A Dependency-Based Conversion of PropBank

Susanne Ekeklint
Växjö University
sek@msi.vxu.se

Joakim Nivre
Växjö University and Uppsala University
nivre@msi.vxu.se

Abstract

As a prerequisite for the investigation of dependency-based methods for semantic role labeling, this paper describes the creation of a dependency-based version of the widely used PropBank, DepPropBank, and discusses some of the issues involved in the integration of syntactic and semantic dependency structures.

1 Introduction

The long-term goal of our research is to investigate the suitability of dependency-based representations for semantic role labeling (SRL). Our research also includes different ways of integrating semantic information into syntactic dependency structures. It has already been established that syntactic information is necessary for accurate SRL (Gildea and Palmer, 2002). It is however still an open issue which type of syntactic information should be used and how this information should be structured. The majority of published experiments on SRL are based on treebanks annotated with phrase structure. For the type of experiments that we wish to conduct, no suitable resource was available, so we decided to create one. In this paper we will therefore describe the creation of a dependency version of PropBank, called DepPropBank, and discuss some of the issues involved in the integration of syntactic and semantic dependency structures.

2 SRL and PropBank

The SRL that we consider is of the predicate-argument type and this type of semantic information can be used in order to improve quality in different natural language processing tasks, such as information retrieval, dialog management, translation or summarization. Typically, any application that needs to recognize entities answering to question words such as "Who", "When", and "Why" can benefit from this type of information. Figure 1 is an example of a sentence containing one predicate (*set*) and the arguments belonging to it. The SRL task can shortly be described as follows:

> Given a sentence the task consists of analyzing the propositions expressed by some target verbs of the sentence. In particular, for each target verb all the constituents in the sentence which fill a semantic role of the verb have to be recognized. It also includes determining which semantic role that each constituent has. (Carreras and Màrques, 2005)

Since it is important for us to be able to compare our experiments to previous work, we decided to create our data sets from PropBank (Palmer et al., 2005). PropBank is the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993), enriched with annotation of predicate-argument relations. PropBank is one of the most widely used resources for SRL experiments, popularized in particular by The CoNLL shared tasks in 2004 and 2005 (Carreras and Màrques, 2005), which have had a large impact on SRL and can be seen as representing the state of the art for this particular task. An annotation unit in PropBank is called a proposition and consists of a verb together with its semantic arguments, classified

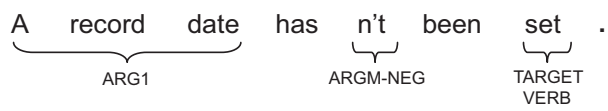

Figure 1: Sentence wsj02wsj_0202 from PropBank labeled with predicate argument-relations.

by numbered verb-specific roles or by general semantic modifier roles. The numbered verb-specific roles are ARG0-ARG5, where for example ARG0 in general corresponds to *agent* and ARG1 to *patient* or *theme*. The general semantic modifiers are adjuncts or functional labels that any verb may take optionally. There are 13 general semantic modifiers, e.g., ARGM-ADV for *general-purpose* and ARGM-NEG for *negation*. The roles are defined according to the role set for each verb, which defines the possible usage of each verb according to VerbNet (Levin, 1993). The PropBank data includes 44631 semantically annotated sentences, with an average of 2.53 propositions per sentence 3.21 arguments per proposition.

3 Dependency-Based SRL

A syntactic dependency graph is a labeled directed graph $G = (V, A_{syn})$, where V is a set of nodes, corresponding to the words of a sentence, and A_{syn} is a set of labeled directed arcs, representing syntactic dependency relations. The basic idea in dependency-based SRL is that we can construct an integrated syntactic-semantic representation by adding a second set A_{sem} of labeled arcs, representing semantic role relations, which gives us a multigraph $G = (V, A_{syn}, A_{sem})$, with two sets of labeled arcs defined on the same set of nodes. The SRL task can then be defined as the task of deriving A_{sem} given V and A_{syn}. In order to perform experiments based on PropBank, we therefore needed to convert the representations in the original Penn Treebank and PropBank to integrated syntactic-semantic dependency graphs. The result of this conversion is what we call DepPropBank.

4 DepPropBank

When designing the conversion from PropBank to DepPropBank we have had three different, partly conflicting requirements in mind:

1. We want to use the converted representations for machine learning experiments on dependency-based SRL, as described in the previous section. (Learnability)

2. We want to preserve the information in the original PropBank as precisely as possible. (Faithfulness)

3. We want to integrate syntactic and semantic relations as closely as possible. (Integration)

These requirements are not always compatible, and different trade-offs are possible. Therefore we have decided to create three different versions of DepPropBank, using three different models for integrating semantic information with syntactic dependency structures, investigating various degrees of tight and loose coupling in the integration. In formal terms, this amounts to three different algorithms for creating the set A_{sem} of semantic relations, given the set of nodes V, the set A_{syn} of syntactic relations, and the original PropBank annotation. The benefit of having three different versions is that we can empirically investigate the impact of different representational choices on SRL accuracy. We call the three different versions DepPropBank 1, 2, and 3.

However, before we could start to integrate the semantic information we needed to convert the syntactic phrase structures in the Penn Treebank to dependency structures. This was done using the freely available conversion program Penn2Malt.[1] This conversion is far from perfect but sufficiently precise for our current purposes. In the future we may instead decide to use the recently developed pennconverter (Johansson and Nugues, 2007),[2] which provides an improved conversion that, among other things, takes empty categories into account.

The next step was to relate the semantic annotation in PropBank to the phrase structure representations in the Penn Treebank. Figure 2 shows how the proposition for the target verb *set* from PropBank is integrated with the corresponding phrase structure from the Penn Treebank.

[1] http://w3.msi.vxu.se/∼nivre/research/Penn2Malt.html
[2] http://nlp.cs.lth.se/pennconverter/

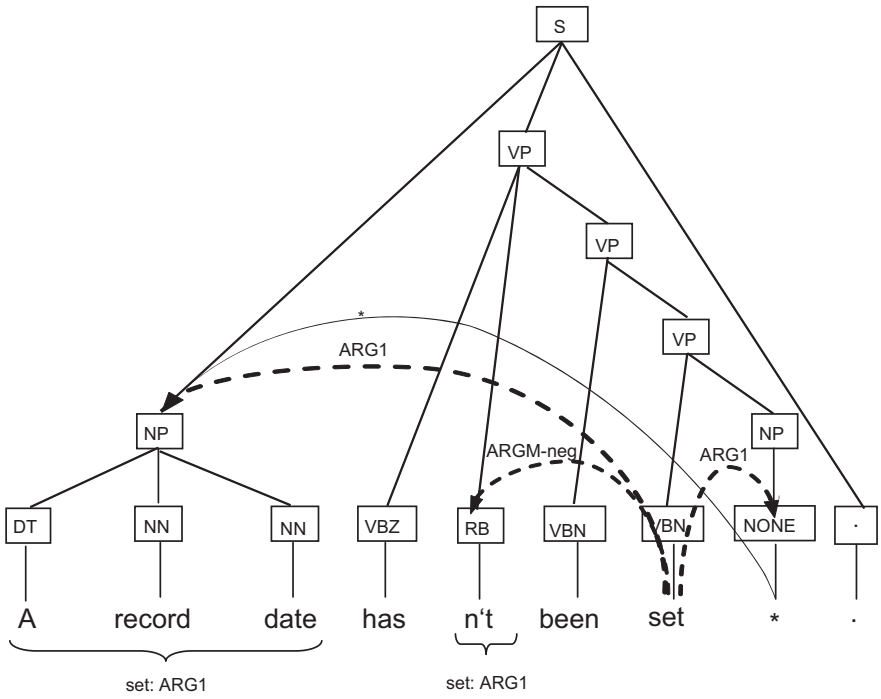

Figure 2: The phrase structure representation of sentence wsj02wsj_0202 in PropBank

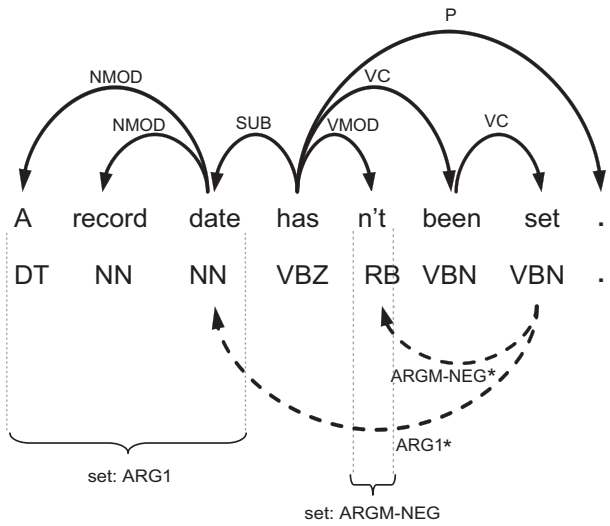

Figure 3: The dependency structure representation of sentence wsj02wsj_0202 in DepPropBank 1.

Since a dependency representation only contains terminal nodes (words), we needed to map these references to word sequences, also taking into account empty categories and co-indexation. The original PropBank annotation identifies predicates and arguments by referring to nodes in the syntactic annotation of the Penn TreeBank. An argument in the PropBank representation can be composed of several subtrees in the syntactic representation. We will refer to a sequence of words included in an argument as the *span* of that argument.

4.1 DepPropBank 1

Given that we have identified all the argument spans associated with a given predicate (and their semantic roles), we can extend the dependency graph generated by the syntactic conversion by adding arcs for semantic roles. In the first version of DepPropBank, this was done in the following way:

> Given an argument span s of predicate p with semantic role r:
>
> 1. For every word w within s that does not have its syntactic head within s, add an arc $p \xrightarrow{r^*} w$.
> 2. For every word w within s that has its syntactic head within s, assume that w belongs to the semantic spans of its syntactic head.

Figure 3 shows a dependency graph where the syntactic arcs in A_{syn}, drawn above the words, form a tree as usual, and where the semantic arcs in A_{sem} are represented by dotted arcs below the words. Note that the semantic arc labeled ARG1* only points to the syntactic head *date*, while the semantic argument span includes the whole syntactic subtree rooted at this node. We use the superscript $*$ on semantic arc labels to indicate that the argument relation extends transitively to syntactic descendants of the head.

Unfortunately, the first version of DepPropBank does not give an adequate representation of all the arguments in PropBank. The problem lies in the assumption that all syntactic dependents belong to the samt semantic spans as their head. This assumption holds for about 86% of all arguments in PropBank (given the current syntactic dependency conversion), but the remaining 14% require a more complex representation, where the internal semantic dependency structure of an argument does not necessarily coincide with its syntactic dependency structure. Figure 4 shows a sentence which has one correctly inherited syntactic subtree and one incorrect.

Looking at this result in a positive way, we can say that as many as 86% of the semantic subtrees have an exact match with the syntactic subtrees within their respective spans, in a representation where every semantic argument is represented by a single arc in A_{sem}. Experiments with this data set should therefore at least be interesting as a baseline for further experiments.

4.2 DepPropBank 2

The second version of DepPropBank 2 was created to solve the problem with the arguments that run outside the intended span. The semantic arcs in A_{sem} wee in this version simply added as follows:

> Given an argument span s of predicate p with semantic role r, add an arc $p \xrightarrow{r} w$ for every word w within s.

Figure 5 shows the same sentence fragment as figure 4, although this time with the representation of DepPropBank 2. Note the absence of the superscript $*$ on semantic role labels to indicate that each arc concerns only the word itself, not its syntactic descendants. The semantic representation has a very loose coupling to the syntactic structure in this version and the obvious drawback of version 2 is the flattening of the semantic structures. However, the representation has the advantage that there is always a single arc connecting each word in a semantic argument span to its predicate. Since there is an average of 2.5 propositions per sentence, of which several have partially or completely overlapping arguments, assigning hierarchical structures to semantic arguments would require a multigraph also for the semantic representation, where two nodes can be connected by more than one (semantic) arc. We could have solved this problem in several ways (for example by adding extra features to the labels and keeping the arcs as they were), but for machine learning experiments we found this particular representation promising.

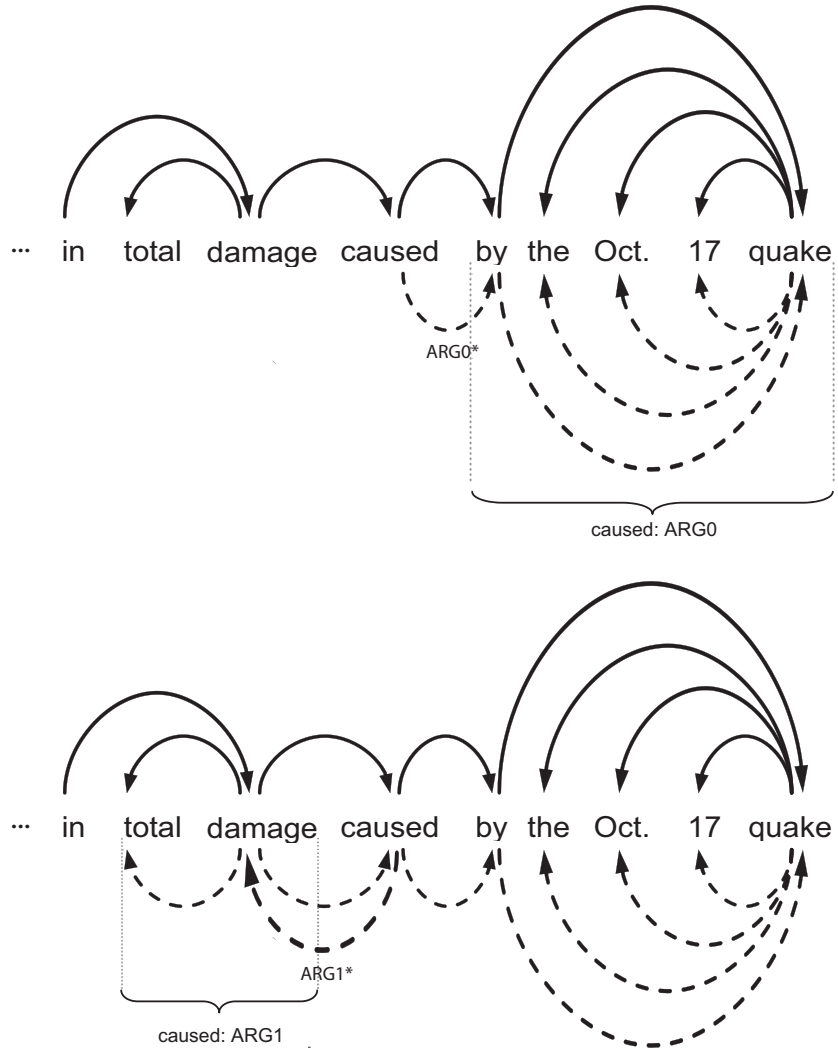

Figure 4: A good (top) and a bad (bottom) match between syntactic and semantic structure for arguments in DepPropBank 1.

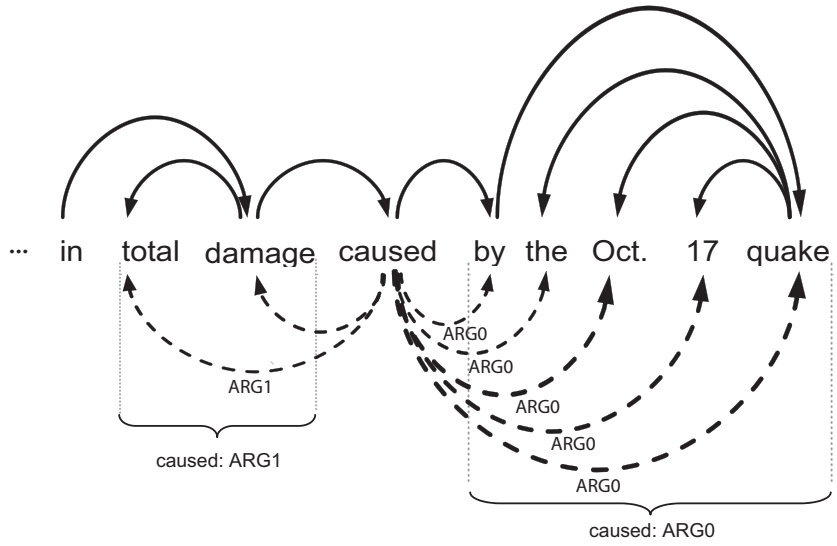

Figure 5: Flat semantic argument structure in DepPropBank 2.

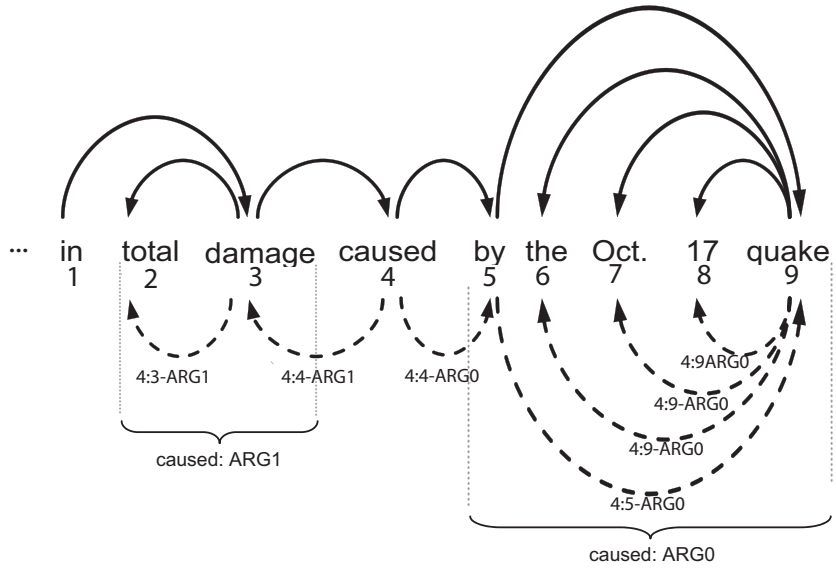

Figure 6: Hierarchical semantic argument structure in DepPropBank 3.

4.3 DepPropBank 3

Comparing DepPropBank 1 and 2 with respect to our three overall requirements, we can say that DepPropBank 1 maximizes syntactic-semantic integration (at the expense of faithfulness), while DepPropBank 2 maximizes faithfulness (at the expense of integration). From the point of view of learnability, both versions facilitate learning by minimizing path lengths in the semantic part of the graph (all paths being of length one), while DepPropBank 1 in addition minimizes the number of semantic arcs that need to be inferred (one arc per argument). In the third version, DepPropBank 3, the idea is to jointly maximize faithfulness and integration, possibly at the expense of learnability. The semantic arcs in A_{sem} were in this version added as follows:

> Given an argument span s of predicate p with semantic role r:
>
> 1. For each word w within s that does not have its syntactic head within s, add an arc $p \overset{i:i-r}{\rightarrow} w$, where i is the index (linear position) of p.
> 2. For each word w within s that has its syntactic head within s, add an arc $h \overset{i:j-r}{\rightarrow} w$, where h is the syntactic head of w, and i and j are the indices (linear positions) of p and h, respectively.

The advantage of this representation is that it has a strong integration of the semantic and syntactic structure without losing any of the information in the original annotation. The downside is the more complex graphs that we have to handle from a machine learning perspective. In fact, (V, A_{sem}) now needs to be a multi-graph, since it is possible to have two nodes connected by more than one arc. Moreover, the labels must encode the index of the predicate, which may be connected to a word by a path of arbitrary length. Figure 6 illustrates the more complex graphs of DepPropBank 3.

5 Conclusion

The three different versions of DepPropBank will allow us to empirically investigate the trade-off between integration, faithfulness and learnability in dependency-based SRL. Starting from the baseline of DepPropBank 1, which poses the simplest learning problem but where 14% of the arguments cannot be retrieved correctly, we can move on to the more faithful but also more complex representations in DepPropBank 2 and 3.

Since our data sets are derived from PropBank, we are also able to compare our results with the state of the art in SRL. In addition, we can investigate whether dependency-based representations give a better fit between argument spans and syntactic units than phrase structure representations. Finally, it is worth nothing that our models are applicable to languages that have treebanks annotated with dependency structure but not phrase structure, such as Czech (Böhmova et al., 2003) and Danish (Kromann, 2003), among others.

References

Alena Böhmová and Jan Hajič and Eva Hajičová and and Barbora Hladká. 2003. The Prague Dependency Treebank: A Three-Level Annotation Scenario In Anne Abeillé (ed.) *Treebanks: Building and Using Parsed Corpora*, Kluwer.

Xavier Carreras and Lluis Màrquez 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Dan Gildea and Martha Palmer 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL-2002*.

Richard Johansson and Pierre Nugues 2007. Extended Constituency-to-Dependency Conversion for English. In *Proceedings of NODALIDA-2007*.

Matthias Trautner Kromann. 2003. The Danish Dependency Treebank and the DTAG Treebank Tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

Beth Levin 1993. *English Verb Classes and Alternations: A Preliminary Investigation* The University of Chicago Press

Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics, 19*

Martha Palmer, Daniel Gildea and Paul Kingsbury 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1), 71–106.