TPLP **19** (5–6): 1021–1037, 2019. © Cambridge University Press 2019
doi:10.1017/S1471068419000334

Using Answer Set Programming for Commonsense Reasoning in the Winograd Schema Challenge

ARPIT SHARMA
Arizona State University, Tempe, USA
(*e-mail:* asharm73@asu.edu)

submitted 30 July 2019; accepted 31 July 2019

Abstract

The Winograd Schema Challenge (WSC) is a natural language understanding task proposed as an alternative to the Turing test in 2011. In this work we attempt to solve WSC problems by reasoning with additional knowledge. By using an approach built on top of graph-subgraph isomorphism encoded using Answer Set Programming (ASP) we were able to handle 240 out of 291 WSC problems. The ASP encoding allows us to add additional constraints in an elaboration tolerant manner. In the process we present a graph based representation of WSC problems as well as relevant commonsense knowledge.

KEYWORDS: Answer Set Programming, Winograd Schema Challenge, Commonsense Reasoning, Graph-Subgraph Isomorphism

1 Introduction

The Winograd Schema Challenge (WSC) (Levesque et al. 2011) is a natural language understanding task. It is made up of special types of pronoun resolution problems. Each WSC problem consists of a sequence of sentences (currently 1-3) which contain a definite pronoun. 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 concerned pronoun. Additionally, two answer choices for the question are also provided. The answer choices are always present in the sentences. The goal in the WSC challenge is to determine the correct answer choice. Following is an example WSC problem.

Sentences: The fish ate the worm. **It**$_{pronoun}$ was tasty
Question: What was tasty? **Answer Choices:** a) fish b) worm

A WSC problem also specifies an "alternate word" for a "special word" in the sentences. Replacing the "special word" by the "alternate word" changes the resolution of the pronoun. In the example above, the special word is *tasty* and the alternate word is *hungry*. Thus every schema represents a pair of coreference resolution problems that are almost identical but have different answers. Based on our analysis of how people solve the WSC problems, it suggests that for solving them a program would have to use relevant world knowledge. For example to solve the above question, the knowledge that *'something that is eaten may be tasty'* is needed.

Earlier attempts to solve the challenge are mainly based on two different approaches. Works such as (Schüller 2014) and Bailey et al. (2015) solve 8 and 72 WSC problems

respectively by reasoning with the explicitly provided knowledge. Such works presented algorithms which take a WSC problem and a suitable knowledge as input and produce the solution of the problem. Other attempts however utilize the recent advancement in the field of neural language modelling. For example the language model in Radford et al. (2019) correctly answered 193 out of 273 WSC problems by predicting the more plausible answer choice based on the support generated by a language model trained on large body of text.

In this work we attempt to solve the WSC by reasoning with additional knowledge. We define an algorithm which is built on top of graph-subgraph isomorphism (Cordella et al. 2004). By using an Answer Set Programming (ASP) (Baral 2003; Gelfond and Lifschitz 1988) based implementation of the algorithm, we were able to tackle 240 out of 291 WSC problems. The motivation behind using ASP is that we would like the process of adding new constraints to be easier. It plays an important part in the isomorphism detection step of the algorithm where the nodes in two graphs are paired based on a set of constraints. Adding new constraints in other high level languages such as python would take one to delve deep into the code to identify the actual place of injection whereas it can be easily accomplished in ASP by writing a new constraint anywhere in the code. The main contributions of this work are summarized below.

- a graph based representations of WSC sentences and commonsense knowledge,
- Winograd Schema Challenge Reasoning (WiSCR) algorithm,
- an ASP implementation of the WiSCR algorithm, and
- an experimental evaluation of the implementation showing that it handles 240 out of 291 WSC problems. This is accomplished by performing three experiments, one of which involves an automatic approach to extract knowledge from text.

The rest of the paper is organized as follows. Sections 2 and 3 describe graphical representations of a WSC problem and a piece of knowledge. Section 4 details the reasoning algorithm and its ASP implementation. Section 5 presents the evaluation results of the ASP implementation. Section 6 provides the literature review. Finally Section 7 presents our conclusion.

2 Graphical Representation of a WSC Problem

Graphical meaning representations are popular for natural languages such as English. It is because of their simplicity, readability and ability to be easily processed, that in the recent years there has been a significant amount of progress (Sharma et al. 2015a; Banarescu et al. 2013) in defining graphical representations for natural language and development of systems which can automatically parse a natural language text into those representations. Inspired by such representations, in this work we use a graphical schema to represent the sentences in a WSC problem, and a piece of knowledge.

In the following section, 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 (part-of-speech) 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

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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:

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

Example 1. Let us consider the sequence of English sentences $\mathcal{S}$ = (*'The man could not lift his son because he was so weak.'*) where $\mathcal{S}$ contains only one sentence. Then,

$$\mathbb{T}(\mathcal{S}) = \{The\_1, man\_2, could\_3, not\_4, lift\_5, his\_6, son\_7, because\_8, he\_9, was\_10, so\_11, weak\_12\}.$$

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 (Part-Of-Speech) 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 *'The man could not lift his son because he was so weak.'* The set of tokens in the sequence is as shown in the Example 1. Then an example of a mapping produced by a POS tagging function is,

$f_\mathcal{S}^{pos}(The\_1) = other$
$f_\mathcal{S}^{pos}(man\_2) = noun$
$f_\mathcal{S}^{pos}(could\_3) = verb$
$f_\mathcal{S}^{pos}(not\_4) = adverb$
$f_\mathcal{S}^{pos}(lift\_5) = verb$
$f_\mathcal{S}^{pos}(his\_6) = pronoun$
$f_\mathcal{S}^{pos}(son\_7) = noun$
$f_\mathcal{S}^{pos}(because\_8) = other$
$f_\mathcal{S}^{pos}(he\_9) = pronoun$
$f_\mathcal{S}^{pos}(was\_10) = verb$
$f_\mathcal{S}^{pos}(so\_11) = adverb$
$f_\mathcal{S}^{pos}(weak\_12) = adjective$

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 $f_\mathcal{S}^{class}$ maps an element of $\mathbb{T}(\mathcal{S})$ to an element in a set $\mathbb{C}$, i.e., $f_\mathcal{S}^{class} : \mathbb{T}(\mathcal{S}) \rightarrow \mathbb{C}$ where the set $\mathbb{C}$ is a union of three sets $\mathbb{C}_1$, $\mathbb{C}_2$ and $\{\phi\}$ such that,

• $\mathbb{C}_1 = \{c \mid c \text{ is the lemmatized}[1] \text{ form of } w \text{ where } w\_i \in \mathbb{T}(\mathcal{S}) \text{ and } f_\mathcal{S}^{pos}(w\_i) \in \{verb, adverb, adjective\}\}$

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

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

• $\mathbb{C}_2 = \{object, person, group, location, quantity, shape, animal, plant, cognition, communication, event, feeling, act, motive, phenomenon, possession, process, relation, state, time\}$[2]

and,

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

Example 3. Let us consider the sequence of English sentences *'The man could not lift his son because he was so weak.'* The set of tokens in the sequence is as shown in the Example 1. Also let a mapping produced by a POS tagging function is as shown in the Example 2 above. Then an example of a mapping produced by a class mapping function is,

$f_{\mathcal{S}}^{class}(The\_1) = \phi$
$f_{\mathcal{S}}^{class}(man\_2) = person$
$f_{\mathcal{S}}^{class}(could\_3) = can$
$f_{\mathcal{S}}^{class}(not\_4) = not$
$f_{\mathcal{S}}^{class}(lift\_5) = lift$
$f_{\mathcal{S}}^{class}(his\_6) = person$
$f_{\mathcal{S}}^{class}(son\_7) = person$
$f_{\mathcal{S}}^{class}(because\_8) = \phi$
$f_{\mathcal{S}}^{class}(he\_9) = person$
$f_{\mathcal{S}}^{class}(was\_10) = be$
$f_{\mathcal{S}}^{class}(so\_11) = so$
$f_{\mathcal{S}}^{class}(weak\_12) = weak$

Definition 4 (A Formal Representation of a Sequence of One or More English Sentences). Let $\mathcal{S}$ be a sequence of 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 $w\_i \in \mathbb{V}_1\}$

The nodes in $\mathbb{V}_1$ are called *instance* nodes and the nodes in $\mathbb{V}_2$ are called *class* nodes. $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$, 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 $v_2 \in \mathbb{V}_1$, where (v_1, v_2) represents a directed edge between the nodes v_1 and v_2,
• $(v_1, v_2) \in \mathbb{E}_2$ if $v_1 \in \mathbb{V}_1$ and $v_2 \in \mathbb{V}_2$, where (v_1, v_2) represents a directed edge between the nodes v_1 and v_2,

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

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

• 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 $v \neq v_2$. This means that v_1 has only one class node as its successor. This is because a concept can be of one type only in this representation.

$f : \mathbb{E} \rightarrow \mathbb{L} \cup \{instance\_of\}$, 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 *‘The man could not lift his son because he was so weak.’*, POS mapping shown in Example 2 and class mapping shown in Example 3. Then a representation of the sentences is shown in Figure 1. All the edges labels other than *instance_of* part of a predefined set of binary relations between two nodes (as mentioned in the Definition 4). In this work, these relations are from the relations in a semantic parser called K-Parser (Sharma et al. 2015a).

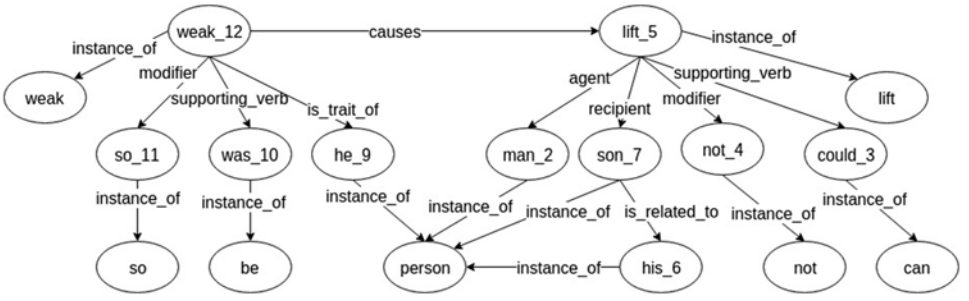

Fig. 1: A Graphical Representation of Sequence of Sentences in a WSC Problem, *“The man could not lift his son because he was so weak.”*

3 Graphical Representation of a Piece of Knowledge

The WSC corpus was created in a way that each problem in it requires an additional knowledge. Let us consider the following WSC example.

> **Sentence:** The man could not 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 being weak prevents her to lift someone else”*. This knowledge can be written as, *“**if** person1 can not lift someone because person2 is weak **then** person1 **is same as** person2”*. Intuitively, it means that if *person2* being weak prevents *person1* from lifting something then *person1* is same as *person2*. Such a knowledge is made up of two parts. The first part is an *if-condition*, which consists of an English sentence, i.e., *‘person1 can not lift someone because person2 is weak’*. The second part of the knowledge is the consequent of the *if-condition*. The consequent is always an *‘is same as’* commutative

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

relationship between two words (e.g., *person*1 and *person*2 above) in the sentence. Such a knowledge and its graphical representation are formally defined below.

Definition 5 (**A Piece of Knowledge**)**.** A piece of 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 $\mathcal{S}$, $x, y \in \mathbb{T}(S)$, $f_{\mathcal{S}}^{pos}(x) = noun$ and $f_{\mathcal{S}}^{pos}(y) = noun$, where $f_{\mathcal{S}}^{pos}$ is a POS tagging function.

Example 5. An example of a piece of knowledge is, **IF** *'person1 can not lift someone because person2 is weak'* **THEN** *person1_1* **is same as** *person2_7*.

Definition 6 (**A Graphical Representation of a Piece of Knowledge**)**.** Let $\mathcal{K}$ = '**IF** $\mathcal{S}$ **THEN** x **is same as** y' be a piece of 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
-

$$f_{\mathcal{K}}((v_1, v_2)) = \begin{cases} f_{\mathcal{S}}((v_1, v_2)) & \text{if } (v_1, v_2) \in \mathbb{E}_{\mathcal{S}} \\ \text{“}is\_same\_as\text{”} & \text{Otherwise} \end{cases}$$

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

Example 6. An example of a representation of a piece of knowledge is shown in Figure 2.

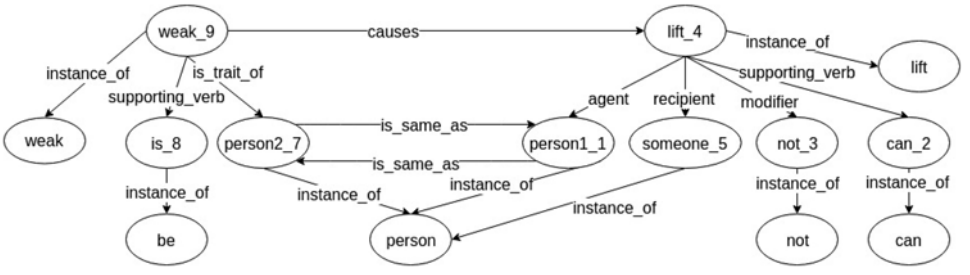

Fig. 2: Graphical Representation of the Knowledge, "**IF** *person1 can not lift someone because person2 is weak* **THEN** *person1_1 is same as person2_7*"

4 Reasoning with Commonsense Knowledge

In this work we defined a reasoning algorithm for solving the WSC problems. The algorithm takes graphical representations of a WSC problem and a piece of knowledge 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 defined the 'most natural resolution' and the answer of a WSC problem with respect to the graphical representations of a WSC problem and a piece of knowledge 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 $\mathcal{S}$, $\mathcal{G}'_{\mathcal{S}} = (\mathbb{V}'_{\mathcal{S}}, \mathbb{E}'_{\mathcal{S}}, f'_{\mathcal{S}})$

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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 piece of 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., $(a,i) \in \mathbb{E}_{\mathcal{S}}$, $(b,i) \in \mathbb{E}_{\mathcal{K}}$, $f_{\mathcal{S}}((a,i)) = instance\_of$ 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 7. Let us consider the representation of a piece of knowledge shown in the Figure 2, the representation of the sentences in a WSC problem as shown in the Figure 1. Then, according to the Definition 7, following is the value of the set of node pairs (i.e., $\mathbb{M}$).

$\mathbb{M} = \{$(*weak_12*, *weak_9*), (*lift_5*, *lifts_4*), (*he_9*, *person2_7*),(*man_2*, *person1_1*, (*son_7*, *someone_5*), (*was_10*, *is_8*), (*not_4*, *not_3*), (*could_3*, *can_2*)$\}$. We can see that (*he_9*, *person2_7*)$\in \mathbb{M}$ and (*man_2*, *person1_1*)$\in \mathbb{M}$, $(person1\_1, person2\_7)$ is an edge in the graphical representation of the knowledge with label is_same_as. Then, according to the Definition 7 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 piece of 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 8. Let us consider the representation of a piece of knowledge from Figure 2, the representation of WSC sentences from Figure 1, the token for pronoun to resolve is *'he_9'*, the tokens for answer choices are *'man_2'* and *'son_8'*. 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.

4.1 Winograd Schema Challenge Reasoning (WiSCR) Algorithm

Input to the 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

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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

Behavior of the Algorithm:

STEP 1: In this step a subgraph of $\mathcal{G}_{\mathcal{S}}$ is extracted. Let the extracted subgraph be named $\mathcal{G}_{\mathcal{S}}'$. $\mathcal{G}_{\mathcal{S}}'$ contains all the nodes which are not class nodes in $\mathcal{G}_{\mathcal{S}}$. All the edges which connect such nodes are also extracted. An example of the output of the Step 1 is shown in the Figure 3. 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.

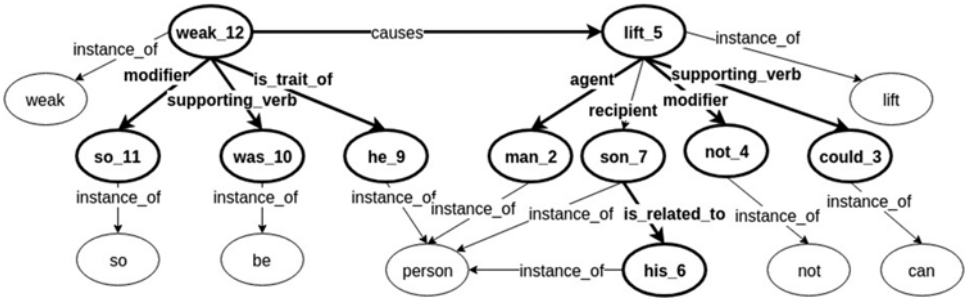

Fig. 3: An Example of Step 1 Output of the WiSCR Algorithm with Respect to the WSC Sentence *"The man could not lift his son because he was so weak."*

STEP 2: In this step a subgraph of $\mathcal{G}_{\mathcal{K}}$ is extracted. Let the extracted subgraph be 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 *'is_same_as'*. An example of the output of the Step 2 is shown in the Figure 4. The entire graph in the figure is the representation of a piece of knowledge (as shown in Figure 2) and the highlighted part of the graph is the subgraph extracted in this step.

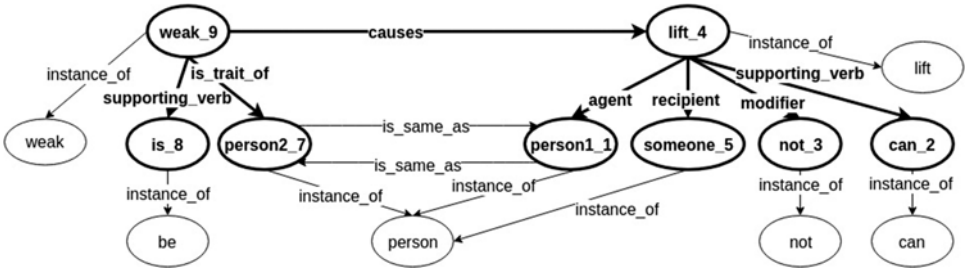

Fig. 4: Example of Step 2 Output of the WiSCR Algorithm

STEP 3: In this step, all 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

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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 a directed 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 a directed 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 piece of 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. Proof can be found in the supplementary material corresponding to this paper at the TPLP archives. □

4.2 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 of using ASP to implement the WiSCR algorithm is to make the process of adding new constraints easier. In this section, first we present the details of the ASP encoding of the inputs to WiSCR algorithm and an ASP implementation

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

of the WiSCR algorithm. Then we show, with the help of examples, how the current implementation can be easily updated to include new constraints.

4.2.1 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 piece of 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 piece of knowledge is represented as a graph. It is encoded in ASP by using a ternary predicate $\texttt{has\_k(h}_1\texttt{,l}_1\texttt{,t}_1\texttt{)}$, where $\texttt{h}_1$ and $\texttt{t}_1$ are two nodes and $\texttt{l}_1$ is an edge label of the directed edge from $\texttt{h}_1$ to $\texttt{t}_1$. 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.

4.2.2 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.

4.2.3 ASP implementation of the Step 2 of WiSCR Algorithm

In Step 2 of the WiSCR algorithm a subgraph of the graphical representation of a piece of 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.

4.2.4 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.

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

```
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).
```

`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.

4.2.5 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.
```

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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. This part of the algorithm is implemented in python. Let us call the python procedure which implements this part as AnswerFinder. AnswerFinder takes as input the answers generated by the ASP code and outputs the final answer based on the following conditions.

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

The WiSCR algorithm requires graph-subgraph isomorphism detection as a sub-module. Graph-subgraph isomorphism[3] is an NP-Complete problem. In recent times, there has been remarkable progress made in computing answer sets efficiently. Some of the popular answer set solvers are SModels[4], CModels[5] and Clingo[6]. In this work we used Clingo, which use techniques similar to the ones used in SAT solvers (Lin and Zhao 2004). The rest of the steps in the algorithm can be performed in polynomial time.

4.2.6 Adding New Constraints

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 and knowledge 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.
Knowledge: **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*.

[3] https://en.wikipedia.org/wiki/Subgraph_isomorphism_problem
[4] http://www.tcs.hut.fi/Software/smodels/
[5] http://www.cs.utexas.edu/users/tag/cmodels/
[6] http://potassco.sourceforge.net/

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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 piece of 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

(i) the facts of the form $has\_s(h_1, l_1, t_1)$ and $has\_k(h_2, l_2, t_2)$,
(ii) a fact of the form $pronoun(p)$,
(iii) two facts of the form $ans\_ch1(a_1)$ and $ans\_ch2(a_2)$,
(iv) 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 piece of 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 python procedure defined in Section 4.2.5. 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. Proof is present in the supplementary material at the TPLP archives. □

5 Experimental Evaluation of the WiSCR Algorithm

The main goal of the evaluation process is to 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 specified formats. We evaluated a corpus[7] of 291 WSC problems. In this section we present the three experiments which we performed to validate the WiSCR algorithm and our findings with respect to those experiments.

Experiment 1: In this experiment we manually created the input graphical representations of the WSC sentences and the needed knowledge. We found that the WSC

[7] Avaiable at https://tinyurl.com/y22ykz5p

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

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 type 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 likely to be true than the other. For example, **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: In this experiment we considered the 240 WSC problems that are handled by the WiSCR algorithm. The needed knowledge for all the 240 problems was manually written in the *'**IF** S **THEN** x **is same as** y'* format as mentioned in the Definition 5. Both, the WSC problems and the needed knowledge were automatically converted into graphs by using two K-Parser wrappers. The details of the K-Parser wrappers are provides in the paragraph below. 200 (82.98%) out of 240 problems were correctly answered in this experiment by the WiSCR algorithm. The remaining 40 problems were not answered because of syntactic dependency parsing errors and part-of-speech errors while generating the representations.

Two wrappers over K-Parser were developed as part of this work. The first translates a sequence of sentences into a graphical representation that satisfies the Definition 4. K-Parser produces a graph for an input English sequence of sentences. The only two differences between the K-Parser output and the representation in Definition 4 is that in K-Parser output there are two levels of class nodes instead of one and the K-Parser output contains semantic roles of entities. So, as part of this wrapper the two levels of classes was reduced to one by keeping the *superclasses* of *noun* and *pronoun* words and by keeping the classes which represent the lemmatized form of other types of nodes. The semantic roles are not considered by the wrapper. The second wrapper is used to

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

translate a knowledge of the form ***IF*** S ***THEN*** x ***is same as*** y where S is a sentence and x, y are tokens in S. In this wrapper the same modifications to the K-Parser output of S are made as were in the wrapper 1 along with the addition of two extra edges. An edge from the node representing x to a node representing y was added and labeled as *is_same_as* and another edge from y to x with same label is also added.

Experiment 3: In this experiment we used a technique to automatically extract the knowledge that is needed for the WSC problems which were correctly represented by using K-Parser. 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. 2015b). The idea there is to extract a set of sentences (by using a search engine) which are similar to the original WSC sentences in terms of the actions and properties in it. Such sentences are then parsed with the help of K-Parser to extract the knowledge. For example, a sentence extracted for the Winograd sentence shown in Figure 1 is *"She could not lift him because she is weak."*. And the knowledge extracted from the above sentence is *"IF person1 could not lift someone because person2 is weak THEN person1_1 is same as person2_7"*. Because of the limited availability of search engine access, the sentences similar to only 120 WSC sentences could be extracted. Those sentences are then passed to a rule based knowledge extraction module. The module uses the K-Parser outputs to find the patterns which satisfy the kind of knowledge handled by our reasoning algorithm.

6 Related Work

Over the years various approaches have been proposed to solve the Winograd Schema Challenge by using additional knowledge. Such works include the ones which focus on defining the reasoning theories (Bailey et al. 2015; Schüller 2014; Richard-Bollans et al. 2018; Wolff 2018). These approaches mention the need of additional knowledge and reasoning, but they suffer from the issue of low coverage on the WSC corpus.

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. 2018), and the ones which perform knowledge extraction with respect to a pre-populated knowledge base (Isaak and Michael 2016). These approaches rely on the heuristic procedures. More recently, composition embedding (Liu et al. 2017) and statistical language modelling (Radford et al. 2019) based approaches have been used to address the challenge. The later recently reported the state of the art accuracy (70.70%) on the overall corpus. 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 is 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.

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press

7 Conclusion

In this work, we attempted to solve the Winograd Schema Challenge by reasoning 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 how an approach built on top of graph-subgraph isomorphism encoded in ASP is able to tackle 240 out of 291 WSC problems. We presented how the ASP implementation of the algorithm allows us to add new constraints easily. It also makes the current implementation to easily generalize by adding new rules.

Supplementary material

To view supplementary material for this article, please visit https://doi.org/10.1017/S1471068419000334.

References

Bailey, D., Harrison, A., Lierler, Y., Lifschitz, V., and Michael, J. 2015. The winograd schema challenge and reasoning about correlation. In *In Working Notes of the Symposium on Logical Formalizations of Commonsense Reasoning*.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. 178–186.

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

Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence 26,* 10, 1367–1372.

Emami, A., De La Cruz, N., Trischler, A., Suleman, K., and Cheung, J. C. K. 2018. A knowledge hunting framework for common sense reasoning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1949–1958.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*. Vol. 88. 1070–1080.

Isaak, N. and Michael, L. 2016. Tackling the winograd schema challenge through machine logical inferences. In *STAIRS*. Vol. 284. 75–86.

Levesque, H. J., Davis, E., and Morgenstern, L. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. Vol. 46. 47.

Lin, F. and Zhao, Y. 2004. Assat: Computing answer sets of a logic program by sat solvers. *Artificial Intelligence 157,* 1-2, 115–137.

Liu, Q., Jiang, H., Evdokimov, A., Ling, Z.-H., Zhu, X., Wei, S., and Hu, Y. 2017. 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)*. 2344–2350.

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

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog 1*, 8.

Richard-Bollans, A., Gomez Alvarez, L., and Cohn, A. G. 2018. The role of pragmatics in solving the winograd schema challenge. In *Proceedings of the Thirteenth International Symposium on Commonsense Reasoning (Commonsense 2017)*. CEUR Workshop Proceedings.

Schüller, P. 2014. Tackling winograd schemas by formalizing relevance theory in knowledge graphs. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Sharma, A., Vo, N., Aditya, S., and Baral, C. 2015a. Identifying various kinds of event mentions in k-parser output. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. 82–88.

Sharma, A., Vo, N. H., Aditya, S., and Baral, C. 2015b. Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *IJCAI*. 1319–1325.

Wolff, J. G. 2018. Interpreting winograd schemas via the sp theory of intelligence and its realisation in the sp computer model. *arXiv preprint arXiv:1810.04554*.

https://doi.org/10.1017/S1471068419000334 Published online by Cambridge University Press