# Actions Speak Louder than Words: Evaluating Parsers in the Context of Natural Language Understanding Systems for Human-Robot Interaction

**Sandra Kübler, Rachael Cantrell, Matthias Scheutz**
Indiana University
{skuebler,rcantrel,mscheutz}@indiana.edu

## Abstract

The standard ParsEval metrics alone are often not sufficient for evaluating parsers integrated in natural language understanding systems. We propose to augment intrinsic parser evaluations by extrinsic measures in the context of human-robot interaction using a corpus from a human cooperative search task. We compare a constituent with a dependency parser on both intrinsic and extrinsic measures and show that the conversion to semantics is feasible for different syntactic paradigms.

## 1 Introduction

Human-robot interactions (HRI) in *natural language* (Scheutz et al., 2007) pose many challenges for natural language understanding (NLU) systems, for humans expect robots to (1) generate quick responses to their request, which requires all processing to be done in real-time, (2) to rapidly integrate perceptions (e.g., to resolve referents (Brick and Scheutz, 2007)), and (3) to provide backchannel feedback indicating whether they understood an instruction, often before the end of an utterance. As a result, NLU systems on robots must operate *incrementally* to allow for the construction of meaning that can lead to robot action before an utterance is completed (e.g., a head-turn of the robot to check for an object referred to by the speaker). Hence, the question arises how one can best evaluate NLU components such as parsers for robotic NLU in the context of HRI.

In this paper, we argue that intrinsic parser evaluations, which evaluate parsers in isolation, are insufficient for determining their performance in HRI contexts where the ultimate goal of the NLU system is to generate the correct actions for the robot in a timely manner. For high performance of a parser with respect to intrisic measures does not imply that the parser will also work well with the other NLU components. A correct but overly complex parse passed to the semantic analysis unit, for example, may not result in the correct meaning interpretation and will thus fail to generate correct actions. Similarly, fragmented input from the speech recognizer may not lead to any parsable sequence of words, again likely resulting in incorrect robot behavior. Hence, we need an extrinsic evaluation to determine the utility and performance of a parser *in the context of other NLU components at the level of semantics and action execution.*

To this end, we introduce an evaluation architecture that can be used for extrinsic evaluations of NLU components and demonstrate its utility for parser evaluation using state-of-the-art parsers for each of the two main parsing paradigms: the Berkeley *constituent parser* (Petrov and Klein, 2007) and MaltParser (Nivre et al., 2007b), a *dependency parser*. The evaluation compares intrinsic and extrinsic measures on the *CReST corpus* (Eberhard et al., 2010), which is representative of a broad class of collaborative instruction-based tasks envisioned for future robots (e.g., in search and rescue missions). To our knowledge, no previous extrinsic parser evaluation used conversions to semantic/action representations, which can be performed for different parser types and are thus ideally suited for comparing parsing frameworks. Moreover, no previous work has presented a combined intrinsic-extrinsic evaluation where the extrinsic evaluation uses full-fledged semantic/action representations in an HRI context.

## 2 Previous Work

Evaluating different types of parsers is challenging for many reasons. For one, intrinsic evaluation measures are often specific to the type of parser. The ParsEval measures (precision and recall) are the standard for constituent parsers, attachment scores for dependency parsing. Yet, none of these measures is ideal: the ParsEval measures have been widely criticized because they favor flat annotation schemes and harshly punish attachment errors (Carroll et al., 1998). Additionally, there is no evaluation scheme that can compare the performance of constituent and dependency parsers, or parsers using different underlying grammars. Converting constituents into dependencies (Boyd and Meurers, 2008), evens out differences between underlying grammars. However, it is well known that the conversion into a different format is not straightforward. Clark and Curran (2007), who convert the CCGBank to DepBank, report an F-score of 68.7 for the conversion on gold data. Conversions into dependencies have been evaluated on the treebank side (Rehbein and van Genabith, 2007), but not on the parser side; yet, the latter is critical since parser errors result in unpredicted structures and thus conversion errors.

Intrinsic parsing quality has been shown to be insufficient for comparing parsers, and adding extrinsic measures to the evaluation can lead to inconclusive results, in comparing two dependency parsers (Mollá and Hutchinson, 2003), three constituent parsers (Preiss, 2002), and for a deep and a partial parser (Grover et al., 2005).

We propose to use intrinsic and extrinsic measures together to assess tradeoffs for parsers embedded in NLU systems (e.g., low-intrinsic/high-extrinsic quality is indicative of parsers that work well in challenging systems, while high-intrinsic/low-extrinsic quality is typical of high-performance parsers that are difficult to interface).

## 3 An Evaluation Framework for HRI

For evaluation, we propose the robotic `DIARC` architecture (Scheutz et al., 2007) which has been used successfully in many robotic applications. In addition to components for visual perception and action execution, `DIARC` consists of five NLU components. The first two components, a speech recognizer, and a disfluency filter which filters out common vocal distractors (“uh”, “um”, etc.) and common fillers (“well”, “so”, etc.) will not be used here. The third component optionally performs trigram-based part of speech (POS) tagging. The fourth component, the parser to be evaluated, which produces the constituent tree or dependency graph used by the fifth component, the $\lambda$ converter, to produce formal semantic representations. If the semantic representation indicates that a command needs to be executed, the command is passed on to an action interpreter (which then retrieves an existing action script indexed by the command or, if no such script is found, forwards the request to a task planner, which will plan a sequence of actions to achieve it (Schermerhorn et al., 2009)).

The semantic conversion process makes use of combinatorial categorial grammar (CCG) tags associated with lexical items, which are essentially part-of-speech tags enriched with information about the word’s arguments. Given a word and the appropriate CCG tag, the corresponding semantic representations are retrieved from a semantic lexicon. These representations are $\lambda$-expressions expressed in a fragment of first-order dynamic logic sufficiently rich to capture the language of (action) instructions from the corpus (c.f. e.g., (Goldblatt, 1992)). Expressions are repeatedly combined using $\beta$-reduction until all words are converted and (preferably) only one $\lambda$-free formula is left (Dzifcak et al., 2009).

For example, the sentence “do you see a blue box?” is translated as $\texttt{check-and-answer}(\exists x.see(\texttt{self}, x) \wedge box(x) \wedge blue(x))$. `check-and-answer` is an action that takes a formula as an argument, checks its truth (if possible), and causes the robot to reply with “yes” or “no” depending on the outcome of the check operation[1].

The conversion from dependency graphs to semantic representations is straightforward: When a dependent is attached to a head, the dependent is added to the CCG tag, resulting in a convenient format for semantic conversion. Then each node is looked up in the dictionary, and the definition is used to convert the node. For the example above, the parse graph indicates that “a” and “blue” are syntactic arguments of “box”, “you” and “a blue box” are arguments of “see”, and the clause “you see a blue box” is an argument of “do”. Based on the lexical definitions, the phrase “a blue box” is combined into the expression $(\exists x.box(x) \wedge blue(x))$. As argument of the verb “see”, it is then combined into the expression $(\exists x.see(\texttt{self}, x) \wedge box(x) \wedge blue(x))$, and ultimately $\texttt{check\&answer}(\exists x.see(\texttt{self}, x) \wedge box(x) \wedge blue(x))$.

The conversion for constituent trees is less straightforward since it is more difficult to automatically identify the head of a phrase, and to connect the arguments in the same way. We use a slightly different method: each node in the

[1] `self` is a deictic referent always denoting the robot.

tree is looked up in the dictionary for a suitable word/CCG tag combination given the words dominated by the node's daughters. The $\lambda$ conversions are then performed for each sentence after the parser finishes producing a parse tree.

## 4 Experimental Setup

For parser evaluations, we use an HRI scenario where processing speed is critical (often more important even than accuracy) as humans expect timely responses of the robot. Moreover, a parser's ability to produce fragments of a sentence (instead of failing completely) is highly desirable since the robot can ask clarification questions (if it knows where the parse failed) as opposed to offline processing tasks as humans are typically willing to help. This is different from a corpus, where no clarification question can be asked. *Correctness* here is determined by correct semantic interpretations that can be generated in the semantic analysis based on the (partial) parses. While these aspects are often of secondary importance in many NLU systems, they are essential to a robotic NLU architecture. Since we experiment with a new corpus that has not been used in parsing research yet, we also present an intrinsic evaluation to give a reference point to put the parsers' performance into perspective with regard to previous work.

More specifically, we investigate two points: (1) Given that spoken commands to robots are considerably shorter and less complex than newspaper sentences, is it possible to use existing resources, i.e., the Penn Treebank (Marcus et al., 1993), for training the parsers without a major decrease in accuracy? And (2), are constituent or dependency parsers better suited for the NLU architecture described above, in terms of accuracy and speed?

To answer these questions, we carried out two experiments: (1) The intrinsic evaluation. This is split into two parts: one that compares constituent and dependency parsers on our test data when both parsers were trained on the Penn Treebank; and one that compares the parsers trained on a small in-domain set. (2) The extrinsic evaluation, which compares the two parsers in the NLU architecture, is also based on in-domain training data.

**Intrinsic and extrinsic measures:** For the first experiment we use standard intrinsic parsing measures: for the constituent parser, we report labeled precision (LP), labeled recall (LR), and labeled F-score (LF); for the dependency parser the labeled attachment score (LAS). The second experiment uses the accuracy of the logical forms and the correct action interpretation and execution as a measure of quality. For this experiment, we also report the processing time, i.e., how much time the complete system requires for processing the test set from the text input to the output of logical forms.

**Data sets:** For the intrinsic evaluation, we used the Penn Treebank. For the constituent experiments, we used the treebank with grammatical functions since the semantic construction requires this information. The only exception is the experiment using the Berkeley parser with the Penn Treebank: Because of memory restrictions, we could not use grammatical functions. For the dependency parser, we used a dependency version of the Penn Treebank created by *pennconverter* (Johansson and Nugues, 2007).

For the in-domain experiments (intrinsic and extrinsic), we used CReST (Eberhard et al., 2010), a corpus of natural language dialogues obtained from recordings of humans performing a *cooperative*, *remote search task*. The multi-modal corpus contains the speech signals and transcriptions of the dialogues, which are additionally annotated for dialogue structure, disfluencies, POS, and syntax. The syntactic annotation covers both constituent annotation based on the Penn Treebank annotation scheme and dependencies based on the dependency version of the Penn Treebank. The corpus consists of 7 dialogues, with 1,977 sentences overall. The sentences are fairly short; average sentence length is 6.7 words. We extracted all commands (such as "walk into the next room"), which our robot can handle, and used those 122 sentences as our test set. We performed a 7-fold cross validation, in which one fold consists of all test sentences (i.e. commands) from one of the 7 dialogues. All the other folds combined with the declarative sentences from all dialogues served as training data. The number of commands per dialogue varies so the evaluation was performed on the set of all test sentences rather than averaged over the 7 folds.

**Parsers:** We use both state-of-the-art constituent and dependency parsers: As constituent parser, we chose the Berkeley parser (Petrov and Klein, 2007), a parser that learns a refined PCFG grammar based on latent variables. We used grammars based on 6 split-merge cycles.

|  | Berkeley parser |  |  |  | MaltParser |  |
|---|---|---|---|---|---|---|
| training data | POS acc. | LP | LR | LF | POS acc. | LAS |
| Penn | 86.9 | 47.2 | 44.8 | 46.0 | 88.1 | 40.6 |
| CReST | 67.8 | 56.7 | 48.9 | 52.5 | 92.8 | 70.5 |

Table 1: The results of the intrinsic evaluation.

For the dependency parser, we used MaltParser (Nivre et al., 2007b), a pseudo-projective dependency parser, which has reached state-of-the-art results for all languages in the CONLL 2007 shared task (Nivre et al., 2007a). We decided to use version 1.1 of MaltParser, which allows the use of memory-based learning (MBL) in the implementation of TiMBL[2]. MBL has been shown to work well with small training sets (cf., (Banko and Brill, 2001)). MaltParser was used with the Nivre algorithm and the feature set that proved optimal for English (Nivre et al., 2007b). TiMBL parameters were optimized for each experiment in a non-exhaustive search. When trained on the Penn Treebank, the parser performed best using MVDM, 5 nearest neighbors, no feature weighting, and Inverse Distance class weighting. For the experiments on the dialogue corpus, the default settings proved optimal. Since MaltParser requires POS-tagged input, we used the Markov model tagger TnT (Brants, 1999) to tag the test sentences for dependency parsing; the Berkeley parser performs POS tagging in the parsing process.

For the experiment based on the complete NLU architecture, we used an incremental reimplementation of the Nivre algorithm called Mink (Cantrell, 2009) as dependency parser. Mink uses the WEKA implementation of the C4.5 decision tree classifier (Hall et al., 2009) as guide. The confidence threshold for pruning is 0.25, and the minimum number of instances per leaf is 2.

## 5 Results

The results of the **intrinsic parser evaluation** are shown in Table 1. The POS tagging results for TnT (for MaltParser) are unexpected: the small in-domain training set resulted in an increase of accuracy of 4.7 percent points. The result for the POS tagging accuracy of the Berkeley parser trained on CReST is artificially low because the parser did not parse 9 sentences, which resulted in missing POS tags for those sentences. All of the POS tagging results are lower than the TnT accuracy of 96.7%, reported for the Penn Treebank (Brants, 1999). This is due to either out-of-domain data or the small training set for the training with CReST.

When the parsers were trained on the Penn Treebank, the very low results for both parsers (46.0 F-score, 40.6 LAS) show clearly that pre-existing resources cannot be used for training. The low results are due to the fact that the test set consists almost exclusively of commands, a sentence type that, to our knowledge, does not occur in the Penn Treebank. A comparison between ParsEval measures and LAS is difficult. We refrained from converting the constituent parse to dependencies for evaluation because it is unclear how reliable the conversion for parser output is.

The results for the Berkeley parser trained on the dialogue data from CReST are better than the results trained on the Penn Treebank. However, even with training on in-domain data, the F-score of 52.5 is still considerably lower than state-of-the-art results for in-domain parsing of the Penn Treebank. This is partly due to our inclusion of grammatical functions in the parsing process as well as in the evaluation. Thus, the parsing task is more difficult than in other experiments. Another possible reason for the low performance is the size of the training set. We must assume that the Berkeley parser requires a larger training set to reach good results. This is corroborated by the fact that this parser did not find any parse for 9 sentences. The dependency parser performs equally badly when trained on the Penn Treebank (40.6 LAS). However, when it is trained on in-domain data, it reaches an LAS of 70.5, which corroborates the assumption that TiMBL performs well with small data sets.

An error analysis of the parser output based on the CReST training shows that one frequent type of error results from differing lexical preferences between the Penn Treebank and the CReST domain. The word “left”, for example, is predominantly used as a verb in the Penn Treebank, but as an adverb or noun in the dialogue corpus, which results in frequent POS tagging errors and subse-

[2]`http://ilk.uvt.nl/timbl/`

```
( (S (VP (VB hold) (PRT (RP on)) (S (VP (VB let) (S (NP (PRP me)) (VP (VB pick) (PRT (RP
up)) (NP (DT those) (JJ green) (NNS boxes)))))))) )
```

Figure 1: Constituent parse for "hold on let me pick up those green boxes".

quent parsing errors.

For the **extrinsic evaluation** in the context of the NLU system, we report *exact match* accuracy for the logical forms. Since the semantic conversion fails on unexpected parser output, the quantitative semantic evaluation is based only on syntactically-correct sentences, although partially-correct parses are instructive examples, and thus are included in the discussion. More parses were *almost* correct than perfectly so: 27% were perfectly correct for the constituent parser, and 30% for the dependency parser.

Of these, 90% of dependency graphs were correctly semantically combined. while just 64% of constituent trees were correctly combined. Mink was also faster: Averaged over a range of sentence lengths and complexities, the NLU system using Mink was roughly twice as fast as the one with the Berkeley parser. Averaged over 5 runs of 100 sentences each, Mink required approx. 180 ms per sentence, the Berkeley parser approx. 270 ms.

The most egregious problem area involves a typical phenomenon of spontaneous speech that an utterance does not necessarily correspond to a sentence in the syntactic sense: Many utterances contain multiple, independent phrases or clauses, e.g., "hold on let me pick up those green boxes", as a single utterance. The ideal translation for this utterance is: $wait(listener); get(speaker, \{x|green(x) \wedge box(x)\})$ where ";" is the sequencing operator.

The constituent parse for the utterance is shown in Figure 1. This parse is partially correct, but the two commands are not treated as a conjunction of clauses; instead, the second command is treated as subordinate to the first one, This analysis results in the argument structure shown in Table 2, where each phrase takes its phrasal constituents as arguments. The semantic definitions and CCG tags are shown in Table 3. Some definitions do not have the same number of arguments as the CCG tags, in particular the verb "pick" with its raised subject, which will be applied by the semantics of the verb "let". The correspondence between the constituent parse and semantics output is shown in Table 4. The dependency parse is shown in Figure 2. The two commands are correctly analyzed as independent clauses.

| Phr.:Head | Arguments |
|---|---|
| VP:hold | (PRT=on,S) |
| VP:let | (S) |
| S | (NP=me,VP) |
| VP:pick | (PRT=up,NP) |
| NP | (DT=those,JJ=green,NNS=boxes) |

Table 2: The argument structure based on the constituent parse.

| Token | Arg. Str. | Semantics |
|---|---|---|
| hold | S/RP | $\lambda x.wait(x)$ |
| on | RP | $on$ |
| let | S/NP/S | $\lambda x.\lambda X.X(x)$ |
| me | NP | $speaker$ |
| pick | S/RP/NP | $\lambda x.\lambda y.\lambda z.pick(x, y, z)$ |
| up | RP | $up$ |
| those | NP/NP | $\lambda X.\{x \| X(x)\}$ |
| green | NP/NP | $\lambda X.\lambda x.green(x) \wedge X(x)$ |
| boxes | NP | $box$ |

Table 3: Semantics for the example sentence.

| Head | Dependents |
|---|---|
| HOLD | on |
| LET | me, pick |
| PICK | up, boxes |
| BOXES | those, green |

Table 5: Syntactic head/dependent relationships.

The parse results in the syntactic head and dependent relationships and the semantic head and dependent relationships for the words in the utterance, constructed from the definitions in Table 5. In the semantic analysis, "pick" is similar to the syntactic analysis in that it takes a noun phrase and a particle as its arguments. This results in the following combination: $\lambda x.\lambda y.\lambda z.pick(up, z, y)$ (up) (those green boxes)[3]. The first application applies "up" to $x$, resulting in the analysis: $\lambda y.\lambda z.pick(up, z, y)$ (those green boxes) which in turn is converted into: $\lambda z.pick(up, z, those\_green\_boxes)$.

[3]Here, "those green boxes" is a human-convenient shorthand for its full semantic definition.

| | Constituency | Semantic |
|---|---|---|
| 1 | $NP_1$:boxes (DT=those,JJ=green) | $\{x \mid green(x) \wedge boxes(x)\}$ |
| 2 | $VP_1$:pick (PRT=up,$NP_1$) | $\lambda z.pick(up, z, \{x \mid green(x) \wedge box(x)\})$ |
| 3 | $S_1$($NP_2$=speaker,$VP_1$) | $pick(up, speaker, \{x \mid green(x) \wedge box(x)\})$ |
| 4 | $VP_2$:let ($S_1$) | $pick(up, speaker, \{x \mid green(x) \wedge box(x)\})$ |
| 5 | $S_2$($VP_2$) | $pick(up, speaker, \{x \mid green(x) \wedge box(x)\})$ |
| 6 | $VP_3$:hold (PRT=on,$S_2$) | $wait(pick(up, speaker, \{x \mid green(x) \wedge box(x)\})) \Leftarrow$ **error** |
| 7 | $S_4$($VP_3$) | |

Table 4: Correspondence between the constituent parse and the semantics output.

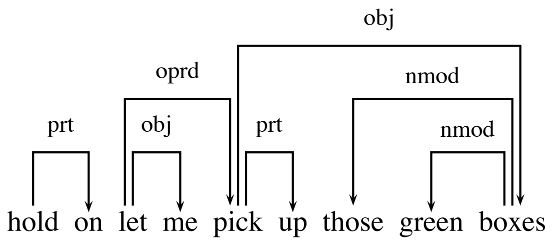

Figure 2: The dependency analysis.

Here we find a systematic difference between the syntactic analysis and the intended semantic one: While syntactically, the adjective “green” is dependent on the head “boxes”, it is the opposite in the semantic analysis. The definition of “boxes” indicates that it is a predicate that takes as an argument an abstract entity “x”, representing the real-world item that has the property of being a box. This predicate, $box(x)$, is itself then applied to the predicate “green”, which has the definition $\lambda X.\lambda x.green(x) \wedge X(x)$. The variable $X$ represents the concept that will be applied. This application produces $\lambda x.green(x) \wedge box(x))$. Thus a conversion rule reverses dependencies within noun phrases.

## 6 Discussion

The results show that a considerable number of sentences could be parsed but not converted correctly to logical form because of the way certain information is represented in the parses. Additionally, a small difference in the parsers’ behavior, namely MaltParser’s ability to provide partial parses, resulted in a large difference in the usability of the parsers’ output – partial parses are not only better than parse failures, but may even be the expected outcome in an HRI settings, since they can be successfully translated to logical form.

While the same parser performed better under both intrinsic and extrinsic evaluation, this may not necessarily always be the case (see section 2). It is possible that one parser provides imperfect parses when evaluated intrinsically but the information is presented in a form that can be used by higher applications. This occurred in our experiment in the case of the dependency parser, whose partial parses could be converted in completely correct semantic representations. I.e., while the parse may not be completely correct with regard to the gold standard, it may still provide enough information to use for the higher component so that no information loss ensues.

One advantage of our extrinsic evaluation is that the conversion to semantics can be performed for a wide range of different syntactic annotations. While previous evaluations stayed within one parsing framework (e.g., dependency parsing), our evaluation included a constituent and a dependency parser (this evaluation can be extended to “deeper” parsers such as HPSG parsers). Additionally, the conversion to semantics involves a wide range of syntactic phenomena, thus providing a high granularity compared to extrinsic evaluations in information retrieval, where only specific sentence parts (e.g., noun phrases) are targeted.

## 7 Conclusions

We introduced a novel, semantics-based method for comparing the performance of different parsers in an HRI setting and evaluated our method on a test corpus collected in a human coordination task.

The experiments emphasize the importance of performing an extrinsic evaluation of parsers in typical application domains. While extrinsic evaluations may depend on the application domain, it is important to show that parsers cannot be used off-the-shelf based on intrinsic evaluations. To estimate the variance of parsers, it is important to establish a scenario of different applications in which parsers can be tested. An NLU component in an HRI setting is an obvious candidate since the conversion to semantics is possible for any syntactic paradigm, and the HRI setting requires evaluation metrics, such as the time behavior or the incrementality of the parser, which are typically not considered.

## Acknowledgment

This work was in part supported by ONR grants #N00014-10-1-0140 and #N00014-07-1-1049.

## References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL-EACL'01*, pages 26–33, Toulouse, France.

Adriane Boyd and Detmar Meurers. 2008. Revisiting the impact of different annotation schemes on PCFG parsing: A grammatical dependency evaluation. In *Proceedings of the ACL Workshop on Parsing German*, Columbus, OH.

Thorsten Brants. 1999. *Tagging and Parsing with Cascaded Markov Models*. DFKI, Universität des Saarlandes.

Timothy Brick and Matthias Scheutz. 2007. Incremental natural language processing for HRI. In *Proceedings of the Second ACM IEEE International Conference on Human-Robot Interaction*, pages 263–270, Washington D.C.

Rachael Cantrell. 2009. Mink: An incremental data-driven dependency parser with integrated conversion to semantics. In *Student Workshop at RANLP*, Borovets, Bulgaria.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of LREC 1998*, pages 447–454, Granada, Spain.

Stephen Clark and James Curran. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of ACL 2007*, Prague, Czech Republic.

Juraj Dzifcak, Matthias Scheutz, and Chitta Baral. 2009. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'09)*, Kobe, Japan.

Kathleen Eberhard, Hannele Nicholson, Sandra Kübler, Susan Gunderson, and Matthias Scheutz. 2010. The Indiana "Cooperative Remote Search Task" (CReST) Corpus. In *Proceedings of LREC-2010*, Valetta, Malta.

Robert Goldblatt. 1992. Parallel action: Concurrent dynamic logic with independent modalities. *Studia Logica*, 51(3/4):551–578.

Claire Grover, Mirella Lapata, and Alex Lascarides. 2005. A comparison of parsing technologies for the biomedical domain. *Natural Language Engineering*, 11:27–65.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Diego Mollá and Ben Hutchinson. 2003. Intrinsic versus extrinsic evaluations of parsing systems. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, Budapest, Hungary.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL 2007*, Prague, Czech Republic.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL'07*, Rochester, NY.

Judita Preiss. 2002. Choosing a parser for anaphora resolution. In *Proceedings of DAARC*, Lisbon, Portugal.

Ines Rehbein and Josef van Genabith. 2007. Treebank annotation schemes and parser evaluation for German. In *Proceedings of EMNLP-CoNLL 2007*, pages 630–639, Prague, Czech Republic.

Paul Schermerhorn, J Benton, Matthias Scheutz, Kartik Talamadupula, and Rao Kambhampati. 2009. Finding and exploiting goal opportunities in real-time during plan execution. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis.

Matthias Scheutz, Paul Schermerhorn, James Kramer, and David Anderson. 2007. First steps toward natural human-like HRI. *Autonomous Robots*, 22(4):411–423.