionnet: Fusing via fully-aware attention with application to machine comprehension”, arXiv preprint arXiv:1711.07341 (2017).

Isaak, N. and L. Michael, “Tackling the winograd schema challenge through machine logical inferences.”, in “STAIRS”, vol. 284, pp. 75–86 (2016).

Johnson, T., “Natural language computing: the commercial applications”, The Knowledge Engineering Review **1**, 3, 11–23 (1984).

Joshi, M., E. Choi, D. S. Weld and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension”, arXiv preprint arXiv:1705.03551 (2017).

Kimmig, A., S. H. Bach, M. Broecheler, B. Huang and L. Getoor, “A short introduction to probabilistic soft logic”, in “NIPS Workshop on probabilistic programming: Foundations and applications”, vol. 1, p. 3 (2012).

Kollar, T., D. Berry, L. Stuart, K. Owczarzak, T. Chung, L. Mathias, M. Kayser, B. Snow and S. Matsoukas, “The alexa meaning representation language”, in “Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)”, vol. 3, pp. 177–184 (2018).

Krishnamurthy, J., P. Dasigi and M. Gardner, “Neural semantic parsing with type constraints for semi-structured tables”, in “Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing”, pp. 1516–1526 (2017).

Leacock, C. and M. Chodorow, “Combining local context and wordnet similarity for word sense identification”, WordNet: An electronic lexical database **49**, 2, 265–283 (1998).

Lenat, D. B., “Cyc: A large-scale investment in knowledge infrastructure”, Communications of the ACM **38**, 11, 33–38 (1995).

Leone, N., G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri and F. Scarcello, “The DLV system for knowledge representation and reasoning”, ACM Transactions on Computational Logic (TOCL) **7**, 3, 499–562 (2006).

Levesque, H. J., E. Davis and L. Morgenstern, “The winograd schema challenge.”, in “Aaai spring symposium: Logical formalizations of commonsense reasoning”, vol. 46, p. 47 (2011).

Liang, P., “Lambda dependency-based compositional semantics”, arXiv preprint arXiv:1309.4408 (2013).

Liang, P., “Learning executable semantic parsers for natural language understanding”, Communications of the ACM **59**, 9, 68–76 (2016).

Lin, T., O. Etzioni and J. Fogarty, “Identifying interesting assertions from the web.”, in “CIKM”, edited by D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu and J. J. Lin, pp. 1787–1790 (ACM, 2009), URL `http://dblp.uni-trier.de/db/conf/cikm/cikm2009.html#LinEF09`.

Lin, T., Mausam and O. Etzioni, “Commonsense from the web: Relation properties.”, in “AAAI Fall Symposium: Commonsense Knowledge”, vol. FS-10-02 of *AAAI Technical Report* (AAAI, 2010), URL `http://dblp.uni-trier.de/db/conf/aaaifs/aaaifs2010-02.html#LinME10`.

Lin, T., Mausam and O. Etzioni, “Entity linking at web scale”, in “Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction”, AKBC-WEKEX ’12, pp. 84–88 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2012), URL `http://dl.acm.org/citation.cfm?id=2391200.2391216`.

Lin, Z., H. T. Ng and M.-Y. Kan, “A pdtb-styled end-to-end discourse parser”, Natural Language Engineering pp. 1–34 (2014).

Litkowski, K., “The preposition project corpora”, Tech. rep., Technical Report 13-01. Damascus, MD: CL Research (2013).

Liu, C.-W., R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin and J. Pineau, “How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation”, arXiv preprint arXiv:1603.08023 (2016a).

Liu, H. and P. Singh, “Conceptneta practical commonsense reasoning tool-kit”, BT technology journal **22**, 4, 211–226 (2004).

Liu, Q., H. Jiang, A. Evdokimov, Z.-H. Ling, X. Zhu, S. Wei and Y. Hu, “Cause-effect knowledge acquisition and neural association model for solving a set of winograd schema problems”, in “Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)”, pp. 2344–2350 (2017).

Liu, Q., H. Jiang, Z.-H. Ling, X.-D. Zhu, S. Wei and Y. Hu, “Combing context and commonsense knowledge through neural networks for solving winograd schema problems”, CoRR, abs/1611.04146 (2016b).

Liu, X., P. He, W. Chen and J. Gao, “Multi-task deep neural networks for natural language understanding”, arXiv preprint arXiv:1901.11504 (2019).

Lyu, C. and I. Titov, “Amr parsing as graph prediction with latent alignment”, arXiv preprint arXiv:1805.05286 (2018).

MacCartney, B. and C. D. Manning, *Natural language inference* (Stanford University Stanford, 2009).

Matthiessen, C. and J. A. Bateman, *Text generation and systemic-functional linguistics: experiences from English and Japanese* (Pinter Publishers, 1991).

Mihaylov, T., P. Clark, T. Khot and A. Sabharwal, “Can a suit of armor conduct electricity? a new dataset for open book question answering”, arXiv preprint arXiv:1809.02789 (2018).

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

Minsky, M., *Society of mind* (Simon and Schuster, 1988).

Mishra, B. D., L. Huang, N. Tandon, W.-t. Yih and P. Clark, “Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension”, arXiv preprint arXiv:1805.06975 (2018).

Niemelä, I. and P. Simons, “Smodelsan implementation of the stable model and well-founded semantics for normal logic programs”, in “International Conference on Logic Programming and NonMonotonic Reasoning”, pp. 420–429 (Springer, 1997).

Oepen, S., M. Kuhlmann, Y. Miyao, D. Zeman, S. Cinková, D. Flickinger, J. Hajic and Z. Uresova, “Semeval 2015 task 18: Broad-coverage semantic dependency parsing”, in “Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)”, pp. 915–926 (2015).

Ouyang, J. and K. McKeown, “Towards automatic detection of narrative structure”, in “Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC14)(Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., eds.),(Reykjavik, Iceland), European Language Resources Association (ELRA)”, vol. 2 (2014).

Palmer, M., D. Gildea and P. Kingsbury, “The proposition bank: An annotated corpus of semantic roles”, Computational linguistics **31**, 1, 71–106 (2005).

Palmer, M., D. Gildea and N. Xue, “Semantic role labeling”, Synthesis Lectures on Human Language Technologies **3**, 1, 1–103 (2010).

Parikh, A. P., O. Täckström, D. Das and J. Uszkoreit, “A decomposable attention model for natural language inference”, arXiv preprint arXiv:1606.01933 (2016).

Pasupat, P. and P. Liang, “Compositional semantic parsing on semi-structured tables”, arXiv preprint arXiv:1508.00305 (2015).

Poon, H., “Grounded unsupervised semantic parsing”, in “Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, vol. 1, pp. 933–943 (2013).

Prakash, A., A. Sharma, A. Mitra and C. Baral, “Combining knowledge hunting and neural language models to solve the winograd schema challenge”, in “Proceedings of the ACL 2019”, (Association for Computational Linguistics, 2019).

Pustejovsky, J., “The syntax of event structure”, Cognition **41**, 1, 47–81 (1991).

Rajpurkar, P., R. Jia and P. Liang, “Know what you don’t know: Unanswerable questions for squad”, arXiv preprint arXiv:1806.03822 (2018).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, “Squad: 100,000+ questions for machine comprehension of text”, arXiv preprint arXiv:1606.05250 (2016).

Rao, S., D. Marcu, K. Knight and H. Daumé III, “Biomedical event extraction using abstract meaning representation”, BioNLP 2017 pp. 126–135 (2017).

Rashkin, H., M. Sap, E. Allaway, N. A. Smith and Y. Choi, “Event2mind: Commonsense inference on events, intents, and reactions”, arXiv preprint arXiv:1805.06939 (2018).

Ritter, A., Mausam, O. Etzioni and S. Clark, “Open domain event extraction from twitter”, in “Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining”, KDD ’12, pp. 1104–1112 (ACM, New York, NY, USA, 2012), URL `http://doi.acm.org/10.1145/2339530.2339704`.

Roemmele, M., C. A. Bejan and A. S. Gordon, “Choice of plausible alternatives: An evaluation of commonsense causal reasoning.”, in “AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning”, pp. 90–95 (2011).

Sap, M., R. LeBras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith and Y. Choi, “Atomic: An atlas of machine commonsense for if-then reasoning”, arXiv preprint arXiv:1811.00146 (2018).

Schüller, P., “Tackling winograd schemas by formalizing relevance theory in knowledge graphs”, in “Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning”, (2014).

Sharma, A., *Solving winograd schema challenge: Using semantic parsing, automatic knowledge acquisition and logical reasoning* (Arizona State University, 2014).

Sharma, A. and C. Baral, “Automatic extraction of events-based conditional commonsense knowledge”, in “Proceedings of the 3rd Workshop on Knowledge Extraction from Text at the AAAI”, (2016).

Sharma, A., N. V. Ha, S. Aditya and C. Baral, “Towards addressing the winograd schema challenge - building and using a semantic parser and a knowledge hunting module.”, in “Proceedings of Twenty-Fourth International Joint Conference on Artificial Intelligence”, (AAAI, 2015a).

Sharma, A., N. H. Vo, S. Aditya and C. Baral, “Identifying various kinds of event mentions in k-parser output”, in “Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT”, pp. 82–88 (2015b).

Sharma, A., N. H. Vo, S. Aditya and C. Baral, “Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module.”, (2015c).

Sharma, A., N. H. Vo, S. Gaur and C. Baral, “An approach to solve winograd schema challenge using automatically extracted commonsense knowledge”, in “2015 AAAI Spring Symposium Series”, (2015d).

Singh, P., T. Lin, E. T. Mueller, G. Lim, T. Perkins and W. L. Zhu, “Open mind common sense: Knowledge acquisition from the general public”, in “OTM Confederated International Conferences” On the Move to Meaningful Internet Systems””, pp. 1223–1237 (Springer, 2002).

Singhal, A., “Introducing the knowledge graph: things, not strings”, Official Google Blog, May (2012).

Swanson, R., E. Rahimtoroghi, T. Corcoran and M. A. Walker, “Identifying narrative clause types in personal stories”, in “Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL),(Philadelphia, PA, USA)”, pp. 171–180 (2014).

Tandon, N., G. de Melo, F. Suchanek and G. Weikum, “Webchild: harvesting and organizing commonsense knowledge from the web”, in “Proceedings of the 7th ACM international conference on Web search and data mining”, pp. 523–532 (ACM, 2014).

Tandon, N., G. De Melo and G. Weikum, “Deriving a web-scale common sense fact database.”, (2011).

Tandon, N., G. de Melo and G. Weikum, “Webchild 2.0: fine-grained commonsense knowledge distillation”, Proceedings of ACL 2017, System Demonstrations pp. 115–120 (2017).

Templeton, M. and J. Burger, “Problems in natural-language interface to dbms with examples from eufid”, in “Proceedings of the first conference on Applied natural language processing”, pp. 3–16 (Association for Computational Linguistics, 1983).

Trinh, T. H. and Q. V. Le, “A simple method for commonsense reasoning”, arXiv preprint arXiv:1806.02847 (2018).

Trischler, A., T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman and K. Suleman, “Newsqa: A machine comprehension dataset”, arXiv preprint arXiv:1611.09830 (2016).

Turing, A. M., “Computing machinery and intelligence”, Mind **59**, 236, 433–460 (1950).

Van Durme, B. and L. Schubert, “Open knowledge extraction through compositional language processing”, in “Proceedings of the 2008 Conference on Semantics in Text Processing”, pp. 239–254 (Association for Computational Linguistics, 2008).

Wang, C., S. Pradhan, N. Xue, X. Pan and H. Ji, “Camr at semeval-2016 task 8: An extended transition-based amr parser”, Proceedings of SemEval pp. 1173–1178 (2016).

Wang, W., M. Yan and C. Wu, “Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering”, arXiv preprint arXiv:1811.11934 (2018).

Weissenborn, D., G. Wiese and L. Seiffe, "Making neural qa as simple as possible but not simpler", arXiv preprint arXiv:1703.04816 (2017).

Weston, J., A. Bordes, S. Chopra and T. Mikolov, "Towards ai-complete question answering: a set of prerequisite toy tasks", arXiv preprint arXiv:1502.05698 (2015).

Woods, W. A., "Progress in natural language understanding: an application to lunar geology", in "Proceedings of the June 4-8, 1973, national computer conference and exposition", pp. 441–450 (ACM, 1973).

Xu, K., Y. Feng, S. Reddy, S. Huang and D. Zhao, "Enhancing freebase question answering using textual evidence", arXiv preprint arXiv:1603.00957 (2016).

Zettlemoyer, L. S. and M. Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars", arXiv preprint arXiv:1207.1420 (2012).

APPENDIX A

APPENDIX TO CHAPTER 4

A.1 Proof of Theorem 1

The proof of Theorem 1 is done using a set of lemmas. In this sections we present those lemmas and then use them to prove Theorem 1.

Lemma 1 *Let $\mathcal{G}_\mathcal{S} = (\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$ be a graphical representation of the sequence of sentences in a WSC problem. Then, Step 1 of the WiSCR Algorithm extracts a subgraph $\mathcal{G}'_\mathcal{S}$ of $\mathcal{G}_\mathcal{S}$ such that $\mathcal{G}'_\mathcal{S} = (\mathbb{V}'_\mathcal{S}, \mathbb{E}'_\mathcal{S}, f'_\mathcal{S})$ where $\mathbb{V}'_\mathcal{S} = \mathbb{V}_\mathcal{S} - \mathbb{V}^c_\mathcal{S}$, $\mathbb{V}^c_\mathcal{S}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{S}$, $f'_\mathcal{S} = f_\mathcal{S}$, $\mathbb{E}'_\mathcal{S} = \mathbb{E}_\mathcal{S} - \mathbb{E}^c_\mathcal{S}$, and $e \in \mathbb{E}^c_\mathcal{S}$ if $f(e) = instance\_of$.*

Proof 3 *According to the Step 1 of the WiSCR algorithm, given a graph $\mathcal{G}_\mathcal{S} = (\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$, a subgraph of it is extracted. Let $\mathcal{G}'_\mathcal{S} = (\mathbb{V}'_\mathcal{S}, \mathbb{E}'_\mathcal{S}, f'_\mathcal{S})$ be the extracted subgraph. $\mathbb{V}'_\mathcal{S}$ contains all the nodes from $\mathcal{G}_\mathcal{S}$ which are not class nodes, i.e., $\mathbb{V}'_\mathcal{S} = \mathbb{V}_\mathcal{S} - \mathbb{V}^c_\mathcal{S}$, $\mathbb{V}^c_\mathcal{S}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{S}$. Also, $\mathbb{E}'_\mathcal{S}$ contains all the edges between the nodes in $\mathbb{V}'_\mathcal{S}$. So, by Definition 4 $\mathbb{E}'_\mathcal{S} = \mathbb{E}_\mathcal{S} - \mathbb{E}^c_\mathcal{S}$ where $e \in \mathbb{E}^c_\mathcal{S}$ if $f(e) = instance\_of$. Furthermore, no new edges or nodes are added to $\mathcal{G}'_\mathcal{S}$ so $f'_\mathcal{S}$ = $f_\mathcal{S}$.*

Hence, the step 1 of the WiSCR Algorithm extract a subgraph $\mathcal{G}'_\mathcal{S}$ from $\mathcal{G}_\mathcal{S}$ such that if $\mathcal{G}_\mathcal{S} = (\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$ then $\mathcal{G}'_\mathcal{S} = (\mathbb{V}'_\mathcal{S}, \mathbb{E}'_\mathcal{S}, f'_\mathcal{S})$ where $\mathbb{V}'_\mathcal{S} = \mathbb{V}_\mathcal{S} - \mathbb{V}^c_\mathcal{S}$, $\mathbb{V}^c_\mathcal{S}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{S}$, $f'_\mathcal{S} = f_\mathcal{S}$, $\mathbb{E}'_\mathcal{S} = \mathbb{E}_\mathcal{S} - \mathbb{E}^c_\mathcal{S}$, and $e \in \mathbb{E}^c_\mathcal{S}$ iff $f(e) = instance\_of$.

Lemma 2 *Let $\mathcal{G}_\mathcal{K} = (\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$ be a graphical representation of a knowledge (By Definition 6). Then, Step 2 of the WiSCR Algorithm extracts a subgraph $\mathcal{G}'_\mathcal{K}$ from $\mathcal{G}_\mathcal{K}$ such that $\mathcal{G}'_\mathcal{K} = (\mathbb{V}'_\mathcal{K}, \mathbb{E}'_\mathcal{K}, f'_\mathcal{K})$ where $\mathbb{V}'_\mathcal{K} = \mathbb{V}_\mathcal{K} - \mathbb{V}^c_\mathcal{K}$, $\mathbb{V}^c_\mathcal{K}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{K}$, $f'_\mathcal{K} = f_\mathcal{K}$, $\mathbb{E}'_\mathcal{K} = \mathbb{E}_\mathcal{K} - \mathbb{E}^c_\mathcal{K}$, and $e \in \mathbb{E}^c_\mathcal{K}$ if $f(e) \in \{instance\_of, is\_same\_as\}$.*

Proof 4 *According to the Step 2 of the WiSCR algorithm, given a graphical representation of a knowledge $\mathcal{G}_\mathcal{K} = (\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$, a subgraph of it is extracted. Let $\mathcal{G}'_\mathcal{K} = (\mathbb{V}'_\mathcal{K}, \mathbb{E}'_\mathcal{K}, f'_\mathcal{K})$ be the extracted subgraph. $\mathbb{V}'_\mathcal{K}$ contains all the nodes from $\mathcal{G}_\mathcal{K}$ which are not class nodes, i.e., $\mathbb{V}'_\mathcal{K} = \mathbb{V}_\mathcal{K} - \mathbb{V}^c_\mathcal{K}$, $\mathbb{V}^c_\mathcal{K}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{K}$. Also, $\mathbb{E}'_\mathcal{K}$ contains all the edges between the nodes in $\mathbb{V}'_\mathcal{K}$ except the ones labeled as 'is_same_as'. So, by Definition 6 $\mathbb{E}'_\mathcal{K} = \mathbb{E}_\mathcal{K} - \mathbb{E}^c_\mathcal{K}$, and $e \in \mathbb{E}^c_\mathcal{K}$ if $f(e) \in \{instance\_of, is\_same\_as\}$. Furthermore, no new edges or nodes are added to $\mathcal{G}'_\mathcal{K}$ so $f'_\mathcal{K}$ = $f_\mathcal{K}$.*

Hence, the step 2 of the WiSCR Algorithm extract a subgraph $\mathcal{G}'_\mathcal{K}$ from $\mathcal{G}_\mathcal{K}$ such that if $\mathcal{G}_\mathcal{K} = (\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$ then $\mathcal{G}'_\mathcal{K} = (\mathbb{V}'_\mathcal{K}, \mathbb{E}'_\mathcal{K}, f'_\mathcal{K})$ where $\mathbb{V}'_\mathcal{K} = \mathbb{V}_\mathcal{K} - \mathbb{V}^c_\mathcal{K}$, $\mathbb{V}^c_\mathcal{K}$ is a set of all the class nodes in $\mathcal{G}_\mathcal{K}$, $f'_\mathcal{K} = f_\mathcal{K}$, $\mathbb{E}'_\mathcal{K} = \mathbb{E}_\mathcal{K} - \mathbb{E}^c_\mathcal{K}$, and $e \in \mathbb{E}^c_\mathcal{K}$ if $f(e) \in \{instance\_of, is\_same\_as\}$

Lemma 3 *Let $\mathcal{G}_\mathcal{S}$ = $(\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$ be a graphical representation of a sequence of sentences in a WSC problem, $\mathcal{G}'_\mathcal{S}$ = $(\mathbb{V}'_\mathcal{S}, \mathbb{E}'_\mathcal{S}, f'_\mathcal{S})$ be a subgraph of $\mathcal{G}_\mathcal{S}$ such that $\mathbb{V}'_\mathcal{S} = \mathbb{V}_\mathcal{S} - \mathbb{V}^c_\mathcal{S}$ where $\mathbb{V}^c_\mathcal{S}$ is the set of all the class nodes in $\mathcal{G}_\mathcal{S}$, $f'_\mathcal{S} = f_\mathcal{S}$ and $\mathbb{E}'_\mathcal{S} = \mathbb{E}_\mathcal{S} - \mathbb{E}^c_\mathcal{S}$ where $e \in \mathbb{E}^c_\mathcal{S}$ iff $f_\mathcal{S}(e)$ = "instance_of". Let $\mathcal{G}_\mathcal{K}$ = $(\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$ be a graphical representation of a knowledge where $f_\mathcal{K}$ is defined using $f_\mathcal{S}$, $\mathcal{G}'_\mathcal{K}$ = $(\mathbb{V}'_\mathcal{K}, \mathbb{E}'_\mathcal{K}, f'_\mathcal{K})$ be a subgraph of $\mathcal{G}_\mathcal{K}$ such that $\mathbb{V}'_\mathcal{K} = \mathbb{V}_\mathcal{K} - \mathbb{V}^c_\mathcal{K}$ where $\mathbb{V}^c_\mathcal{K}$ is the set of all the class nodes in $\mathcal{G}_\mathcal{K}$, $f'_\mathcal{K} = f_\mathcal{K}$ and $\mathbb{E}'_\mathcal{K} = \mathbb{E}_\mathcal{K} - \mathbb{E}^c_\mathcal{K}$ where $e \in \mathbb{E}^c_\mathcal{K}$ iff $f_\mathcal{K}(e) \in \{is\_same\_as, instance\_of\}$. Then, Step 3 of the WiSCR algorithm extracts all possible sets of node pairs of the form (a,b) such that either there does not exist such a non-empty set or if $\mathbb{M}_i$ is one such non-empty set then,*

- *for each $(a,b) \in \mathbb{M}_i$, $a \in \mathbb{V}'_\mathcal{S}$ and $b \in \mathbb{V}'_\mathcal{K}$,*

- *for each* $(a,b) \in \mathbb{M}_i$*, a and b are instances of same class, i.e.,* $(a,i) \in \mathbb{E}_{\mathcal{S}}$*,* $(b,i) \in \mathbb{E}_{\mathcal{K}}$*,* $f_{\mathcal{S}}((a,i)) = instance\_of$ *and* $f_{\mathcal{K}}((b,i)) = instance\_of$
- *if for every pair (a,b)*$\in \mathbb{M}_i$*, a is replaced by b in* $\mathbb{V}'_{\mathcal{S}}$ *then* $\mathcal{G}'_{\mathcal{K}}$ *becomes a subgraph of the node-replaced* $\mathcal{G}'_{\mathcal{S}}$

Proof 5 *(i) Given a graphical representation of the sentences in a WSC problem (say* $\mathcal{G}_{\mathcal{S}}$ = $(\mathbb{V}_{\mathcal{S}},\mathbb{E}_{\mathcal{S}},f_{\mathcal{S}})$*) and Lemma 1, the Step 1 of the WiSCR algorithm produces a subgraph of* $\mathcal{G}_{\mathcal{S}}$ *(say* $\mathcal{G}'_{\mathcal{S}}$ = $(\mathbb{V}'_{\mathcal{S}},\mathbb{E}'_{\mathcal{S}},f'_{\mathcal{S}})$*) such that* $\mathbb{V}'_{\mathcal{S}} = \mathbb{V}_{\mathcal{S}} - \mathbb{V}^c_{\mathcal{S}}$ *where* $\mathbb{V}^c_{\mathcal{S}}$ *is a set of all the class nodes in* $\mathcal{G}_{\mathcal{S}}$*,* $f'_{\mathcal{S}} = f_{\mathcal{S}}$ *and* $\mathbb{E}'_{\mathcal{S}} = \mathbb{E}_{\mathcal{S}} - \mathbb{E}^c_{\mathcal{S}}$ *where* $e \in \mathbb{E}^c_{\mathcal{S}}$ *if* $f_{\mathcal{S}}(e) = instance\_of$*.*

(ii) Given a graphical representation of a knowledge (say $\mathcal{G}_{\mathcal{K}}$ = $(\mathbb{V}_{\mathcal{K}},\mathbb{E}_{\mathcal{K}},f_{\mathcal{K}})$*) and 2, the step 2 of the WiSCR algorithm produces a subgraph of* $\mathcal{G}_{\mathcal{K}}$ *(say* $\mathcal{G}'_{\mathcal{K}}$ = $(\mathbb{V}'_{\mathcal{K}},\mathbb{E}'_{\mathcal{K}},f'_{\mathcal{K}})$*) such that* $\mathbb{V}'_{\mathcal{K}} = \mathbb{V}_{\mathcal{K}} - \mathbb{V}^c_{\mathcal{K}}$ *where* $\mathbb{V}^c_{\mathcal{K}}$ *is a set of all the class nodes in* $\mathcal{G}_{\mathcal{K}}$*,* $f'_{\mathcal{K}} = f_{\mathcal{K}}$ *and* $\mathbb{E}'_{\mathcal{K}} = \mathbb{E}_{\mathcal{K}} - \mathbb{E}^c_{\mathcal{K}}$ *where* $e \in \mathbb{E}^c_{\mathcal{K}}$ *if* $f_{\mathcal{K}}(e) \in \{instance\_of, is\_same\_as\}$*.*

(iii) Given $\mathcal{G}'_{\mathcal{S}}$ *and* $\mathcal{G}'_{\mathcal{K}}$ *are the graphs generated by the steps 1 and 2 of the WiSCR algorithm respectively, then according to the Step 3 of the WiSCR algorithm, it extracts all possible graph-subgraph isomorphisms between* $\mathcal{G}'_{\mathcal{S}}$ *and* $\mathcal{G}'_{\mathcal{K}}$*. In other words, it extracts all possible sets of pairs of the form* (a,b) *such that either there does not exist such a non-empty set or if* $\mathbb{M}_i$ *is one such non-empty set then,*

- *for each* $(a,b) \in \mathbb{M}_i$*,* $a \in \mathbb{V}'_{\mathcal{S}}$ *and* $b \in \mathbb{V}'_{\mathcal{K}}$*,*
- *for each* $(a,b) \in \mathbb{M}_i$*, a and b are instances of same class, i.e.,* $(a,i) \in \mathbb{E}_{\mathcal{S}}$*,* $(b,i) \in \mathbb{E}_{\mathcal{K}}$*,* $f_{\mathcal{S}}((a,i))$ = $instance\_of$ *and* $f_{\mathcal{K}}((a,i)) = instance\_of$*, and*
- *if for every pair* $(a,b) \in \mathbb{M}_i$*, a is replaced by b then* $\mathcal{G}'_{\mathcal{K}}$ *becomes a subgraph of the node-replaced* $\mathcal{G}'_{\mathcal{S}}$

Theorem 1 *Let* $\mathcal{S}$ *be a sequence of sentences in a WSC problem* $\mathcal{P}$*,* $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}},\mathbb{E}_{\mathcal{S}},f_{\mathcal{S}})$ *be a graphical representation of* $\mathcal{S}$*, p be a node in* $\mathcal{G}_{\mathcal{S}}$ *such that it represents the pronoun to be resolved in* $\mathcal{P}$*,* a_1 *and* a_2 *be two nodes in* $\mathcal{G}_{\mathcal{S}}$ *such that they represent the two answer choices for* $\mathcal{P}$*, and* $\mathcal{G}_{\mathcal{K}} = (\mathbb{V}_{\mathcal{K}},\mathbb{E}_{\mathcal{K}},f_{\mathcal{K}})$ *be a graphical representation of a knowledge such that* $f_{\mathcal{K}}$ *is defined using* $f_{\mathcal{S}}$*. Then, the Winograd Schema Challenge Reasoning (WiSCR) algorithm outputs,*

- a_1 *as the answer of* $\mathcal{P}$*, if only* a_1 *provides the 'most natural resolution' (By Definition 7) for p in* $\mathcal{G}_{\mathcal{S}}$*,*
- a_2 *as the answer of* $\mathcal{P}$*, if only* a_2 *provides the 'most natural resolution' for p in* $\mathcal{G}_{\mathcal{S}}$*,*
- *No answer otherwise*

Proof 6 *If* $\mathcal{G}_{\mathcal{S}} = (\mathbb{V}_{\mathcal{S}},\mathbb{E}_{\mathcal{S}},f_{\mathcal{S}})$ *is a graphical representation of the sequence of sentences in a WSC problem then by Lemma 1, we have that*
Step 1 of the WiSCR Algorithm extract a subgraph $\mathcal{G}'_{\mathcal{S}}$ *from* $\mathcal{G}_{\mathcal{S}}$ *such that* $\mathcal{G}'_{\mathcal{S}} = (\mathbb{V}'_{\mathcal{S}},\mathbb{E}'_{\mathcal{S}},f'_{\mathcal{S}})$ *where* $\mathbb{V}'_{\mathcal{S}} = \mathbb{V}_{\mathcal{S}} - \mathbb{V}^c_{\mathcal{S}}$*,* $\mathbb{V}^c_{\mathcal{S}}$ *is a set of all the class nodes in* $\mathcal{G}_{\mathcal{S}}$*,* $f'_{\mathcal{S}} = f_{\mathcal{S}}$*,* $\mathbb{E}'_{\mathcal{S}} = \mathbb{E}_{\mathcal{S}} - \mathbb{E}^c_{\mathcal{S}}$*, and* $e \in \mathbb{E}^c_{\mathcal{S}}$ *if* $f(e) = instance\_of$*.*

If $\mathcal{G}_{\mathcal{K}}$=$(\mathbb{V}_{\mathcal{K}},\mathbb{E}_{\mathcal{K}},f_{\mathcal{K}})$ *is a graphical representation of a knowledge then by Lemma 2, we have that*

Step 2 of the WiSCR Algorithm extracts a subgraph $\mathcal{G}'_{\mathcal{K}}$ from $\mathcal{G}_{\mathcal{K}}$ such that $\mathcal{G}'_{\mathcal{K}} = (\mathbb{V}'_{\mathcal{K}}, \mathbb{E}'_{\mathcal{K}}, f'_{\mathcal{K}})$ where $\mathbb{V}'_{\mathcal{K}} = \mathbb{V}_{\mathcal{K}} - \mathbb{V}^c_{\mathcal{K}}$, $\mathbb{V}^c_{\mathcal{K}}$ is a set of all the class nodes in $\mathcal{G}_{\mathcal{K}}$, $f'_{\mathcal{K}} = f_{\mathcal{K}}$, $\mathbb{E}'_{\mathcal{K}} = \mathbb{E}_{\mathcal{K}} - \mathbb{E}^c_{\mathcal{K}}$, and $e \in \mathbb{E}^c_{\mathcal{K}}$ if $f(e) \in \{instance\_of, is\_same\_as\}$.

If $\mathcal{G}_{\mathcal{S}}$, $\mathcal{G}_{\mathcal{S}}'$, $\mathcal{G}_{\mathcal{K}}$ and $\mathcal{G}_{\mathcal{K}}'$ are inputs to the Step 3 of the WiSCR algorithm then by Lemma 3, we have that
Step 3 of the WiSCR algorithm produces all the possible sets of node pairs of the form (a,b) such that either there does not exist such a non-empty set or if $\mathbb{M}_i$ is one such non-empty set then,

- *for each $(a,b) \in \mathbb{M}_i$, $a \in \mathbb{V}'_{\mathcal{S}}$ and $b \in \mathbb{V}'_{\mathcal{K}}$, and*
- *for each $(a,b) \in \mathbb{M}_i$, a and b are instances of same class, i.e., $(a,i) \in \mathbb{E}_{\mathcal{S}}$, $(b,i) \in \mathbb{E}_{\mathcal{K}}$, $f_{\mathcal{S}}((a,i)) = instance\_of$ and $f_{\mathcal{K}}((a,i)) = instance\_of$*
- *if for every pair (a,b)$\in \mathbb{M}_i$, a is replaced by b then $\mathcal{G}'_{\mathcal{K}}$ becomes an induced subgraph of $\mathcal{G}'_{\mathcal{S}}$*

If $p \in \mathbb{V}_{\mathcal{S}}$ represents the pronoun to be resolved, $a_1, a_2 \in \mathbb{V}_{\mathcal{S}}$ represent the two answer choices. Then by Step 4 of the WiSCR algorithm and for each possible non-empty set of pairs (say $\mathbb{M}_i$) produced by Step 3, we have that

1. *a_1 as an answer if,*
   - *$(p,n_1) \in \mathbb{M}_i$,*
   - *$(a_1,n_2) \in \mathbb{M}_i$,*
   - *$(n_1,n_2) \in \mathbb{E}_{\mathcal{K}}$ and $f_{\mathcal{K}}((n_1,n_2)) = is\_same\_as$, or $(n_2,n_1) \in \mathbb{E}_{\mathcal{K}}$ and $f_{\mathcal{K}}((n_2,n_1)) = is\_same\_as$, and*
   - *there does not exist an n and an x ($x \neq a_1$) such that $(x,n) \in \mathbb{M}_i$ and either $f_{\mathcal{K}}((n,n_1)) = is\_same\_as$ or $f_{\mathcal{K}}((n_1,n)) = is\_same\_as$.*
2. *a_2 as an answer if,*
   - *$(p,n_1) \in \mathbb{M}_i$,*
   - *$(a_2,n_2) \in \mathbb{M}_i$,*
   - *$(n_1,n_2) \in \mathbb{E}_{\mathcal{K}}$ where $f_{\mathcal{K}}((n_1,n_2)) = is\_same\_as$, or $(n_2,n_1) \in \mathbb{E}_{\mathcal{K}}$ where $f_{\mathcal{K}}((n_2,n_1)) = is\_same\_as$, and*
   - *there does not exist an n and an x ($x \neq a_2$) such that $(x,n) \in \mathbb{M}_i$ and either $f_{\mathcal{K}}((n,n_1)) = is\_same\_as$ or $f_{\mathcal{K}}((n_1,n)) = is\_same\_as$.*
3. *not answer is produced if neither a_1 nor a_2 are found as an answer*

Then, the Step 4 of the WiSCR algorithm outputs a_1 as the final answer if only a_1 is found as an answer with respect to the possible node pairs extracted in the Step 3. The Step 4 of the WiSCR algorithm outputs a_2 as the final answer if only a_2 is found as an answer with respect to the possible node pairs extracted in the Step 3. The Step 4 of the algorithm does not answer anything otherwise.
By definition of 'most natural resolution' and above details of the Step 4 of the WiSCR algorithm, we have that

- *a_1 is the answer of $\mathcal{P}$, if only a_1 provides the 'most natural resolution' (By Definition 7) for p in $\mathcal{G}_{\mathcal{S}}$,*

- *a_2 is the answer of $\mathcal{P}$, if only a_2 provides the 'most natural resolution' for p in $\mathcal{G}_\mathcal{S}$,*
- *No answer otherwise*

The theorem is proved.

A.2 Proof of Theorem 2

Theorem 2 *Let $\mathcal{S}$ be a sequence of sentences in a WSC problem $\mathcal{P}$, $\mathbb{T}(S)$ be the set of tokens in $\mathcal{S}$, $p \in \mathbb{T}(S)$ be the token which represents the pronoun to be resolved, $a_1, a_2 \in \mathbb{T}(S)$ be two tokens which represent the two answer choices, $\mathcal{G}_\mathcal{S} = (\mathbb{V}_\mathcal{S}, \mathbb{E}_\mathcal{S}, f_\mathcal{S})$ be a graphical representation of $\mathcal{S}$, and $\mathcal{G}_\mathcal{K} = (\mathbb{V}_\mathcal{K}, \mathbb{E}_\mathcal{K}, f_\mathcal{K})$ be a representation of a knowledge such that $f_\mathcal{K}$ is defined using $f_\mathcal{S}$. Also, $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ be the AnsProlog program for WiSCR algorithm and* ANSWERFINDER *be the sub-algorithm defined in Section 4.3.2. Then, the WiSCR algorithm produces an answer x to the input WSC problem iff $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ and* ANSWERFINDER *together output the answer x.*

Proof 7 *(i) Given the ASP encoding of a graphical representation of the sequence of sentences in a WSC problem, the rules `s11-s13` extract a subgraph such that it contains only the non class nodes from the original graphs and the edges which connect them. The nodes of the subgraph are represented using the predicate `node_G_s` and the edges are represented using the binary predicate `edge_G_S`. In other words, the rules `s11-s13` implement the Step 1 of the WiSCR algorithm.*
(ii) Similar to (i) the rules `s21-s23` implement the Step 2 of the WiSCR algorithm.
(iii) Given the outputs of the rules `s11-s23`, and the ASP representations of the sequence of sentences in a WSC problem and a knowledge, the rules `s31-s37` first generate all possible matching pairs corresponding to the nodes of the graph of WSC sentences and the graph of knowledge, then a set of constraints are used to remove the possibilities which do not represent an isomorphism between the subgraphs of WSC sentences and knowledge. In other words, the rules `s31-s37` implement the Step 3 of the WiSCR algorithm.
(iv) Given an output of the rules `s31-s37`, and the ASP representations of the sequence of sentences in a WSC problem and a knowledge, the rules `s41-s49`
- *output $ans(a_1)$ if $matches(p, n_1)$, $matches(a_1, n_2)$ are true and $has\_k(n_1,$"is_same_as"$, n_2)$ or $has\_k(n_2,$"is_same_as"$, n_1)$ is true, and there does not exist an n and an x ($x \neq a_1$) such that $matches(x, n)$ is true and either $has\_k(n_1,$"is_same_as"$, n)$ or $has\_k(n,$"is_same_as"$, n_1)$ is true.*
- *output $ans(a_2)$ if $matches(p, n_1)$, $matches(a_2, n_2)$ are true and $has\_k(n_1,$"is_same_as"$, n_2)$ or $has\_k(n_2,$"is_same_as"$, n_1)$ is true, and there does not exist an n and an x ($x \neq a_2$) such that $matches(x, n)$ is true and either $has\_k(n_1,$"is_same_as"$, n)$ or $has\_k(n,$"is_same_as"$, n_1)$ is true.*
- *do not satisfy the current interpretation*

If more than one answers are produced and all of them correspond to one answer then ANSWERFINDER *sub-algorithm outputs that as the final answer. Otherwise if zero answers are produced, or not all among the multiple answers correspond to a common answer then the* ANSWERFINDER *sub-algorithm does not output anything.*

In other words, the rules `s41-s49` and the ANSWERFINDER *sub-algorithm together implement the step 4 of the WiSCR algorithm.*

By (i), (ii), (iii) and (iv), the WiSCR algorithm produces an answer x to the input WSC problem iff $\Pi(\mathcal{G}_S, \mathcal{G}_K, p, a_1, a_2)$ and ANSWERFINDER *together output the answer x.*
The theorem is proved.