Backbone Extraction and Pruning for Speeding Up a Deep Parser for Dialogue Systems

Myroslava O. Dzikovska
Human Communication Research Centre
University of Edinburgh
Edinburgh, EH8 9LW, United Kingdom,
mdzikovs@inf.ed.ac.uk

Carolyn P. Rosé
Carnegie Mellon University
Language Technologies Institute,
Pittsburgh, PA 15213,
cprose@cs.cmu.edu

Abstract

In this paper we discuss issues related to speeding up parsing with wide-coverage unification grammars. We demonstrate that state-of-the-art optimisation techniques based on backbone parsing before unification do not provide a general solution, because they depend on specific properties of the grammar formalism that do not hold for all unification based grammars. As an alternative, we describe an optimisation technique that combines ambiguity packing at the constituent structure level with pruning based on local features.

1 Introduction

In this paper we investigate the problem of scaling up a deep unification-based parser developed specifically for the purpose of robust interpretation in dialogue systems by improving its speed and coverage for longer utterances. While typical sentences in dialogue contexts are shorter than in expository text domains, longer utterances are important in discussion oriented domains. For example, in educational applications of dialogue it is important to elicit deep explanation from students and then offer focused feedback based on the details of what students say.

The choice of instructional dialogue as a target application influenced the choice of parser we needed to use for interpretation in a dialogue system. Several deep, wide-coverage parsers are currently available (Copestake and Flickinger, 2000; Rosé, 2000; Baldridge, 2002; Maxwell and Kaplan, 1994), but many of these have not been designed with issues related to interpretation in a dialogue context in mind.

The TRIPS grammar (Dzikovska et al., 2005) is a wide-coverage unification grammar that has been used very successfully in several task-oriented dialogue systems. It supports interpretation of fragments and lexical semantic features (see Section 2 for a more detailed discussion), and provides additional robustness through "robust" rules that cover common grammar mistakes found in dialogue such as missing articles or incorrect agreement. These enhancements help parsing dialogue (both spoken and typed), but they significantly increase grammar ambiguity, a common concern in building grammars for robust parsing (Schneider and McCoy, 1998). It is specifically these robustness-efficiency trade-offs that we address in this paper.

Much work has been done related to enhancing the efficiency of deep interpretation systems (Copestake and Flickinger, 2000; Swift et al., 2004; Maxwell and Kaplan, 1994), which forms the foundation that we build on in this work. For example, techniques for speeding up unification in HPSG lead to dramatic improvements in efficiency (Kiefer et al., 1999). Likewise ambiguity packing and CFG backbone parsing (Maxwell and Kaplan, 1994; van Noord, 1997) are known to increase parsing efficiency. However, as we show in this paper, these techniques depend on specific grammar properties that do not hold for all grammars. This claim is consistent with observations of Carroll (1994) that parsing software optimisation techniques tend to be limited in their applicability to the individual grammars they were developed for. While we used TRIPS as our example unification-based grammar, this investigation is important not only for this project, but in the general context of speeding up a wide-coverage unification grammar which incorporates fragment rules and lexical semantics, which may not be immediately provided by other available systems.

In the remainder of the paper, we begin with a brief description of the TRIPS parser and grammar, and motivate the choice of LCFLEX parsing algorithm to provide a fast parsing foundation. We then discuss the backbone extraction and pruning techniques that we used, and evaluate them in comparison with the original parsing algorithm. We conclude with discussion of some implications for implementing grammars that build deep syntactic and semantic representations.

2 Motivation

The work reported in this paper was done as part of the process of developing a dialogue system that incorporates deep natural language understanding. We needed a grammar that provides lexical semantic interpretation, supports parsing fragmentary utterance in dialogue, and could be used to start development without large quantities of corpus data. TRIPS fulfilled our requirements better than similar alternatives, such as LINGO ERG (Copestake and Flickinger, 2000) or XLE (Maxwell and Kaplan, 1994).

TRIPS produces logical forms which include semantic classes and roles in a domain-independent frame-based formalism derived from FrameNet and VerbNet (Dzikovska et al., 2004; Kipper et al., 2000). Lexical semantic features are known to be helpful in both deep (Tetreault, 2005) and shallow interpretation tasks (Narayanan and Harabagiu, 2004). Apart from TRIPS, these have not been integrated into existing deep grammars. While both LINGO-ERG and XLE include semantic features related to scoping, in our applications the availability of semantic classes and semantic role assignments was more important to interpretation, and these features are not currently available from those parsers. Finally, TRIPS provides a domain-independent parse selection model, as well as rules for interpreting discourse fragments (as was also done in HPSG (Schlangen and Lascarides, 2003), a feature actively used in interpretation.

While TRIPS provides the capabilities we need, its parse times for long sentences (above 15 words long) are intolerably long. We considered two possible techniques for speeding up parsing: speeding up unification using the techniques similar to the LINGO system (Copestake and Flickinger, 2000), or using backbone extraction (Maxwell and Kaplan, 1994; Rosé and Lavie, 2001; Briscoe and Carroll, 1994). TRIPS already uses a fast unification algorithm similar to quasi-destructive unification, avoiding copying during unification.[1] However, the TRIPS grammar retains the notion of phrase structure, and thus it was more natural to chose to use backbone extraction with ambiguity packing to speed up the parsing.

As a foundation for our optimisation work, we started with the freely available LCFLEX parser (Rosé and Lavie, 2001). LCFLEX is an all-paths parser that uses left-corner prediction and ambiguity packing to make all-paths parsing tractable, and which was shown to be efficient for long sentences with somewhat less complex unification augmented context-free grammars. We show that all-paths parsing with LCFLEX is not tractable for the ambiguity level in the TRIPS grammar, but that by introducing a pruning method that uses ambiguity packing to guide pruning decisions, we can achieve significant improvements in both speed and coverage compared to the original TRIPS parser.

3 The TRIPS and LCFLEX algorithms

3.1 The TRIPS parser

The TRIPS parser we use as a baseline is a bottom-up chart parser with lexical entries and rules represented as attribute-value structures. To achieve parsing efficiency, TRIPS uses a best-first beam search algorithm based on the scores from a parse selection model (Dzikovska et al., 2005; Elsner et al., 2005). The constituents on the parser's agenda are grouped into buckets based on their scores. At each step, the bucket with the highest scoring constituents is selected to build/extend chart edges. The parsing stops once N requested analyses are found. This guarantees that the parser returns the N-best list of analyses according to the parse selection model used, unless the parser reaches the chart size limit.

[1]Other enhancements used by LINGO depend on disallowing disjunctive features, and relying instead on the type system. The TRIPS grammar is untyped and uses disjunctive features, and converting it to a typed system would require as yet undetermined amount of additional work.

In addition to best-first parsing, the TRIPS parser uses a chart size limit, to prevent the parser from running too long on unparseable utterances, similar to (Frank et al., 2003). TRIPS is much slower processing utterances not covered in the grammar, because it continues its search until it reaches the chart limit. Thus, a lower chart limit improves parsing efficiency. However, we show in our evaluation that the chart limit necessary to obtain good performance in most cases is too low to find parses for utterances with 15 or more words, even if they are covered by the grammar.

The integration of lexical semantics in the TRIPS lexicon has a major impact on parsing in TRIPS. Each word in the TRIPS lexicon is associated with a semantic type from a domain-independent ontology. This enables word sense disambiguation and semantic role labelling for the logical form produced by the grammar. Multiple word senses result in additional ambiguity on top of syntactic ambiguity, but it is controlled in part with the use of weak selectional restrictions, similar to the restrictions employed by the VerbNet lexicon (Kipper et al., 2000). Checking semantic restrictions is an integral part of TRIPS parsing, and removing them significantly decreases speed and increases ambiguity of the TRIPS parser (Dzikovska, 2004). We show that it also has an impact on parsing with a CFG backbone in Section 4.1.

3.2 LCFLEX

The LCFLEX parser (Rosé and Lavie, 2001) is an all-paths robust left corner chart parser designed to incorporate various robustness techniques such as word skipping, flexible unification, and constituent insertion. Its left corner chart parsing algorithm is similar to that described by Briscoe and Carroll (1994). The system supports grammatical specification in a unification framework that consists of context-free grammar rules augmented with feature bundles associated with the non-terminals of the rules. LCFLEX can be used in two parsing modes: either context-free parsing can be done first, followed by applying the unification rules, or unification can be done interleaved with context-free parsing. The context free backbone allows for efficient left corner predictions using a pre-compiled left corner prediction table, such as that described in (van Noord, 1997). To enhance its efficiency, it incorporates a provably optimal ambiguity packing algorithm (Lavie and Rosé, 2004).

These efficiency techniques make feasible all-path parsing with the LCFLEX CARMEL grammar (Rosé, 2000). However, CARMEL was engineered with fast all-paths parsing in mind, resulting in certain compromises in terms of coverage. For example, it has only very limited coverage for noun-noun compounding, or headless noun phrases, which are a major source of ambiguity with the TRIPS grammar.

4 Combining LCFLEX and TRIPS

4.1 Adding CFG Backbone

A simplified TRIPS grammar rule for verb phrases and a sample verb entry are shown in Figure 1. The features for building semantic representations are omitted for brevity. Each constituent has an assigned category that corresponds to its phrasal type, and a set of (complex-valued) features.

The backbone extraction algorithm is reasonably straightforward, with CFG non-terminals corresponding directly to TRIPS constituent categories. To each CFG rule we attach a corresponding TRIPS unification rule. After parsing is complete, the parses found are scored and ordered with the parse selection model, and therefore parsing accuracy in all-paths mode is the same or better than TRIPS accuracy for the same model.

For constituents with subcategorized arguments (verbs, nouns, adverbial prepositions), our backbone generation algorithm takes the subcategorization frame into account. For example, the TRIPS VP rule will split into 27 CFG rules corresponding to different subcategorization frames: VP → V_intr, VP → V_NP NP, VP → V_NP_CP NP CP, *etc.* For each lexical entry, its appropriate CFG category is determined based on the subcategorization frame from TRIPS lexical representation. This improves parsing efficiency using the prediction algorithms in TFLEX operating on the CFG backbone. The version of the TRIPS grammar used in testing contained 379 grammar rules with 21 parts of speech (terminal symbols) and 31 constituent types (non-terminal symbols), which were expanded into 1121 CFG rules with 85 terminals and 36 non-terminals during backbone extraction.

We found, however, that the previously used tech-

```
(a) ((VP (SUBJ ?!subj) (CLASS ?lf))
      -vp1-role .99
      (V (LF ?lf) (SUBJ ?!subj) (DOBJ ?dobj)
                  (IOBJ ?iobj) (COMP3 ?comp3))
      ?iobj ?dobj ?comp3)
(b) ((V (agr 3s) (LF LF::Filling)
      (SUBJ (NP (agr 3s)))
      (DOBJ (NP (case obj))) (IOBJ -) (COMP3 -)))
```

Figure 1: (a) A simplified VP rule from the TRIPS grammar; (b) a simplified verb entry for a transitive verb. Question marks denote variables.

nique of context-free parsing first followed by full re-unification was not suitable for parsing with the TRIPS grammar. The CFG structure extracted from the TRIPS grammar contains 43 loops resulting from lexical coercion rules or elliptical constructions. A small number of loops from lexical coercion were both obvious and easy to avoid, because they are in the form $N \rightarrow N$. However, there were longer loops, for example, $NP \rightarrow SPEC$ for sentences like "John's car" and $SPEC \rightarrow NP$ for headless noun phrases in sentences like "I want three". LCFLEX uses a re-unification algorithm that associates a set of unification rules with each CFG production, which are reapplied at a later stage. To be able to apply a unification rule corresponding to $N \rightarrow N$ production, it has to be explicitly present in the chart, leading to an infinite number of N constituents produced. Applying the extra CFG rules expanding the loops during re-unification would complicate the algorithm significantly. Instead, we implemented loop detection during CFG parsing.

The feature structures prevent loops in unification, and we considered including certain grammatical features into backbone extraction as done in (Briscoe and Carroll, 1994). However, in the TRIPS grammar the feature values responsible for breaking loops belonged to multi-valued features (6 valued in the worst case), with values which may depend on other multiple-valued features in daughter constituents. Thus adding the extra features resulted in major backbone size increases because of category splitting. This can be remedied with additional pre-compilation (Kiefer and Krieger, 2004), however, this requires that all lexical entries be known in advance. One nice feature of the TRIPS lexicon is that it includes a mechanism for dynamically adding lexical entries for unknown words from wide-coverage lexicons such as VerbNet (Kipper et al., 2000), which would be impractical to use in pre-compilation.

Therefore, to use CFG parsing before unification in our system, we implemented a loop detector that checked the CFG structure to disallow loops. However, the next problem that we encountered is massive ambiguity in the CFG structure. Even a very short phrase such as "a train" had over 700 possible CFG analyses, and took 910 msec to parse compared to 10 msec with interleaved unification. CFG ambiguity is so high because noun phrase fragments are allowed as top-level categories, and lexical ambiguity is compounded with semantic ambiguity and robust rules normally disallowed by features during unification. Thus, in our combined algorithm we had to use unification interleaved with parsing to filter out the CFG constituents.

4.2 Ambiguity Packing

For building semantic representations in parallel with parsing, ambiguity packing presents a set of known problems (Oepen and Carroll, 2000). One possible solution is to exclude semantic features during an initial unification stage, use ambiguity packing, and re-unify with semantic features in a post-processing stage. In our case, we found this strategy difficult to implement, since selectional restrictions are used to limit the ambiguity created by multiple word senses during syntactic parsing. Therefore, we chose to do ambiguity packing on the CFG structure only, keeping the multiple feature structures associated with each packed CFG constituent.

To begin to evaluate the contribution of ambiguity packing on efficiency, we ran a test on the first 39 utterances in a hold out set not used in the formal evaluation below. Sentences ranged from 1 to 17 words in length, 16 of which had 6 or more words. On this set, the average parse time without ambiguity packing was 10 seconds per utterance, and 30 seconds per utterance on utterances with 6 or more words. With ambiguity packing turned on, the average parse time decreased to 5 seconds per utterance, and 13.5 seconds per utterance on the utterances with more than 6 words. While this evaluation showed that ambiguity packing improves parsing efficiency, we determined that further enhancements were necessary.

4.3 Pruning

We added a pruning technique based on the scoring model discussed above and ambiguity packing to enhance system performance. As an illustration, consider an example from a corpus used in our evaluation where the TRIPS grammar generates a large number of analyses, "we have a heart attack victim at marketplace mall". The phrase "a heart attack victim" has at least two interpretations,"a [N1 heart [N1 attack [N1 victim]]]" and "a [N1 [N1 heart [N1 attack]] [N1 victim]]". The prepositional phrase "at marketplace mall" can attach either to the noun phrase or to the verb. Overall, this results in 4 basic interpretations, with additional ambiguity resulting from different possible senses of "have".

The best-first parsing algorithm in TRIPS uses parse selection scores to suppress less likely interpretations. In our example, the TRIPS parser will chose the higher-scoring one of the two interpretations for "a heart attack victim", and use it first. For this NP the features associated with both interpretations are identical with respect to further processing, thus TRIPS will never come back to the other interpretation, effectively pruning it. "At" also has 2 possible interpretations due to word sense ambiguity: LF::TIME-LOC and LF::SPATIAL-LOC. The former has a slightly higher preference, and TRIPS will try it first. But then it will be unable to find an interpretation for "at Marketplace Mall", and backtrack to LF::SPATIAL-LOC to find a correct parse.

Without chart size limits the parser is guaranteed to find a parse eventually through backtracking. However, this algorithm does not work quite as well with chart size limits. If there are many similarly-scored constituents in the chart for different parts of the utterance, the best-first algorithm expands them first, and the the chart size limit tends to interfere before TRIPS can backtrack to an appropriate lower-scoring analysis.

Ambiguity packing offers an opportunity to make pruning more strategic by focusing specifically on competing interpretations for the same utterance span. The simplest pruning idea would be for every ambiguity packed constituent to eliminate the interpretations with low TRIPS scores. However, we need to make sure that we don't prune constituents that are required higher up in the tree to make a parse. Consider our example again.

The constituent for "at" will be ambiguity packed with its two meanings. But if we prune LF::SPATIAL-LOC at that point, the parse for "at Marketplace Mall" will fail later. Formally, the competing interpretations for "at" have *n*on-local features, namely, the subcategorized complement (time versus location) is different for those interpretations, and is checked higher up in the parse. But for "a heart attack victim" the ambiguity-packed interpretations differ only in local features. All features associated with this NP checked higher up come from the head noun "victim" and are identical in all interpretations. Therefore we can eliminate the low scoring interpretations with little risk of discarding those essential for finding a complete parse. Thus, for any constituent where ambiguity-packed non-head daughters differ only in local features, we prune the interpretations coming from them to a specified prune beam width based on their TRIPS scores.

This pruning heuristic based on local features can be generalised to different unification grammars. For example, in HPSG pruning would be safe at all points where a head is combined with ambiguity-packed non-head constituents, due to the locality principle. In the TRIPS grammar, if a trips rule uses subcategorization features, the same locality principle holds. This heuristic has perfect precision though not complete recall, but, as our evaluation shows, it is sufficient to significantly improve performance in comparison with the TRIPS parser.

5 Evaluation

The purpose of our evaluation is to explore the extent to which we can achieve a better balance between parse time and coverage using backbone parsing with pruning compared to the original best-first algorithm. For our comparison we used an excerpt from the Monroe corpus that has been used in previous TRIPS research on parsing speed and accuracy (Swift et al., 2004) consisting of dialogues s2, s4, s16 and s17. Dialogue s2 was a hold out set used for pilot testing and setting parameters. The other three dialogues were set aside for testing. Altogether, the test set contained 1042 utterances, ranging from 1 to

45 words in length (mean 5.38 words/utt, st. dev. 5.7 words/utt). Using our hold-out set, we determined that a beam width of three was an optimal setting. Thus, we compared TFLEX using a beam width of 3 to three different versions of TRIPS that varied only in terms of the maximum chart size, giving us a version that is significantly faster than TFLEX overall, one that has parse times that are statistically indistinguishable from TFLEX, and one that is significantly slower. We show that while lower chart sizes in TRIPS yield speed ups in parse time, they come with a cost in terms of coverage.

5.1 Evaluation Methodology

Because our goal is to explore the parse time versus coverage trade-offs of two different parsing architectures, the two evaluation measures that we report are average parse time per sentence and probability of finding at least one parse, the latter being a measure estimating the effect of parse algorithm on parsing coverage.

Since the scoring model is the same in TRIPS and TFLEX, then as long as TFLEX can find at least one parse (which happened in all but 1 instances on our held-out set), the set returned will include the one produced by TRIPS. We spot-checked the TFLEX utterances in the test set for which TRIPS could not find a parse to verify that the parses produced were reasonable. The parses produced by TFLEX on these sentences were typically acceptable, with errors mainly stemming from attachment disambiguation problems.

5.2 Results

We first compared parsers in terms of probability of producing at least one parse (see Figure 2). Since the distribution of sentence lengths in the test corpus was heavily skewed toward shorter sentences, we grouped sentences into equivalence classes based on a range of sentence lengths with a 5-word increment, with all of the sentences over 20 words aggregated in the same class. Given a large number of short sentences, there was no significant difference overall in likelihood to find a parse. However, on sentences greater than 10 words long, TFLEX is significantly more likely to produce a parse than any of the TRIPS parsers (evaluated using a binary logistic regression, $N = 334$, $G = 16.8$, $DF = 1$, $p < .001$). Furthermore, for sentences greater than 20 words long, no form of TRIPS parser ever returned a complete parse.

Parser	<= 20 words	>= 6 words
TFLEX	6.2 (20.2)	29.1 (96.3)
TRIPS-1500	2.3 (5.4)	6.9 (8.2)
TRIPS-5000	7.7 (30.2)	28.1 (56.4)
TRIPS-10000	22.7 (134.4)	107.6 (407.4)

Table 1: The average parse times for TRIPS and TFLEX on utterances 6 words or more.

Next we compared the parsers in terms of average parse time on the whole data set across equivalence classes of sentences, assigned based on Aggregated Sentence Length (see Figure 2 and Table 1). An ANOVA with Parser and Aggregated Sentence Length as independent variables and Parse Time as the dependent variable showed a significant effect of Parser on Parse Time ($F(3, 4164) = 270.03$, $p < .001$). Using a Bonferroni post-hoc analysis, we determined that TFLEX is significantly faster than TRIPS-10000 ($p < .001$), statistically indistinguishable in terms of parse time from TRIPS-5000, and significantly slower than TRIPS-1500 ($p < .001$). Since none of the TRIPS parsers ever returned a parse for sentences greater than 20 words long, we recomputed this analysis excluding the latter. We still find a significant effect of Parser on Parse Time ($F(3, 4068) = 18.6$, $p < .001$). However, a post-hoc analysis reveals that parse times for TFLEX, TRIPS-1500, and TRIPS-5000 are statistically indistinguishable for this subset, whereas TFLEX is significantly faster than TRIPS-10000 ($p < .001$). See Table 1 for for parse times of all four parsers. Since TFLEX and TRIPS both spent 95% of their computational effort on sentences with 6 or more words, we also include results for this subset of the corpus.

Thus, TFLEX presents a superior balance of coverage and efficiency especially for long sentences (10 words or more) since for these sentences it is significantly more likely to find a parse than any version of TRIPS, even a version where the chart size is expanded to an extent that it becomes significantly slower (i.e., TRIPS-10000). And while TRIPS-1500 is consistently faster than the other parsers, it is not significantly faster than TFLEX on sentences 20

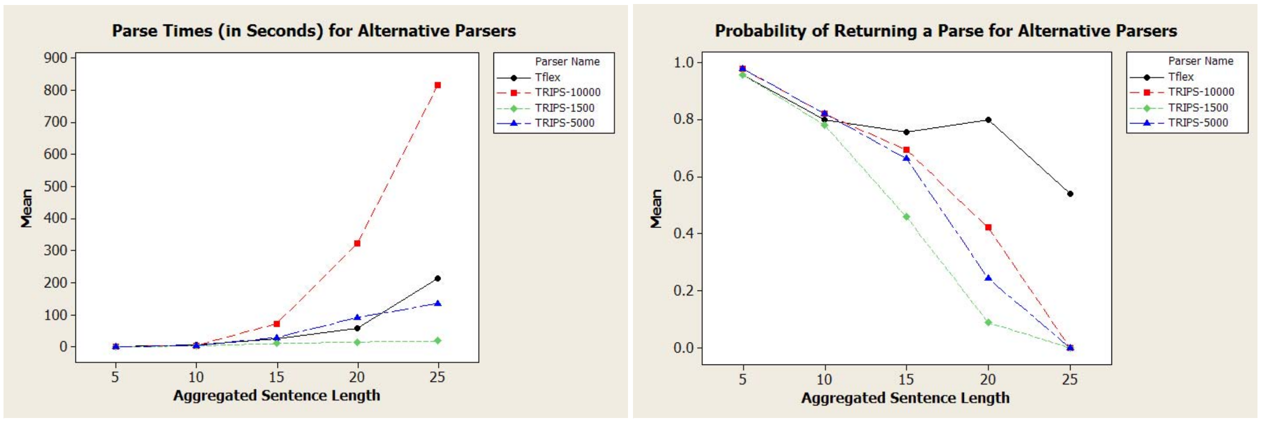

Figure 2: Parse times and probability of getting a parse depending on (aggregated) sentence lengths. 5 denotes sentences with 5 or fewer words, 25 sentences with more than 20 words.

words long or less, which is the subset of sentences for which it is able to find a parse.

5.3 Discussion and Future Work

The most obvious lesson learned in this experience is that the speed up techniques developed for specific grammars and unification formalisms do not transfer easily to other unification grammars. The features that make TRIPS interesting – the inclusion of lexical semantics, and the rules for parsing fragments – also make it less amenable to using existing efficiency techniques.

Grammars with an explicit CFG backbone normally restrict the grammar writer from writing grammar loops, a restriction not imposed by general unification grammars. As we showed, there can be a substantial number of loops in a CFG due to the need to cover various elliptical constructions, which makes CFG parsing not interleaved with unification less attractive in cases where we want to avoid expensive CFG precompilation. Moreover, as we found with the TRIPS grammar, in the context of robust parsing with lexical semantics the ambiguity in a CFG backbone grows large enough to make CFG parsing followed by unification inefficient. We described an alternative technique that uses pruning based on a parse selection model.

Another option for speeding up parsing that we have not discussed in detail is using a typed grammar without disjunction and speeding up unification as done in HPSG grammars (Kiefer et al., 1999). In order to do this, we must first address the issue of integrating the type of lexical semantics that we require with HPSG's type system. Adding lexical semantics while retaining the speed benefits obtained through this type system would require that the semantic type ontology be expressed in the same formalism as the syntactic types. We plan to further explore this option in our future work.

Though longer sentences were relatively rare in our test set, using the system in an educational domain (our ultimate goal) means that the longer sentences are particularly important, because they often correspond to significant instructional events, specifically answers to deep questions such as “why” and “how” questions. Our evaluation has been designed to show system performance with utterances of different length, which would roughly correspond to the performance in interpreting short and long student answers. Since delays in responding can de-motivate the student and decrease the quality of the dialogue, improving handling of long utterances can have an important effect on the system performance. Evaluating this in practice is a possible direction for future work.

6 Conclusions

We described a combination of efficient parsing techniques to improve parsing speed and coverage with the TRIPS deep parsing grammar. We showed that context-free parsing was inefficient on a backbone extracted from an existing unification grammar, and demonstrated how to make all-path parsing more tractable by a new pruning algorithm based on ambiguity packing and local features, generalisable to other unification grammars. We demonstrated that our pruning algorithm provides better efficiency-coverage balance than the best-first parsing with chart limits utilised by the TRIPS parser, and discussed the design implications for other robust parsing grammars.

Acknowledgements

We thank Mary Swift and James Allen for their help with the TRIPS code and useful comments. This material is based on work supported by grants from the Office of Naval Research under numbers N000140510048 and N000140510043.

References

J. Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

T. Briscoe and J. Carroll. 1994. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

J. Carroll. 1994. Relating complexity to practical performance in parsing with wide-coverage unification grammars. In *Proceedings of ACL-2004*.

A. Copestake and D. Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC-2000*, Athens, Greece.

M. O. Dzikovska, M. D. Swift, and J. F. Allen. 2004. Building a computational lexicon and ontology with framenet. In *LREC workshop on Building Lexical Resources from Semantically Annotated Corpora*.

M. Dzikovska, M. Swift, J. Allen, and W. de Beaumont. 2005. Generic parsing for multi-domain semantic interpretation. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT-05)*.

M. O. Dzikovska. 2004. *A Practical Semantic Representation For Natural Language Parsing*. Ph.D. thesis, University of Rochester.

M. Elsner, M. Swift, J. Allen, and D. Gildea. 2005. Online statistics for a unification-based dialogue parser. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT-05)*.

A. Frank, B. Kiefer, B. Crysmann, M. Becker, and U. Schafer. 2003. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of ACL 2003*.

B. Kiefer and H.-U. Krieger. 2004. A context-free approximation of head-driven phrase structure grammar. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology*. Kluwer.

B. Kiefer, H. Krieger, J. Carroll, and R. Malouf. 1999. Bag of useful techniques for efficient and robust parsing. In *Proceedings of ACL 1999*.

K. Kipper, H. T. Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the 7th Conference on Artificial Intelligence and of the 12th Conference on Innovative Applications of Artificial Intelligence*.

A. Lavie and C. P. Rosé. 2004. Optimal ambiguity packing in context-free parsers with interleaved unification. In H. Bunt, J. Carroll, and G. Satta, editors, *Current Issues in Parsing Technologies*. Kluwer Academic Press.

J. T. Maxwell and R. M. Kaplan. 1994. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

S. Narayanan and S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

S. Oepen and J. Carroll. 2000. Ambiguity packing in constraint-based parsing - practical results. In *Proceedings of NAACL'00*.

C. P. Rosé and A. Lavie. 2001. Balancing robustness and efficiency in unification-augmented context-free parsers for large practical applications. In J. Junqua and G. Van Noord, editors, *Robustness in Language and Speech Technology*. Kluwer Academic Press.

C. Rosé. 2000. A framework for robust semantic interpretation. In *Proceedings 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

D. Schlangen and A. Lascarides. 2003. The interpretation of non-sentential utterances in dialogue. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Japan, May.

D. Schneider and K. F. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of COLING-ACL'98*.

M. Swift, J. Allen, and D. Gildea. 2004. Skeletons in the parser: Using a shallow parser to improve deep parsing. In *Proceedings of COLING-04*.

J. Tetreault. 2005. *Empirical Evaluations of Pronoun Resolution*. Ph.D. thesis, University of Rochester.

G. van Noord. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3):425–456.