Abstract

The Winograd Schema Challenge was proposed as an alternative to the Turing test in 2011. It is a text based question answering challenge which, according to its creators, “requires world knowledge and default reasoning abilities”. In this work, we performed a comprehensive study of the problems in the challenge from the perspective of identifying the types of knowledge which are needed to solve them. We identified 12 knowledge types to address the entire challenge corpus. We also defined a logical reasoning algorithm to tackle 10 out of 12 knowledge types. The algorithm covers 82.47% of all the problems in the dataset. Furthermore, we show how the overall approach generalizes on a well known pronoun resolution corpus which is inspired from the Winograd Schema Challenge.

1. Introduction

The Winograd Schema Challenge (WSC) [Levesque et al., 2011] consists of text based question answering problems and has been proposed as an alternative to the Turing test. Following is an example problem in the challenge.

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

In IJCAI 2016, Nuance Communications Inc. sponsored a contest to solve the challenge. Six systems competed in the contest but the low accuracy (58.3%) of the best performing system [Liu et al., 2016] shows that there is still a long way to go before actually solving the challenge.

Similar to many of the other approaches [Emami et al., 2018a, Liu et al., 2017, Isaak and Michael, 2016, Sharma et al., 2015b, 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 WSC example shown above. To answer the given question one needs the knowledge which links the property of being “weak” and “the inability to lift”.

Commonsense Knowledge Types Identification and Reasoning for the Winograd Schema Challenge

Anonymous authors
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 paper, 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 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.

Consequently, the main contributions of this paper are categorization of commonsense knowledge and development of a reasoning algorithm to handle the categories with respect to the Winograd Schema Challenge. There are various works [Bailey et al., 2015, Schüller, 2014, Sharma et al., 2015b] which support the underlying motivation of using knowledge and reasoning to solve the challenge. Only a few tried (with limited coverage) to identify and categorize (implicitly or explicitly) the types of knowledge that are needed to solve the challenge. Furthermore, these approaches do not focus on processing the natural language text. Instead, they assume that a given natural language text can be automatically translated into a representation of their choice. In this paper, we progress towards addressing these issues. We also show that the overall approach generalizes well by performing an experiment on a held out set of problems from the dataset published by Rahman and Ng [Rahman and Ng, 2012]. The dataset is inspired from the WSC corpus. In simple words, the major contributions of this work can be summarized in the following points.

- A semantic graph based formal representation of a WSC sentence and a question.
- Identification and graph based representation of various knowledge types.
- A logical reasoning algorithm to handle the knowledge types.
- An experimental evaluation of the overall approach on a held out set of problems.

### 2. Formal Representations and Knowledge Types Identification

Two of the main contributions of this work include formal representation of text and identification of different types of commonsense knowledge. In this section we provide details of our efforts in that direction.

We used a logical commonsense reasoning approach. A prerequisite of using such an approach is to formally represent an input problem (a sentence and a question in a WSC

\(^1\) Available at: [https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WS.html](https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WS.html)
problem) and the needed knowledge. Over the years, various works have focused on representing natural language text. Abstract Meaning Representation (AMR) [Banarescu et al., 2013] is an outcome of one such work. It depicts the meaning of a text by transforming it into a rooted, edge and leaf labeled directed graph. The nodes in the graph represent concepts and the edges represent the semantic dependencies between the concepts. An example of an AMR graph is as shown in Figure 1. Knowledge Parser (K-Parser) [Sharma et al., 2015a] and TRIPS parser [Allen et al., 2008] are other works on formally defining and automatically translating an English text into its graphical meaning representation.

Effectiveness of the above mentioned meaning representations in several applications [Rao et al., 2017, Mostafazadeh et al., 2016, Sharma et al., 2015b] inspired us to use a graphical formal representation in this work. There are two main advantages of using such a representation, (1) it is easily used to develop a formal reasoning result (as shown in a later section), and (2) there are several off the shelf parsers which can assist in generating it. There has been a number of semantic parsing works which automatically transform a text into a graphical semantic representation (e.g., K-Parser, TRIPS and JAMR [Flanigan et al., 2014]). In this work, we used K-Parser\(^2\) (details in the Experiments section).

Following are the main components of the semantic graphs used here.

• **Action Nodes:** These nodes in a semantic graph of a text correspond to the concrete verbs (non auxiliary\(^3\) verbs) in the text. For example the node labeled \(\text{lift}_5\) in the graph shown in Figure 2 is an action node. \("_5\) in the node label represents the word index of the verb \(\text{lift}\) in the text.

• **Entity Nodes:** These nodes correspond to the nouns and the pronouns in a text. For example the nodes \(\text{he}_9\), \(\text{man}_2\), \(\text{son}_7\) and \(\text{his}_6\) in Figure 2 are entity nodes. Similar to the action nodes, \("_9\) in the \("\text{he}_9\) node represents the index of the word \("\text{he}\) in the text.

• **Property Nodes:** These are the nodes which correspond to the adjectives and the adverbs in a given text. For example the node \(\text{so}_{11}\) is a property node in the graph.

\(^2\) Available at: https://github.com/arpit7123/K-Parser-JAR.git

\(^3\) http://www.chompchomp.com/terms/auxiliaryverb.htm
shown in Figure 2. Also, “_11” in the “so_11” node represents the index of the word “so” in the text.

- **Conceptual Class Nodes:** These nodes represent the conceptual classes or types of the action, entity and property nodes. For example in Figure 2 the conceptual class node labeled lift represents the type of the action node lift_5 and the class node labeled entity represents the type of all the entity nodes. By default all the entity nodes have a type called entity. The class for each action node is the base form of the verb used to generate the action node. A conceptual class node may be associated with more than one action, entity or property nodes. For example if there are two action nodes labeled lift_4 and lifted_6 in a representation then both have a common class node which is labeled as lift.

- **Semantic Relations:** The labels of the directed edges which connect action, entity, property and conceptual class nodes are called semantic relations. They define the semantic dependency between different nodes in the representation. There are seven types of semantic relations defined based on the types of end nodes connected by them. They are action-action, action-entity, action-property, property-action, entity-entity, entity-property and action/entity/property-concept. An example of a property-action semantic relation (prevents) is shown in Figure 2.

2.1 Representation of a Sentence and a Question

Based on the above components, following are the formal definitions of a representation of a sentence and a question with respect to this work.

**Definition 1 (A Representation of a Sentence (G_S)).** A representation of an input sentence (S) is a non-empty, rooted, edge labeled directed acyclic graph (G_S) such that,

- each node is an action, entity, property or conceptual class node.
- every two nodes in G_S which are semantically dependent on each other are connected through a semantic relation taken from a predefined set of relation labels.
- a conceptual class node exists in G_S only if there is a corresponding action, entity or a property node is in G_S.

An example of a sentence’s representation is shown in Figure 2.

**Definition 2 (A Representation of a Question (G_Q)).** A formal representation of an input question (Q) is a non-empty, rooted, edge labeled directed acyclic graph (G_Q) such that,

- all the properties of G_S (from Definition 1) are also satisfied by G_Q.
- the entity nodes in G_Q, which correspond to the “Wh” words in Q, are relabeled as “q_#” where “#” represents the index of the respective “Wh” word in Q. Each such node is called an “unknown node.”
2.2 Identification and Representation of Commonsense Knowledge Types

As a part of this work, we performed a comprehensive study of the WSC problems and identified the commonsense knowledge needed to correctly solve them. Let us consider an example WSC problem,

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

**Question:** Who was weak?, **Answer:** man

The representations of the above problem are as shown in Figures 2 and 3. It is clear that the knowledge needed to answer the question must make a connection between the inability of an entity to lift and the entity being weak. Such knowledge can be written in plain English as “weak y prevents y lifts” where y refers to a placeholder for an entity. Intuitively, the knowledge states that if ‘an entity ($e_1$) being weak’ prevents ‘another entity
Anonymous authors

(e_2) to lift’ then possibly both e_1 and e_2 are referring to the same entity. We used this intuition to identify 12 distinct knowledge types as mentioned in the Table 1. The first 10 types are motivated by the different kinds of interactions between actions and properties. For example, the prevents relation in the example shown in Figure 4. The remaining two are based on the likelihood of one proposition with respect to another, and multiple pieces knowledge (see Table 1 for examples). Each knowledge of types 1 through 10 have three main parts. Let us consider the weak-prevents-lift knowledge mentioned above to understand the representation of each part in detail.

Knowledge: “weak y prevents y lifts”

The first part (say P1) corresponds to weak y. The second part (say P2) corresponds to prevents, and the third part (say P3) corresponds to y lifts. Continuing the idea of using a graphical representation, we formulated the commonsense knowledge also into a rooted, edge and node labeled, directed acyclic graph made up of four kinds of nodes (action, entity, property and conceptual class) and semantic relations between the nodes. Parts P1 and P3 of the knowledge are treated as regular sentences and transformed into separate graphs as per the Definition 1. Part P2 however represents a semantic relationship between the other two parts and hence it is transformed into a semantic relation (prevents) between the root nodes of the representations of parts (1) and (3). The co-referencing nodes in the graphs of parts (1) and (3) (i.e., y_2 and y_4) are merged into a single node in the final representation (i.e., y_2). A representation of the weak-prevents-lift knowledge and its generation flow is shown in Figure 4.

Figure 4: A representation of the knowledge “weak y prevents y lifts”
<table>
<thead>
<tr>
<th>Knowledge Type</th>
<th>Example</th>
<th># Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Property prevents Action</td>
<td>Sent: The man could not lift his son because he was so weak. <strong>Know</strong>: weak y prevents y lifts</td>
<td>16</td>
</tr>
<tr>
<td>2. Action1 prevents Action2</td>
<td>Sent: Beth didn’t get angry with Sally, who had cut her off, because she stopped and apologized. <strong>Ques</strong>: Who apologized? <strong>Know</strong>: y apologize prevents x gets angry with y</td>
<td>6</td>
</tr>
<tr>
<td>3. Action1 causes Action2</td>
<td>Sent: The older students were bullying the younger ones, so we rescued them. <strong>Ques</strong>: Whom did we rescue? <strong>Know</strong>: y was bullied causes y was rescued</td>
<td>41</td>
</tr>
<tr>
<td>4. Property causes Action</td>
<td>Sent: Jim comforted Kevin because he was so upset. <strong>Ques</strong>: Who was upset? <strong>Know</strong>: y was upset causes y was comforted</td>
<td>27</td>
</tr>
<tr>
<td>5. Action causes Property</td>
<td>Sent: I took the water bottle out of the backpack so that it would be handy. <strong>Ques</strong>: What would be handy? <strong>Know</strong>: y is taken out of backpack causes handy y</td>
<td>13</td>
</tr>
<tr>
<td>6. Property1 causes Property2</td>
<td>Sent: In July, Kamchatka declared war on Yakutsk. Since Yakutsk’s army was much better equipped and ten times larger, they were victorious within weeks. <strong>Ques</strong>: Who was victorious? <strong>Know</strong>: y’s army was larger causes y was victorious</td>
<td>4</td>
</tr>
<tr>
<td>7. Action1 followed by Action2</td>
<td>Sent: The customer walked into the bank and stabbed one of the tellers. He was immediately taken to the emergency room. <strong>Ques</strong>: Who was taken to the emergency room? <strong>Know</strong>: y was stabbed followed by y was taken to emergency room</td>
<td>17</td>
</tr>
<tr>
<td>8. Action followed by Property</td>
<td>Sent: Sam broke both his ankles and he’s walking with crutches. But a month or so from now they should be better. <strong>Ques</strong>: What should be better? <strong>Know</strong>: y was broken followed by better y</td>
<td>2</td>
</tr>
<tr>
<td>9. Property followed by Action</td>
<td>Sent: Thomson visited Cooper’s grave in 1765. At that date he had been dead for five years. <strong>Ques</strong>: Who had been dead for five years? <strong>Know</strong>: y was dead followed by x visits y’s grave</td>
<td>2</td>
</tr>
<tr>
<td>10. Co-existing Action(s) and Property(s)</td>
<td>Sent: Steve follows Fred’s example in everything. He influences him hugely. <strong>Ques</strong>: Who is influenced? <strong>Know</strong>: y follows and y was influenced</td>
<td>112</td>
</tr>
<tr>
<td>11. Statement1 is more likely than Statement2</td>
<td>Sent: Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like dogs. <strong>Ques</strong>: What looked like dogs? <strong>Know</strong>: Sheep looks like a dog is more likely than Shepherd looks like a dog</td>
<td>26</td>
</tr>
<tr>
<td>12. Multiple Knowledge</td>
<td>Sent: Mary tucked her daughter Anne into bed, so that she could work. <strong>Ques</strong>: Who is going to work? <strong>Know 1</strong>: x tuck y into bed causes y sleeps <strong>Know 2</strong>: daughter of y sleeps prevents y is disturbed <strong>Know 3</strong>: y is not disturbed causes y can work</td>
<td>25</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>—</strong></td>
<td><strong>291</strong></td>
</tr>
</tbody>
</table>

Table 1: Table of Knowledge Types

In this work our focus was to develop a reasoning algorithm for the first 10 knowledge types in the Table 1. Therefore, based on the above explanation, a knowledge of type 1 through 10 and their representation are formally defined in the Definition 3 and 4 respectively.

**Definition 3 (A Knowledge (K)).** A knowledge is a string made up of three parts such that,

- the first and the third parts are regular English sentences with placeholder variables for entities, and
• the second part is a semantic relation (causes, prevents or followed by) that connects the other two parts

**Definition 4 (A Representation of Knowledge (G_K)).** Let K be a knowledge (as per Definition 3). Then, G_K is a representation of K such that,

• G_K is a non-empty, rooted, edge and node labeled, directed acyclic graph
• the first and the third parts of K are transformed into semantic graphs by Definition 1 and added to G_K, and
• the second part of K defines an edge from the root of the semantic graph of the first part to the root of semantic graph of the second part of K in G_K, and
• the co-referencing nodes in the semantic graphs of the first and the third parts of K are merged into one node in G_K.

An example of a representation of a knowledge is shown in Figure 4.

### 3. Reasoning Algorithm for the WSC

As part of this work we have developed a logical reasoning algorithm to solve the WSC problems. The algorithm takes the formal representations of a WSC problem (a sentence and a question) and a knowledge as inputs and deduces an answer to the question if it is entailed by the inputs. Following are the two main phases of the reasoning algorithm.

#### 3.1 Phase 1: Merged Representation Generation

In this phase an input sentence’s representation and a representation of a knowledge are combined to generate a merged representation. The intuition behind is to update the representation of the input sentence with the information contained in the knowledge. Let G_S be an input sentence’s representation (By Definition 1) and G_K be a representation of a knowledge (By Definition 4). Then, following operations are performed in this phase.

1. A node x in G_S is said to be a constant node in G_S if there exists an edge labeled instance_of from x to another node in G_S. Similarly, a node x in G_K is said to be a constant node in G_K if there exists an edge labeled instance_of from x to another node in G_K. Intuitively, this operation identifies all the action, entity and property nodes in the input sentence’s and the representation of a knowledge and tags them as constant nodes.

2. A constant node x in G_S is said to have a constant parent p in G_S if there exists an edge from p to x, where p is a constant node in G_S. Similarly, a constant node y in G_K is said to have a constant parent p1 in G_K if there exists an edge from p1 to y, where p1 is a constant node in G_K. Intuitively, this operation identifies and tags the constant nodes which have constant parent node(s) in the input sentence’s and the representation of a knowledge.
3. Similar to the step 2, a constant node $x$ in $G_S$ is said to have a constant child $c$ in $G_S$ if there exists an edge from $x$ to $c$ in $G_S$, where $c$ is also a constant node in $G_S$. Similarly, a constant node $y$ in $G_K$ is said to have a constant child $c$ in $G_K$ if there exists an edge from $y$ to $c$ in $G_K$, where $c$ is also a constant node in $G_K$. Intuitively, this operation identifies and tags the constant nodes which have constant child node(s) in the input sentence’s and the representation of a knowledge.

4. A constant node $y$ in $G_S$ is said to be a cross domain sibling of a constant node $x$ in $G_K$, if there exists an outgoing edge labeled `instance_of` from $x$ to a node $i$ in $G_K$, and there also exists an outgoing edge labeled `instance_of` from $y$ to a node $i$ in $G_S$. Intuitively, it means that both $x$ and $y$ are of the same type.

5. A constant node $y$ in $G_S$ is said to be a cross domain clone of a constant node $x$ in $G_K$ if all of the below mentioned conditions are satisfied,
   - $y$ is a cross domain sibling of $x$
   - If $(p_j, r_j, x)$ is an edge in $G_K$, then $(p_j', r_j, y)$ is an edge in $G_S$ such that $p_j'$ is a cross domain clone of $p_j$ from $G_K$ to $G_S$, and
   - If $(x, r_k, c_k)$ is an edge in $G_K$, then $(y, r_k, c'_k)$ is an edge in $G_S$ such that $c'_k$ is a cross domain sibling of $c_k$ from $G_K$ to $G_S$

The constant parent and child nodes information is used in the last two conditions above to handle the possibilities where the $x$ and $y$ do not have any constant parent or child.

Intuitively, a node $y$ in $G_S$ is said to be a cross domain clone of a node $x$ in $G_K$ if both $x$ and $y$ are mirror images of each other in all the aspects (parents, children and type) except their labels. In other words, $x$ and $y$ cross domain siblings if all the conditions below are satisfied.
   - both $x$ and $y$ are of same type, and
   - both $x$ and $y$ have same number of child nodes which are connected to $x$ and $y$ through same semantic relations and the children are of same types, and
   - both $x$ and $y$ have same number of parent nodes which are connected to $x$ and $y$ through same semantic relations and the parents are cross domain clones of each other.

The outcomes of the above mentioned operations are used to generate a merged representation as defined below in Definition 5.

**Definition 5 (Merged Representation ($G_M$)).** Let $G_S$ be a representation of a sentence and $G_K$ be a representation of a knowledge. A merged representation $G_M$, is obtained by performing the following set of operations on $G_S$ and $G_K$.

1. copy all the nodes and edges from $G_S$ into $G_M$

2. if each constant node in $G_K$ has a cross-domain clone in $G_S$, then for each constant node $v_i$ in $G_K$, for every two distinct constant nodes $v'_j$ and $v'_k$ in $G_S$ such that $v'_j$ and $v'_k$ are a cross domain clones of $v_i$, do the following if the respective conditions are satisfied.
Anonymous authors

(a) add \((p_j, r_j, v'_k)\) to \(G_M\) if \((p_j, r_j, v'_j) \in G_S\)

(b) add \((v'_k, r_j, c_j)\) to \(G_M\) if \((v'_j, r_j, c_j) \in G_S\)

Intuitively,

- a sentence’s representation is copied as is in a merged representation, and
- if two distinct nodes \(v'_j\) and \(v'_k\) in a sentence’s representation are found to be the mirror images (cross domain clone) of a single node in a representation of a knowledge then edges are added in the merged representation such that the sets of parents and children of both \(v'_j\) and \(v'_k\) in the merged representation are union of their parents and children in the sentence’s representation respectively.

According to the Definition 5, Figure 5 shows the merged representation generated for a sentence’s and a knowledge’s representation shown in Figures 2 and 4 respectively.

![Figure 5: An example of a merged representation](image)

### 3.2 Phase 2: Answer Retrieval

Answer retrieval is the second and the final phase of the reasoning algorithm. In this phase, the answer to a given question is retrieved from a merged representation. The intuition for the retrieval is to project a question’s representation on the merged representation and retrieve the nodes from the merged representation which are cross domain clones of the unknown nodes in the question’s representation.

Let \(G_M\) be a merged representation and \(G_Q\) be a representation of an input question. Then, following operations are performed in this phase which are similar to the merged representation generation phase.

1. Constant nodes in both \(G_M\) and \(G_Q\) are identified
2. Nodes with constant parents and constant children are identified from both \(G_M\) and \(G_Q\)
3. Cross-domain siblings of the nodes in $G_Q$ are identified from $G_M$ by using constant
nodes and constant parent/child nodes identified previously.

4. Cross-domain clones of the nodes in $G_Q$ are identified from $G_M$ by using the previously
identified properties.

The outputs of the above operations are then used to retrieve the answer(s) to the input
question. An answer to a question is as defined below in Definition 6.

**Definition 6 (An Answer (ans($q_i, x$))).** Let $G_S$, $G_Q$, $G_K$ be representations of a sentence,
a question and a knowledge respectively and $G_M$ be the merged representation of $G_S$ and $G_K$
(by Definition 5). We say that if each constant node in $G_Q$ has a cross-domain clone node
in $G_M$, $q_i$ is an unknown node in $G_Q$ (i.e. $(q_i, \text{instance_of}, q) \in G_Q$), and a node $x$ in $G_M$
is a cross domain clone of $q_i$, then $x$ is an answer corresponding to the unknown node, $q_i$,
in $G_Q$.

Intuitively, all the nodes which are cross-domain clones of the unknown nodes in a
question’s representation are retrieved as the final answers with respect to the corresponding
unknown nodes. Based on the above definition, following are the answers for the $G_S$, $G_Q$,
$G_K$ and $G_M$ shown in Figures 2, 3, 4 and 5 respectively.

$$\text{ans}(q_1, \text{he, 9})$$
$$\text{ans}(q_1, \text{man, 2})$$

### 3.3 ASP Implementation of the Reasoning Algorithm

An Answer Set Programming (ASP)[Gelfond and Lifschitz, 1988, Baral, 2003] based imple-
mentation of the reasoning algorithm was also done as part of this work. The implementa-
tion being a direct translation of the theory, is not shown in this paper. The implementation
code along with a worked out example is available at [https://bit.ly/2QMyBAM](https://bit.ly/2QMyBAM).

### 4. Experiments

#### 4.1 Datasets Used

We used two datasets in this work. Firstly, we used the original WSC corpus available
on its official web page\(^1\). From this point onward, we will refer to this dataset as $\text{WSC}_{og}$
because it is a version of the real WSC dataset proposed in [Levesque et al., 2011]. The
corpus used in this work consists of 291 question answering problems. Secondly, we used the
Definite Pronoun Resolution Dataset\(^4\) which was published by Rahman and Ng [Rahman
and Ng, 2012]. It consists of WSC like problems but they are more relaxed in terms of the
knowledge needed. For example, in many problems the corpus statistics (such as google
search result comparisons) play an important part in identifying the correct co-referent of
a pronoun. We randomly selected a set of 200 problems from the test partition (564 QA
problems) of the dataset. We refer to this partition as $\text{WSC}_{dpr}$ from this point onward. The focus of this work not being the extraction of commonsense knowledge, the knowledge

---

\(^1\) Available at: [http://www.hlt.utdallas.edu/~vince/data/emnlp12/](http://www.hlt.utdallas.edu/~vince/data/emnlp12/)

\(^4\) Available at: [http://www.hlt.utdallas.edu/~vince/data/emnlp12/](http://www.hlt.utdallas.edu/~vince/data/emnlp12/)
required for the problems in WSC\textsubscript{dpr} was manually generated. Due to the manual effort associated with creating the required knowledge we limited ourselves to 200 problems.

The details of the evaluations setups and the results of the main contributions of this work are as shown in the sections below.

4.2 Evaluation of Knowledge Type Identification

4.2.1 Setup:

We performed a comprehensive analysis of the problems in the WSC\textsubscript{og} dataset and identified the knowledge required to solve each of them. The identified knowledge was then manually categorized into the 12 types as shown in the Table 1. The knowledge types identified from the analysis of the WSC\textsubscript{og} corpus were used to categorize the WSC\textsubscript{dpr}.

4.2.2 Results:

All 291 of the problems in the WSC\textsubscript{og} corpus are categorized based on the 12 knowledge types defined in this work. The number of problems in each knowledge category are as mentioned in the Table 1. As per the WSC\textsubscript{dpr} corpus, the 12 knowledge types covered a total of 164 out of the 200 problems (82%).

4.2.3 Remarks:

The knowledge types 2, 6, 8 and 9 combined were found to cover less than 5% of the challenge problems in the WSC\textsubscript{og} dataset. This shows that the dataset may not be well balanced with respect to the types of knowledge needed. As the dataset was manually created by selecting examples from different literature resources so the unbalanced distribution shows that the knowledge of types 2, 6, 8 and 9 are not very commons among those resources. Similar issues were observed in the WSC\textsubscript{dpr} dataset where the types 2, 6, 8 and 9 covered only 2% of the problems. About 38% of the problems in the WSC\textsubscript{og} dataset were found to be of type 10. Type 10 is similar to correlation based knowledge. The results show that there are many problems which can be benefited by using just correlation based knowledge.

4.3 Evaluation of Reasoning Algorithm and the Input Representation

In this work, we used K-Parser\textsuperscript{2} to translate a natural language text into the desired graphical representations. This is done by transforming the output of K-Parser according to the Definitions 1 and 2. We evaluated the quality of the representations by testing their effect on the overall reasoning accuracy. The reasoning algorithm defined in this work uses the semantic graphical representations of a sentence, a question and a knowledge to produce an answer. We calculated the accuracy of the overall reasoning algorithm and categorized the errors caused due to the representation errors, and the reasoning errors. Though the extraction of the required knowledge is not a contribution of this paper, to show that the knowledge mentioned in this paper can of course be extracted, we use a semi automatic approach to extract knowledge for solving a selected set of examples from the WSC\textsubscript{og} corpus. The knowledge extraction process is as mentioned in the setup below. For the WSC\textsubscript{dpr} corpus, we manually created the needed knowledge.
4.3.1 Setup for WSC\textsubscript{og}:

The reasoning algorithm developed in this work handles the top 10 categories of knowledge mentioned in the Table 1. There are 240 WSC\textsubscript{og} problems which fall under those categories. We selected 100 out of 240 problems to test our reasoning algorithm. The selection was motivated by the availability of the knowledge. We used a semi automated knowledge extraction to extract knowledge for this work. The extraction is inspired from the work done in [Sharma and Baral, 2016]. There the idea is to extract a set of sentences (from a search engine) which are similar to the original Winograd Schema Challenge sentences in terms of actions and properties. Such sentences are then parsed with the help of a semantic parser to extract knowledge. For example, a sentence extracted for the Winograd sentence shown in Figure 2 is “She could not lift it because she is a weak girl.”. And the knowledge extracted from the above sentence is “weak prevents y lifts”.

Because of the main focus of this work not being the automatic knowledge extraction, and limited availability of search engine access, the sentences similar to the Winograd Schema Challenge sentences are manually extracted. The extracted sentences are then passed to a rule based knowledge extraction module. The module uses the graphical semantic representations of an input sentence (generated by K-Parser) to find the patterns which satisfy the knowledge types handled by our reasoning framework.

4.3.2 Results for WSC\textsubscript{og}:

By using steps 1 and 2 of the knowledge extraction, we were able to retrieve sentences, which may contain knowledge, for 161 WSC problems. Out of 161, the required knowledge for 100 problems was extracted in the third step. The reasoning system was able to answer all the 100 WSC problems correctly.

4.3.3 Remarks for WSC\textsubscript{og}:

The absolute accuracy of the reasoning algorithm empirically supports the correctness of the algorithm. The ad-hoc knowledge extraction algorithm we used in this work shows that the required knowledge can be extracted from text. The low performance (100 out of 291) of the knowledge extraction algorithm indicates that it needs an improvement which could be provided by exploiting the knowledge type information.

4.3.4 Setup for WSC\textsubscript{dpr}:

The 10 knowledge categories handled by our algorithm cover 138 out of 200 problems. For evaluating the reasoning algorithm on the dataset, we manually wrote the needed knowledge in the graphical format for each one of the 138 problems.

4.3.5 Results for WSC\textsubscript{dpr}:

Out of the 138 problems in WSC\textsubscript{dpr}, 111 (80.45\%) were correctly answered by the reasoning algorithm, 2 (1.45\%) were incorrectly answered because of the parsing error with respect to the input sentences, and 25 (18.1\%) were unanswered again because of the errors in parsing the input sentences. No incorrect answers were reported because of an error in the reasoning algorithm.
4.3.6 Remarks for WSC\textsubscript{dpr}:

The absence of reasoning algorithm errors supports the correctness of the algorithm. Incorrect representations due to parsing errors in the input sentences were found to be the only sources of errors. The main reasons for such errors were incorrect part of speech labelling of words. For example close in “too close” is labeled as a verb instead of an adjective. The incorrect syntactic dependency parsing was found to be another cause of errors. Finally, the inability of the semantic parser to identify phrasal verbs in the text (such as “fall off”) was another cause of errors.

4.4 Additional Remarks: Need for Multiple Pieces of Knowledge

Our current reasoning algorithm focuses on using one piece of knowledge to infer an answer of a given question. Based on our analysis we found that various WSC problems require multiple knowledge (See Type 12 in Table 1). Below is an example of such a WSC problem where three different pieces of knowledge are needed.

<table>
<thead>
<tr>
<th>Sentence:</th>
<th>Mary tuck her daughter Anne into bed, so that she could work.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question:</td>
<td>Who is going to work?</td>
</tr>
<tr>
<td>Answer:</td>
<td>Mary.</td>
</tr>
<tr>
<td>Knowledge 1:</td>
<td>( x ) tuck ( y ) into bed \textit{causes} ( y ) sleeps</td>
</tr>
<tr>
<td>Knowledge 2:</td>
<td>daughter of ( y ) sleeps \textit{prevents} ( y ) is disturbed</td>
</tr>
<tr>
<td>Knowledge 3:</td>
<td>( y ) is not disturbed \textit{causes} ( y ) can work</td>
</tr>
</tbody>
</table>

Updating the reasoning algorithm to accommodate multiple knowledge is an interesting avenue which needs to be explored next along with the automated extraction of knowledge. As an initial step in the direction, we have identified 25 WSC problems which require multiple knowledge.

5. Related Work

Over the years various approaches have been proposed to solve the Winograd Schema Challenge problems by using additional knowledge. Such works include the ones which focus on defining the reasoning theories [Bailey et al., 2015, Schüller, 2014]. Though these approaches mention the need of additional knowledge, their focus revolves around the reasoning aspect of the problem. It drives such approaches towards a strict knowledge representation which is more compatible with their reasoning algorithm. Whereas in this work we identify knowledge types which can be read as plain English phrases. We also provide the graphical representation which can be generated for a knowledge phrase by using existing semantic parsing frameworks.

Another set of approaches address the knowledge extraction and reasoning with it in a joint method. Such approaches include the ones which use on the fly knowledge extraction [Sharma et al., 2015b, Emami et al., 2018a], and the ones which perform knowledge extraction with respect to a pre-populated knowledge base [Isaak and Michael, 2016].

More recently, composition embedding [Liu et al., 2017] and statistical language modelling [Trinh and Le, 2018] based approaches have been used to address the challenge. The
first won the challenge in IJCAI 2016 and the later reported the state of the art accuracy on the overall corpus in June 2018. Such approaches try to capture the knowledge in the form of word and sentences embedding and later use it to infer which phrase is more probable. This helps in the cases where the needed knowledge is based on the possible correlation between two terms for example “a ball is kicked” where there is a correlation between kicked and ball. But it may not be able to infer that “worm is tasty” for the Winograd Schema Challenge problem “Fish ate the worm. It was tasty.”. On the other hand it is more possible that it finds “fish is tasty” more probable because “fish” and “tasty” has higher chances of occurring in the same context in text corpora.

The end-to-end systems [Emami et al., 2018b] which cover the entire WSC corpus also suffer from the low accuracy. This is because they do not account for the different types of knowledge and unintentionally treat every knowledge in a similar way in their reasoning algorithm. Whereas in this work we designed a reasoning algorithm for 10 types of knowledge by carefully analyzing the knowledge types. The remaining two types require different kind of reasoning which can be implemented by exploiting their knowledge type information.

As far as the knowledge types identification is concerned, there are various available knowledge bases that contain different types of knowledge. Most of them do not have the knowledge needed to solve the WSC challenge. However, knowledge bases such as the narrative chains schemas [Chambers and Jurafsky, 2009] and more recently ATOMIC [Sap et al., 2018] contain a subset of the types defined in this paper. The narrative chains contain the stative knowledge about the subjects and objects of actions. For example a person x is convicted and then x is sentenced. This knowledge resembles Type 7 in the Table 1. On the other hand ATOMIC contains 9 different types of knowledge. The types defined in ATOMIC are covered as part of the Types 3, 5, 7, 8 and 9 in the Table 1. The remaining types in the table represent the new types. The idea of crowd-sourcing and neural inference (as used in ATOMIC) can also be used to extract knowledge of the types defined in this paper.

6. Conclusion

This paper focuses on the identification and representation of different types of knowledge, representation of a WSC problem and development of a logical reasoning algorithm for addressing the Winograd Schema Challenge. We identified and defined 12 such knowledge types, covering 100% of the WSC corpus and 82% of a WSC inspired corpus . The representations defined in this work can be easily extracted by using the existing semantic parsers. And the reasoning framework defined in this work handles 10 out of 12 knowledge types. We performed an experimental evaluation of the reasoning algorithm on two distinct datasets to show its correctness and the ability to generalize. We also used an ad-hoc knowledge extraction approach and showed that the knowledge required can be extracted from text. Though the fully-automatic extraction of the required knowledge still remains a problem to be addressed but the recent advancement in the commonsense knowledge generation by crowd-sourcing (as in ATOMIC) and this work takes us one step closer by making important contributions towards identifying the knowledge types and developing a reasoning algorithm to handle those types.
References


Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu 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)*, pages 2344–2350, 2017.


