An AI using Construction Grammar: Automatic Acquisition of Knowledge about Words

Denis Kiselev
HIKIMA.NET, Sapporo, Japan

Keywords: Natural Language Processing, Knowledge Representation and Reasoning, Cognitive Systems, Explainable AI, Construction Grammar, Winograd Schema.

Abstract: This paper deals with an AI implementation that uses knowledge in an original Construction Grammar (CG) format for deep understanding of text. CG is a means of processing knowledge pieces—aka constructions—that describe the form and meaning of text parts. Understanding consists in automatically finding in the text the knowledge that constructions contain and in creating knowledge networks that reflect the text information structure. Deeper understanding is achieved by propagating knowledge within the network, i.e. some constructions can share with others information about syntax, semantics, pragmatics and other text properties. A shortcoming of this information-rich method is limited coverage: only text for which a CG database is available can be understood; that database due to its complexity most often needs to be made up manually. The author attempts to increase the coverage by implementing automatic acquisition of word knowledge from sources such as an external (non-CG) knowledge base and formatting the knowledge as CG constructions. The resulting CG database has been used in evaluation experiments to understand the Winograd Schema (WS)—a test for AI. A 28% increase in accurate coverage and opportunities for further improvement are observed.

1 INTRODUCTION

This section introduces the issue the proposed knowledge acquisition method tackles: the limited coverage of text by deeper understanding, such as in the case of the computational CG. In the context of contemporary Natural Language Processing (NLP) the above issue can be presented as follows.

An NLP history survey (Brock, 2018) shows two major research areas: those featuring "recognition" and "reasoning". Among the former Brock (2018) mentions "neural networks coupled to large data stores", e.g. deep learning. For such approaches—aka statistical NLP—the major criterion for "making judgement" is the count of character combinations (words etc.) recognized in a large data store (e.g. in a collection of user input). The latter (reasoning) approaches—aka Natural Language Understanding (NLU)—aim for deep understanding of human cognition (Micelli et al., 2009) and attempt to infer by processing knowledge such as syntax, semantics and common sense. Computational CG is among such NLU approaches; Kiselev (2017) introduces CG as a formalism and describes its practical applications.

Both deep NLU and statistical NLP have strengths accompanied by weaknesses as shown below.

Modern statistical NLP methods are usually "trained" on large data (e.g., millions of word occurrences) so they can process (i.e. their knowledge database is enough to cover) considerably large inputs (Zhang et al., 2018). One problem with such methods is limited accuracy when it comes to difficult understanding tasks involving deep semantics or common sense reasoning (Mitkov, 2014; Richard-Bollans et al., 2018). To illustrate this problem: a number of statistical NLP methods (Peng et al., 2015; Sharma et al., 2015; Liu et al., 2017; Emami et al., 2018) can process considerably large quantities of WS sentences, however with limited accuracy; Bailey et al. (2015) describe WS as a test for AI. Another problem with statistical NLP, e.g. Machine Learning (ML), is limited transparency: it is often impossible or impractical to find out what exact "thoughts" led an AI of this kind to a certain decision (Brock, 2018). To find that out one would need to look at millions of nodes (words, etc) and their numeric weights (resulting from mathematical manipulations of word occurrence counts, etc) that activated certain networks. This limited explainability is a reason to call statistical

Kiselev, D.
An AI using Construction Grammar: Automatic Acquisition of Knowledge about Words.
DOI: 10.5220/0008865902890296
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 289-296
ISBN: 978-989-758-395-7; ISSN: 2184-433X

NLP AI with neural networks "a black box" (Brock, 2018).

Differently from statistical NLP, deep NLU approaches, e.g. CG ones (Kiselev, 2017; Raghuram et al., 2017), show high accuracy on the WS but can process (i.e. their knowledge database is enough to cover) limited numbers of WS sentences. This limited coverage results from more difficulty—than in the case of statistical NLP—to automatically create a database because the database has to reflect a richer information structure of text. Deep NLU requires more human effort to (manually) create the knowledge base. As for the explainability, the CG-based AI "thoughts" are usually clear from the data structure and/or reasoning rules used, as for instance in the case of the Kiselev (2017) and the AI method the present paper proposes. Implementations of this type are referred to as explainable AI (aka XAI).

To sum it up, statistical NLP (such as ML) demonstrates large coverage but limited accuracy and limited explainability, deep NLU (such as computational CG) demonstrates limited coverage but high accuracy and sufficient explainability.

2 CURRENT RESEARCH STAGE AND TARGETS

This section identifies the AI implementation the proposed knowledge acquisition method has built upon, and outlines future knowledge acquisition research prospects. The section ends with the question to be empirically answered by the present paper.

The author tackles the NLU coverage issue, explained in 1, by further improving an existing AI implementation. That AI (Kiselev, 2017) has shown high accuracy in answering questions from a WS dataset, but its CG database is limited to that dataset only and no automatic knowledge acquisition is implemented. That database is manually populated with information structures called constructions, each of which contains knowledge about the form and meaning of a word, a phrase or a sentence. To put it differently, the database has three types of constructions: word, phrase and sentence ones. It is important to note that the same constructions can be reused to understand text pieces with similar form and meaning.

This paper describes the current research stage at which the author looks into automatic acquisition of word constructions only. They are added to the database of the above AI and used for WS understanding. Automatic acquisition of phrase and sentence constructions are future stages of the research.

The paper is to answer the following question. Does automatically adding word constructions to the existing database result in any increase of the accurate coverage for unknown sentences? This question is important because the word constructions are meant to become components of phrase and sentence constructions by satisfying syntactic, semantic and pragmatic requirements (listed "inside" phrase and sentence constructions). The answer to the question contributes to the practical value of the proposed method applied to understanding the WS, an especially hard (Winograd, 1980) task for AI.

3 TEXT UNDERSTANDING

Before explaining how the knowledge is acquired, it is important to explain how the present AI implementation understands text.

3.1 Knowledge Networks

First, word constructions from the database are used to find known words. A construction is, simply put, a collection of `feature => value` pairs. This is a template so instead of the `feature` and `value` any meaningful information can be used. That information is to describe form and/or meaning of text pieces of three types (recall the construction types mentioned in 2). For instance, a familiar word "phone" was found because the value (i.e. the word spelling) in the pair `string => phone` matched "phone" in the text. The boldface words within boxes in Fig. 1 represent constructions for the known words: the implementation now knows e.g. that "he" is a personal pronoun. It knows because the construction that matched "he" has the pair `type => personal` and the construction also says that "he" is a pronoun.

Next, the AI checks if the found known words (or, technically, the word constructions that have matched the spelling) fit into known phrase structures. Each phrase construction in the database represents a known phrase type, e.g. the verb phrase (VP) for "call George" (Fig. 1). These two words fit into the VP because the AI has found matching features and values. For instance, the values `action` in the word construction for "call" and `entity` in that for "George" have matched the same values in the VP construction; for space considerations the figure does not show every feature and value.

As all form and meaning requirements of the VP to its potential components were similarly met by the two word constructions (i.e. certain features and values were found matching), the constructions became

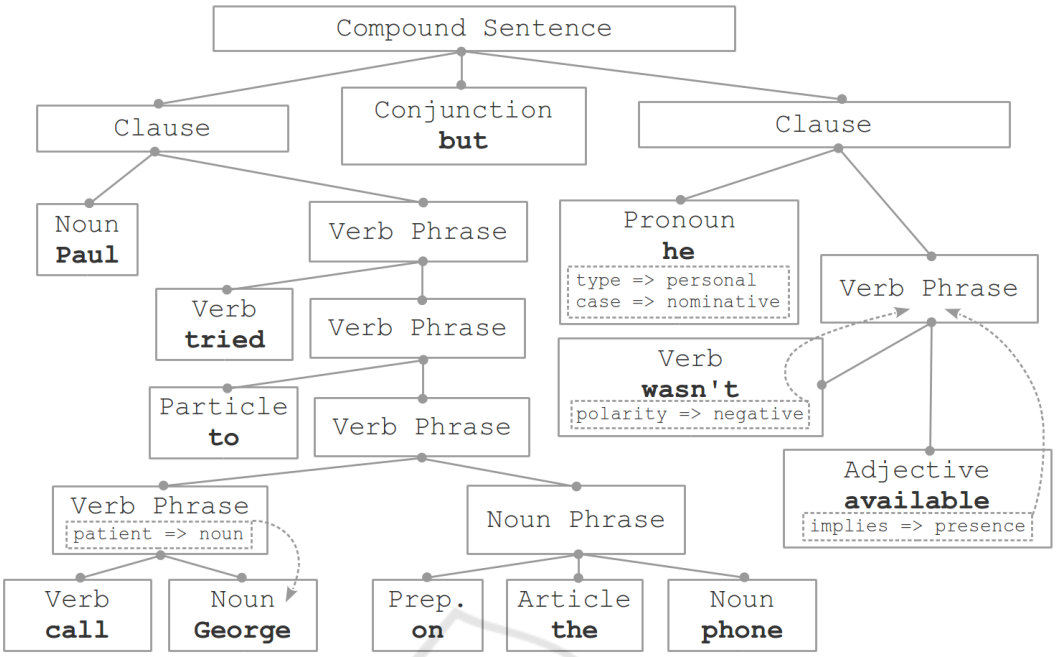

Figure 1: The knowledge network generated to find what "he" refers to in the WS sentence "Paul tried to call George on the phone but he wasn't available"; a concise illustration. Solid line boxes represent CG constructions.

components[1] of the VP. This VP construction has been (manually) designed to know its semantic structure `patient => noun` (Fig. 1): the noun denotes the patient, i.e. the receiver of the action. This semantic knowledge has been propagated from the VP to the "George" word construction, i.e. the AI has copied knowledge from the above feature-value pair. In this way the implementation has understood phrases and now knows that e.g. "call George" is a VP, "call" is an `action` and "George" is an `entity` as well as the `patient` (recipient) for "call". In other words, knowledge networks for phrases have been formed. In Fig. 1 the network components are connected by lines with circles on ends, dashed arrows show how knowledge propagated (i.e.was copied by the AI).

Sentences are understood the same way as phrases. Constructions for the understood phrases fit into the known sentence structure if features and values match. Knowledge listed as feature-value pairs is copied from some constructions to others. Technical details of what conditions feature-value pairs must meet so they match or so they are copied the way the construction designer desires are given by Kiselev (2017) . However, for the present AI some feature-value pairs have been discarded to make phrase and sentence constructions lighter, and regular expressions (regex)[2] are incorporated into constructions for more matching versatility. For instance, the same construction can be designed to match multiple word forms or parts of speech (POS).

Another feature of the present AI is recursive generation of alternative parses. Some cases for doing so are as follows. In English there is a number of words that, depending on the context, can be different POS, although the spelling is the same. If such word is used in a sentence and matches constructions for different POS, alternative sentence parses (each containing the construction for the different POS) are generated. If the same group of words in a sentence matches multiple constructions for different types of phrases, alternative sentence parses are also generated. At present the AI is set to output parses more extensively covered by the CG database knowledge, or simply parses with the larger number of matching constructions.

It is not true that the tree-like structure representing the knowledge network for the whole sentence in Fig. 1 has to be a syntactic tree generated by means of statistical NLP—athough, that is also implementable. In other words, what types of phrases the sentence is to consist of does not have to be determined by how many phrases of some type are usually found in large collections of data. By creating feature-value pairs the construction designer is free to have any construction

[1]Becoming components means getting marked by the AI as components and put in a specially allocated memory container that is traceable, retrievable and printable.

[2]Programming tools used for manipulating character strings, e.g. finding words or changing morphemes.

convey practically any meaning, match another construction and form a network that can have practically any syntactic structure. This kind of freedom is referred to by van Trijp (2017) as "chopping down the syntax tree".

By forming the sentence network exemplified in Fig. 1 the AI has understood, for instance, the following. The sentence (Fig. 1) has a clause, this clause has a VP, and this VP implies negated presence. This implication has been understood because the pairs `polarity => negative` and `implies => presence` from the "wasn't" and "available" constructions respectively were copied to the VP construction—copied because they met certain conditions as mentioned above, and met the conditions because the constructions had been manually designed to do so. In terms of human reasoning, this knowledge propagation can be described like so. As "wasn't" is negative and "available" implies presence and the two words form a VP (the way shown in Fig. 1), that VP implies negated presence. This is one example of the way CG knowledge propagation can in itself be looked upon as a chain of reasoning. The knowledge propagation implemented by the author draws upon the CG remarkable quality of feature and value propagation referred to by Steels (2017) as "percolation" and "merging".

The following CG concept should also be noted. It is not true that each single construction can describe only one unique piece of text. Constructions are meant to be reusable for (i.e. can describe) text pieces with similar forms and meanings. For instance a single `VP` construction can be "filled with" a variety of actual verbs and nouns from the text. The same concept applies to sentence constructions. This is what makes CG different from annotated corpora where a particular tag is attached to one word or one group of them.

3.2 Inference Rules

Rules are instructions for the present AI implementation to follow when more complex reasoning—such as the anaphora resolution task exemplified by the caption for Fig. 1—is involved. A rule defines a reasoning pattern as explained below. Kiselev (2017) has shown that WS anaphora resolution can be handled without using rules. However, that results in construction data proliferation: the need to use a larger number of long constructions each of which describes a separate sentence as a whole. Using rules for the present AI aims for shorter and more versatile constructions.

A rule has condition and action parts: the action is taken if the condition is met. Below is a human language gloss of a rule for anaphora resolution in the Fig. 1 sentence and in similar sentences; the construction data dealt with can be looked up by locating dashed boxes and arrows. *In a compound sentence a clause of which implies negated presence, into a part that is a nominative case personal pronoun, insert* `refers_to =>` *pointing to a part that denotes the patient (the receiver of the action).* To gloss it further: the sentence speaks about not being present, so "he" refers to "George". This is just one possible resolution of anaphora, a construction designer is free to come up with others.

4 KNOWLEDGE ACQUISITION

The knowledge (i.e. word constructions) acquisition targets unknown words. A word is unknown if to match its spelling (recall 3.1) the AI has tried every word construction available from the database, but all unsuccessfully. In this case it does not even know if some character string between spaces can be a word. So the acquisition task can be divided into two steps. First, identifying words as such. Next, making word constructions for them.

4.1 Identifying Words

For lack of CG tools covering the morphology and syntax of large textual data, the author has used the external tool TreeTagger[3] to identify words and tag them as POS. TreeTagger is a probabilistic (Hidden Markov Model) POS tagger using decision trees (Schmid, 1994). The decision algorithm features analyzing morphemes of the word in question and available POS tags for words preceding it, which demonstrates rather high precision with a limited amount of training data (Schmid, 1999). TreeTagger has been incorporated into the present AI and has been configured to output in the Penn Treebank format which attempts (Taylor et al., 2003) to utilize fewer, more universal tags and observes a standard list (Santorini, 1990) of thirty-six POS tags displayed in Table 3 (see APPENDIX). For instance "call" (as used in the Fig. 1 sentence) according to this list is tagged `VB` i.e. a verb, base form.

4.2 Making Word Constructions

After an unknown word (or, technically, a character string) is identified as a POS, a construction for it is

[3]Available at http://www.cis.uni-muenchen.de/schmid/tools/TreeTagger/ retrieved on May 3, 2019.

```
CONSTRUCTION => call_VB
___Text_Form
ID => VB
lemma => call
string => calls|called
to_match => ID
___Text_Meaning
ID => A
to_match => ID
wn_senses => 28
```

Figure 2: A word construction example: a minimal construction for the verb "call".

generated. By tradition (De Saussure, 2011) a construction describes a linguistic sign of a certain form (e.g., a string of characters that are a word) and the meaning corresponding to that form. Constructions automatically generated by the present AI have two major parts `Text_Form` and `Text_Meaning` (Fig. 2). To identify the form and meaning of a word (so the identifier can match one in other constructions to form networks), the two parts have pairs such as `ID => VB` ("verb, base form") and `ID => A` ("action"), see Fig. 2.

The notation in Fig. 2 is interpreted as follows: the value `call_VB` means this is a construction for the word "call" and its POS tag is `VB`; the `ID` values `VB` and `A` result from the TreeTagger output the way described later; the `lemma` pair is also made out of the TreeTagger output and shows the basic form of the word; `calls|called` (given as mere examples) enable the construction to match the former or the latter verb form in a text, if needed; `to_match => ID` is a "dummy" pair used to tell the AI to look for matches only in the `ID` pair of this word construction and of phrase or sentence constructions (when matches in other form or meaning features are not required for forming a network); the `wn_senses` pair points to word senses as described later.

As mentioned above, the identifiers `VB` and `A` (Fig. 2) result—i.e. are automatically made—from the TreeTagger output. The word "call" as it is used in the sentence (Fig. 1) is parsed by TreeTagger as `VB`. The meaning identifier `A` (Fig. 2) corresponds to the form identifier `VB` (as a verb denotes an action). Meaning tags that correspond to other POS are also generated. The form and meaning identifiers are used so they can match the same ones in, for instance, a phrase construction that may say that its prospective parts must match those identifiers; in this way the AI understands what words are parts of what phrases etc., recall 3.1. Table 3 (see APPENDIX) lists and describes form and meaning identifiers automatically inserted into word constructions.

The `wn_senses => 28` pair (Fig. 2) says that there are data for twenty-eight senses of the verb "call" in WordNet (Miller, 1998), a publicly available lexical database with look-up tools. WordNet has been incorporated into the present AI and the number of senses is automatically inserted into the word construction. For each of the senses WordNet lists synonyms and other related words; the total number of senses like `28` facilitates faster (parallel) processing of the data for each sense. The author uses these WordNet sense data so the same inference rules (such as that exemplified in 3.2) can be applied to text parts with similar meanings, however in-depth research into that kind of rule application is left for the future.

At present minimal constructions like one shown in Fig. 2 are automatically generated, additional feature-value need to be entered manually.

5 EVALUATION EXPERIMENTS

5.1 Setting

The purpose of the experiments is to answer the question (posed in 2) about the coverage increase. To further detail the question: is there any increase in the number of accurately formed sentence networks due to the described automatic generation of word constructions? To gloss the question for more clarity: how many sentence constructions (i.e. their `ID` values, etc.) are correctly matched by word and phrase ones; do the formed sentence networks correctly reflect the syntax and semantics? As the phrase and sentence constructions come from the limited database (introduced in 2) it is interesting how large a portion of the whole WS collection the database knowledge can cover—in the experiments the database is used to understand the whole WS collection, differently from Kiselev (2017) who applied that database to a part of the WS collection.

The whole WS collection of 150 items or 300 statement-question sets (as of the time the experiments were preformed) was utilized. That collection was obtained from a publicly available online source[4]. For more clarity the original collection format that uses slashes and square brackets was changed (by writing and executing an additional program) into that shown in Fig. 3, the item numbering corresponds to the original collection numbering.

The CG database mentioned above was originally designed to cover 7 items or 14 statement-question sets (Kiselev, 2017) from the WS collection. In the

[4]http://www.cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.html retrieved on June 20, 2019.

```
- 52 -
The fish ate the worm. It was tasty.
What was tasty?
The fish ate the worm. It was hungry.
What was hungry?
```

Figure 3: A WS as it is used in the experiments. There are two statement-question sets: the statement part is directly followed by the question. A WS is preceded by its number.

Table 1: Construction quantities in the database before the automatic generation of word constructions.

Word Constructions	79
Phrase Constructions	19
Sentence Constructions	13

experiments the database is applied to all of the 150 items or 300 sets; recall that the same constructions can be reused to understand text pieces with similar form and meaning. Table 1 lists the quantities for constructions of each type in the database. Before the experiments the database was formatted so the existing constructions could form networks with the new automatically generated ones. For instance, the new form and meaning notation for `ID` pairs (described in 4.2) was incorporated into constructions.

First, the AI applied the database without automatically generating word constructions. Next, it applied the same database again and while doing so automatically added constructions for unknown words.

5.2 Results

In Table 2 the row TTL gives the total number of statement-question sets in which at least one sentence network was formed, i.e. at least one sentence construction matched phrase and/or sentence ones; the row Accurate gives the part of the above TTL where the sentence network correctly—from the human prospective—reflects the syntax and semantics; the row Accurate Coverage % shows the share of the above Accurate count in the number of all the WS collection statement-question sets (i.e. in 300).

By comparing the entries in the row Accurate Coverage % a 28% increase can be observed.

5.3 Discussion

The above results are not meant to extensively cover the whole WS collection, they are rather a step towards that—the following steps being phrase and sentence knowledge acquisition as mentioned in 2.

An increase in accurate coverage of sentences without automatically generating phrase and sentence constructions can be considered encouraging and speaks in favor of CG scalability to a certain extent: phrase and sentence constructions have proven reusable on the WS.

Table 2: Numbers and percentages for sentence networks formed with and without the automatic generation of word constructions: **Proposed** and **Baseline** respectively.

	Baseline	**Proposed**
TTL	17	170
Accurate	17	101
Accurate Coverage %	6%	34%

Along with an encouragement there is room for improvement. Namely, by comparing the figures for TTL and Accurate under Proposed in Table 2 it is clear that 69 sentence networks were not parsed correctly. The reason is that some constructions may have been designed excessively versatile, i.e. allowing matches of unwanted POS, word forms etc. There is a need for a better look into using a larger number of non-versatile constructions as opposed to a smaller number of versatile ones.

The recursive generation of alternative parses is rather time-consuming (about 25 seconds for a statement-question set with an especially large number of parses) so the implementation of parallel processing for variant parses may be needed.

6 CONCLUSIONS AND FUTURE WORK

This paper has described a step towards automating the hard task of understanding the WS by means of deep NLU, featuring an original explainable AI that utilizes CG and automatically generates word knowledge. That knowledge generation augmented phrase and sentence knowledge already existing in a database and led to accurate understanding of the sentence structure in a larger portion of the WS text. However, a more complete unsupervised understanding necessitates automatic generation of not only word knowledge but also knowledge about the phrase and sentence form and meaning. Research into this kind of knowledge generation is on the long-term agenda. The short-term agenda includes research into the appropriate degree of the construction versatility and into parallel processing for alternative parses of text.

REFERENCES

Bailey, D., Harrison, A. J., Lierler, Y., Lifschitz, V., and Michael, J. (2015). The winograd schema challenge

and reasoning about correlation. In *2015 AAAI Spring Symposium Series*.

Brock, D. C. (2018). Learning from artificial intelligence's previous awakenings: The history of expert systems. *AI Magazine*, 39(3):3–15.

De Saussure, F. (2011). *Course in general linguistics*. Columbia University Press.

Emami, A., Trischler, A., Suleman, K., and Cheung, J. C. K. (2018). A generalized knowledge hunting framework for the winograd schema challenge. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 25–31.

Kiselev, D. (2017). A construction-grammar approach to computationally solving the winograd schema: Implementation and evaluation. In *2017 AAAI Spring Symposium Series*, Stanford University, Palo Alto, USA.

Liu, Q., Jiang, H., Evdokimov, A., Ling, Z.-H., Zhu, X., Wei, S., and Hu, Y. (2017). Cause-effect knowledge acquisition and neural association model for solving a set of winograd schema problems. In *The Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

Micelli, V., van Trijp, R., and De Beule, J. (2009). Framing fluid construction grammar. In *the 31th annual conference of the cognitive science society*, pages 3023–3027.

Miller, G. (1998). *WordNet: An electronic lexical database*. MIT press.

Mitkov, R. (2014). *Anaphora resolution*. Routledge.

Peng, H., Khashabi, D., and Roth, D. (2015). Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

Raghuram, V., Trott, S., Shen, K., Goldberg, E., and Oderberg, S. (2017). Semantically-driven coreference resolution with embodied construction grammar. In *2017 AAAI Spring Symposium Series*, Stanford University, Palo Alto, USA.

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

Santorini, B. (1990). *Part-of-speech tagging guidelines for the Penn Treebank Project*. University of Pennsylvania, School of Engineering and Applied Science.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees, intl. In *Conference on New Methods in Language Processing. Manchester, UK*.

Schmid, H. (1999). Improvements in part-of-speech tagging with an application to german. In *Natural language processing using very large corpora*, pages 13–25. Springer.

Sharma, A., Vo, N. H., Gaur, S., and Baral, C. (2015). An approach to solve winograd schema challenge using automatically extracted commonsense knowledge. In *2015 AAAI Spring Symposium Series*.

Steels, L. (2017). Basics of fluid construction grammar. *Constructions and frames*, 9(2):178–225.

Taylor, A., Marcus, M., and Santorini, B. (2003). The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer.

van Trijp, R. (2017). A computational construction grammar for english. In *2017 AAAI Spring Symposium Series*, Stanford University, Palo Alto, USA.

Winograd, T. (1980). What does it mean to understand language? *Cognitive science*, 4(3):209–241.

Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

APPENDIX

Table 3: The columns **No.**, **Form ID** and **POS Explanation** respectively give the item number, the part-of-speech (POS) identifier (ID) tag and the explanation as they are in the Penn Treebank Project (Santorini, 1990). The column **Meaning ID** lists IDs showing a meaning for each POS (e.g., the meaning "action" for a verb). The **Meaning ID**s: A = action, E = entity, P = property, U = currently unspecified. This simplistic meaning identification meets the current research needs but may be reconsidered in the future.

No.	Form ID	Meaning ID	POS Explanation
1.	CC	U	Coordinating conjunction
2.	CD	U	Cardinal number
3.	DT	U	Determiner
4.	EX	U	Existential there
5.	FW	U	Foreign word
6.	IN	U	Preposition or subordinating conjunction
7.	JJ	P	Adjective
8.	JJR	P	Adjective, comparative
9.	JJS	P	Adjective, superlative
10.	LS	U	List item marker
11.	MD	U	Modal
12.	NN	E	Noun, singular or mass
13.	NNS	E	Noun, plural
14.	NP	E	Proper noun, singular
15.	NPS	E	Proper noun, plural
16.	PDT	U	Predeterminer
17.	POS	U	Possessive ending
18.	PP	U	Personal pronoun
19.	PP$	U	Possessive pronoun
20.	RB	P	Adverb
21.	RBR	P	Adverb, comparative
22.	RBS	P	Adverb, superlative
23.	RP	U	Particle
24.	SYM	U	Symbol
25.	TO	U	to
26.	UH	U	Interjection
27.	VB	A	Verb, base form
28.	VBD	A	Verb, past tense
29.	VBG	U	Verb, gerund or present participle
30.	VBN	U	Verb, past participle
31.	VBP	A	Verb, non-3rd person singular present
32.	VBZ	A	Verb, 3rd person singular present
33.	WDT	U	Wh-determiner
34.	WP	U	Wh-pronoun
35.	WP$	U	Possessive wh-pronoun
36.	WRB	U	Wh-adverb