LTH: Semantic Structure Extraction using Nonprojective Dependency Trees

Richard Johansson and **Pierre Nugues**
Department of Computer Science, Lund University, Sweden
{richard, pierre}@cs.lth.se

Abstract

We describe our contribution to the SemEval task on Frame-Semantic Structure Extraction. Unlike most previous systems described in literature, ours is based on dependency syntax. We also describe a fully automatic method to add words to the FrameNet lexical database, which gives an improvement in the recall of frame detection.

1 Introduction

The existence of links between grammatical relations and various forms of semantic interpretation has long been observed; grammatical relations play a crucial role in theories of *linking*, i.e. the realization of the semantic arguments of predicates as syntactic units (Manning, 1994; Mel'čuk, 1988). Grammatical relations may be covered by many definitions but it is probably easier to use them as an extension of dependency grammars, where relations take the form of arc labels. In addition, some linguistic phenomena such as *wh*-movement and discontinuous structures are conveniently described using dependency syntax by allowing *nonprojective* dependency arcs. It has also been claimed that dependency syntax is easier to understand and to teach to people without a linguistic background.

Despite these advantages, dependency syntax has relatively rarely been used in semantic structure extraction, with a few exceptions. Ahn et al. (2004) used a post-processing step to convert constituent trees into labeled dependency trees that were then used as input to a semantic role labeler. Pradhan et al. (2005) used a rule-based dependency parser, but the results were significantly worse than when using a constituent parser.

This paper describes a system for frame-semantic structure extraction that is based on a dependency parser. The next section presents the dependency grammar that we rely on. We then give the details on the frame detection and disambiguation, the frame element (FE) identification and classification, and dictionary extension, after which the results and conclusions are given.

2 Dependency Parsing with the Penn Treebank

The last few years have seen an increasing interest in dependency parsing (Buchholz and Marsi, 2006) with significant improvements of the state of the art, and dependency treebanks are now available for a wide range of languages. The parsing algorithms are comparatively easy to implement and efficient: some of the algorithms parse sentences in linear time (Yamada and Matsumoto, 2003; Nivre et al., 2006).

In the semantic structure extraction system, we used the Stanford part-of-speech tagger (Toutanova et al., 2003) to tag the training and test sentences and MaltParser, a statistical dependency parser (Nivre et al., 2006), to parse them.

We trained the parser on the Penn Treebank (Marcus et al., 1993). The dependency trees used to train the parser were created from the constituent trees using a conversion program (Johansson and Nugues, 2007)[1]. The converter handles most of the secondary edges in the Treebank and encodes those edges as (generally) nonprojective dependency arcs. Such information is available in the Penn Treebank in the form of empty categories and secondary edges, it is however not available in the output of traditional constituent parsers, although there have been some attempts to apply a post-processing step to predict it, see Ahn et al. (2004), *inter alia*.

Figures 1 and 2 show a constituent tree from the Treebank and its corresponding dependency tree. Note that the secondary edge from the *wh*-trace to *Why* is converted into a nonprojective `PRP` link.

3 Semantic Structure Extraction

This section describes how the dependency trees are used to create the semantic structure. The system

[1]Available at `http://nlp.cs.lth.se/pennconverter`

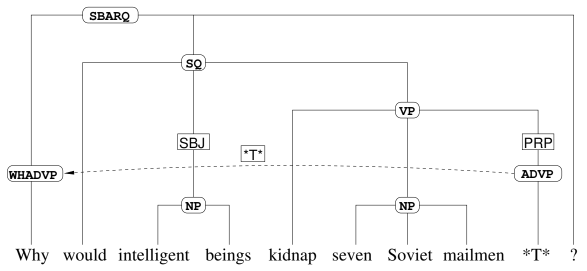
SBARQ
SQ
VP
SBJ
T
PRP
WHADVP
ADVP
NP
NP
Why would intelligent beings kidnap seven Soviet mailmen *T* ?

Figure 1: A constituent tree from the Penn Treebank.

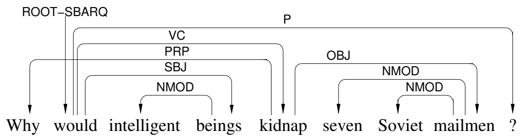

Figure 2: Converted dependency tree.

is divided into two main components: frame detection and disambiguation, and frame element detection and classification.

3.1 Frame Detection and Disambiguation

3.1.1 Filtering Rules

Since many potential target words appear in senses that should not be tagged with a frame, we use a filtering component as a first step in the frame detection. We also removed some words (especially prepositions) that caused significant performance degradation because of lack of training data. With the increasing availability of tagged running text, we expect that we will be able to replace the filtering rules with a classifier in the future.

- *have* was retained only if it had an object,
- *be* only if it was preceded by *there*,
- *will* was removed in its modal sense,
- *of course* and *in particular* were removed,
- the prepositions *above*, *against*, *at*, *below*, *beside*, *by*, *in*, *on*, *over*, and *under* were removed unless their head was marked as locative,
- *after* and *before* were removed unless their head was marked as temporal,
- *into*, *to*, and *through* were removed unless their head was marked as direction,
- *as*, *for*, *so*, and *with* were always removed,
- since the only sense of *of* was PARTITIVE, we removed it unless it was preceded by *only*, *member*, *one*, *most*, *many*, *some*, *few*, *part*, *majority*, *minority*, *proportion*, *half*, *third*, *quarter*, *all*, or *none*, or if it was followed by *all*, *group*, *them*, or *us*.

We also removed all targets that had been tagged as support verbs for some other target.

3.1.2 Sense Disambiguation

For the target words left after the filtering, we used a classifier to assign a frame, following Erk (2005). We trained a disambiguating SVM classifier on all ambiguous words listed in FrameNet. Its accuracy was 84% on the ambiguous words, compared to a first-sense baseline score of 74%.

The classifier used the following features: target lemma, target word, subcategorization frame (for verb targets only), the set of dependencies of the target, the set of words of the child nodes, and the parent word of the target.

The subcategorization frame feature was formed by concatenating the dependency labels of the children, excluding subject, parentheticals, punctuation and coordinations. For instance, for *kidnap* in Figure 2, the feature is `PRP+OBJ`.

3.1.3 Extending the Lexical Database

Coverage is one of the main weaknesses of the current FrameNet lexical database – it lists only 10,197 lexical units, compared to 207,016 word–sense pairs in WordNet 3.0 (Fellbaum, 1998). We tried to remedy this problem by training classifiers to find words that are related to the words in a frame.

We designed a feature representation for each lemma in WordNet, which uses a sequence of identifiers for each synset in its hypernym tree. All senses of the lemma were used, and the features were weighted with respect to the relative frequency of the sense. Using this feature representation, we trained an SVM classifier for each frame that tells whether a lemma belongs to that frame or not.

The FrameNet dictionary could thus be extended by 18,372 lexical units. If we assume a Zipf distribution and that the lexical units already in FrameNet are the most common ones, this would increase the coverage by up to 9%. In the test set, the new lexical units account for 53 out of the 808 target words our system detected (6.5%). We roughly estimated the precision to 70% by manually inspecting 100 randomly selected words in the extended dictionary.

This strategy is most successful when the frame is equivalent to one or a few synsets (and their subtrees). For instance, for the frame MEDICAL_CONDITION, we can add the complete subtree of the synset *pathological state*, resulting in 641 new lemmas referring to all sorts of diseases. On the other hand, the strategy also works well for motion verbs (which often exhibit complex patterns of polysemy): 137 lemmas could be added to the SELF_MOTION frame. Examples of frames with frequent errors are LEADERSHIP, which includes many insects (probably because the most frequent sense of *queen* in SemCor is the queen bee), and FOOD, which included many chemical substances as well as inedible plants and animals.

3.2 Frame Element Extraction

Following convention, we divided the FE extraction into two subtasks: argument identification and argument classification. We did not try to assign multiple labels to arguments. Figure 3 shows an overview. In addition to detecing the FEs, the argument identification classifier detects the dependency nodes that should be tagged on the layers other than the frame element layer: SUPP, COP, NULL, EXIST, and ASP. The ANT and REL labels could be inserted using simple rules. Similarly to Xue and Palmer (2004),

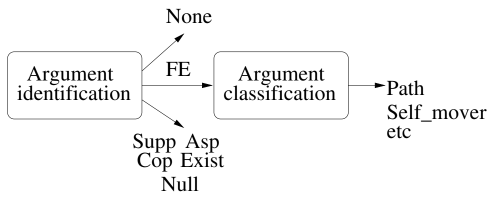

Figure 3: FE extraction steps.

we could filter away many nodes before the argument identification step by assuming that the arguments for a given predicate correspond to a subset of the dependents of the target or of its transitive heads.

Both classifiers were implemented using SVMs and use the following features: target lemma, voice (for verb targets only), subcategorization frame (for verb targets only), the set of dependencies of the target, part of speech of the target node, path through the dependency tree from the target to the node, position (before, after, or on), word and part of speech for the head, word and part of speech for leftmost and rightmost descendent.

In the path feature, we removed steps through verb chains and coordination. For instance, in the sentece *I have seen and heard it*, the path from *heard* to *I* is only SBJ↓ and to *it* OBJ↓.

3.3 Named Entity Recognition

In addition to the frame-semantic information, the SemEval task also scores named entities. We used YamCha (Kudo and Matsumoto, 2003) to detect named entities, and we trained it on the SemEval full-text training sets. Apart from the word and part of speech, we used suffixes up to length 5 as features. We think that results could be improved further by using an external NE tagger.

4 Results

The system was evaluated on three texts. Table 1 shows the results for frame detection averaged over the test texts. In the Setting colums, the first shows whether Exact or Partial frame matching was used by the evaluation script, and the second whether Labels or Dependencies were used. Table 2 compares the results of the system using the extended dictionary with one using the orignal FrameNet dictionary, using the Partial matching and Labels scoring. The extended dictionary introduces some noise and thus lowers the precision slightly, but the effects on the recall are positive. Table 3 shows the aver-

Table 1: Results for frame detection.

Setting		Recall	Precision	$F1$
E	L	0.528	0.688	0.597
P	L	0.581	0.758	0.657
E	D	0.549	0.715	0.621
P	D	0.601	0.784	0.681

Table 2: Comparison of dictionaries.

Dictionary	Recall	Precision	$F1$
Original	0.550	0.767	0.634
Extended	0.581	0.758	0.657

aged precision, recall, and $F1$ measures for different evaluation parameters. The third column shows whether named entities were used (Y) or not (N). Interestingly, the scores are higher for the semantic dependency graphs than for flat labels, while the two other teams generally had higher scores for flat labels. We believe that the reason for this is that we used a dependency parser, and that the rules that we used to convert dependency nodes into spans may have produced some errors. It is possible that the figures would have been slightly higher if our program produced semantic dependency graphs directly.

Table 3: Results for frame and FE detection.

Setting			Recall	Precision	$F1$
E	L	Y	0.372	0.532	0.438
P	L	Y	0.398	0.570	0.468
E	D	Y	0.389	0.557	0.458
P	D	Y	0.414	0.594	0.488
E	L	N	0.364	0.530	0.432
P	L	N	0.391	0.570	0.464
E	D	N	0.384	0.561	0.456
P	D	N	0.411	0.600	0.488

5 Conclusion and Future Work

We have presented a system for frame-semantic structure extraction that achieves promising results. While most previous systems have been based on constituents, our system relies on a dependency parser. We also described an automatic method to add new units to the FrameNet lexical database.

To improve labeling quality, we would like to apply constraints to the semantic output so that semantic type and coreness rules are obeyed. In addition, while the system described here is based on pipelined classification, recent research on semantic role labeling has shown that significant performance improvements can be gained by exploiting interdependencies between arguments (Toutanova et al., 2005). With an increasing amount of running text annotated with frame semantics, we believe that this insight can be extended to model interdependencies between frames as well.

Our motivation for using dependency grammar is that we hope that it will eventually make semantic structure extraction easier to implement and more theoretically well-founded. How to best design the dependency syntax is also still an open question. Ideally, all arguments would be direct dependents of the predicate node and we could get rid of the sparse and brittle *Path* feature in the classifier.

References

David Ahn, Sisay Fissaha, Valentin Jijkoun, and Maarten de Rijke. 2004. The university of Amsterdam at Senseval-3: Semantic roles and logic forms. In *Proceedings of SENSEVAL-3*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the CoNLL-X*.

Katrin Erk. 2005. Frame assignment as word sense disambiguation. In *Proceedings of IWCS 6*.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

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

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *ACL-2003*.

Chistopher Manning. 1994. Ergativity: Argument structure and grammatical relations.

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

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser generator for dependency parsing. In *Proceedings of LREC*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *ACL-2005*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL 2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT-03*.