Using Ellipsis Detection and Word Similarity for Transformation of Spoken Language into Grammatically Valid Sentences

Using Ellipsis Detection and Word Similarity for Transformation of Spoken Language into Grammatically Valid Sentences

Manuel Giuliani
fortiss GmbH
Munich, Germany
giuliani@fortiss.org

Thomas Marschall
fortiss GmbH
Munich, Germany
marschat@in.tum.de

Amy Isard
University of Edinburgh
Edinburgh, UK
amyi@inf.ed.ac.uk

Abstract

When humans speak they often use grammatically incorrect sentences, which is a problem for grammar-based language processing methods, since they expect input that is valid for the grammar. We present two methods to transform spoken language into grammatically correct sentences. The first is an algorithm for automatic ellipsis detection, which finds ellipses in spoken sentences and searches in a combinatory categorial grammar for suitable words to fill the ellipses. The second method is an algorithm that computes the semantic similarity of two words using WordNet, which we use to find alternatives to words that are unknown to the grammar. In an evaluation, we show that the usage of these two methods leads to an increase of 38.64% more parseable sentences on a test set of spoken sentences that were collected during a human-robot interaction experiment.

1 Introduction

Computer systems that are designed to interact verbally with humans need to be able to recognise and understand human speech. In this paper we use as an example the robot bartender JAMES (Joint Action for Multimodal Embodied Social Systems),[1] shown in Figure 1. The robot is able to take drink orders from customers and to serve drinks. It is equipped with automatic speech recognition, to understand what the human is saying, and it has a grammar, to parse and process the spoken utterances.

The JAMES robot grammar was initially very restricted, and therefore during grammar development as well as during the user studies that we conducted (Foster et al., 2012; Giuliani et al., 2013; Keizer et al., 2013), we experienced situations in which the robot was not able to process the spoken input by humans, because they spoke sentences with grammatical structures that could not be parsed by the grammar, they used words that were not part of the grammar, or they left out words. We had for example cases where humans approached the robot and used a sentence with an ellipsis ("*I want Coke*", but the grammar expected a determiner in front of "*Coke*") or a sentence with a word that was unknown to the grammar ("*I need a water*", but "*need*" was not part of the grammar's word list). In these cases, the robot was unable to process and to respond to the spoken utterance by the human. Of course, these shortcomings can be overcome by extending the grammar, but even with a much more sophisticated grammar there will always be instances of unexpected language, and we believe that our approach can be very useful in extending the coverage of a grammar during testing or user studies.

Figure 1: The robot bartender of the JAMES project interacting with a customer.

Therefore, we present an approach to transform unparseable spoken language into sentences that a given grammar can parse. For ellipsis detection,

[1] http://www.james-project.eu

we present in Section 3.1 a novel algorithm that searches for ellipses in a sentence and computes candidate words to fill the ellipsis with words from a grammar. In Section 3.2, we show how we use WordNet (Miller, 1995) to find replacements for words that are not in the robot's grammar. In Section 4 we evaluate our approach with a test set of 211 spoken utterances that were recorded in a human-robot interaction (HRI) experiment, and the grammar for processing used in the same experiment.

2 Related Work

The work described in this paper draws on research and techniques from three main areas: the automatic detection of ellipses in sentences, calculation of semantic similarity between two words using WordNet, and spoken language processing. This section provides a summary of relevant work in these areas.

2.1 Ellipsis Detection

There is a wide range of research in ellipsis detection in written language, where different types of ellipses are widely defined, such as *gapping*, *stripping* or *verb phrase ellipsis* (Lappin, 1996). For example, an ellipsis occurs when a redundant word is left out of succeeding sentences, such as the words "*want to have*" in the sentence "*I want to have a water, and my friend a juice*", which are omitted in the second part of the sentence.

The detection of verb phrase ellipses (VPE) is a subfield of ellipsis detection that has received much attention. For VPE detection, researchers have used machine learning algorithms which were trained on grammar-parsed corpora, for example in the works of (Hardt, 1997), (Nielsen, 2004a; Nielsen, 2004b), and (Smith and Rauchas, 2006). Other approaches for ellipsis detection rely on symbolic processing of sentences, which is similar to our work. (Haddar and Hamadou, 1998) present a method for ellipsis detection in the Arabic language, which is based on an augmented transition network grammar. (Egg and Erk, 2001) present a general approach for ellipsis detection and resolution that uses a language for partial description of λ-terms called Constraint Language for Lambda Structures.

2.2 WordNet-based Semantic Similarity Calculation

WordNet is used in many varied natural language processing applications, such as word sense disambiguation, determining the structure of texts, text summarisation and annotation, information extraction and retrieval, automatic indexing, lexical selection, and the automatic correction of word errors in text. In our work, we use WordNet to find similar or synonym words. In previous work, researchers have proposed several methods to generally compute the semantic relatedness of two words using WordNet. (Budanitsky and Hirst, 2006) review methods to determine semantic relatedness. Newer examples for WordNet-based calculation of semantic similarity are the works by (Qin et al., 2009), (Cai et al., 2010), (Liu et al., 2012), and (Wagh and Kolhe, 2012).

2.3 Spoken Language Processing

Our work addresses the processing of spoken language, which differs from the processing of written language in that spoken language is more often elliptical and grammatically incorrect. Previous work in this area has attempted to address this issue at different levels of processing. (Issar and Ward, 1993) presented the CMU speech processing system that supports recognition for grammatically ill-formed sentences. (Lavie, 1996) presents GLR*, a grammar-based parser for spoken language, which ignores unparseable words and sentence parts and instead looks for the maximal subset of an input sentence that is covered by the grammar.

Other researchers in this area have designed grammar-based approaches for incremental spoken language processing: (Brick and Scheutz, 2007) present RISE, the robotic incremental semantic engine. RISE is able to process syntactic and semantic information incrementally and to integrate this information with perceptual and linguistic information. (Kruijff et al., 2007) present an approach for incremental processing of situated dialogue in human-robot interaction, which maintains parallel interpretations of the current dialogue that are pruned by making use of the context information. (Schlangen and Skantze, 2009) describe a "general, abstract model of incremental dialogue processing", where their goal is to provide principles for designing new systems for incremental speech processing.

3 Approach

Our goal in this paper is to transform spoken utterances which cannot be parsed by our grammar into grammar-valid sentences. During this process, we have to make sure that the changes to the input sentence do not change its meaning. In this section, we show how we implement ellipsis detection and semantic similarity computation in order to achieve this goal. We present our ellipsis detection algorithm in Section 3.1. Section 3.2 explains our implementation of WordNet-based word similarity computation.

3.1 Ellipsis Detection Algorithm

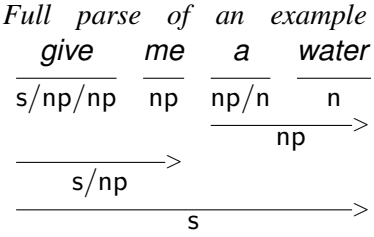
We use the OpenCCG parser (White, 2006), which is based on Combinatory Categorial Grammar (Kruijff and Baldridge, 2004; Steedman, 2000), to parse the output of our speech recognition system. We use the properties of CCGs to solve a problem that often occurs during parsing of spoken language. In our evaluation (Section 4) we use a test set (Section 4.1) of spoken sentences that was collected during one of our human-robot interaction studies (Foster et al., 2012) and the CCG (Section 4.2) that was used in the same study. In the test set, we found that speakers leave out words. For example, one speaker said *I want water* to order a drink. The grammar used in the experiment assumed that every noun is specified by an article; the grammar was only able to parse the sentence *I want a water*. Just to remind you, of course this particular example could have been solved by rewriting the grammar, but at the time of running the experiment it was not possible to us to change the grammar. Furthermore, we argue that there will always be examples of the above described situation where experiment participants use grammatical structures or words that cannot be processed by the used grammar. Thus, we present an algorithm that automatically finds ellipses in sentences and suggests words from the grammar that could be used to fill the ellipses.

To illustrate our approach, we will use the example sentence *give me a water*. Example (1) shows the words of the example sentence with their assigned categories from the used CCG, and Example (2) shows the parsed sentence. In the examples, we use the category symbols s for sentence, n for noun, and np for noun phrase. In Example (2) the symbol > denotes the used CCG forward application combinator.

(1) *CCG lexicon entries*

a. *give* := s / np / np
b. *me* := np
c. *a* := np / n
d. *water* := n

(2) *Full parse of an example sentence*



The algorithm consists of two parts: **(i)** search for ellipses in the sentence and selection of the most relevant ellipsis, and **(ii)** computation of the category for the word that will fill the chosen ellipsis.

(i) Ellipsis Search

In order to find the ellipsis in the sentence, our algorithm assumes that to the left and to the right of the ellipsis, the spoken utterance consists of sentence parts that the grammar can parse. In our example, these sentence parts would be *I want* to the left of the ellipsis and *water* to the right. In order to automatically find the sentence part to the right, we use the following algorithm, which we present in Listing 1 in a Java-style pseudo code: The algorithm uses the method tokenize() to split up the string that contains the utterance into an array of single words. It then iterates through the array and builds a new sentence of the words in the array, using the method buildString(). This sentence is then processed by the parser. If the parser finds a parse for the sentence, the algorithm returns the result. Otherwise it cuts off the first word of the sentence and repeats the procedure. This way, the algorithm searches for a parseable sentence part for the given utterance from the left to the right until it either finds the right-most parseable sentence part or there are no more words to parse. In order to find the left-most parseable sentence part, we implemented a method findParseReverse(), which parses sentence parts from right to left.

One has to consider that our method for ellipsis detection can falsely identify ellipses in certain sentence constellations. For example, if the word *like* in the sentence *I would like a water* is left out and given to our ellipsis detection algorithm, it would falsely find an ellipsis between *I* and *would*, and another ellipsis between *would*

Listing 1: Ellipsis detection algorithm.

```
Result findParse(String utterance) {
  words[] = tokenize(utterance);
  for (i = 0; i < words.length; i++) {
    String sentence = buildString(words[
        i], words.length);
    Result parse = parse(sentence);
    if (parse != null) {
      return parse;
    }
  }
  return null;
}
```

and *a*. The reason for the detection of the first ellipsis is that the categories for *I* and *would* cannot be combined together. *would* and *like* have to be parsed first to an auxiliary verb-verb construct. This construct can then be combined with the pronoun *I*. To overcome this problem, we first compute the category for each found ellipsis. Then we find a word for the ellipsis with the simplest category, which is either an atomic category or a functional category with fewer functors than the other found categories, add it to the original input sentence, and parse the output sentence. If the output sentence cannot be parsed, we repeat the step with the next found ellipsis.

(ii) Ellipsis Category Computation

After the algorithm has determined the ellipsis in an utterance, it computes the category of the word that will fill the ellipsis. The goal here is to find a category which the grammar needs to combine the sentence parts to the left and right of the ellipsis. For example, the left part of our example utterance *I want* has the category s/np and the right part *water* has the category n. Hence, the category for the missing word needs to be np/n, because it takes the category of the right sentence part as argument and produces the category np, which is the argument of the category of the left sentence part.

Figure 2 shows the processing sequence diagram of our algorithm for computing the category of an ellipsis. In the diagram, left and right stand for the categories of the sentence parts that are to the left and right of the ellipsis. The predicates symbolise functions: isEmpty(category) checks if a category is empty, atom(category) checks if a category is atomic, compl(category) checks if a category is complex and has a slash operator that faces toward the ellipsis. The predicate arg(category) returns the argument of a complex category. Rectangular boxes symbolise steps where the algorithm builds the result category for the missing word. The algorithm determines the category with the following rules:

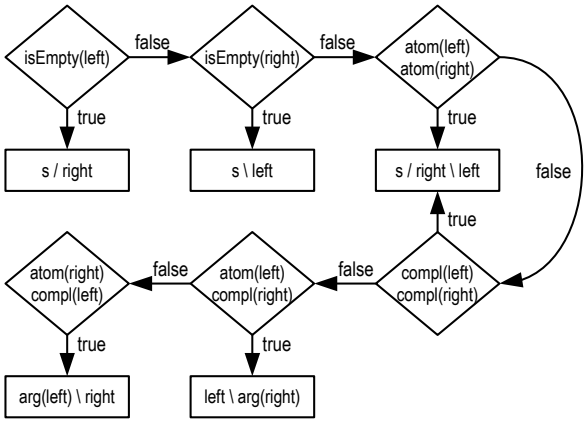


Figure 2: Processing sequence of the category computation algorithm.

- if the categories to the left or to the right of the ellipsis are empty, the ellipsis category is s/right or s\left, respectively,
- if the categories to the right and to the left of the ellipsis are atomic, the ellipsis category is s/right\left,
- if the categories to the right and to the left of the ellipsis are both complex and have a slash operator facing toward the ellipsis, the ellipsis category is s/right\left,
- if the category to the left of the ellipsis is atomic and to the right of the ellipsis is complex, the ellipsis category is left\arg(right),
- if the category to the right of the ellipsis is atomic and to the left of the ellipsis is complex, the ellipsis category is arg(left)\right.

After the computation of the ellipsis category, we use the OpenCCG grammar to select words to fill the ellipsis. This step is straightforward, because the grammar maintains a separate word list with corresponding categories. Here, we benefit from the usage of a categorial grammar, as the usage of a top-down grammar formalism would have meant a more complicated computation in this step.

3.2 WordNet-based Word Substitution

Spoken language is versatile and there are many ways to express one's intentions by using differ-

ent expressions. Thus, in grammar-based spoken language processing it often happens that sentences cannot be parsed because of words that are not in the grammar. To overcome this problem, we use WordNet (Miller, 1995) to find semantically equivalent replacements for unknown words. WordNet arranges words in sets of synonyms called synsets which are connected to other synsets by a variety of relations, which differ for each word category. The most relevant relations for our work are: for nouns and verbs *hyperonyms* (e.g., drink is a hyperonym of juice) and *hyponyms* (e.g., juice is a hyponym of drink), and for adjectives we use the *similar to* relation.

Our implementation of word substitution executes two steps if a word is unknown to the grammar: (1) look-up of synonyms for the unknown words. The unknown word can be substituted with a semantically similar word directly, if the synset of the unknown word contains a word, which is known to the grammar. (2) Computation of similar words in the WordNet hyperonym/hyponym hierarchy. If the synset of the unknown word does not contain a substitution, we compute if one of the hyperonyms of the unknown word has a hyponym which is known to the grammar. Here, one has to be careful not to move too far away from the meaning of the unknown word in the WordNet tree, in order not to change the meaning of the originally spoken sentence. Also, the computation of the similar word should not take too much time. Therefore, in our implementation, we only substitute an unknown word with one of its hyperonym/hyponym neighbours when the substitution candidate is a direct hyponym of the direct hyperonym of the unknown word.

4 Evaluation

The goal of this evaluation is to measure how many spoken sentences that our grammar cannot parse can be processed after the transformation of the sentences with our methods. In Section 4.1 we present the test set of spoken sentences that we used in the evaluation. In Section 4.2 we give details of the used grammar. As mentioned above, both, the test set as well as the grammar, were taken from the human-robot interaction study reported in (Foster et al., 2012). Section 4.3 summarises the details of the evaluation procedure. Finally, we present the evaluation results in Section 4.4 and discuss their meaning in Section 4.5.

4.1 Test Set

As test set for the evaluation, we used the spoken utterances from the participants of the human-robot interaction experiment reported in (Foster et al., 2012). In the experiment, 31 participants were asked to order a drink from the robot bartender shown in Figure 1. The experiment consisted of three parts: in the first part, participants had to order drinks on their own, in the second and third part, participants were accompanied by a confederate in order to have a multi-party interaction with the robot. The spoken utterances in the test set were annotated by hand from video recordings of the 93 drink order sequences. Please refer to (Foster et al., 2012) for a detailed description of the experiment.

Table 1 shows an overview of the test set. In total, it contains 211 unique utterances; the experiment participants spoke 531 sentences of which some sentences were said repeatedly. We divided the test set into the following speech acts (Searle, 1965): Ordering (“*I would like a juice please.*”), Question (“*What do you have?*”), Greeting (“*Hello there.*”), Polite expression (“*Thank you.*”), Confirmation (“*Yes.*”), Other (“*I am thirsty.*”).

4.2 Grammar

The grammar that we used in this evaluation was also used in the robot bartender experiment (Foster et al., 2012). This grammar is limited in its scope, because the domain of the experiment—the robot hands out drinks to customers—was limited as well. Overall, the lexicon of the grammar contains 92 words, which are divided into the following part of speech classes: 42 verbs, 11 nouns, 10 greetings, 6 pronouns, 5 prepositions, 4 adverbs, 4 determiners, 3 quantifiers, 3 confirmations, 2 relative pronouns, 1 conjunction, 1 polite expression.

4.3 Procedure

For the evaluation, we implemented a programme that takes our test set and automatically parses each sentence with four different settings, which are also presented in Table 1: **(1)** parsing with the grammar only, **(2)** application of ellipsis detection and word filling before parsing, **(3)** application of WordNet similarity substitution before parsing, **(4)** application of a combination of both methods before parsing. Please note that for alternative **(4)** the sequence in which the methods

Speech act	No. utt	(1) CCG	(2) Ell. det.	(3) WordNet	(4) Ell. + WordNet
Ordering	143	34	16	-	1
Question	19	1	-	-	-
Greeting	18	4	1	-	-
Polite expression	14	1	-	-	-
Confirmation	5	4	-	-	-
Other	12	-	-	-	-
Total	211	44	17	-	1

Table 1: Overview for test set and evaluation. Column **No. utt** contains the number of test utterances per speech act. Column **(1) CCG** shows the number of utterances that were directly parsed by the grammar. Columns **(2) Ell. det.**, **(3) WordNet**, and **(4) Ell. + WordNet** show how many utterances were additionally parsed using the ellipsis detection , WordNet substitution, and combination of both modules.

are applied to a given sentence can make a difference to the parsing result. In this evaluation, we used both possible sequences: first ellipsis detection followed by WordNet substitution method or vice versa.

4.4 Results

Table 1 shows the result of the evaluation procedure. The grammar parses 44 sentences of the 211 test set sentences correctly. By using the ellipsis detection algorithm, 17 additional sentences are parsed. The usage of the WordNet substitution algorithm yields no additionally parsed sentences. The combination of both methods (in this case, first ellipsis detection, then WordNet substitution) leads to the correct parse of one additional sentence.

4.5 Discussion

The evaluation results show that the application of our ellipsis detection algorithm leads to an increase of successfully parsed sentences of 38.64%. In the class of ordering sentences, which was the most relevant for the human-robot interaction experiment from which we used the evaluation test set, the number of successfully parsed sentences increases by 47.06%. Compared to this, the usage of WordNet substitution alone does not lead to an increase in parseable sentences. The one case in which the combination of ellipsis detection and WordNet substitution transformed an unparseable sentence into a grammatically valid sentence is interesting: here, the experiment participant said "*I need Coke.*" to order a drink from the robot. This sentence contained the word "*need*", which was not in the grammar. WordNet has the synonym "*want*" in the synset for the word "*need*". However, the sentence "*I want Coke.*" was also not parseable, because the grammar expected an article in front of every noun. The ellipsis detection algorithm was able to find the missing article in the sentence and filled it with an article "*a*" from the grammar, leading to a parseable sentence "*I want a Coke.*".

Although we see an increase in parsed sentences, 150 sentences of the test set were not transformed by our approach. Therefore, we made an analysis for the remaining utterances to find the main causes for this weak performance. We found that the following reasons cause problems for the grammar (with number of cases in brackets behind each reason):

- *Word missing in grammar (81).* The participant used a word that was not in the grammar. For example, users ordered drinks by saying "*One water, please.*" , but the grammar did not contain "*one*" as an article. This result shows that the WordNet similarity substitution has the potential to lead to a large increase in parseable sentences. However as mentioned above, there is a risk of changing the meaning of a sentence too much when allowing the replacement of words which are only vaguely similar to the unknown word.

- *Sentence structure (25).* Some participants said sentences that were either grammatically incorrect or had a sentence structure that was not encoded in the grammar. For example one participant tried to order a juice by saying "*Juice for me.*". Additionally, some participants asked questions ("*Do you have coke?*"). For the latter, please note that it was not part of the HRI experiment, from which we use

the test set, that the experiment participants should be allowed to ask questions to the robot.

- *Unnecessary fill words (22).* Some experiment participants used unnecessary fill words that did not add meaning to the sentence, for example one participant said "*Oh come on, I only need water*" to order a drink.

- *Sentence not related to domain (22).* Some participants said sentences that were contrary to the given experiment instructions. For example, some participants asked questions to the robot ("*How old are you?*") and to the experimenter ("*Do I need to focus on the camera?*"), or complained about the robot's performance ("*You are not quite responsive right now.*"). Clearly, these sentences were out of the scope of the grammar.

5 Conclusion

We presented two methods for transforming spoken language into grammatically correct sentences. The first of these two approaches is an ellipsis detection, which automatically detects ellipses in sentences and looks up words in a grammar that can fill the ellipsis. Our ellipsis detection algorithm is based on the properties of the combinatory categorial grammar, which assigns categories to each word in the grammar and thus enables the algorithm to find suitable fill words by calculating the category of the ellipsis. The second approach for sentence transformation was a WordNet-based word similarity computation and substitution. Here, we used the synsets of WordNet to substitute words that are unknown to a given grammar with synonyms for these words. In an evaluation we showed that the usage of ellipsis detection leads to an increase of successfully parsed sentences of up to 47.06% for some speech acts. The usage of the WordNet similarity substitution does not increase the number of parsed sentences, although our analysis of the test set shows that unknown words are the most common reason that sentences cannot be parsed.

Our approach was specifically implemented to help circumventing problems during development and usage of grammars for spoken language processing in human-robot interaction experiments, and the example grammar was a very restricted one. However, we believe that our method can also be helpful with more extensive grammars, and for developers of dialogue systems in other areas, such as telephone-based information systems or offline versions of automatic smartphone assistants like Apple's Siri.

In the future, we will refine our methodology. In particular, the WordNet similarity substitution is too rigid in its current form. Here, we plan to loosen some of the constraints that we applied to our algorithm. We will systematically test how far away from a word one can look for suitable substitutes in the WordNet hierarchy, without losing the meaning of a sentence. Furthermore, we plan to add a dialogue history to our approach, which will provide an additional source of information—besides the grammar—to the ellipsis detection method. Finally, we plan to work with stop word lists to filter unnecessary fill words from the input sentences, since these proved also to be a reason for sentences to be unparseable.

Acknowledgements

This research was supported by the European Commission through the project JAMES (FP7-270435-STREP).

References

Timothy Brick and Matthias Scheutz. 2007. Incremental natural language processing for hri. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 263–270. ACM New York, NY, USA.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Songmei Cai, Zhao Lu, and Junzhong Gu. 2010. An effective measure of semantic similarity. In *Advances in Wireless Networks and Information Systems*, pages 9–17. Springer.

Markus Egg and Katrin Erk. 2001. A compositional account of vp ellipsis. *Technology*, 3:5.

Mary Ellen Foster, Andre Gaschler, Manuel Giuliani, Amy Isard, Maria Pateraki, and Ronald P. A. Petrick. 2012. Two people walk into a bar: Dynamic multi-party social interaction with a robot agent. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI 2012)*, October.

Manuel Giuliani, Ronald P.A. Petrick, Mary Ellen Foster, Andre Gaschler, Amy Isard, Maria Pateraki, and Markos Sigalas. 2013. Comparing task-based and

socially intelligent behaviour in a robot bartender. In *Proceedings of the 15th International Conference on Multimodal Interfaces (ICMI 2013)*, Sydney, Australia, December.

Kais Haddar and A Ben Hamadou. 1998. An ellipsis detection method based on a clause parser for the arabic language. In *Proceedings of the International Florida Artificial Intelligence Research Society FLAIRS98*, pages 270–274.

Daniel Hardt. 1997. An empirical approach to vp ellipsis. *Computational Linguistics*, 23(4):525–541.

Sunil Issar and Wayne Ward. 1993. Cmu's robust spoken language understanding system. In *Proceedings of the Third European Conference on Speech Communication and Technology (Eurospeech 1993)*.

Simon Keizer, Mary Ellen Foster, Oliver Lemon, Andre Gaschler, and Manuel Giuliani. 2013. Training and evaluation of an MDP model for social multi-user human-robot interaction. In *Proceedings of the SIGDIAL 2013 Conference*, pages 223–232, Metz, France, August.

Geert-Jan M. Kruijff and Jason Baldridge. 2004. Generalizing dimensionality in combinatory categorial grammar. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, August.

Geert-Jan M. Kruijff, Pierre Lison, Trevor Benjamin, Henrik Jacobsson, and Nick Hawes. 2007. Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction. In Luis Seabra Lopes, Tony Belpaeme, and Stephen J. Cowley, editors, *Symposium on Language and Robots (LangRo 2007)*, Aveiro, Portugal, December.

Shalom Lappin. 1996. The interpretation of ellipsis. *The Handbook of Contemporary Semantic Theory*, 397:399.

Alon Lavie. 1996. *GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language*. Ph.D. thesis, Carnegie Mellon University.

Hongzhe Liu, Hong Bao, and De Xu. 2012. Concept vector for semantic similarity and relatedness based on wordnet structure. *Journal of Systems and Software*, 85(2):370–381.

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

Leif Arda Nielsen. 2004a. Verb phrase ellipsis detection using automatically parsed text. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 1093. Association for Computational Linguistics.

Leif Arda Nielsen. 2004b. Verb phrase ellipsis detection using machine learning techniques. *Amsterdam Studies in the Theory and History of Linguistic Science*, page 317.

Peng Qin, Zhao Lu, Yu Yan, and Fang Wu. 2009. A new measure of word semantic similarity based on wordnet hierarchy and dag theory. In *Proceedings of the International Conference on Web Information Systems and Mining2009 (WISM 2009)*, pages 181–185. IEEE.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*.

John R. Searle. 1965. What is a speech act? In Robert J. Stainton, editor, *Perspectives in the Philosophy of Language: A Concise Anthology*, pages 253–268.

Lindsey Smith and Sarah Rauchas. 2006. A machine-learning approach for the treatment of vp ellipsis. In *Proceedings of the Seventeenth Annual Symposium of the Pattern Recognition Association of South Africa*, November.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Kishor Wagh and Satish Kolhe. 2012. A new approach for measuring semantic similarity in ontology and its application in information retrieval. In *Multidisciplinary Trends in Artificial Intelligence*, pages 122–132. Springer.

Michael White. 2006. CCG chart realization from disjunctive inputs. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 12–19, Sydney, Australia, July. Association for Computational Linguistics.