# Tackling Domain-Specific Winograd Schemas with Knowledge-Based Reasoning and Machine Learning

Suk Joon Hong
201298823

Supervised by Dr Brandon Bennett (School of Computing)

Submitted in accordance with the requirements for the
module MATH5872M: Dissertation in Data Science and Analytics
as part of the degree of

Master of Science in Data Science and Analytics

The University of Leeds, School of Mathematics

September 2020

# Abstract

The Winograd Schema Challenge (WSC) is an alternative to the Turing Test designed to test *thinking* ability of a machine (Levesque et al., 2012). WSC is a coreference resolution task requiring background knowledge to be used, which can be an evidence of *thinking* (Levesque et al., 2012). WSC can be tackled by knowledge-based reasoning and machine learning, and each method has limitations. In knowledge-based reasoning, building a knowledge base for general domain is too expensive (Bailey et al., 2015; Sharma, 2019). In machine learning, it lacks ability to explain the reason for the answer, and especially a deep learning approach, as a machine learning, has randomness (Dror et al., 2019; Kocijan et al., 2019).

Our aim is to tackle Winograd schemas in a specific domain by using both knowledge-based reasoning and machine learning. Our research covers the following three steps — defining a domain, developing a model and evaluating the experiments.

In the first step, we introduce a keyword approach to identify a domain in Winograd schemas and focus on the *thanking* domain (containing a word related to thanking). We used WinoGrande (Sakaguchi et al., 2019) which is a large Winograd schema data set to extract the domain-specific Winograd schemas. In the thanking domain specified by the keywords, the domain-specific high-level patterns were found. To our best knowledge, it is the first work to use keywords to specify a domain in WSC.

In the second step, we develop an advanced domain-specific high-level knowledge-based reasoning approach which uses these high-level patterns by modifying the approach of Sharma (2019). Furthermore, to mitigate limitations of knowledge-based reasoning, we propose a simple ensemble method to combine knowledge-based reasoning and machine learning. As a machine learning method, we chose Bidirectional Encoder Representations from Transformers (BERT) (Kocijan et al., 2019) which is a deep learning method for its high performance, and we combined it with our knowledge-based reasoning approach.

In the third step, we propose a 'robust' accuracy by improving the method of Trichelair et al. (2018) which reduces the probability of making a correct prediction by chance. In our experiments on the thanking domain, our ensemble approach gave a better performance than each single method in terms of both accuracy and the 'robust' accuracy.

The methods used in each step are not limited to the thanking domain, and these can also be used in other tasks such as Choice Of Plausible Alternatives (COPA) (Roemmele et al., 2011). As our research focuses on one specific domain with the small data set, future work needs to be done with larger data set to tackle many other domains in WSC.

# Acknowledgments

I would like to thank Dr Brandon Bennett for his supervising. His feedbacks and the discussions with him helped me to think critically about the previous approaches related to the Winograd Schema Challenge.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 The Winograd Schema Challenge (WSC)

The Winograd Schema Challenge (**WSC**) is proposed as an alternative of the Turing Test to measure whether a machine can think (Levesque et al., 2012). According to Levesque et al. (2012), this challenge is a coreference resolution task to predict what the pronoun in a sentence refers to with the special form of a Winograd schema. The following example is from the original WSC data set (**WSC273**) (Levesque et al., 2012), and it is a representative Winograd schema (Levesque et al., 2012, p.554, my italics. boldness mine):

1. The trophy doesn't fit in the brown suitcase because **it** is too *large*.
    - **Candidates for the pronoun:** the trophy / the suitcase, **answer:** the trophy
2. The trophy doesn't fit in the brown suitcase because **it** is too *small*.
    - **Candidates for the pronoun:** the trophy / the suitcase, **answer:** the suitcase

As shown in the example, a Winograd schema that Levesque et al. (2012) propose refers to a pair of sentences that are almost same, but what the pronoun in each sentence indicates is different. In this example, as the authors state, the pronoun "it" in the first sentence refers to "the trophy", but the same pronoun in the second sentence refers to "the suitcase", which can be easily answered by non-experts.

Levesque et al. (2012, p.554) explain why this difference is possible by introducing the concepts of "*special* word" and "*alternative* word". According to the authors, these concepts mean that what the pronoun refers to changes depending on whether the special word is used or the alternative word is used. In this example, the authors specify that the special word is "large" and the alternative word is "small", and we can see that these two words are the only different parts between the two sentences.

In addition to these characteristics, Levesque et al. (2012) want background knowledge to be used to resolve Winograd schemas as WSC is to see the evidence of *thinking*. The authors assume that a machine is considered thinking if it is able to solve the problem by using knowledge that does not appear in the problem. In this sense, the authors point out that a Winograd schema should not be resolved by statistical association between special word (or alternative word) and a candidate in the same sentence.

For instance, as Levesque et al. (2012) intend, the example of the trophy and the suitcase seems to meet this *thinking* condition as the trophy and the suitcase would have a similar statistical association with large/small. The authors specify that it would require some background knowledge similar to the fact that something that is too large is not able to fit into the thing that is very small.

On the other hand, there can be a pronoun resolution case that does not require background knowledge in order to know the answer. An example sentence is given by Rahman and Ng (2012, p.781):

Lions eat zebras because they are predators.

In this example, the candidates for the pronoun "they" are the lions and the zebras, and the special word can be "predators". Since the lions would have higher statistical association with "predators", this pronoun ambiguity can be easily resolved without any other background knowledge. According to the conditions given by Levesque et al. (2012), this sentence should not be a Winograd schema sentence as it does not require *thinking*.

## 1.2 The Turing Test v. WSC

The Turing Test was introduced earlier than WSC to measure *thinking* ability of a machine (Turing, 1950). In the Turing Test, an interrogator communicates with two participants composed of one human and one machine, and if the interrogator cannot distinguish between the machine and the human, that machine is considered to *think* (Turing, 1950). The Turing Test contributes to introducing a behavioural evaluation on machine's intelligence instead of philosophical approaches (Turing, 1950; Levesque et al., 2012).

However, the Turing Test reveals some problems. Levesque et al. (2012) point out that Turing Test forces a machine to pretend to be a human, which is based on deception. The other point that the authors make is that the evaluation is subjective because interrogators can give different judgements even on the same dialogue.

Different from the Turing Test, Levesque et al. (2012) designed WSC in a way that native English speakers who are not experts can resolve Winograd Schemas and there is a definite answer for every Winograd schema. The authors argue that these characteristics of WSC would remove the room for subjectiveness. In addition, as the authors state, WSC does not need to deceive people as well.

## 1.3 The contribution

This thesis has the following four contributions:

- We introduced a keyword approach to specify a domain in Winograd schemas, and the thanking domain was chosen as an example. To our best knowledge, our approach is the first work to specify a domain by keywords and investigate patterns within the domain.
- We developed a high-level domain-specific knowledge-based reasoning approach by modifying Sharma's (2019) approach.
- We proposed a simple method of combining knowledge-based reasoning and machine learning which outperforms each single method.
- We designed 'robust' accuracy to evaluate solutions for Winograd schemas by upgrading Trichelair et al.'s (2018) method.

This thesis is composed of seven chapters in total. After this introduction chapter, the second chapter gives literature review. Chapter 3 presents how we specified the thanking domain by the keywords. Chapter 4 and 5 introduce the methods we used for the experiments which are the high-level domain-specific knowledge-based reasoning approach, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018; Kocijan et al., 2019) and the ensemble approach. In Chapter 6, evaluation metrics including the 'robust' accuracy, the experiment design and analyses of the results are covered. Lastly, Chapter 7 includes conclusion, other applications, limitations and future work related to this research.

# Chapter 2

# Literature Review

There are two main approaches to tackle Winograd Schema Challenge (WSC) — knowledge-based reasoning and machine learning. In this chapter, three approaches that are relevant to our approach are explained with their limitations. As knowledge-based reasoning approaches, the approaches of Sharma (2019) and Bailey et al. (2015) are covered. As a machine learning approach, Bidirectional Encoder Representations from Transformers (**BERT**) (Devlin et al., 2018; Kocijan et al., 2019) is reviewed.

## 2.1 Sharma's approach to knowledge-based reasoning

The author in this section refers to Sharma (2019). In this section, the author's approach which uses knowledge-based reasoning is reviewed. The author's method and limitations are explained with some examples.

### 2.1.1 Method

The author proposes a knowledge-based reasoning method to tackle Winograd schemas. The author's method is composed of parsing a Winograd schema sentence, defining background knowledge principles and defining rules to derive the answer by reasoning. The strong point of the author's approach is that these three parts can be implemented with the actual programs — K-Parser (Sharma et al., 2015a) and Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988; Mueller, 2014). Each part is explained below along with the implementation methods.

1. **Parsing a Winograd schema sentence:** Whether it is a machine learning method or a knowledge-based reasoning method, a Winograd schema sentence needs to be represented in a way that a model can use. In machine learning, a natural language sentence is normally tokenised and each token is represented as a vector (Mikolov et al., 2013; Devlin et al., 2018). On the other hand, in knowledge-based reasoning, a sentence is represented into a logical form (Bailey et al., 2015; Sharma, 2019). The author develops K-Parser

which transforms a sentence into a graphical representation, and uses it to parse Winograd schema sentences. The author's graphical representation is composed of multiple predicates, and each predicate is a form of *predicate(node1, edge, node2)*. In the author's definition, a node can be token, instance, or class and an edge is a relation between the two nodes. A sentence from WinoGrande (Sakaguchi et al., 2019) that is not addressed in Sharma (2019) can be parsed by K-Parser as:

- The sentence: Kayla cooked sticky white rice for Jennifer, and [she] was thanked for making such delicate rice.
- The output of K-Parser on the sentence using the predicate "`has_s`" (Sharma, 2019, p.9):

```
has_s(rice_5,destination,jennifer_7).
has_s(making_14,recipient,rice_17).
has_s(rice_5,trait,white_4).
has_s(rice_17,trait,such_15).
has_s(cooked_2,agent,kayla_1).
has_s(rice_5,trait,sticky_3).
has_s(cooked_2,recipient,rice_5).
has_s(thanked_12,recipient,she_10).
has_s(thanked_12,passive_supporting_verb,was_11).
has_s(thanked_12,objective,making_14).
has_s(rice_17,trait,delicate_16).
has_s(kayla_1,instance_of,kayla).
has_s(kayla,is_subclass_of,person).
has_s(thanked_12,instance_of,thank).
has_s(thank,is_subclass_of,communication).
has_s(cooked_2,instance_of,cook).
has_s(cook,is_subclass_of,creation).
has_s(white_4,instance_of,white).
has_s(white,is_subclass_of,color_descriptive).
has_s(delicate_16,instance_of,delicate).
has_s(delicate,is_subclass_of,quality_descriptive).
has_s(rice_17,instance_of,rice).
has_s(rice,is_subclass_of,food).
has_s(was-11,instance_of,be).
has_s(be,is_subclass_of,stative).
has_s(she_10,instance_of,she).
has_s(she,is_subclass_of,person).
has_s(jennifer_7,instance_of,jennifer).
has_s(jennifer,is_subclass_of,person).
has_s(making_14,instance_of,make).
has_s(make,is_subclass_of,social).
has_s(rice_5,instance_of,rice).
has_s(rice,is_subclass_of,food).
has_s(cooked_2,next_event,was_11).
has_s(rice_17,semantic_role,creation).
has_s(kayla_1,semantic_role,cook_agent).
has_s(rice_5,semantic_role,food).
has_s(she_10,semantic_role,thanked).
```

- The graph drawn by us that represents the predicates of the sentence given by K-Parser :

After the sentence is transformed into the graphical representation, the author defines which nodes refer to the two candidates and the pronoun. The author uses the three predicates to define them: the predicate `pronoun` for the pronoun and the predicates `ans_ch1` and `ans_ch2` for the candidates. In this example, they can be defined as:

```
pronoun(she_10).
ans_ch1(kayla_1).
ans_ch2(jennifer_7).
```

2. **Defining background knowledge principles:** The author defines background knowledge principles and use them as a knowledge base for reasoning. The author's (p.5) background knowledge principles have a form as:

   **IF** [condition($x$, $y$)] **THEN** $x$ is same as $y$.

   In this form, the author states thats $x$ and $y$ are nouns and condition($x$, $y$) is a sentence which includes the two nouns. For example, the background knowledge principle needed to resolve the sentence can be written in the author's style as:

   - **IF** person1 cooks rice for someone, and person2 is thanked **THEN** person1 is same as person2.
   - This background knowledge principle can be represented in ASP by using the predicate "`has_k`" (Sharma, 2019, p.10) as:

```
has_k(cooked_2,agent,person1_1).
has_k(cooked_2,recipient,rice_3).
has_k(rice_3,destination,someone_5).
has_k(thanked_9,recipient,person2_7).

has_k(person1_1,instance_of,person1).
```

```
    has_k(person2_7,instance_of,person2).
    has_k(someone_5,instance_of,someone).
    has_k(cooked_2,instance_of,cook).
    has_k(rice_3,instance_of,rice).
    has_k(is_8,instance_of,be).
    has_k(thanked_9,instance_of,thank).

    has_k(person1,is_subclass_of,person).
    has_k(person2,is_subclass_of,person).
    has_k(someone,is_subclass_of,person).
    has_k(cook,is_subclass_of,creation).
    has_k(rice,is_subclass_of,food).
    has_k(be,is_subclass_of,stative).
    has_k(thank,is_subclass_of,communication).

    has_k(person1_1,is_same_as,person2_7).
    has_k(person2_7,is_same_as,person1_1).
```

The author uses two ways to acquire such background knowledge principles. The first method the author uses is to define these principles manually. The second method that the author uses is to extract these principles automatically from a search engine (Sharma et al., 2015b; Sharma, 2019).

3. **Reasoning using Answer Set Programming:** The author uses the graphical representations of the Winograd schema sentence and the background knowledge principle to resolve the pronoun. As the first step of the reasoning, the author defines the nodes and the edges from each graphical representation. Then, the goal of the author's reasoning is to find the matching nodes in the sentence which match with the nodes person1 and person2 in the background knowledge principle. From the matches, the author infers that as the background knowledge principle states that person1 is same as person2, the sentence node matching with the background knowledge node person1 would be same as the sentence node matching with the background knowledge node person2.

   In this example, the matches can be represented with the predicate "matches" (Sharma, 2019, p.11) which is composed of the sentence node and the background knowledge node. The following matches are the outputs using the example's sentence representation and background knowledge principle by the author's reasoning method:

```
matches(kayla_1, person1_1).
matches(she_10, person2_7).
matches(jennifer_7, someone_5).
matches(cooked_2, cooked_2).
matches(rice_5, rice_3).
matches(thanked_12, thanked_9)).
matches(cook, cook).
matches(rice, rice).
matches(be, be).
matches(thank, thank).
```

From these matches, it can be derived that `kayla_1` matches with `person1_1` and `she_10` matches with `person2_7`. Thus, the pronoun "she" would refer to "Kayla" in this example.

### 2.1.2 Discussions

The author contributes to tackling Winograd schemas with knowledge-based reasoning mainly in two aspects. The first contribution is that the author's approach can be used in general domains in Winograd schemas. The second contribution is that the author's way of representation and reasoning can be automated by programs such as K-Parser and ASP.

However, the author's approach also has limitations. As the author points out, K-Parser can make errors when a sentence is transformed into a graphical representation, and the author's definition of background knowledge principle cannot be applied to resolve some Winograd Schemas. The following two sentences are examples from WSC273 (my italitcs) that is not addressed in the author's paper, and these examples demonstrate the case when the background knowledge style of the author does not make sense:

- **Sentence1:** Look! There is a *shark* swimming right below that duck! **it** had better get away to safety fast!
  **Candidates:** the shark/ the duck, **Answer:** the duck

- **Sentence 2:** Look! There is a *minnow* swimming right below that duck! **it** had better get away to safety fast!
  **Candidates**: the shark/ the duck, **Answer:** the minnow

- **A background knowledge principle in the rule representation of the author's style :**
  **IF** there is something swimming right below something1 and something2 had better get away to safety. **THEN** something1 **is same as** something2.

In these examples, the author's approach is successful in tackling only one of the two sentences. The correct answer("duck") for the first sentence can be derived from the background knowledge principle, but it predicts the correct answer for the wrong reason. The used background knowledge principle does not explain the true reason for why "duck" in the example had better get away. The condition that something is *below* “duck” in the example does not necessarily mean that “duck” need to get away. What matters in this context is what is something below that “duck” in the example. That is why in the second sentence, the pronoun should refer to “minnow”, not “duck” though that minnow is also below that duck. But the background knowledge principle which cannot explain this real reason would predict the pronoun in the second sentence to be “duck” which is a wrong anwer in this case.

In order to resolve both Winograd schema sentences, ‘high-level’ background knowledge principles are required, which the author’s approach does not use. In this case, some high-level background knowledge such as “**IF** something1 which is *weaker* than something3 is *near* something3, and something2 had better get away to safety **THEN** something1 is same as something2” is needed. Also, the higher-level relationships such as “weaker” or “near” need to be derived from the given sentences. But the author does not define high-level background knowledge principles which do not appear explicitly. In the same way, the author does not derive implicit high-level relationships in a given Winograd schema as well. In other words, The author’s approach considers only low-level knowledge.

In addition to the limitations of K-parser and the low-level knowledge, in terms of acquiring background knowledge principles, the two methods of the author also have limitations. The first method of the author is to define background knowledge principles manually, but it is too costly to define all possible background knowledge principles applicable to general domain. The second method of the author is to extract background knowledge principles from a search engine, but *not* all necessary background knowledge principles can be extracted.

## 2.2 Bailey’s approach to knowledge-based reasoning

The authors in this section refers to Bailey et al. (2015). This section is to explain the authors’ apporach which is also a knowledge-based reasoning method. The method of the authors is explained and the limitations of their method are discussed.

### 2.2.1 Method

The authors suggest a method using correlation and first-order logic (Brachman and Levesque, 2004) to tackle Winograd schemas. The authors (p.18) define a “correlation formula” as:

$$F \oplus G$$

where $F$ and $G$ refer to formulas written in first-order logic which can be used to represent sentences. The authors describe that the correlation formula means correlation exists between $F$ and $G$.

In terms of natural language sentences, the “correlation” that the authors (p.18) define is that either one of the sentences is more plausible if the other sentence is given. The authors suggest that the correlation can be used to explain the answer for Winograd schemas. For example, this is an example from WSC273 that can be explained with the correlation according to the authors' approach:

Jim yelled at Kevin because **he** was so upset.

The pronoun **he** in the sentence needs to refer to either Jim or Kevin, and Jim is the answer for this sentence. If the pronoun is replaced with the answer, the first clause of the example "Jim yelled at Kevin" would be correlated with the second clause "Jim was so upset." In other words, if there is a clause that someone yells, the other clause that the same person is upset is *more plausible*.

Regarding the correlation, the authors (p. 18) give the five inference rules, and as one of them, the implication rule is given as:

$$\frac{\widetilde{\forall}(F \rightarrow G)}{F \oplus G}$$

According to the authors, this rule means that if a formula $G$ can be implied by a formula $F$, the correlation exists between $F$ and $G$.

Using the correlation and the inference rules, the authors can resolve some Winograd schemas. The authors' approach involves the three parts - representing a Winograd schema sentence, defining background knowledge principles and reasoning to derive the answer. In the explanation, an example from WSC273 is used which Sharma (2019) cannot resolve. This example is not addressed in the authors' paper, but it can be resolved by the authors' approach.

1. **The example sentence from WSC273 (the pronoun "she" was replaced with "Mary")**: Mary tucked her daughter Anne into bed, so that [Mary] could work.

   (if this sentence with the candidate "Mary" inserted into the pronoun can be derived by the authors' correlation reasoning, the answer for the pronoun is considered to be "Mary")

2. **Representing a Winograd schema sentence:** The authors represent a Winograd schema sentence into a form of the correlation formula. Different from Sharma (2019), the authors manually represent natural language sentences into the forms of first-order logic, and connect them with the correlation $\oplus$. The example sentence can be represented in the authors' style as:

   $$(\texttt{parent\_to(M,D)} \wedge \texttt{tuck(M,D)}) \oplus \texttt{work(M)}$$

   with the presuppositions: `(1)mary(M)`, `(2)daughter_of_mary(D)`.

3. **Defining background knowledge principles:** The authors define different levels of background knowledge principles while Sharma (2019) gives only low-level knowledge. It means that the authors' method can be used to tackle more complicated Winograd schmeas.

In this case, the relevant background knowledge principles that Sharma (2019, p.14) gives as an example in plain English can be defined in the first-order logic form as:

(3) $\forall x \forall y(\texttt{tuck(x,y)} \rightarrow \texttt{sleep(y)})$

- **It means:** someone who is tucked into bed, may sleep (Sharma, 2019, p.14)

(4) $\forall x \forall y((\texttt{parent\_to(x,y)} \land \texttt{sleep(y)}) \rightarrow \texttt{work(x)})$

- **It means:** someone who's daughter is sleeping may be able to work (Sharma, 2019, p.14)

4. **Reasoning to derive the answer:** The authors use correlation, the inference rules, presuppositions and background knowledge principles to derive the answer. If the authors' approach can derive the representation of the Winograd schema sentence whose pronoun is replaced with one of the candidates, the candidate used in that derived representation is considered to be the answer. The following steps demonstrate the process to resolve the example:

   (5) $\forall x \forall y((\texttt{parent\_to(x,y)} \land \texttt{tuck(x,y)}) \rightarrow \texttt{work(x)})$
   Formula (5) is entailed by (3) and (4).

   (6) $(\texttt{parent\_to(M,D)} \land \texttt{tuck(M,D)}) \rightarrow \texttt{work(M)}$
   Formula (6) is derived by substitution from (5).

   (7) $(\texttt{parent\_to(M,D)} \land \texttt{tuck(M,D)}) \oplus \texttt{work(M)}$
   Formula (7) is derived by the implication rule.

   In this example, the representation of the sentence whose pronoun is replaced with "Mary" can be derived since formula (7) is same as the representation. It means that this Winograd schema sentence can be resolved by Bailey et al.'s (2015) method while the method of Sharma (2019) cannot resolve it.

### 2.2.2 Discussions

As shown in the background knowledge principles in the previous subsection and those in the authors' paper, the authors' correlation method can use different levels of background knowledge principles to tackle Winograd schemas. Different from Sharma's (2019) method, the authors' usage of different levels of background knowledge principles makes it possible to consider implicit reasons in Winograd schemas. In the example of the previous subsection, the reason that "Mary could work" is that "her daughter Anne" would fall asleep because Ann was tucked. This implicit fact can de considered in the authors' method, but not in Sharma's (2019) method.

However, the authors' method has main limitations with respect to automation and unstructured background knowledge principles. In terms of representing a Winograd schema sentence into the correlation formula, this transformation is done by the authors manually. The authors do not give specific rules or an automation tool regarding this representation process while

Sharma (2019) uses K-Parser for the automation. With respect to defining background knowledge principles, the authors do not specify what kinds of knowledge compose their background knowledge principles, which makes it hard for the authors' method to be generalisable. In addition, even though the authors suggest a way to define some background knowledge principles automatically by using predefined ontologies, the authors do not give an automatic method to extract other background knowledge principles needed.

## 2.3 Bidirectional Encoder Representations from Transformers (BERT) to machine learning

Bidirectional Encoder Representations from Transformers (**BERT**), which is a state-of-the-art deep learning (a subset of machine learning) method, is proposed as a language model which can be used for different natural language tasks (Devlin et al., 2018). BERT is widely used in natural language processing as BERT and its variants show high performance on many different tasks (Devlin et al., 2018). BERT and its variants are also used to tackle Winograd Schema Challenge (Kocijan et al., 2019; Sakaguchi et al., 2019).

### 2.3.1 Devlin's BERT as a language model

The authors in this subsection refer to Devlin et al. (2018). In order to give word representations, the authors pre-trains BERT with the two objectives: 1) predicting the masked word(s), 2) predicting the next sentence. The authors argue that the first objective enables word representation to capture more contexts because this method is "bidirectional" (no pagination). The following example is from Levesque et al. (2012, p.554) and this example is modified to the form of the authors' training method for the first objective:

- **input:** The trophy doesn't fit in the brown suitcase because [the] [MASK] is too large.
- **label for [MASK]:** trophy

As "Bidirectional" is in BERT's name, the authors intend to enable BERT to consider contexts from both sides, and as shown in this example, to predict the masked word, both the part before it and the part after it can be considered in the model. In the authors' BERT, it is trained with a large corpus such as English Wikipedia and in a similar way to this example, 15% of words in a sentence are randomly masked for training.

While this example sentence is from WSC273, the authors' BERT is not trained with any Winograd schema data set. The reason for giving this example is to show how BERT's training process is similar to tackle Winograd Schema Challenge. Basically, a well-trained BERT would give higher likelihood to "trophy" compared to "suitcase" in this example, which implies that BERT's training method can be easily compatible for resolving Winograd schemas.

As a result of the pre-training, each word in a sentence can have representations as a vector, which the authors intend to achieve. While previous language models such as Word2Vec (Mikolov et al., 2013) can give only the same representation of a word regardless of its context, as the authors enable BERT to consider context of each word, BERT can give different representations for the same word depending on the context. The following two sentences are from WSC273, and "bank" is used differently in the two sentences (boldness mine):

1. The customer walked into the **bank** and stabbed one of the tellers. He was immediately taken to the police station.
2. Archaeologists have concluded that humans lived in Laputa 20,000 years ago. They hunted for evidence on the river **bank**[ ].

In this example, Word2Vec (Mikolov et al., 2013) would represent "bank" in the two sentences as the same vectors, but BERT would give different representations for each "bank" in the two sentences depending on the contexts. As the first "bank" refers to a financial building and the second "bank" refers to the land near a river, it makes more sense to represent them differently as BERT does.

After the pre-training, the authors then fine-tune the pre-trained BERT with a task-specific supervised learning. The next subsection gives how fine-tuning is used to tackle Winograd Schema Challenge (Kocijan et al., 2019)

### 2.3.2 Kocijan's application of BERT in WSC

Devlin et al. (2018) demonstate that it is more effective to fine-tune a pre-trained BERT to tackle a specific task than training a model from scratch. This BERT's approach which has pre-training and fine-tuning is a form of transfer learning since it uses the trained parameters from one purpose (pre-training) to be initial state of training for another purpose (fine-tuning) (Devlin et al., 2018). Transfer learning is known to be useful especially when a specific task has a small data set. (Devlin et al., 2018).

In Winograd Schema Challenge(WSC), Kocijan et al. (2019) show that using BERT for transfer learning can be effective to tackle Winograd schemas. Kocijan et al. (2019) use the pre-trained BERT (Devlin et al., 2018) for the initial state of fine-tuning, and in the fine-tuning they use data sets similar to Winograd schemas. Kocijan et al. (2019, p.3) also define a WSC-specific loss function for training:

$$L = -\log \mathbb{P}(c_1|s) + \alpha \ \cdot max(0, \log \mathbb{P}(c_2|s) - \log \mathbb{P}(c_1|s) + \beta) \tag{2.1}$$

In this formula, the authors state that $c_1$ is the correct candidate, $c_2$ is the wrong candidate, $s$ is the given sentence, and $\alpha$ and $\beta$ refer to hyperparameters. It needs to be noted that $-\log \mathbb{P}(c_1|s)$ has a negative trend and $\log \mathbb{P}(c_2|s)$ has a positive trend in this formula. As the objective of the training is to reduce the loss, Formula 2.1 can be understood to favor high likelihood of the correct candidate($c_1$) and punish high likelihood of the wrong candidate($c_2$). In this regard, this formula reflects the characteristics of a Winograd schema sentence which has two candidates.

As a result, Kocijan et al. (2019) show that the accuray of their model is $72.5\%$ on WSC273 which is higher than that of BERT (Devlin et al., 2018) by $10.6\%$. From this result, Kocijan et al. (2019) prove that their method of the fine-tuning can upgrade BERT (Devlin et al., 2018) in terms of tackling Winograd schemas.

### 2.3.3 Limitations

BERT seems to be better than a mere luck as it can have accuracy over 70% (Kocijan et al., 2019). It is advantages that BERT, as a machine learning, can resolve certain amount of Winograd schemas without manual work to build a knowledge base (Kocijan et al., 2019). Nevertheless, it needs to be noted that each Winograd schema sentence can be resolved by 50% chance as there are only two candidates (Levesque et al., 2012). In this sense, BERT's performance shown as accuracy alone cannot be a solid ground that the model has a real understanding of the problems (Trichelair et al., 2018).

Moreover, BERT shows weakness to understand function words such as negation (Ettinger, 2019). BERT seems to be not the only machine learning method which lacks understanding of function words as Saba (2018) raises the same issue on another machine learning model. BERT and the other machine learning show the same problem of giving the similar likelihood for the word or the phrase though the function word is changed (Saba, 2018; Ettinger, 2019).

Two examples from WSC273 are given in Table 2.1 to show that BERT does not seem to understand negation. We modified the original sentences in Table 2.1 by adding negation("not") to each original sentence. The sentences with negation type in Table 2.1 are the modified sentences. It is found that each negated sentence's answer is different from the corresponding original sentence's answer. However, it is interesting that Kocijan et al.'s (2019) best performing BERT gives the same predictions on both each original sentence and the corresponding negated sentence. The results of these examples in Winograd schemas also support Ettinger's (2019) finding that BERT does not seem to distinguish function words such as "not" from other normal words. Informally, if two sentences look similar, BERT would give the same predictions regardless of the differences caused by function words such as negation.

Other than the function words, another limitation of BERT is its lack of explaining the reasons for the answer of Winograd schemas as deep learning approaches in general have low interpretability (Kocijan et al., 2019; Kowsari et al., 2019). The only explanation that BERT can give is that one candidate is more likely to be the answer for a given Winograd schema sentence than the other candidate (Kocijan et al., 2019). Even though the current Winograd Schema Challenge(WSC) does not require explanation for the answer (Levesque et al., 2012), explainability can matter in terms of 'thinking' which is what WSC wants to measure. This is supported by the fact that humans have a logical system ("System 2") as well as intution ("System 1") when they think (Kahneman, 2012, p. 13).

| Type | Sentence | Prediction | Answer |
|---|---|---|---|
| Original | Dan had to stop Bill from toying with the injured bird. **He** is very compassionate. | **Dan** | Dan |
| Negation | Dan had to stop Bill from toying with the injured bird. **He** is **not** compassionate. | **Dan** | Bill |
| Original | I can't cut that tree down with that axe; **it** is too small. | **The tree** | The axe |
| Negation | I can't cut that tree down with that axe; **it** is **not** small. | **The tree** | The tree |

*Table 2.1: Two Examples from WSC273 with each variant by negation*

Another limitation of BERT is unpredictability of when it would give correct predictions for a Winograd schema since deep learning which BERT is based on is *not* deterministic (Dror et al., 2019; Kocijan et al., 2019). It means that BERT's prediction can be wrong, but when it would give a wrong prediction cannot be predicted. In addition, in order to update a BERT model, the whole parameters or at least some existing parameters need to be updated by training (Devlin et al., 2018; Kocijan et al., 2019). In spite of this training for the update, it is not guaranteed that the updated BERT would be successful in predicting correctly on a specific example from the test set (Dror et al., 2019; Kocijan et al., 2019).

The other limitation of BERT is that the number of parameters of the large-sized model is 340 million parameters (Devlin et al., 2018; Kocijan et al., 2019). This implies that a large data set and a long training time would be required to train BERT (Devlin et al., 2018; Kocijan et al., 2019).

## 2.4 Summary

For the knowledge-based reasoning methods, building a knowledge base is necessary to tackle Winograd schemas. This is related to the characteristic of knowledge-based reasoning that only when there is a relevant knowledge in the knowledge base, the answer for a Winograd schema can be derived (Bailey et al., 2015; Sharma, 2019). Because of this characteristic, if the answer can be derived, that answer is highly likely to be accurate, but there can also be no answer if the knowledge base has no background knowledge principles needed (Bailey et al., 2015; Sharma, 2019).

In order to build a knowledge base, background knowledge principles need to be defined manually or extracted automatically (Bailey et al., 2015; Sharma, 2019). However, building a knowledge base to cover a general domain manually is too expensive, and even when the automation methods are used, the extracted knowledge is incomplete for a general domain (Bailey et al., 2015; Sharma, 2019). It implies that tackling a specific domain in WSC is more achievable with the more complete knowledge base in the specific domain in terms of knowledge-based reasoning.

The other aspects related to the knowledge-based reasoning methods are automation and levels of background knowledge principles. In terms of the automation, Sharma's (2019) method is more advantageous as it uses K-Parser and ASP for the automation. Regarding the background knowledge principles, Bailey et al. (2015) can use different levels of background knowledge principles, but Sharma (2019) only uses low-level background knowledge principles. Using different levels of knowledge is found to be more useful to tackle more complicated Winograd schemas (Bailey et al., 2015; Sharma, 2019).

In machine learning, on the other hand, the answer can always be predicted based on the likelihood without a knowledge base (Kocijan et al., 2019). BERT can be trained to tackle Winograd schemas with a large corpus or the data sets similar to Winograd schemas, which enables BERT to learn likelihoods for each candidate to be in the pronoun's position (Devlin et al., 2018; Kocijan et al., 2019). Between the two BERT models, the fine-tuned BERT (Kocijan et al., 2019) is found to be more accurate on WSC273 than the original BERT (Devlin et al., 2018).

However, BERT has limitations in that its prediction can be wrong and it cannot give a logical explanation for the answer. Ettinger (2019) and Saba (2018) point out that BERT does not seem to understand function words such as negation, which is one of the reasons for a wrong prediction. This is supported by our finding in Table 2.1 in which BERT does not make a correct prediction when the negation is added or removed from the Winograd schema sentences. In addition, BERT cannot specify which background knowledge is used for the answer, and its only explanation is that one candidate is more plausible than the other candidate (Devlin et al., 2018; Kocijan et al., 2019)

Based on the strong points and the weak points of machine learning and knowledge-based reasoning, our ensemble apporach uses both methods to mitigate each weakness. We also combine good components within the knowledge-based reasoning methods. In our advanced knowledge-based reasoning method, we use the automation methods such as K-Parser and ASP in Sharma's (2019) method, but use different levels of background knowledge principles as in Bailey et al.'s (2015) method. In addition, considering the limitation of building an expensive knowledge base for a general domain, we focus on a specific domain in Winograd schemas.

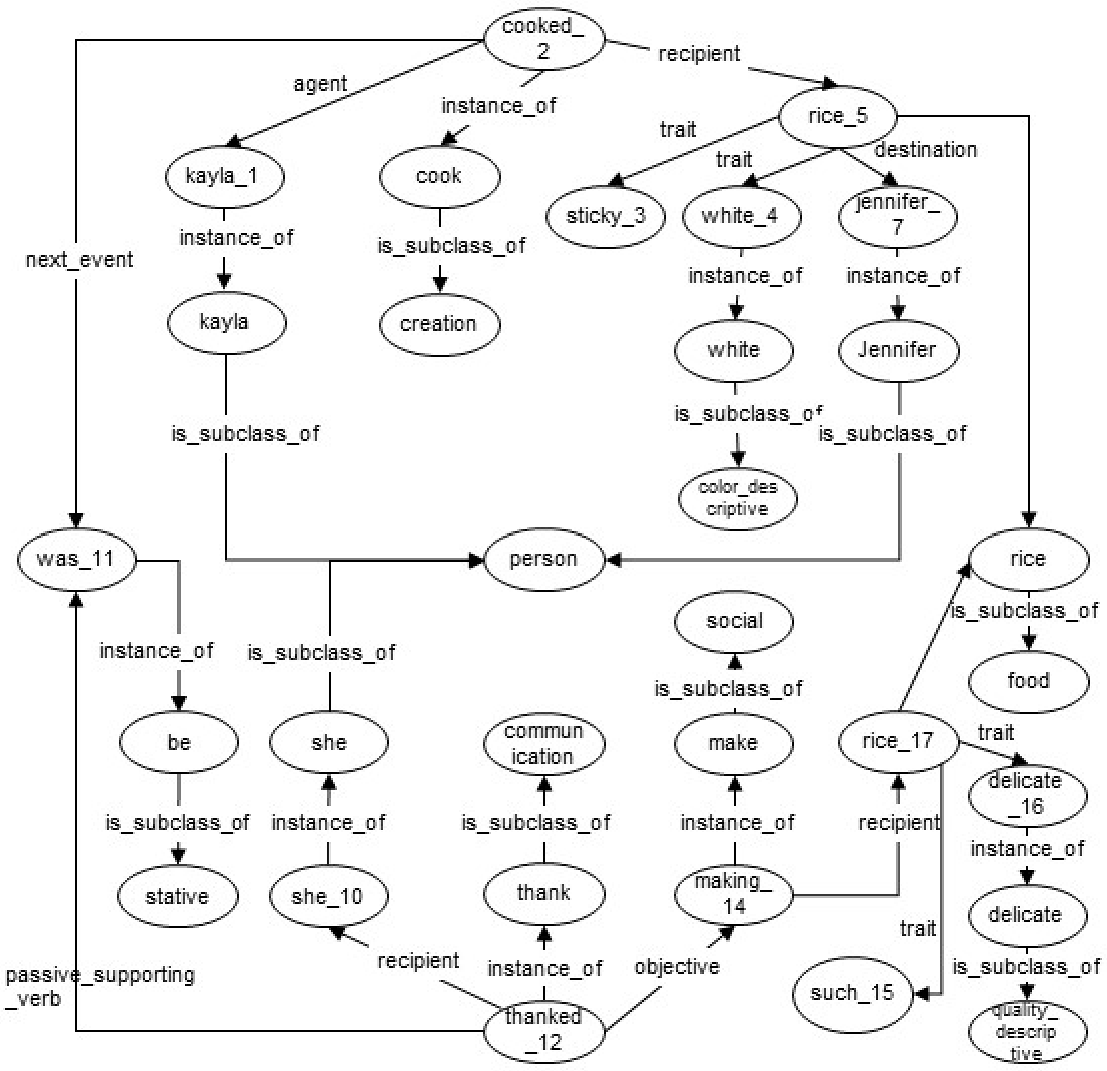

*Figure 2.1: The graphical representation of the sentence which is from the output of the predicates by K-Parser*

# Chapter 3

# Data: Domain-specific Winograd schemas relating to thanking

Winograd schemas can be divided into many domains and a specific domain can be targeted (Levesque et al., 2012). This domain specific approach is especially needed for knowledge-based reasoning. This is because it would cost too much manual works to build a knowledge base covering all domains without targeting a specific domain as discussed in Chapter 2 (Bailey et al., 2015; Sharma, 2019). Furthermore, even when the aim is to cover all domains ultimately, it can be more attainable to divide general domain into many specific domains as a first step and target one domain at a time.

On the other hand, machine learning approaches do not seem to need to target a domain. This is related to the fact that the BERT's variant shows the accuracy around 90% on the original WSC273 which covers general domain (Sakaguchi et al., 2019). This high performance is driven by using a large corpus and the datasets similar to Winograd schemas for the training without manual involvement (Sakaguchi et al., 2019).

Though this is a strong point of machine learning, these high-performing machine learning approaches cannot give a logical explanation for the reason, and they do not seem to understand logical meaning of function words as discussed in Chapter 2 (Saba, 2018; Devlin et al., 2018; Ettinger, 2019; Sakaguchi et al., 2019). In specific domains, these weakness cannot be accepted if these methods are modified to be used in real applications. Suppose these methods become a ground for medical or legal decisions. We cannot accept the machine learning methods' decisions in these important matters without knowing the reasons. In this sense, it is still worthy to use knowledge-based reasoning that can explain reasons as well as machine learning at least in specific domains.

In order to mitiage the difficulty of building a general knowledge base, we propose a method to specify a domain in Winograd schemas by using keywords. As a representative case, we chose a *thanking* domain which is neithor too narrow nor too wide. To our best knowledge, this thesis is the first work to use keywords to specify domains in Winograd schemas and investiage high-level background knowledge principles that are applicable to a specific domain.

## 3.1 Our Keyword approach to identify a domain

We used keywords to identify a domain in Winograd schemas. This keyword approach is based on the assumption that keywords can limit a sentence's high-level semantic meaning. In other words, in our assumption, some keywords are only used to express certain cases in which domain-specific high-level patterns exist.

This keyword approach is advantageous for two reasons. The first advantage is that it can decide whether an unknown sentence belongs to the specific domain or not automatically. It means that there are definite rules to decide the domain of a sentence. The second advantage is that a domain can be decided flexibly depending on the purpose. For instance, a more specific domain within spatial reasoning domain can be investigated.

In this thesis, we chose the thanking domain as a representative case as expressing gratitude appears frequently in daily conversations. After choosing the domain, we needed to extract Winograd schemas that belong to the thanking domain from a Winograd schema data set. We chose to use WinoGrande (Sakaguchi et al., 2019) which is a large data set containing around 44K Winograd schema sentences.This is because the original Winograd data set (WSC273) (Levesque et al., 2012), which is composed of 273 Winograd schema sentences, has only 2 sentences that contain the keyword "thank".

| Keywords | Example sentences | included |
|---|---|---|
| thank | Kayla bought a tablet for Rachel for their birthday last Sunday, and [she] **thanked** them for the thoughtful gift. | Yes |
| thank | Jeffrey's house was on fire and Lawrence put it out, later [he] received a large gift as **thanks**. | Yes |
| thank | Tanya is **thankful** for Laura's generosity because [she] needed a skirt to wear to the party. | Yes |
| grateful | Amy was **grateful** to Patricia for giving her some nausea medication, because [she] felt really sick. | Yes |
| thanks to | The plant was growing and thriving **thanks to** Benjamin but not Donald because [he] had a green thumb. | **No** |
| thanks in no small part to | Kyle received a fantastic education from his teachers **thanks in no small part to** Ian, for whose benefit [he] provided proper guidance. | **No** |

*Table 3.1: The examples sentences from WinoGrande with respect to the keywords*

In the extraction process for the thanking domain, we included the sentences which contain "thank" and "grateful" but excluded ones containing "thanks to" and "thanks in no small part to" from WinoGrande as in Table 3.1. By using "thank", the sentences which include the other related keywords such as "thankful" or "thanks" were automatically identified. However, we found that "thanks to" and "thanks in no small part to" which include "thank" refer to causal relation, *not* gratitude. That is why we excluded them in the thanking domain. As a result, The 171 Winograd schema sentences were identified to belong to the thanking domain.

When specifying a domain, some words were found to have different meanings depending on the context. In Table 3.1, “thanks” appears in both the second sentence and the fifth sentence, but the same word means differently. “Thanks” in the second sentence means gratitude, but the same word in the fifth sentence means causal relation. We were able to distinguish between them by whether “thanks” is a part of the phrases “thanks to” and “thanks in no small part to” which mean causal relations.

In addition, there can be more difficult word sense disambiguation cases which require other methods. Though it does not appear in WinoGrande, there can be a sentence including “gives thanks to” which has “thanks to”. In this case, “thanks” means gratitude though it is part of “thanks to”. As this is an exception of our method, additional measures such as a dependency parsing might be needed for the disambiguation.

## 3.2 Domain-specific patterns in the thanking domain

We found that most of the Winograd schemas in the thanking domain share high-level patterns which are specific to this domain. Among the patterns, some are major patterns which can be applied to more than half of the Winograd schema sentences in the thanking domain, and others are minor patterns which are applicable to a less number of the sentences.

### 3.2.1 Major patterns

| Type | Sentence |
|---|---|
| Pattern 1 | Candidate1 owes candiate2, and (so) pronoun is doing good |
| Pattern 2 | Candidate1 owes candidate2, and (so) pronoun is receiving good |
| Pattern 3 | Candidate 1 does good to candidate2 because pronoun is owing |
| Pattern 4 | Candidate 1 gives thanks to candidate 2 because pronoun is being owed |
| Pattern 5 | Candidate 1 gives thanks to candidate 2 because pronoun is owing |

*Table 3.2: The five major high-level reasoning patterns in the thanking domain*

These high-level patterns can be written in plain English in terms of Winograd schemas which have candidate1, candidate2 and pronoun. Table 3.2 gives the five major high-level reasoning patterns that we found. As the thanking domain is mainly related to expressing gratitude because of the keywords, it is often found that one person is “owing” and the other person is “being owed” in the thanking data set. It follows that the person who is owing would give thanks to or does good to the other. It also implies that the person who is being owed would be thanked or receive something good.

By using the third (“Pattern 3) reasoning pattern in Table 3.2, we can derive another pattern that “Candidate 1 does good to candidate2 because pronoun is *being owed*”. However, no sentence in the thanking domain has this derived pattern, so we excluded this pattern from the major patterns.

The following examples are from WinoGrande related to the thanking domain, and we demonstrate that each example implies one of the patterns:

1. Pattern 1

   - **Example**: Jeffrey’s house was on fire and Lawrence put it out, later [he] brought him a large gift as thanks.
   - **Implication**: Jeffrey **owes** Lawrence, and he is **doing good**.

2. Pattern 2

   - **Example**: Jeffrey’s house was on fire and Lawrence put it out, later [he] received a large gift as thanks.
   - **Implication**: Jeffrey **owes** Lawrence, and he is **receiving good**.

3. Pattern 3

   - **Example**: Jennifer was being taken out by Sarah to a restaurant for [she] wanted to thank the other for helping them in a difficult time.
   - **Implication**: Sarah **does good to** Jennifer because she is **owing**

4. Pattern 4

   - **Example**: Tanya is thankful for Laura’s generosity because [she] bought her a skirt to wear to the party.
   - **Implication**: Tanya **gives thanks to** Laura because she is **being owed**

5. Pattern 5

   - **Example**: Tanya is thankful for Laura’s generosity because [she] needed a skirt to wear to the party.
   - **Implication**: Tanya **gives thanks to** Laura because she is **owing**

It is found that around 77% of the sentences (132/171) can be represented into the only five domain-specific high-level patterns. There are also minor patterns found in the other sentences, and these minor patterns are explained in the next subsection.

| Type | Sentence |
|---|---|
| Pattern 1 | Candidate1 is more thankful than candiate2, because person2 is owing |
| Pattern 2 | Candidate1 is more thankful than candidate2, because person2 is less owing |

*Table 3.3: The two minor high-level reasoning patterns related to the comparative forms in the thanking domain*

### 3.2.2 Minor patterns

Other than the five major patterns, minor patterns are also found in the thanking domain. Some are related to the comparative forms as shown in Table 3.3. Similar to the major patterns, they are variants of the "owing" relationships. These comparative patterns can be applied to the following WinoGrande examples in the thanking domain:

1. Comparative pattern 1
    - **Example**: Lawrence was more grateful for their gift than Benjamin was for theirs because [he] had gotten much more.
    - **Implication**: Lawrence **is more thankful than** Benjamin because he is **owing**.
2. Comparative pattern 2
    - **Example**: Eric was more thankful for the gift than Robert because the gift [he] received was less expensive.
    - **Implication**: Eric **is more thankful than** Robert because he is **less owing**.

Other minor patterns are also found in the thanking domain that has no relationship with "owing". They are related to preference, and Table 3.4 shows these patterns:

| Type | Sentence |
|---|---|
| Pattern 1 | Candidate1 is preferred to candidate2, because person2 is good. |
| Pattern 2 | Candidate1 is preferred to candidate2, because person2 is bad. |

*Table 3.4: The two minor high-level reasoning patterns related to preference in the thanking domain*

The following WinoGrande examples in the thanking domain are given to demonstrate how these preference patterns can be applicable:

1. Preference pattern 1
    - **Example**: I like a thank you note more than a phone call, because the [thing] seems polite.

- **Implication**: The thank you note **is preferred to** the phone call because the thing is **good**.

2. Preference pattern 2

   - **Example**: I like a thank you note more than a phone call, because the [thing] seems rude.
   - **Implication**: The thank you note **is preferred to** the phone call because the thing is **bad**.

## 3.3 Other characteristics in the thanking domain

The 171 Winograd schema sentences in the thanking domain show some characteristics different from the original WSC273. The first characteristic is that the proportion of the candidates refering to human names in the thanking domain are larger than that in WSC273 by around 40%. In our calculation of the proportions, if one of the candidates is an exact name of humans, that is counted as candidates having human names. Table 3.5 gives the exact proportions of human names between the two data sets:

| Data set | Proportion of human names |
|---|---|
| The thanking domain | 94.2% (161/171) |
| WSC273 | 50.9% (139/273) |

*Table 3.5: The proportions of human names between the thanking domain data set and WSC273*

This is related to the fact that gratitude situation has high association with humans. The second characteristic is that not all the Winograd schemas sentences are paired. It is caused by the "*special* words" or the "*alternative* words" (Levesque et al., 2012, p.554) including the keywords such as "thank". These are the numbers in regard to whether the sentences can be paired within the thaning domain:

| Type | Count |
|---|---|
| Paired | 80 |
| Non-paired | 91 |

*Table 3.6: The count of the paired and non-paired sentences in the thanking domain*

# Chapter 4

# Method 1: The advanced high-level knowledge-based reasoning

Our high-level knowledge-based reasoning approach is an advanced version of Sharma's (2019) approach. The motivation of our approach is to mitigate the limitations of the author's approach that it only uses low-level knowledge. As discussed in Chapter 2, using only low-level knowledge requires too many specific background knowledge principles which are not structured, and it cannot resolve some Winograd schemas that need more high-level (abstract) knowledge (Sharma, 2019).

By modifying the approach of Sharma (2019), our approach derives high-level representations of Winograd schemas and uses them with high-level background knowledge principles to derive the answer. As shown in Figure 4.1, we used the author's K-Parser for parsing a Winograd schema sentence and the author's WSC resolution rules for reasoning. But different from the author's approach, we defined the rules to derive the semantic roles and their relationships in the knowledge base. These rules are used to derive the high-level representation of a sentence. Also, our background knowledge principles, which uses the author's format, represent domain-specific higher-level (more abstract) knowledge than those of the author.

The High-level representation, which takes a central role in our approach, is based on Bennett (2020, p.13)'s representation of a Winograd schema sentence:

$$\phi(a, b, p) \equiv ((\alpha(a) \ \wedge \ \beta(b) \ \wedge \rho(a, b)) \ \# \ \pi(p)) \tag{4.1}$$

In this formula, $\alpha$ is a property for the candidate $a$, $\beta$ is a property for the candidate $b$, $\rho$ is a predicate defining the relationship between the candidates, $\#$ refers to the relationship between the candidates' part and the pronoun part (e.g. "because"), and $\pi$ is a property for the pronoun $p$ (Bennett, 2020). This formula is used to transform a Winograd schema sentence into the high-level representation and to represent high-level background knowledge principles.

Before representing a Winograd schema into the formula (4.1) format, we used the two reasoning steps — 1) deriving the domain-specific semantic roles and 2) deriving the domain-

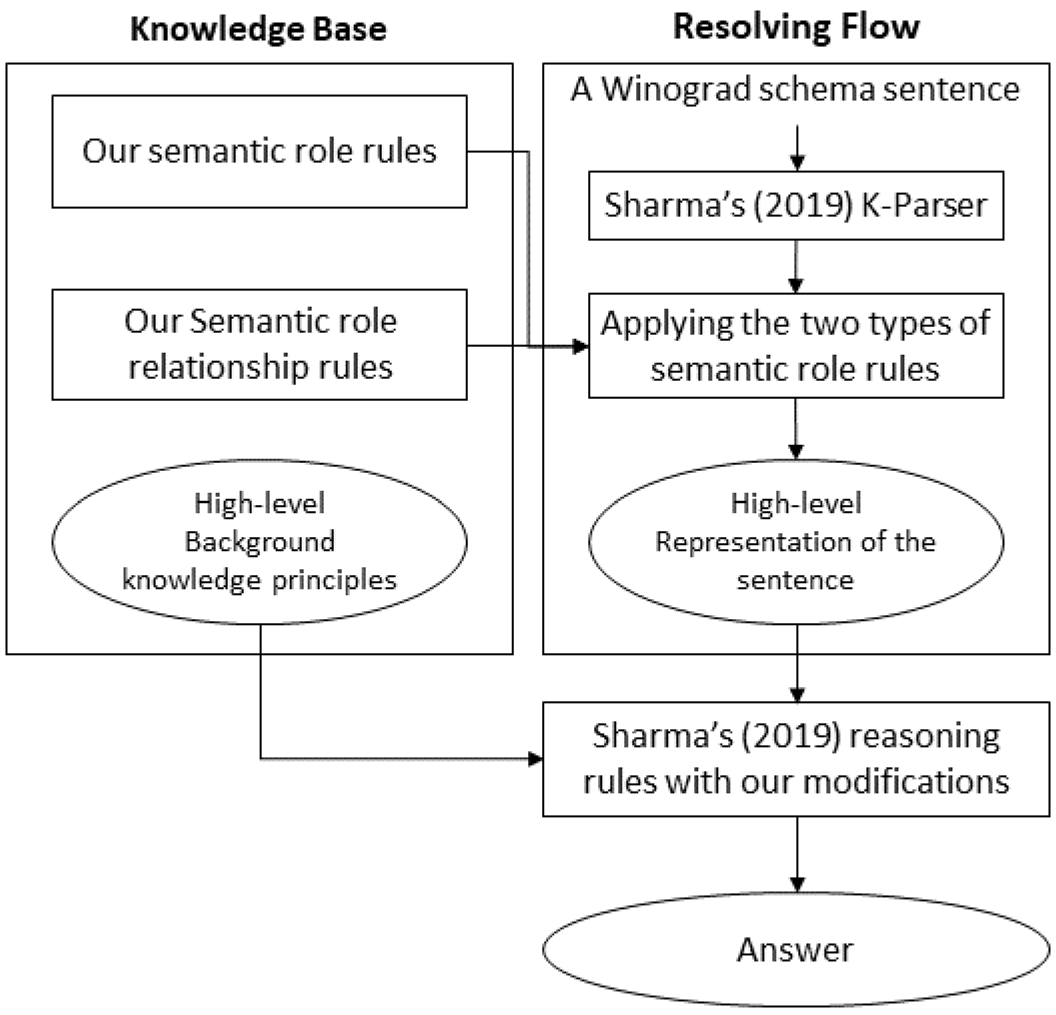
Knowledge Base
Resolving Flow
Our semantic role rules
Our Semantic role relationship rules
High-level Background knowledge principles
A Winograd schema sentence
Sharma's (2019) K-Parser
Applying the two types of semantic role rules
High-level Representation of the sentence
Sharma's (2019) reasoning rules with our modifications
Answer

*Figure 4.1: The architecture of our advanced high-level knowledge-based reasoning method*

specific high-level representation regarding the semantic roles. As in Figure 4.2 where an example from WinoGrande (Sakaguchi et al., 2019) is used, the semantic roles are derived for the two candidates and the pronoun. The semantic roles in this example are "giver", "given" and "being thanked". Using these semantic roles, we can derive the high-level representation used for the formula (4.1). In this example, "$\rho(a, b)$" in the formula (4.1) which means the relationship between the candidates is derived as "Jennifer owes Kayla", and "$\pi(p)$" in the formula (4.1) which means the proporty for the pronoun is derived as "she is receiving good".

The whole process of our domain-specific high-level knowledge-based approach is as follows:

- Building a domain-specific knowledge base using semantic roles

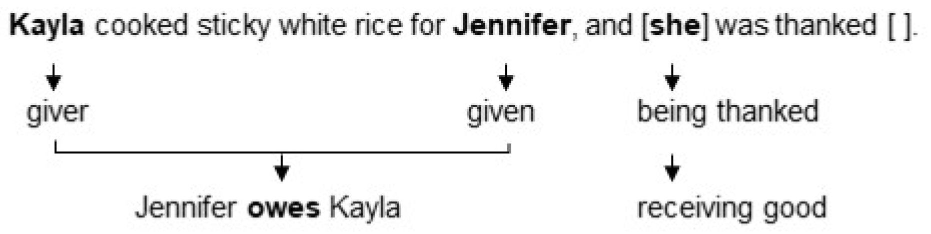
Kayla cooked sticky white rice for Jennifer, and [she] was thanked [ ].
giver
given
being thanked
Jennifer owes Kayla
receiving good

*Figure 4.2: An example from WinoGrande to be represented in high level*

- Transforming a Winograd schema sentence into a high-level representation
- Reasoning by finding matches with a background knowledge principle and deriving the answer

The first process which builds a domain-specific knonwledge base needs to be done manually, and the other two processes can be automated by using K-Parser (Sharma et al., 2015a) and Clingo for ASP.

## 4.1 Building a domain-specific knowledge base using semantic roles

In our knowledge-based reasoning approach, as in its name, a knowledge base is ground to derive the answer. A knowledge base is composed of facts and rules, and quantity and quality of the knowledge base are central to our approach (Brachman and Levesque, 2004). We derived these facts and rules from the thanking domain train set. Our knowledge base can be divided into the following three parts:

- Defining rules to derive semantic roles
- Defining relationships regarding semantic roles
- Defining domain-specific high-level background knowledge principles

The first two parts are related to rules and the third part is related to facts. The rules defined in the first two parts are used to transform a Winograd schema into the high-level representation as in Figure 4.2. The high-level background knowledge principles in the third part are used in the reasoning process to derive the answer. Each part of the knowledge base is explained in each subsection.

### 4.1.1 Defining rules to derive semantic roles

When it comes to building a domain-specific knowledge base, the first step is to define rules to derive semantic roles. These semantic roles are important as they are more generalisable (higher-level) representation than the original parsed representation. In addition, semantic roles function as the basis for defining the relationship between the candidates. This relationship is a key factor to derive the answer in Winograd schemas.

As our approach is to tackle domain-specific Winograd schemas in the thanking domain, we defined our own domain-specific semantic roles. The Winograd schemas in the thanking domain were investigated to find the domain-specifc semantic roles which can derive higher-level representations compared to the original sentences. Interestingly, the six major semantic roles for the candidates and the pronoun were found in the thanking domain. These domain-specific semantic roles are thanker, being thanked, giver, given, helper, and being helped. The following examples give what low-level representations of a sentence can correspond to each semantic role:

- **thanker:** thanking someone, being thankful, being grateful
- **being thanked:** being thanked, receiving a thank you letter
- **giver:** giving something, buying something for someone, cooking something for someone
- **given:** being given something, receiving something, being taken out to a restaurant
- **helper:** helping someone, saving someone, putting out a fire
- **being helped:** being helped, being saved, someone putting out a fire in his/her house

These examples demonstrate that some low-level representations of a sentence can be used to derive the higher-level domain-specific semantic roles. For example, a "giver" can be someone buying or cooking something as well as giving something. In addition, a "helper" can refer to someone who saves a person or puts out a fire as well as helps a person. In these ways, many specific semantic roles can imply domain-specific higher-level (more abstract) semantic roles such as giver and helper.

After finding the high-level semantic roles in the thanking domain, we defined the rules to derive these semantic roles from the K-Parser representations of original Winograd schema sentences. We used ASP to represent these semantic role rules. These rules have a common format which is composed of *IF* and logical *AND*, which means that ":-" refers to *IF* and "," refers to logical *AND* (Merritt, 2017, p.49).

For example, this rule is the first rule of "thanker" which is one of the simple rules:

```
has_s(X, semantic_role, thanker) :-
                has_s(Thank,agent,X),
                has_s(Thank,instance_of,thank).
```

This rule means that **if** Thank's agent is X **and** Thank is instance of thank **then** X's semantic role is thanker. We defined these semantic role rules manually by using the train set in the thanking domain. Under this same format, we used the following four methods to define more generalisable semantic role rules:

1. **The method when the semantic role of only one of the candidates is known:** In our approach, it is required to derive the semantic roles for the two candidates and the pronoun in a sentence in order to derive the answer. It means that if there is no rule to derive the semantic role of at least one of them, no answer can be derived. To prevent this case, if one candidate's semantic role is known, the other's semantic role need to be derived if there is a logical relationship between them.

   As we assume that there would be a certain logical relationship between the two candidates ("$\rho(a,b)$") in the formula (4.1), the semantic role of one of the candidates can be the ground to derive that of the other candidate. The following examples show these logical relationships between the two candidates:

   - one of the candidates: giver $\rightarrow$ the other: given
   - one of the candidates: given $\rightarrow$ the other: giver
   - one of the candidates: helper $\rightarrow$ the other: being helped
   - one of the candidates: being helped $\rightarrow$ the other: helper
   - one of the candidates: thanker $\rightarrow$ the other: being thanked
   - one of the candidates: being thanked $\rightarrow$ the other: thanker

   The fifth example, for instance, means that if the semantic role of one candidate is "thanker" then that of the other would be "being thanked".

   To implement this in ASP, we first defined the rules to check which noun is a candidate for the answer:

   ```
   is_candidate(X) :- ans_ch1(X).
   is_candidate(X) :- ans_ch2(X).
   ```

   where the predicate `ans_ch1(X)` refers to the first candidate and the predicate `ans_ch2(X)` refers to the second candidate. These rules mean that **if** X is the first candidate or the second candidate **then** X is a candidate.

   Using this predicate, we defined the rules which derive the semantic role of one noun from the other's semantic role. This is an example of the rules for "being thanked':

   ```
   has_s(X, semantic_role, being_thanked) :-
           has_s(X1, semantic_role, thanker),
           X != X1,
           is_candidate(X),
           is_candidate(X1).
   ```

where "!= " refers to "not equal to". It means that **if** the semantic role of one of the candidates(X1) is "thanker" **then** the semantic role of the other candidate(X) is "being thanked". In the same way, the semantic rules such as "given" and "being helped" were defined.

2. **When no semantic role of the candidates is known:** In our knowledge-based reasoning method, semantic roles can only be derived by using rules. As a result, in some cases no semantic roles of the candidates are derived, which means that we cannot derive the answer. But it is costly to define every single case into rules.

   This lack of the semantic role rules occurs frequently in terms of "giver" compared to "thanker" or "helper". Since we used the two keywords "thank" and "grateful" to identify the thanking domain, the cases of "thanker" are comparatively limited. It means that most of the "thanker" semantic roles in the thanking domain can be derived by the specific rules. On the other hand, "giver" can cover broad cases such as someone cooking something for someone, and "helper" ('giving' help) can be understood as "giver" in terms of the thanking domain.

   Therefore, we defined the three general semantic role rules to cover the cases of "giver" in the thanking domain. The Syntactic relationships such as "agent", "recipient" and "object" regarding the two candidates were used to define these rules while the verb of a candidate which has a semantic meaning is not specified. The general rules to derive the semantic role is as follows:

   - candidate1 Verb candidate2 (where candidate2 is the recipient of Verb)
   - candidate1 Verb candidate2 (where candidate2 is the object of Verb)
   - candidate1 Verb Something for candidate2

   In ASP, these three rules can be written as:

```
(1) has_s(X, semantic_role, giver) :-
        is_candidate(X),
        is_candidate(Y),
        X != Y,
        has_s(Verb,agent,X),
        has_s(Verb,recipient,Y),
        not has_s(X, semantic_role, thanker),
        not has_s(X, semantic_role, helper).

(2) has_s(X, semantic_role, giver) :-
        is_candidate(X),
        is_candidate(Y),
        X != Y,
        has_s(Verb,agent,X),
        has_s(Verb,object,Y),
        not has_s(X, semantic_role, thanker),
        not has_s(X, semantic_role, helper).

(3) has_s(X, semantic_role, giver) :-
        is_candidate(X),
        is_candidate(Y),
        X != Y,
        has_s(Verb,agent,X),
        has_s(Verb,recipient,Something),
        has_s(Something,destination,Y),
        not has_s(X, semantic_role, thanker),
        not has_s(X, semantic_role, helper).
```

Suppose that X and Y are the two different candidates, and X is neither thanker nor helper. The rules (1) and (2) mean that **if** Verb's agent is X and Verb's recipient or object is Y **then** X's semantic role is giver. The rule (3) is the case when Verb's recipient is Something and Something's destination is Y. If one candidate's semantic role is derived to be giver, then we can derive the other's semantic role as "given" by using the first method (when the semantic role of only one of the nouns is known).

3. **Using synonym**: Synonym can be used to enable the semantic role rules more generalisable (Sharma, 2019). Our implementation for this method has similarity with Sharma (2019)'s method, but they are not exactly same. We used the predicate `syn_check(word, X)` to check whether $X$ is a synonym of the word. In addition, instead of using the external thesauraus, we defined synonymous relations between words using the predicates within the thanking data set. The purpose of this constraint is to exclude synonyms that does not make sense in the thanking domain. This is an example of using the predicates with the two rules and the two facts to show how they can make another rule ("being thanked") more generalisable:

```
syn_check(X,X) :- has_s(_,instance_of,X).
syn_check(X,X) :- syn_check(X,_).
syn_check(receive,get).
syn_check(thank,thanks).

has_s(X, semantic_role, being_thanked) :-
        has_s(Receive,agent,X),
        has_s(Receive,instance_of,Receive_syn),
        has_s(Receive,recipient,Thank),
        has_s(Thank,instance_of,Thank_syn),
        syn_check(receive,Receive_syn),
        syn_check(thank,Thank_syn),
        not has_s(X, semantic_role, not_being_thanked).
```

This rule in the example means that **if** X is agent of a synonym of receive and recipient of the synonym of receive is a synonym of thank **then** X's semantic role is "being thanked". This representation is advantageous as one rule can function as multiple rules. In this example, this one rule can cover the four different cases as each synonym (including its own) of "receive" can be paired with each synonym of "thank" as follows:

```
Receive_syn = {receive, get}.
Thank_syn = {thank, thanks}.
```

$$\texttt{Receive\_syn} \times \texttt{Thank\_syn} = \{(receive, thank), (receive, thanks), (get, thank), (get, thanks)\}$$

4. **Sentiment lexicon**: We used an external sentimental lexicon dictionary (Hu and Liu, 2004) to derive the semantic roles of "good" and "bad" as these general sentiments can also be applied to the specific thanking domain. Since the semantic roles of "good" and "bad" are only used in limited cases in the thanking domain as in Table 3.4 for the minor patterns, these roles are only defined for the pronoun when the candidates have the relationship of "is preferred to" or "is better than".

   For example, this is the example from WinoGrande used in Subsection 3.2.2 which has the relationship of "is preferred to" between the candidates:

   - **Example**: I like a thank you note more than a phone call, because [it] seems polite.
   - **The relationship between the candidates**: The thank you note **is preferred to** the phone call

   In this example, the pronoun "[it]" is the cause of the preference of the thank you note rather than the phone call. Since the pronoun refers to one of the candidates, we can infer that the pronoun's semantic role would be either "good" or "bad" and the lexicon dictionary can be used for this decision. Theses are the rules we defined in the logical programming:

```
has_s(X, semantic_role, good) :-
        1 {has_s(_, is_preferred_to, _);
        has_s(_, is_better_than, _)},
        has_s(X, relation, causer),
        1 {has_s(X,trait,Positive_word);
        has_s(X,complement_word,Positive_word)},
        has_s(Positive_word,instance_of,Positive),
        check_positive(Positive).

has_s(X, semantic_role, bad) :-
        1 {has_s(_, is_preferred_to, _);
        has_s(_, is_better_than, _)},
        has_s(X, relation, causer),
        1 {has_s(X,trait,Negative_word);
        has_s(X,complement_word,Negative_word)},
        has_s(Negative_word,instance_of,Negative),
        check_negative(Negative).
```

The prediates "`check_positive`" and "`check_negative`" are used to check whether a word belongs to either positive or negative sentiment according to the ontology in the lexicon dictionary. This enables us to derive the semantic roles of "good" and "bad" even when the train set is very small because the ontology in the lexicon dictionary can be used.

By using the thanking domain train set and the four methods to make more generalisable rules, we manually defined the rules to derive the semantic roles from the train set in the thanking domain. The following rules are some examples from the defined rules for the major six semantic roles:

1. **thanker**

```
has_s(X, semantic_role, thanker) :-
                    has_s(Thank,agent,X),
                    has_s(Thank,instance_of,thank).

has_s(X, semantic_role, thanker) :-
                    has_s(X,trait,Thankful),
                    has_s(Thankful,instance_of,Thankful_syn),
                    syn_check(thankful,Thankful_syn),
                    not has_s(Thankful,extent,_).
```

2. **being thanked**

```
has_s(X, semantic_role, being_thanked) :-
                    has_s(Thank,recipient,X),
                    has_s(Thank,instance_of,thank).
has_s(X, semantic_role, being_thanked) :-
                    has_s(X1, semantic_role, thanker),
                    X != X1,
                    is_candidate(X),
                    is_candidate(X1).
```

3. **giver**

```
has_s(X, semantic_role, giver) :-
                has_s(Give,agent,X),
                has_s(Give,instance_of,Give_syn),
                syn_check(give,Give_syn),
                not has_s(X,semantic_role,not_thanker),
                not has_s(X,semantic_role,thanker).

has_s(X, semantic_role, giver) :-
                has_s(Cook,agent,X),
                has_s(Cook,instance_of,cook).
```

4. **given**

```
has_s(X, semantic_role, given) :-
                has_s(Give, object, X),
                has_s(Give, instance_of, Y),
                syn_check(give, Y).

has_s(X, semantic_role, given) :-
                has_s(X1, semantic_role, giver),
                X != X1,
                is_candidate(X),
                is_candidate(X1).
```

5. **helper**

```
has_s(X, semantic_role, helper) :-
                has_s(Help,agent,X),
                has_s(Help,instance_of,help).

has_s(X, semantic_role, helper) :-
                has_s(Save,agent,X),
                has_s(Save,instance_of,save).
```

6. **being helped**

```
has_s(X, semantic_role, being_helped) :-
                has_s(Help,recipient,X),
                has_s(Help,instance_of,help).

has_s(X, semantic_role, being_helped) :-
                has_s(X1, semantic_role, helper),
                X != X1,
                is_candidate(X),
                is_candidate(X1).
```

### 4.1.2 Defining relationships regarding semantic roles

Once the semantic roles are derived from the parsed representation of a Winograd schema sentence, the next step is to define the the relationships with respect to these semantic roles. As in the formula (4.1), the following three need to be defined: relationship between the semantic roles of the candidates("$\rho(a, b)$"), relation between the candidates' part and the pronoun part("$\#$"), and the property of the pronoun("$\pi(p)$").

1. **Relationship between the semantic roles of the candidates** ("$\rho(a, b)$" in the formula (4.1)): As discussed in Chapter 3, domain-specific relationships between the candidates' semantic roles are found in the thanking domain. We defined the relationship between the semantic roles mainly related to "owes","does good to" and "gives thanks to". Table 4.1 shows how these relationships can be derived depending on the semantic roles of the candidates and existence of causal relation:

| Semantic relationship | Existence of causal relation | Semantic role | |
|---|---|---|---|
| | | X | Y |
| X owes Y | No | being helped | helper |
| X owes Y | No | given | giver |
| X does good to Y | Yes | helper | being helped |
| X does good to Y | Yes | giver | given |
| X gives thanks to Y | Yes | thanker | being thanked |

*Table 4.1: The major relationships between the candidates' semantic roles*

The following rules are examples to define these major relationships in ASP:

```
(1) has_s(X, owes, Y) :-
          has_s(X, semantic_role, being_helped),
          has_s(Y, semantic_role, helper),
          not has_s(_, relation, causer).

(2) has_s(X, owes, Y) :-
          has_s(X, semantic_role, given),
          has_s(Y, semantic_role, giver),
          not has_s(_, relation, causer).

(3) has_s(X, does_good_to, Y) :-
          has_s(X, semantic_role, helper),
          has_s(Y, semantic_role, being_helped),
          has_s(_, relation, causer).

(4) has_s(X, does_good_to, Y) :-
          has_s(X, semantic_role, giver),
          has_s(Y, semantic_role, given),
          has_s(_, relation, causer).

(5) has_s(X, gives_thanks_to, Y) :-
          has_s(X, semantic_role, thanker),
          has_s(Y, semantic_role, being_thanked),
          has_s(_, relation, causer).
```

The rule (1) means that **if** $X$ is being helped and $Y$ is a helper when there is *no* causal relation **then** $X$ **owes** $Y$. On the other hand, the rule (3) means that **if** $X$ is being helped and $Y$ is a helper when there is a causal relation **then** $X$ **does good to** $Y$. Even though the semantic roles are same between the rules (1) and (3), we defined the relationship in the rule (3) different from the rule (1). It depends on the existence of causal relation. These sentences from WinoGrande (Sakaguchi et al., 2019) show the difference:

(a) Kayla bought a tablet for Rachel for their birthday last Sunday, and [she] was thanked for the thoughtful gift.
   - **The implied relationship:** Rachel **owes** Kayla
   - **The existence of causal relation:** No

(b) Jennifer was taking out Sarah [ ] to a restaurant **for** [she] wanted to thank the other for helping them in a difficult time.
   - **The implied relationship:** Jennifer **does good to** Sarah
   - **The existence of causal relation:** Yes ("**for**" in the sentence)

In the first sentence, the candidates' relationship is considered "owes", but in the second sentence, the relationship is considered "does good to".

2. **Relation between the candidates' part and the pronoun part** ("$\#$" in the formula (4.1)): As discussed in relationship between the semantic roles of the candidates, the causal relationship in a sentence can affect the relationship between the semantic roles. In this thanking domain, we only considered "causal" relation. This is because what matters in this domain is whether the candidates' part explains the pronoun part or the pronoun part explains the candidates' part. We defined the rules to detect the causal relation in a sentence, and these are two of them:

```
(1) has_s(P, relation, causer) :-
        pronoun(P),
        is_candidate(A),
        has_s(Verb1,caused_by,Verb2),
        1 {has_s(Verb1,agent,A);has_s(Verb1,recipient,A)},
        has_s(Verb2,agent,P).

(2) has_s(P, relation, causer) :-
        pronoun(P),
        is_candidate(A),
        has_s(Verb1,caused_by,Verb2),
        1 {has_s(Verb1,agent,A);has_s(Verb1,recipient,A)},
        has_s(Y,supporting_verb,Verb2),
        has_s(Y,agent,P).
```

In the first and second rules,$\{A\ ;\ B\}$ expression was used, which means that $A$ OR $B$ (Schaub, 2018). In this sense, the first rule means that **if** Verb2's agent is the pronoun, Verb1 is caused by Verb2, and Verb1's agent or recipient is a candidate **then** the pronoun part functions as a causer.

3. **Property of the pronoun** ("$\pi(p)$" in the formula (4.1)) : One semantic role can be a subset of another semantic role in the thanking domain. In other words, a low-level semantic role **is a subset of** a high-level semantic role. For example, the semantic role of "giver" **is a subset of** semantic role of "doing good". In this case, "doing good" is a higher representation of "giver". We defined the following rules which give the major subset relations:

(1) doing good

```
has_s(X, semantic_role, doing_good) :-
            has_s(X, semantic_role, giver).
has_s(X, semantic_role, doing_good) :-
```

```
                has_s(X, semantic_role, thanker).
has_s(X, semantic_role, doing_good) :-
                has_s(X, semantic_role, helper).
```

(2) receiving good

```
has_s(X, semantic_role, receiving_good) :-
                has_s(X, semantic_role, given).
has_s(X, semantic_role, receiving_good) :-
                has_s(X, semantic_role, being_thanked).
has_s(X, semantic_role, receiving_good) :-
                has_s(X, semantic_role, being_helped).
```

(3) being owed

```
has_s(X, semantic_role, being_owed) :-
                has_s(X, semantic_role, giver).
has_s(X, semantic_role, being_owed) :-
                has_s(X, semantic_role, helper).
```

(4) owing

```
has_s(X, semantic_role, owing) :-
                has_s(X, semantic_role, given).
has_s(X, semantic_role, owing) :-
                has_s(X, semantic_role, thanker).
has_s(X, semantic_role, owing) :-
                has_s(X, semantic_role, being_helped).
```

### 4.1.3 Defining domain-specific high-level background knowledge principles

In order to resolve a Winograd schema, the knowledge base needs to have background knowledge prinicples as well as the rules for deriving semantic roles and the relationship regarding them. We used Sharma's (2019) representation style of background knowledge, but our background knowledge is represented in *higher level*. Our background knowledge principles are represented with the relationship between the candidates' semantic roles and the high-level semantic role of the pronoun. Table 4.2 gives our major background knowledge principles related to the high-level patterns found in the thanking domain in Table 3.2:

As "high-level" is an important feature of our background knowledge principles, our one high-level background knowledge principle can include multiple low-level background knowledge principles in the style of Sharma (2019). These examples demonstrate this relationship between our high-level background knowledge principles and Sharma's (2019) background knowledge principles:

|   | High-level background knowledge principles |
|---|---|
| 1 | **IF** person1 owes someone, and (so) person2 is doing good **THEN** person1 is same as person2 |
| 2 | **IF** someone owes person1, and (so) person2 is receiving good **THEN** person1 is same as person2 |
| 3 | **IF** person1 does good to someone because person2 is owing **THEN** person1 is same as person2 |
| 4 | **IF** someone gives thanks to person1 because person2 is being owed **THEN** person1 is same as person2 |
| 5 | **IF** person1 gives thanks to someone because person2 is owing **THEN** person1 is same as person2 |

*Table 4.2: The five major high-level background knowledge principles in the thanking domain*

- Our second high-level background knolwedge principle in Table 4.2:
  **IF** someone **owes** person1, and (so) person2 is receiving good **THEN** person 1 is same as person2.

- Sharma's (2019) representation style:
  **IF** person1 cooks rice for someone, and (so) person2 is thanked **THEN** person1 is same as person2. (This example is defined by us according to the author's style.)

In the author's representation in this example, it is only applicable to the case when person1 *cooks*. But our representation in this example can be applicable to diverse cases where person1 *cooks*, *helps*, *gives*, or any situation of someone owing person1.

We extracted the high-level background knowledge principles from the train set in the thanking domain manually. Once they are extracted, these background knowledge principles written in plain English as in Table 4.2 need to be represented into the logical form into the knowledge base. We followed Sharma's (2019) logical form used in ASP which uses the predicate `has_k`, and we defined all the high-level background knowledge principles including the ones in Table 4.2 into the author's logical form. The following example is the logical representation of the first high-level background knowledge principle in Table 4.2:

```
has_k(someone_,owes,person1_).
has_k(person2_,semantic_role,doing_good).

has_k(person1_,instance_of,person1).
has_k(person1,is_subclass_of,person).
has_k(someone_,instance_of,someone).
has_k(someone,is_subclass_of,person).
has_k(person2_,instance_of,person2).
has_k(person2,is_subclass_of,person).
has_k(person1_,is_same_as,person2_).
has_k(person2_,is_same_as,person1_).
```

By using the high-level background knowledge principles defined in the knowledge base, a Winograd schema sentence can be resolved if the sentence is transformed into the high-level representation as in Figure 4.2. Section 4.2 shows how the high-level representation of a sentence can be derived. Section 4.3 gives the detailed reasoning method to derive the answer using the high-level background knowledge principles and the high-level representation of a sentence.

## 4.2 Transforming a Winograd schema sentence into a high-level representation

After building a knowledge base, the second part in our approach is to transform a Winograd schema into a high-level representation. This part involves the three processes: 1) parsing a Winograd schema sentence, 2) deriving the semantic roles and 3) deriving the high-level representations regarding semantic roles. The first process which uses K-Parser to parse a sentence into the graphical representation is also included in Sharma's (2019) approach. But the second and the third processes are our original work to represent a sentence into the domain-specific higher-level representation. The rules to derive the semantic roles and the relationship regarding them in the knowledge base are used in these two processes.

In this section, we use the same example (named as "the rice example") from WinoGrande (Sakaguchi et al., 2019) that is used in the literature review of Sharma's (2019) approach in Section 2.1:

Kayla cooked sticky white rice for Jennifer, and [she] was thanked for making such delicate rice.

This same example is given to demonstrate the difference between our approach and Sharma's (2019) approach.

### 4.2.1 Parsing a Winograd schema sentence by K-Parser

We used the K-Parser to transform a Winograd schema sentence into the graphical representation, which is same as Sharma's (2019) approach. This parsing method was applied to all the Winograd schema sentences in the thanking domain. For example, the rice example was parsed, and the only important part of the result is given below as the whole parsing result of the same sentence is given in Section 2.1:

```
has_s(cooked_2,agent,kayla_1).
has_s(cooked_2,instance_of,cook).
has_s(cooked_2,recipient,rice_5).
has_s(rice_5,destination,jennifer_7).
has_s(thanked_12,recipient,she_10).
has_s(thanked_12,instance_of,thank).
```

As in Sharma's (2019) approach, we also defined the two candidates and the pronoun for all the Winograd schema sentences after the parsing. For example, the candidates and the pronoun in the rice example are defined as:

```
pronoun(she_10).
ans_ch1(kayla_1).
ans_ch2(jennifer_7).
```

Though we used the K-Parser which is an automatic method, we needed to make minor modifications in order to achieve more accurate parsing. Sharma (2019) also points out that the parsing results by K-Parser can be not correct. In order to reduce the incorrect parsing, we implemented the following two methods:

1. **Causal relation:** In the thanking domain, whether a causal relation exists is important. Depending on its existence, the answer can be switched. In most cases, K-Parser seems to identify a causal relation well when there is a expression "because" which are frequently used in the thanking domain. However, K-Parser does not seem to identify a causal relation when other expressions are used to mean a causal relation.

   The following sentences are the examples from WinoGrande([boldness mine]) that causal relations cannot be identified by K-Parser:

   - Jennifer was taking out Sarah out to a restaurant **for** [she] wanted to thank the other for helping them in a difficult time.
   - Adam thanked Dennis for their life **after** [he] fell in the water and was saved by them after almost drowning.
   - During the fire evacuation that was occurring at the office building, Maria thanked Christine **when** [she] extinguished the building.

   Even though the bold words ("for", "after" and "when") in these exammples refer to the causal relations semantically, K-Parser cannot identify these causal relations. For the better parsing, we replaced these words with "because" which K-Parser can identify as causal relation when sentences are put into K-Parser.

2. **Human names:** K-Parser shows two problems regarding human names. The first is that it may not give the predicates that give the relation "is subclass of" between a person name and person(e.g. `has_s(adam,is_subclass_of,person)`). To derive these relations, we put the names in the thanking domain into the predicate `check_person` (e.g. `check_person(adam)`). Then, we defined the following rule:

```
has_s(X,is_subclass_of,person) :-  has_s(_,instance_of,X),
                                   check_person(X).
```

The second problem is that depending on the names put into K-Parser, parsing results could change. For some sentences, we replaced the names with the different names, and put the changed sentences into K-Parser. After the parsing, the changed names were replaced with the original names.

Despite these improvement methods, some parsing results need to be modified. Table 4.3 shows how many parsing results of the 80 paired sentences in the thanking domain needed to be modified manually according to the number of modifications in a sentence:

| Number of modifications within a sentence | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Number of the sentences | 42 | 24 | 4 | 6 | 1 | 3 |

*Table 4.3: The number of the sentences according to the number of modifications made within a sentence*

Among the 80 sentences, more than half of them $(42/80)$ do not need any modifications, and $86.5\%(70/80)$ of the sentences need no modification or modifications made less than $3$. K-Parser's incorrect parsing seems manageable in a small data set as most of the cases need no manual corrections or a small number of corrections. But as the results show, K-Parser can still be improved to achieve more accurate parsing for the future work. In the thanking domain, some modifications were needed because the parsing results often do not specify which word is related to the comparative words such as "more" and "less". In these cases, we manually added the predicates using the edge of "extent" (e.g. `has_s(thankful_4,extent,more_3).`).

### 4.2.2 Deriving the semantic roles

Once a Winograd schema sentence is parsed into the graphical representation by K-Parser (Sharma, 2019), We can derive the semantic roles for the two candidates and the pronoun by using the semantic role rules in the knowledge base. The semantic roles rules that we use are predefined in the knowledge base and specific to the thanking domain. The derivations can be done automatically by reasoning in ASP.

1. The semantic role for the first candidate ("Kayla" in the sentence)

   - A predefined rule from the knowledge base:

     ```
     has_s(X, semantic_role, giver) :-
                     has_s(Cook,agent,X),
                     has_s(Cook,instance_of,cook).
     ```

   - The relevant predicates from the parsed sentence:

     ```
     has_s(cooked_2,agent,kayla_1).
     has_s(cooked_2,instance_of,cook).
     ```

   - The derived semantic role:

```
has_s(kayla_1, semantic_role, giver).
```

2. The semantic role for the second candidate ("Jennifer" in the sentence)

   - A predefined rule from the knowledge base:

   ```
   has_s(X, semantic_role, given) :-
                   has_s(Cook,recipient,Something),
                   has_s(Cook,instance_of,cook),
                   has_s(Something,destination,X).
   ```

   - The relevant predicates from the parsed sentence:

   ```
   has_s(cooked_2,recipient,rice_5).
   has_s(cooked_2,instance_of,cook).
   has_s(rice_5,destination,jennifer_7).
   ```

   - The derived semantic role:

   ```
   has_s(jennifer_7, semantic_role, given).
   ```

3. The semantic role for the pronoun ("she" in the sentence)

   - A predefined rule from the knowledge base:

   ```
   has_s(X, semantic_role, being_thanked) :-
                   has_s(Thank,recipient,X),
                   has_s(Thank,instance_of,thank).
   ```

   - The relevant predicates from the parsed sentence:

   ```
   has_s(thanked_12,recipient,she_10).
   has_s(thanked_12,instance_of,thank).
   ```

   - The derived semantic role:

   ```
   has_s(she_10, semantic_role, being_thanked).
   ```

As a result, with respect to the sentence, the semantic role of the candiate "Kayla" is derived to be "giver", that of the candidate "Jennifer" is derived to "given", and that of the pronoun is derived to be "being thanked".

### 4.2.3 Deriving the high-level representations regarding semantic roles

Once the semantic roles for the two candidates and the pronoun are derived, we can derive the high-level representations with respect to the semantic roles. These high-level representations can be derived according to the formula (4.1) by using the rules defined in Subsection 4.1.2. For example, the high-level representations of the rice example can be derived as follows:

1. **Relationship between the two candidates' semantic roles** ("$\rho(a, b)$" in the formula (4.1)): By using the rules of deriving the relationships between the candidates in Subsection 4.1.2, we can derive the candidates' relationship from the candidates' semantic roles. For example, how the candidates' relationship can be derived from the semantic roles of the rice example is as follows:

   - A predefined rule from the knowledge base:

     ```
     has_s(X, owes, Y) :-  has_s(X, semantic_role, given),
                           has_s(Y, semantic_role, giver),
                           not has_s(_, relation, causer).
     ```

   - The derived semantic roles of the candidates:

     ```
     has_s(jennifer_7, semantic_role, given).
     has_s(kayla_1, semantic_role, giver).
     ```

   - The derived relationship:

     ```
     has_s(jennifer_7, owes, kayla_1)
     ```

   As a result, in the rice example, we can derive that "Jennifer" owes "Kayla".

2. **Relation between the part of the two candidates and the part of the pronoun** ("#" in the formula (4.1)): By using the rules of the causal relation in Subsection 4.1.2, we can derive the relation between the two candidates' part and the pronoun's part. As disccused in the subsection, we only consider the causal relation. If the causal relation exists, the pronoun part explains the candidates' part. On the other hand, If there is no causal relation, the candidates' part explains the pronoun's part.

   In the rice example, no causal relation can be derived by the rules as the sentence has no word such as "because". It means in the sentence the candidates part (the relationship between the candidates: "Jennifer" owes "Kayla") explains the pronoun part ("[she] was thanked").

3. **The pronoun's property** ("$\pi(p)$" in the formula (4.1)): we can derive the high-level representation of the pronoun's property by applying the rules of the subset relations. In the rice example, the high-level property of the pronoun "she" is derived as "receiving good" from the semantic role of "being thanked" as follows:

   - A predefined rule from the knowledge base

     ```
     has_s(X, semantic_role, receiving_good) :-
                        has_s(X, semantic_role, being_thanked).
     ```

   - The derived semantic role for the pronoun:

     ```
     has_s(she_10, semantic_role, being_thanked).
     ```

- The derived semantic role:

  ```
  has_s(she_10, semantic_role, receiving_good).
  ```

## 4.3 Automatic reasoning to derive the answer

After the high-level representations of Winograd schemas are derived by Subsection 4.2, the next step is to use these sentence representations and the domain-specific background knowledge principles to derive the answers. Since there are many domain-specific background knowledge-principles, one Winograd schema sentence representation needs to be compared with each one of the background knowledge principles. Each time we applied the reasoning rules defined by Sharma (2019) that are modified by us in ASP. As a result of these attempts, if one answer is derived, then that would be the final answer. If all the attempts give no answer, the final answer would be no answer. In the case of multiple answers, if the answers are same, that same answer would be the final answer, but if the answers are not same, the final answer would be multiple answers.

As mentioned in the previous paragraph, we modified the reasoning rules of Sharma (2019), and this paragraph gives the background of the modifiations. As discusssed in Section 2.1, the aim of the author's reasoning rule is to find matching nodes between a sentence and a background knowledge principle, so that the token matching with the person1 in the background knowledge principle can be the answer. This is based on the author's background knowledge format that the person1 is same as person2. Some of the author's reasoning rules define the constraints that what nodes can be a match, and it is these rules that are related to our two modifications.

Regarding the constraint rules of Sharma (2019), we implemented tho following two modification measures. Firstly, as the author points out that the K-Parser outputs need to be adjusted to the reasoning rules, we made a change to one of the reasoning rules instead of adjusting the K-Parser outputs. While the author reduces the two classes of each token in K-parser into one class, we changed the rule ("s36")(Sharma, 2019, p.10) to remove the constraints to the pronoun and the candidates, and the following is the modified rule:

```
:- matches(X,Y), ans_ch1(X1), ans_ch2(X2), pronoun(X3),
   X != X1, X != X2, X != X3,
   has_k(Y,instance_of,C), not has_s(X,instance_of,C).
```

By applying this modified constraint rules, the K-Parser outputs can be used without the author's measure to reduce the level of classes.

In addition to the first modification, our second modification is to add one constraint rule. As Sharma (2019) does not use the semantic roles for reasoning, we needed to add one more rule to ensure that if the two nodes are a match, their semantic roles are same. Our added rule is defined as:

```
:- matches(X,Y), has_k(Y,semantic_role,C),
   not has_s(X,semantic_role,C).
```

The rice example is given to demonstrate how the reasoning method is applied in the case of one Winograd sentence and one relevant background knowledge. Since the high-level representation of the rice example is derived in Subsection 4.2, we need to use a relevant bakgroun knowledge principle for the reasoning. The relevant background knowledge principle can be **IF** someone ”owes” person1 and person2 is ”receiving good” **THEN** person1 is same as person2. This can be represented in ASP as:

```
has_k(someone_,owes,person1_).
has_k(person2_,semantic_role,receiving_good).

has_k(person1_,instance_of,person1).
has_k(person1,is_subclass_of,person).
has_k(someone_,instance_of,someone).
has_k(someone,is_subclass_of,person).
has_k(person2_,instance_of,person2).
has_k(person2,is_subclass_of,person).

has_k(person1_,is_same_as,person2_).
has_k(person2_,is_same_as,person1_).
```

By applying the Sharma’s (2019) reasoning rules modified by us, some of the matches predicates can be derived as:

```
matches(kayla_1, person1_).
matches(jennifer_7, someone_).
matches(she_10, person2_).
matches(person, person).
matches(receiving_good, receiving_good).
```

As our high-level background knowledge principle defines that person1 is same as person2 as the author’s style does, “Kayla” in the rice example can be derived as the answer. This is because “Kayla” and “she” in the rice example match with person1 and person2 in the background knowledge principle respectively.

## 4.4 Summary

Our knowledge-based reasoning method uses Sharma (2019)’s framework, but our approach is more advanced to tackle domain-specific Winograd schemas. Firstly, We set a domain and defined semantic roles that are relevant to resolve the pronoun while K-parser (Sharma et al., 2015a) only gives the semantic roles related to general cases. In addition, Sharma (2019) does not use the semantic role for reasoning to resolve a Winograd schema. Secondly, our method is advantageous as higher-level background knowledge is more generalisable. More Winograd schemas can be resolved by one background knowledge principle. Thirdly, our method does not only resolve the pronoun, but also it gives both specific and abstract reasons for the answer.

# Chapter 5

# Methods 2 & 3: BERT and our ensemble approach

In order to tackle the Winograd schemas in the thanking domain, we also used a machine learning method and an ensemble method as well as the domain-specific high-level knowledge-based reasoning method in Chapter 4. As a machine learning method, we used Kocijan et al.'s (2019) BERT explained in Chapter 2. As an ensemble method, we used our original method to combine the knowledge-based reasoning method and the machine learning method. The sections in this chapter gives how each method is used to resolve the domain-specific Winograd schemas.

## 5.1 BERT

We chose to use BERT as a machine learning method to resolve the Winograd schemas in the thanking domain for two reasons. The first reason is that BERT and its variants give the high accuracies including the highest accuracy on the original WSC273 data set (Kocijan et al., 2019; Sakaguchi et al., 2019; Kocijan et al., 2020). The second reason is that the BERT methods can be used in the thanking domain without any modification since they are designed to tackle *general* Winograd schemas (Kocijan et al., 2019; Sakaguchi et al., 2019; Kocijan et al., 2020).

Among the BERT and its variants, we chose Kocijan et al.'s (2019) BERT that is fine-tuned on the data sets similar to Winograd schemas. This model's accuracy on the WSC 273 data set is $72.5\%$ which is higher than that of Devlin et al.'s (2019) BERT which is *not* fine-tuned by $10.6\%$ (Kocijan et al., 2019). Even though Sakaguchi et al.'s (2019) BERT which is trained on WinoGrande gives the current best accuracy on the WSC 273 data set, we did not choose this model because our thanking domain data set is extracted from WinoGrande. This is to ensure that the test set in the thanking domain is not used in the training by any means.

Considering that using fine-tuning can improve BERT's performance on Winograd schemas (Kocijan et al., 2019; Sakaguchi et al., 2019), we also fine-tuned Kocijan et al.'s (2019) BERT with the train set in the *thanking* domain. This fine-tuned model was used to see the difference with Kocijan et al.'s (2019) BERT which is only fine-tuned with Winograd schema sentences in general domain. The train set, the test set and the validation set in the thanking domain are defined depending on each experiment designed in Chapter 6. The train set is used for the fine-tuning and the validation set is used to decide when to stop learning in order to avoid overfitting.

We used Kocijan et al.'s (2019) repository (LINK) to implement and fine-tune the authors' BERT model with respect to the thanking domain. We used Pytorch for the implementation and the fine-tuning since the authors also use Pytorch. The Winograd schema sentences in the thanking domain were transformed into the format the authors use by replacing the pronoun with the mask token.

## 5.2 Combining both the knowledg-based reasoning method and BERT

We propose an ensemble method that combines machine learning and knowledge-based reasoning. Our motivation is to mitigate each approach's limitation by combining them. As discussed in Chapter 2, machine learning's weakness is that it is common for machine learning to give both correct and wrong predictions, and especially in the case of deep learning, we cannot give a logical explanation why it gives a wrong prediction (Devlin et al., 2018; Kocijan et al., 2019). On the other hand, knowledge-based reasoning' weakness is that it cannot give any answer if there is no match in the knowledge base, and building a knowledge base to cover all possible cases is expensive (Bailey et al., 2015; Sharma, 2019).

To mitigate these limitations and improve performance, our ensemble approach has two principles. The first principle is that if the knowledge-based reasoning method and the machine learning method give a prediction that is not same, the prediction from the knowledge-based reasoning has the priority as its prediction is based on the definite logical explanation. The second principle is that if the knowledge-based reasoning gives 'no prediction', the machine learning method's prediction is chosen as an answer instead of leaving the Winograd schema sentence unresolved (unanswered).

Informally, these principles can be understood by humans' behaviour. Humans can solve a problem based a logical explanation, but some problems can be solved by intuition without the logical interpretation in spite of the bias caused by intuition (Kahneman, 2012). For example, if a sentence is given for humans to decide whether the sentence is gramatically correct, even when they have no grammatical knowledge (similar to the knowledge base), they can say that the sentence *looks normal or not* (similar to the likelihood in machine learning) by intuition.

Based on the two principles, our ensemble method is designed as shown in Figure 5.1. Suppose a Winograd schema sentence is given. The first step is to use the knowledge-based reasoning method *first* to resolve the sentence. If it gives a single prediction, that answer will be the final answer. However, If it gives no answer or multiple answers, we use the machine learning method (BERT) as it can always give an answer. In this case, the answer from BERT would be the final answer.

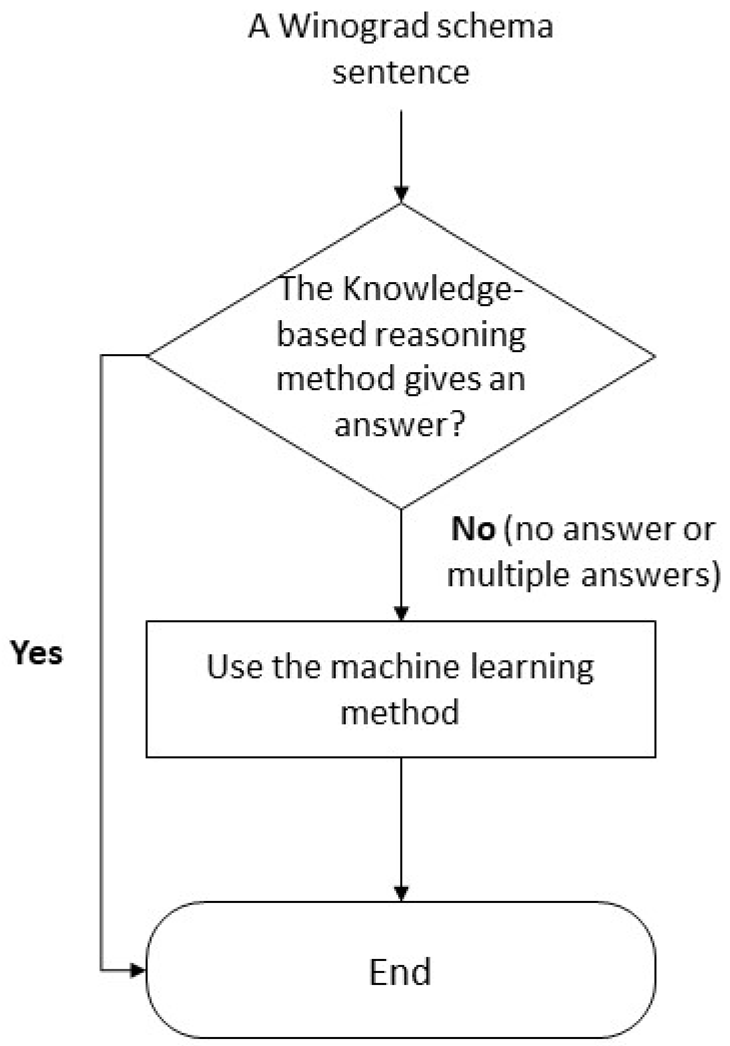
A Winograd schema sentence
The Knowledge-based reasoning method gives an answer?
No (no answer or multiple answers)
Yes
Use the machine learning method
End

*Figure 5.1: An algorithmic flow of combining knowledge-based reasoning and machine learning*

# Chapter 6

# Evaluation

In this chapter, we evaluate the methods to tackle Winograd schemas in the thanking domain. Evaluation metrics, experiment design and analyses of the results are given in each section.

## 6.1 Evaluation metrics

We used the three evaluation metrics: accuracy, a 'conditioned' accuracy and a 'robust' accuracy considering the characteristics of the Winograd Schema Challenge (WSC). The last two metrics are defined by us which are specific to evaluating Winograd schemas. One of the characteristics that we considered is that Winograd schemas have no fixed classes, but each Winograd schema has the two candidates (classes) different from those of the other Winograd schemas (Levesque et al., 2012). These following two examples are from WinoGrande in the thanking domain, and they are from *different* Winograd schemas:

1. Brian loaned their phone to Neil because theirs broke, and [he] thanked them for being so generous.
    - **Candidates for the pronoun:** Brian / Neil, **answer:** Neil
2. Tanya loved the washer that Mary bought for her, so [she] received a thank you note.
    - **Candidates for the pronoun:** Tanya / Mary, **answer:** Mary

These examples demonstrate that the candidates (classes) are different depending on the Winograd schema. That is why the commonly used evaluation metrics in a classification task such as precision and recall cannot be used in WSC.

With respect to our evaluation metrics, accuracy is same as what is used in other WSC-related researches (Kocijan et al., 2020), but the other two metrics are modified. The 'conditioned' accuracy is a slight variant of accuracy, and we defined the 'robust' accuracy in terms of WSC by improving the method of Trichelair et al. (2018). The subsections give each metric's explanation and motivation.

### 6.1.1 Accuracy and the ‘conditional’ accuracy

Accuracy is widely used in evaluating Winograd schemas, and it can be understood as the proportion of the correct answers out of all the questions (Kocijan et al., 2020). In terms of accuracy in Winograd schemas, the case of “no answer” or multiple answers needs to be considered as knowledge-based reasoning methods do not always give a single answer (Rahman and Ng, 2012; Sharma, 2019). In this sense, the accuracy used in Winograd schemas (Kocijan et al., 2020) can be written as:

$$\frac{Correct\ prediction}{Correct\ prediction + Wrong\ prediction + No\ answer + Multiple\ answers}.$$

Even though both “no answer” and wrong prediction are not correct prediction, “no answer” seems more useful than wrong prediction. In the case of no answer, another model would have one more chance to make a prediction. In this sense, “no answer” can be better than wrong prediction, and this is the reason why we introduce the ‘conditional’ accuracy as an evaluation metric. We are inspired by the conditional accuracy in decision lists (Abney, 2007) and define the conditional accuracy in Winograd Schemas as:

$$\frac{Correct\ prediction}{Correct\ prediction + Wrong\ prediction\ (excluding\ no\ answer\ and\ multiple\ answers)}.$$

In the conditional accuracy, “no answer” and multiple answers are excluded in the denominator. It means that the ‘conditional’ accuracy is calculated on the *condition* that predictions are correct or wrong with “no answer” and multiple answers excluded.

### 6.1.2 ‘Robust’ accuracy

Accuracy alone cannot be a solid ground for a model’s performance in WSC (Trichelair et al., 2018). A Winograd schema can be resolved with a chance of $50\%$ since there are only two candidates (Levesque et al., 2012). To double-check whether a correct prediction is not out of luck, Trichelair et al. (2018) suggest generating a varient of the original Winograd schema sentence by switching the nouns. If a model predicts correctly on both the original sentence and its variant, the chance of being correct out of luck goes down. As the success chance is $50\%$ equally, this probabilty to be successful twice continuosly would follow a binomial distribution (DeGroot and Schervish, 2012) and can be calculated as: $\binom{2}{2} \times (\frac{1}{2})^2 \times (1 - \frac{1}{2})^0 = 1 \times (\frac{1}{4}) \times 1 = 0.25$. Despite its usefulness, the chance of $0.25$ seems to be still a high probability.

To lower the chance of being correct out of luck by large margin, we introduce ‘robust’ accuracy by adding three more variants of the original sentence. These variants are generated by replacing each name in the sentence with the randomly chosen name. As WSC is a pronoun resolution task, the randomly chosen names should follow the gender of the pronoun. In the following example, since the pronoun is “she”, all the chosen names are female names. To implement this, we created the male name pool and the femal name pool separately from the names that appear in the thanking domain set. In addition, when the two names in the sentence are replaced, the randomly chosen names should not be same. An example of switching and replacing the names from the thanking data set orginated from WinoGrande (Sakaguchi et al., 2019, boldness mine) is given below:

- **Original sentence**: **Kayla** cooked sticky white rice for **Jennifer**, and [she] was thanked for making such delicate rice.
- **The nouns swithed**: **Jennifer** cooked sticky white rice for **Kayla**, and [she] was thanked for making such delicate rice.
- **The nouns replaced 1**: **Tanya** cooked sticky white rice for **Kayla**, and [she] was thanked for making such delicate rice.
- **The nouns replaced 2**: **Erin** cooked sticky white rice for **Tanya**, and [she] was thanked for making such delicate rice.
- **The nouns replaced 3**: **Lindsey** cooked sticky white rice for **Christine**, and [she] was thanked for making such delicate rice.

By adopting this replacing method, we can have the original sentence and four more variants of the sentence. Only when a model's predictions are correct on all these five sentences, its prediction on the original sentence is considered 'robustly' correct. Thus, the probability of being correct on all of these by chance decreases further. As this probabilty also follows the binomial distribution with the same logic of calculating the probability of the method of Trichelair et al. (2018), our method's probability can be calculated as: $\binom{5}{5} \times (\frac{1}{2})^5 \times (1 - \frac{1}{2})^0 = 1 \times (\frac{1}{2})^5 \times 1 \approx 0.03$, which is below $0.05$. The four variants are only used to evaluate whether the prediction on the original sentence is 'robustly' correct, and thus thses variants do not increase the number of the test examples in the evaluation. The limitation of this method is that this method can only be used when the nouns that the pronoun refers to are names. In the thanking data set, the proportion of human names for the candidates is $94.2\%$ as shown in Table 3.5, which seems to useful in this case.

## 6.2 Experiment Design

Two experiments were implementend to see the models' performance in different conditions. For the first and second experiments, the $80$ sentences in Table 3.6 that can be paired in the thanking domain were used excluding the non-paired sentences. The purpose of the first experiment is to see the performance when each sentence in the train set has similarity with one of the test set sentences. On the other hand, the second experiment is to evaluate models when no sentence in the train has similarity with any of the test set sentences. In other words, a pair of

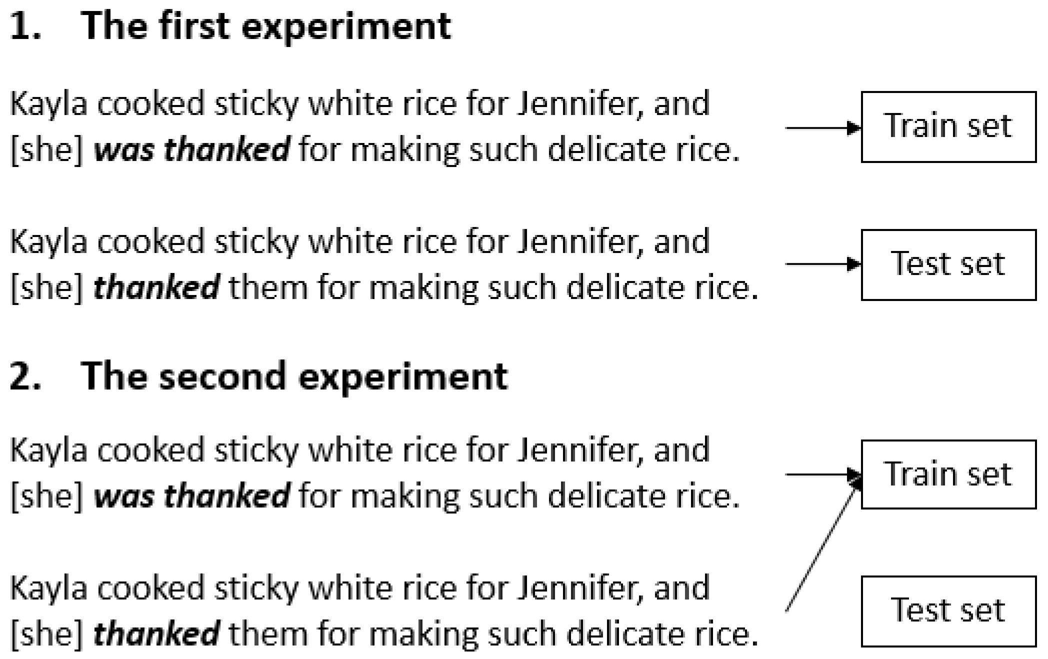
1. The first experiment
Kayla cooked sticky white rice for Jennifer, and [she] was thanked for making such delicate rice.
Train set
Kayla cooked sticky white rice for Jennifer, and [she] thanked them for making such delicate rice.
Test set
2. The second experiment
Kayla cooked sticky white rice for Jennifer, and [she] was thanked for making such delicate rice.
Train set
Kayla cooked sticky white rice for Jennifer, and [she] thanked them for making such delicate rice.
Test set

*Figure 6.1: An example of splitting the data set for the first and second experiments*

similar sentences have to be together either in train set or in test set. Figure 6.1 shows these relationships.

To achieve each experiment's purpose, the thanking data set was split in the following ways. In the first experiment, each pair of the sentences needed to be separated, and thus the thanking data set was split into 50% train set and 50% test set (50:50 splitting is the only way to implement the first experiment). Within each pair of the sentences, one of them was randomly put into train set and the other was put into test set. In the second experiment, each pair of the sentences should *not* be separated in splitting. Under this condition, the paired data set was split into 50% train set and 50% test set randomly. 50:50 splitting was also chosen in the second experiment as smaller proportion of test set such as 20% would lead to the test set containing too small sentences such as 16. This small data set is highly like to have more bias, and thus it would be more objective to use 50% for the test set which has 40 sentences.

The validation set was defined for each experiment to avoid overfitting in neural networks methods such as BERT. In general, the more training is done, the more the model would be fitted to the train set, but this does not necessarily mean that the model is fitted to generalisable data (Kocijan et al., 2019). In order to make the model generalisable, the performance on the validation set was checked at each epoch (one epoch refers to when the whole train set is used for training), and we chose the model at the epoch when the performance on the validation set is highest. In the first and second experiments, the 91 non-paired sentences were used for the validation set in order not to reduce the number of the sentences in the train set and the test set.

## 6.3 Results

### 6.3.1 The overall results

Table 6.1 shows the results of the first experiment. The model combining the knowledge-based reasoning method and BERT gave the best accuracy as 90% and the best robust accuracy as 85%. In terms of the conditional accuracy, it is the knowledge-based reasoning method that gave the best score as 96.7%.

| Model | Accuracy | 'Robust' accuracy | 'Conditional' accuracy |
|---|---|---|---|
| BERT (Kocijan et al., 2019) | 70.0% (28/40) | 62.5% (25/40) | 70.0% (28/40) |
| BERT fine-tuned with the train set | 47.5% (19/40) | 42.5% (17/40) | 47.5% (19/40) |
| Knowledge-based reasoning | 72.5% (29/40) | 72.5% (29/40) | **96.7**% (29/30) |
| Knowledge-based reasoning + BERT(Kocijan et al., 2019) | **90.0**% (36/40) | **85.0**% (34/40) | 90.0% (36/40) |

*Table 6.1: The results of the first experiment*

In Table 6.2, the results of the second experiment showed the same patterns as the first experiment. In terms of the accuracy and the robust accuracy, the model that combines the knowledge-based reasoning and BERT gave the best scores as 80% and 72.5% respectively. On the other hand, the knowledge-based reasoning method gave the best conditional accuracy as 93.8%.

| Model | Accuracy | 'Robust' accuracy | 'Conditional' accuracy |
|---|---|---|---|
| BERT (Kocijan et al., 2019) | 77.5% (31/40) | 70.0% (28/40) | 77.5% (31/40) |
| BERT fine-tuned with the train set | 75.0% (30/40) | 70.0% (28/40) | 75.0% (30/40) |
| Knowledge-based reasoning | 37.5% (15/40) | 37.5% (15/40) | **93.8**% (15/16) |
| Knowledge-based reasoning + BERT(Kocijan et al., 2019) | **80.0**% (32/40) | **72.5**% (29/40) | 80.0% (32/40) |

*Table 6.2: The results of the second experiment*

Overall, it can be understood that the model combining the knowledge-based reasoning and BERT outperforms the other models in terms of accuracy and 'robust' accuracy. It can be explained by the fact that this approach can complement each single model's weakness. With respect to BERT, its weakness is found in 'robust' accuracy and modified precision of the two experiments. In both experiements, the ‘robust’ accuracies are decreased compared to the corresponding accuracies by $5 \sim 7.5\%$. In addition, the conditional accuracies of BERT are lower than those of the knowledge-based reasoning method. On the other hand, the knowledge-based reasoning method alone has weakness in that the accuracies and ‘robust’ accuracies can be lower than those of BERT. The following four subsections give the detailed analysis of each method.

### 6.3.2 Analysis of the result of BERT

BERT(Kocijan et al., 2019) is a different model from the fine-tuned BERT as it was not fine-tuned with the train set of the thanking domain. As this model is free from the effect of the train set, its accuracies on both experiments have smaller difference than those of the fine-tuned BERT. While the fine-tuned BERT’s difference is $27.5\%$, its difference is only $7.5\%$.

In terms of its accuracy compared to the other models, it showed the best performance among the single model approaches in both experiments as $70\%$ and $77.5\%$ respectively. Even though its accuracies are still lower than the model combining the knowledge-based reasoning and BERT, its good performance implies that transfer learning using broader domains in Winograd schemas can also be effective in the specific thanking domain.

However, its robust accuracies were decreased compared to the corresponding accuracies in both experiments by $7.5\%$ while the knowledge-based reasoning method showed no difference between them. It implies that BERT, as a machine learning method, can be misled by bias in the data. Table 6.3 shows the case where it predicted correctly on the original but predicted wrongly on some of the variants. It is an interesting case because the only change made to the variants was the names. Even though it is obvious to humans that the change of the names does not change the semantic relationships in the original sentence, this change could mislead the machine learninng method to make a wrong decision. Nevertheless, in most cases except the three cases in each experiment, BERT seems to understand the fundamental patterns to give a correct answer as it predicted correctly on the original sentences and their variants in the rest of the cases.

### 6.3.3 Analysis of the result of the fine-tuned BERT

In the first experiment, Table 6.4 shows that the validation accuracy is highest at the first epoch and decreases from the second epoch. We chose the model at the first epoch and tested this model on the test set. The test accuracy is $47.50\%$ which is even lower than $50\%$. This accuracy is lower than that of BERT (Kocijan et al., 2019) that was not fine-tuned with the train set by $22.5\%$.

This low accuracy on the test set suggests that BERT is vulerable to bias in the data. In this case of the first experiment, its low performance seems to be related to the similarity between the train set and the test set. Different from a normal classification task where similar sentences would be classified as the same class, this is not the case in Winograd schemas. As shown

| Type | Sentence | Prediction |
|---|---|---|
| Original | Patricia is known to be less gracious than Lindsey because she is just thankful for everything | Correct |
| Switched | Lindsey is known to be less gracious than Patricia because she is just thankful for everything | **Wrong** |
| Replaced 1 | Jennifer is known to be less gracious than Natalie because she is just thankful for everything | Correct |
| Replaced 2 | Sarah is known to be less gracious than Kayla because she is just thankful for everything | Correct |
| Replaced 3 | Sarah is known to be less gracious than Tanya because she is just thankful for everything | **Wrong** |

*Table 6.3: The BERT's predictions in the second experiment on an example of WinoGrande and its variants*

| Epoch | Train accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| 1 | 100.00% | **90.11%** | **47.50%** |
| 2 | 100.00% | 86.81% | - |
| 3 | 100.00% | 86.81% | - |
| 4 | 100.00% | 86.81% | - |
| 5 | 100.00% | 87.91% | - |
| 6 | 100.00% | 90.11% | - |
| 7 | 100.00% | 90.11% | - |
| 8 | 100.00% | 90.11% | - |
| 9 | 100.00% | 90.11% | - |
| 10 | 100.00% | 90.11% | - |

*Table 6.4: The fine-tuned BERT's accuracies on the train set, validation set and test set at each epoch in the first experiment*

in Table 6.5, a pair of similar sentences were separated into either train set or test set, but the answer is *opposite*. In this example, the model is highly like to predict the answer for the test set same as that of the train set, which is "Kayla". That would be the wrong prediction.

In the second experiment, the fine-tuing result at each epoch is given in Table 6.6. As in the first experiment, the validation accuracy at the first epoch was the highest, and thus we chose the model at this epoch. With this model, the test accuracy of the second experiment was 75.0% which is is slightly lower than that of BERT(Kocijan et al., 2019) by 2.5%.

Compared to the first experiment, this result in the second experiment implies that at least the fine-tuning is not harmful in this case. This is also compatible with the finding of Kocijan et al. (2019) that fine-tuning with the paired data set improves the model more than fine-tuning with the non-paired data set. As seen in Table 6.7, when the paired data set is used in the training, a model is expected to learn the cases when the similar sentences are used as inputs,

| Sentence | Train/Test set | Prediction | Answer |
|---|---|---|---|
| "Kayla cooked sticky white rice for Jennifer, and [she] *was thanked* for making such delicate rice." | Train set | Kayla | Kayla |
| "Kayla cooked sticky white rice for Jennifer, and [she] *thanked* them for making such delicate rice." | Test set | **Kayla** | Jennifer |

*Table 6.5: An example of similarity bias in the first experiment (The example sentences are from WinoGrande (my italics))*

| Epoch | Train accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| 1 | $100.00\%$ | $\mathbf{86.81}\%$ | $\mathbf{75.00}\%$ |
| 2 | $100.00\%$ | $75.82\%$ | - |
| 3 | $100.00\%$ | $47.25\%$ | - |
| 4 | $100.00\%$ | $52.75\%$ | - |
| 5 | $100.00\%$ | $58.24\%$ | - |
| 6 | $100.00\%$ | $60.44\%$ | - |
| 7 | $100.00\%$ | $58.24\%$ | - |
| 8 | $100.00\%$ | $59.34\%$ | - |
| 9 | $100.00\%$ | $60.44\%$ | - |
| 10 | $100.00\%$ | $61.54\%$ | - |

*Table 6.6: The fine-tuned BERT's accuracies on the train set, validation set and test set at each epoch in the second experiment*

but the answers can be different. This seems to be the reason why fining-tuning with the paired data set would give the better accuracy.

Same limitations were found in both experiments with the fine-tuned BERT. The 'robust' accuracies were lower than the corresponding accuracies by $5\%$, which was also found in the results of BERT(Kocijan et al., 2019). Regardless of fine-tuning, BERT appears to be affected by little changes that does not affect any semantic relationship. Furthermore, training converged too early at the first epoch in both experiments. It seems to be mainly due to the small size of the train data set that has only $40$ sentences. In that sense, we *cannot* conclude that fine-tuning with the domain-specific train set is less effective than fine-tuning with the general Winograd Schemas' train set. Future study needs to be done regarding this area with more domain-specific Winograd schema train set.

| Sentence | Train/Test set | Prediction | Answer |
|---|---|---|---|
| "Kayla cooked sticky white rice for Jennifer, and [she] *was thanked* for making such delicate rice." | Train set | Kayla | Kayla |
| "Kayla cooked sticky white rice for Jennifer, and [she] *thanked* them for making such delicate rice." | **Train set** | **Jennifer** | Jennifer |

*Table 6.7: An example of ther similar sentences in the train set of the second experiment (The example sentences are from WinoGrande (my italics))*

### 6.3.4 Analysis of the result of knowledge-based reasoning

In the results, the performance of the knowledge-based reasoning varies depending on the experiment and the evaluation metric. This method showed the large difference in terms of accuracy between the first and the second experiments. The large difference is also found between accuracy and the conditional accuracy in each experiment.

Regarding the difference between the first and the second experiments, the accurcy of the first experiement is 72.5% which is larger than that of the second experiment by 35%. This difference is mainly related to the degree of similarity between the train set and the test set. As in the first experiment, high similarity between the train set and the test set is advantageous in terms of the knowledge-based reasoning method, which is opposite to the case of the fine-tuned BERT. As the semantic role rules are defined by the train set, these rules can only be applied to the sentences in the test set that are similar to those in the train set.

The second difference between accuracy and the conditional accuracy is related to the knowledge-based reasoning method's high conditional accuracy. This method gave the highest conditional accuracies in both experiments as 96.7% and 93.8% respectively while its accuracy is 72.5% and 37.5%. It is interesting that in the second experiment, its conditional accuracy is over 90% though its accuracy is only 37.5%. This fact demonstrates that some of the unanswered Winograd schema sentences can decrease the accuracy, but if there is a match in the knowledge base, this method gives an answer that is highly likely to be correct.

### 6.3.5 Analysis of the result of combining knowledge-based reasoning and BERT

In both experiments, the method of combining knowledge-based reasoning and BERT gave the best accuray and the robust accuracy. In the first experiment, the accuracy gap between this ensemble method and the second best method (the knowledge-based reasoning method) is 17.5%. On the other hand, the accuracy gap between the ensemble model and the second best model ( Kocijan et al.'s (2019) BERT) in the second experiment is only 2.5% which is smaller. The similar trend is also found in terms of the robust accuracy.

The high performance of this ensemble method is related to the characteristics of knowledge-based reasoning and machine learning which mitigate each weakness. For the knowledge-based reasoning method, if there is a match in the knowledge base, its prediction is highly likely to be correct, which is over 90%. By applying the knowledge-based reasoning method first, it can reduce the limitation of the machine learning method that its prediction can be wrong with higher possibility. On the other hand, if there is no match in the knowledge base, the knowledge-based reasoning method does not give any prediction which could be wrong. This limitation of the unanswered sentences can be reduced by giving the sentence a second chance to be resolved by the machine learning method. For the machine learning method, though it could give a wrong prediction, it can always make a prediction by comparing the likelihoods. As the machine learning method would give correct predictions on at least some of the unanswered Winograd schemas, it is better than no predictions for them in terms of accuracy.

These strong points of the ensemble model were found in the experiments. An example from WinoGrande in the thanking domain is given to demonstrate that the limitation of the machine learning method can be reduced:

- Sentence: Craig wrote a thank you note to Dennis because [he] sent him a thoughtful gift to him.
- Candidates: Craig / Dennis, Answer: Dennis
- The machine learning method's prediction: Craig (**incorrect**)
- The knowledge-based reasoning method's prediction: Dennis (**correct**)

This example shows that the knowledge-based reasoning method which is applied first can predict correctly on a sentence that the machine learning method predicts incorrectly on. There is another example from WinoGrande in the thanking domain that shows how the limitation of the knowledge-based method can be mitigated:

- Sentence: Natalie sat in the therapist's office and thanked Christine for the observation, because [she] was a grateful patient.
- Candidates: Natalie / Christine, Answer: Natalie
- The machine learning method's prediction: Natalie (**correct**)
- The knowledge-based reasoning method's prediction: **No answer**

As shown in this example, an unanswered Winograd schema sentence by the knowledge-based reasoning method can be resolved by the machine learning method.

# Chapter 7

# Conclusion

Our reasearch aims at tackling Winograd schemas in the *thanking* domain. Regarding this task, we contribute to the four main areas. The first contribution is that we propose a keyword approach to specify a domain in WSC. To our best knowledge, this is a novel approach in terms of WSC. The second contribution is that we suggest a domain-specific high-level knowledge-based reasoning method which is an advanced modification of Sharma's (2019) approach. The third contribution is that we propose a simple ensemble method of combining knowledge-based reasoning and machine learning. The fourth contribution is that we suggest a 'robust' accuracy by improving the method of Trichelair et al. (2018) to achieve more objective evaluation.

Regarding the first contribution, this keyword approach seems to be able to identify more specific domains compared to the previously defined domains such as causal relations. Specifically in the thanking domain, using keywords were successful in identifying Winograd schema sentences that belong to the thanking domain. It is an interesting finding that more than $75\%$ of the Winograd schema sentences in the thanking domain can be represented into the only five domain-specific higher-level (more abstract) patterns.

With respect to the second contribution, our domain-specific high-level knowledge-based reasoning approach has two main advantages compared to Sharma's (2019) approach. Firstly, our approach can resolve more Winograd schemas in that our background knowledge principles are in higher-level (more abstract). This would be true under the condition that the same train set is used to build a knowledge base in both methods. Secondly, our approach can derive the semantic roles and their relationship as well as resolve Winograd schemas. Considering that *thinking* can be defined as using background knowledge to derive something which is the pronoun resolution in WSC (Levesque et al., 2012), these derived semantic roles and their relationship could be the outputs of *thinking* as well. Basically, predefined rules that can be seen as background knowledge are used for these semantic derivations.

The simple method of combining knowledge-based reasoning and machine learning, which is the third contribution, turns out to be more accurate than a single method. In the two experiements, this ensemble method showed the best accruacy and 'robust' accuracy. Its good performance can be related to its similarity to a human intelligence. As humans use both reasoning and intuition (Kahneman, 2012), this ensemble method uses both knowledge-based reasoning and machine learning, which can mitigate each single method's weakness.

As the fourth contribution, 'robust' accuracy can be more objective than just accuracy or Trichelair et al.'s (2019) switching method. In terms of probability of being correct by chance, the probability of 'robust' accuracy is lower than 0.05 which is the lowest among them. This metric is also useful to demonstrate randomness or unstableness of machine learning. For example, in the two experiements the 'robust' accuracies are lower than the accuracies when the models include a machine learning method.

## 7.1 Other appllications

Other than these four contributions, some of our approaches can be used to tackle other tasks such as Choice Of Plausible Alternaties (COPA) (Roemmele et al., 2011). This task is to choose more plausible alternative among the two with respect to a given sentence (Roemmele et al., 2011). Firstly, we can use the keyword approach to identify a COPA question that belongs to a thanking domain. The following example is from the COPA data set and it is extracted by the keyword "thank":

I wrote a thank-you note to my grandmother.

- Alternative 1: She became forgetful.
- Alternative 2: She sent me a gift.

Secondly, we can make a small modification to the domain-specific high-level knowlege-based reasoning in order to derive the answer in COPA. It is interesting that in this example, as we derive semantic roles in WSC, we can derive the major semantic roles from the given sentence and the second alternative in COPA. In the given sentence, the semantic role of "I" is "thanker" and that of "grandmother" is "being thanked", and in the second alternative, the semantic role of "She" is "giver" and that of "me" is "given". In this way, by using the semantic roles that our approach uses, some examples of this task might be resolved with the slight modifications of the reasoning rules.

## 7.2 Limitations and future work

Our research has some limitations. The first limitation is related to the small number of the sentences. Our thanking data set only contains 171 Winograd schema sentences, and the sentences used in the two experiments are the only paired data set which has 80 sentences. For the future work, as smaller data set is vulnerable to data bias, more Winograd schemas related to thanking domain can be collected. Also, the non-paired data set can also be used for an experiment to see the difference.

The second limitation is that we only targeted the thanking domain in this research, which is a small part of the whole Winograd schemas. For the future work, it needs to be researched that what other domains can be extracted by keywords. Also, we need to study whether these other domains have some domain-specific high-level patterns as they were shown in the thanking domain.

# Bibliography

Abney, S. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman and Hall/CRC.

Bailey, D., Harrison, A., Lierler, Y., Lifschitz, V. and Michael, J. 2015. The Winograd Schema Challenge and Reasoning about Correlation In: 2015 *AAAI Spring Symposium Series, 23-25 March 2015, USA*.

Bennett, B. 2020. *Logical Analysis of Winograd Schemas*. Unpublished.

Brachman R.J. and Levesque H.J. 2004. *Knowledge Representation and Reasoning*. Morgan Kaufmann.

DeGroot, M.H. and Schervish, M.J. 2012. *Probability and Statistics*. 4th ed. Pearson.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs].*

Dror, R., Shlomov, S. and Reichart, R. 2019. Deep Dominance - How to Properly Compare Deep Neural Models In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.*, pp.2773–2785.

Ettinger, A. 2019. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *arXiv:1907.13528 [cs].*

Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In: *Proceedings of International Logic Programming Conference and Symposium.*, pp.1070-1080.

Hu, M. and Liu, B. 2004. Mining and Summarizing Customer Reviews. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004).*

Kahneman, D. 2012. *Thinking, fast and slow*. London: Penguin.

Kocijan, V., Cretu, A.-M., Camburu, O.-M., Yordanov, Y. and Lukasiewicz, T. 2019. A Surprisingly Robust Trick for Winograd Schema Challenge. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.*, pp.4837–4842.

Kocijan, V., Lukasiewicz, T., Davis, E., Marcus, G. and Morgenstern, L. 2020. A Review of Winograd Schema Challenge Datasets and Approaches. *arXiv:2004.13831 [cs].*

Kowsari, K., Meimandi, K.J., Heidarysafa, M., Mendu, S., Barnes, L.E. and Brown, D.E. 2019. Text Classification Algorithms: A Survey. *Information*. **10**(4).

Levesque, H., Davis, E. and Morgenstern, L. 2012. The Winograd Schema Challenge. In: *The 13th International Conference on Principles of Knowledge Representation and Reasoning, 10-14 June 2012, Italy*.

Merritt, D. 2017. *Adventure in Prolog*. Independently published.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in Neural Information Processing Systems 26*., pp.3111–3119.

Mueller, E.T. 2014. *Commonsense Reasoning: An Event Calculus Based Approach*. Morgan Kaufmann.

Rahman, A. and Ng, V. 2012. Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In: *EMNLP-CoNLL* .

Roemmele, M., Bejan, C.A. and Gordon, A.S. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In: *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning, 21-23 March 2011, USA*.

Saba, W.S. 2018. Is there a 'Simple' Machine Learning Method for Commonsense Reasoning? A Short Commentary on Trinh & Le (2018). *arXiv:1810.00521 [cs].*

Sakaguchi, K., Bras, R.L., Bhagavatula, C. and Choi, Y. 2019. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *arXiv:1907.10641 [cs].*

Schaub, T. 2018. *Answer Set Solving in Practice*. [Online]. [Accessed 3 September 2020]. Available from: https://www.cs.uni-potsdam.de/ torsten/Potassco/Slides/introduction.pdf

Sharma, A., Vo, N., Aditya, S. and Baral, C. 2015. Identifying Various Kinds of Event Mentions in K-Parser Output. In: *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*., pp.82–88.

Sharma, A., Vo, N.H., Aditya, S. and Baral, C. 2015. Towards Addressing the Winograd Schema Challenge - Building and Using a Semantic Parser and a Knowledge Hunting Module. In: *IJCAI 2015*., pp.1319-1325.

Sharma, A. 2019. Using Answer Set Programming for Commonsense Reasoning in the Winograd Schema Challenge. *arXiv:1907.11112 [cs].*

Trichelair, P., Emami, A., Trischler, A., Suleman, K. and Cheung, J.C.K. 2018. How Reasonable are Common-Sense Reasoning Tasks: A Case-Study on the Winograd Schema Challenge and SWAG. *arXiv:1811.01778 [cs, stat]*.

Turing, A.M. 1950. Computing Machinery and Intelligence. *mind*. **59**(236), pp.433-460.

# Appendix A

# Additional Codes

## A.1 Our knowledge base in ASP

### A.1.1 Additional rules to derive the semantic roles

1. More thanker

```
has_s(X, semantic_role, more_thanker) :-
        has_s(X,trait,Thankful),
        has_s(Thankful,extent,More),
        has_s(More,instance_of,more),
        has_s(Thankful,instance_of,Thankful_syn),
        syn_check(thankful, Thankful_syn).
```

2. Less thanker

```
has_s(X, semantic_role, less_thanker) :-
        has_s(X1,semantic_role,more_thanker),
        X != X1,
        is_candidate(X),
        is_candidate(X1),
        not has_s(X,semantic_role,more_thanker).
```

3. Not thanker

```
has_s(X, semantic_role, not_thanker) :-
        has_s(X,trait,Thankful),
        has_s(Thankful,negative,_),
        has_s(Thankful,instance_of,Thankful_syn),
        syn_check(thankful,Thankful_syn),
        not has_s(Thankful,extent,_).
```

```
has_s(X, semantic_role, not_thanker) :-
        has_s(Give,agent,X),
        has_s(Give,instance_of,Give_syn),
        syn_check(give,Give_syn),
        has_s(Give,recipient,Card),
        has_s(Card,trait,Thank),
        has_s(Thank,instance_of,Thank_syn),
        syn_check(thank,Thank_syn),
        has_s(Give,negative,_).
```

4. More given

```
has_s(X, semantic_role, more_given) :-
        has_s(Receive, agent, X),
        has_s(Receive, recipient, Something),
        has_s(Receive, instance_of, Receive_syn),
        syn_check(receive, Receive_syn),
        has_s(Something, trait, More),
        has_s(More, instance_of, more).

has_s(X, semantic_role, more_given) :-
        has_s(Receive, agent, X),
        has_s(Receive, instance_of, Receive_syn),
        syn_check(receive, Receive_syn),
        has_s(Receive,dependent,More),
        has_s(More, instance_of, more).
```

5. Less given

```
has_s(X, semantic_role, less_given) :-
        has_s(X1,semantic_role,more_given),
        X != X1,
        is_candidate(X),
        is_candidate(X1),
        not has_s(X,semantic_role,more_given).

has_s(X, semantic_role, less_given) :-
        has_s(Receive, agent, X),
        has_s(Receive, instance_of, Receive_syn),
        syn_check(receive, Receive_syn),
        has_s(Receive,dependent,Less),
        has_s(Less, instance_of, less).
```

6. More liked (passive)

```
has_s(X, semantic_role, more_liked) :-
        has_s(Like,recipient,X),
        has_s(Like,instance_of,like),
        has_s(Like,modifier,More),
        has_s(More,instance_of,more).
```

7. Less liked (passive)

```
has_s(X, semantic_role, less_liked) :-
        has_s(X1,semantic_role,more_liked),
        X != X1,
        is_candidate(X),
        is_candidate(X1).
```

### A.1.2 Additional rules to derive the relationship between the candidates

1. More thankful than

```
has_s(X, more_thankful_than, Y) :-
        has_s(X, semantic_role, more_thanker),
        has_s(Y, semantic_role, less_thanker).
```

2. Not gives thanks to

```
has_s(X, not_gives_thanks_to, Y) :-
        has_s(X, semantic_role, not_thanker),
        has_s(Y, semantic_role, not_being_thanked),
        has_s(_, relation, causer).
```

3. Is more given than

```
has_s(X, is_more_given_than, Y) :-
        has_s(X, semantic_role, more_given),
        has_s(Y, semantic_role, less_given).
```

4. Is preferred to

```
has_s(X, is_preferred_to, Y) :-
        has_s(X, semantic_role, more_liked),
        has_s(Y, semantic_role, less_liked).
```

### A.1.3 Additional rules to derive high-level representation of the pronoun

1. Owing

```
has_s(X, semantic_role, owing) :-
        has_s(X, semantic_role, more_given).
```

2. Less owing

```
has_s(X, semantic_role, less_owing) :-
        has_s(X, semantic_role, not_thanker).
has_s(X, semantic_role, less_owing) :-
        has_s(X, semantic_role, less_given).
```