Theory-Supporting Treebanks

Joakim Nivre

Växjö University
School of Mathematics and Systems Engineering
E-mail: Joakim.Nivre@msi.vxu.se

1 Introduction

The question of how treebank annotation schemes should be related to linguistic theories has been debated as long as treebanks have existed. Historically speaking, it is probably true to say that there has been a development from mostly theory-neutral annotation schemes to more theoretically oriented frameworks or even annotation schemes tailored to one specific theory (cf. Nivre [12]). However, given the considerable cost involved in developing treebanks, it is still important to discuss which kind of treebank is most useful to the research community as a whole.

The main argument in favor of theory-neutral annotation seems to be that the treebank can be used by a larger group of researchers, working within different theoretical frameworks. On the other hand, there is always a risk that the annotation contains too little information or too many compromises to be really useful for anyone. In analogy with this, it can be argued that theory-specific treebanks are more useful for people working within the selected theoretical framework but have the disadvantage of shutting out people from other research traditions.

To some extent, I believe that the drawbacks of both approaches have been exaggerated, mainly because the possibilities of converting between different kinds of annotation have been underexploited. For example, the Penn Treebank has sometimes been criticized for not providing enough information to support research within specific theoretical frameworks. Nevertheless, it has proven possible to convert this treebank into a variety of different formats, often with surprisingly high accuracy. For example, the treebank has been converted into dependency structures (Lin [9], Collins [4, 5], Xia [16], Xia and Palmer [17]), LFG f-structures (Cahill et al. [3]), and GLARF representations (Meyers et al. [11]). Similarly, the German NEGRA treebank has been converted into LTAG representations and topological frame representations (Frank [6]), as well as dependency structures (Bohnet [1]).

However, I also believe that the usefulness of treebanks can be improved if these conversions do not have to be performed post hoc. Thus, even in the cases of successful conversion cited above, the conversion was always less than 100% accurate, because the source annotation lacked some information that would have been needed in order to support a completely accurate conversion. Therefore, it may be worth including possible conversions as a requirement in the design of treebank annotation schemes. Instead of asking whether the annotation should be theory-specific and, if so, which theory should be chosen, we should ask what different kinds of (theory-specific) annotation we want to *support* through automatic conversion from an underlying internal representation which in itself need not correspond to the representation of any particular theory.

In this paper, I will illustrate the idea of a theory-supporting treebank in the context of a Swedish treebank project, which is still at the stage of designing its annotation scheme. In particular, I will demonstrate how a basic annotation scheme based on phrase structure trees with labeled edges, of the kind used in the German TIGER project (Brants et al. [2]) and advocated for Swedish by Volk and Gustafson-Capkova [15], can be used to support a theoretical framework based on dependency grammar. I will do this by presenting an algorithm that converts the phrase structure annotation to a dependency tree and discuss under what conditions this conversion succeeds. However, before we turn to a detailed consideration of the conversion algorithm (section 3), I want to discuss some more general issues in the design of a theory-supporting treebank annotation scheme (section 2).

2 Theory-Supporting Annotation Schemes

In the ideal case, a theory-supporting treebank can be seen as a way of getting n different theory-specific treebanks for the price of 1, in the sense that the labor-intensive annotation work only needs to be performed once for the underlying theory-supporting annotation, while the n different theory-specific treebanks can be derived automatically using well-defined conversion algorithms.

In practice, the price will normally be higher than 1, since the underlying representation needs to be more informative than an ordinary annotation scheme in order to support conversion to other annotation schemes, which means that the amount of work needed to produce the annotation will be greater than for ordinary treebanks.

However, as long as the amount of work does not grow proportionally with the number of schemes supported, there could still be a considerable reduction in costs. Moreover, the mere existence of well-defined conversion algorithms will make it easier to systematically compare analyses couched in different theoretical frameworks, which could be seen as a benefit even without the cost reduction.

To take a concrete example, the Swedish Treebank Symposium 2002 [14] was devoted to a discussion of different possible annotation schemes to use in a project to build a large Swedish treebank. The outcome of the discussion was that at least three different kinds of annotation were favored by different research groups:

1. Chunks and clauses (C): Annotation of non-recursive constituent structures (chunks) and clause boundaries.

2. Phrase structure (P): Annotation of hierarchical constituent structure in the form of phrase structure trees.

3. Dependency structure (D): Annotation of dependency structure in the form of (labeled) dependency trees.

One way of satisfying the requirements of different research groups would then be to set up a theory-supporting annotation scheme, depicted schematically in Figure 1, where S stands for source annotation and C_C, C_P and C_D denote conversions from this source annotation to the target annotations C, P and D, respectively.

$$\begin{tikzcd}
 & S \arrow[dl, "C_C"] \arrow[d, "C_P"] \arrow[dr, "C_D"] & \\
C & P & D
\end{tikzcd}$$

Figure 1: Theory-supporting treebank annotation scheme for Swedish

In order to build a Swedish treebank along these lines, we would therefore need to define a source annotation scheme S that could support accurate conversion to each of the three target schemes (C, P, D). As already indicated, I believe that the role of S could be played by an annotation scheme similar to the one used in the German TIGER project (Brants et al. [2]) and advocated for Swedish by Volk and Gustafson-Capkova [15].

Given that the source annotation scheme is based on phrase structure, the conversion to C and P would in most cases be very simple, whereas the conversion to dependency trees is less straightforward. For example, Bohnet [1] reports an accuracy rate of 74% in converting from the NEGRA annotation scheme (the predecessor of TIGER) to dependency structures of the kind used as surface representations in Meaning-Text Theory (cf. Mel’čuk and Pertsov [10]). I hope to show in section 3 that the accuracy can be improved considerably if the conversion can be taken into account when defining the source and target annotation schemes.

In general, we can define a theory-supporting annotation scheme as consisting of a triple $\langle S, T, C \rangle$, where:

- S is a source annotation scheme.
- $T = \{T_1, \ldots, T_m\}$ is a set of target annotation schemes T_i.
- $C = \{C_1, \ldots, C_m\}$ is a set of mappings $C_i : S \to T_i$.

This definition gives rise to a whole range of questions that need to be adequately answered before we can put the idea into practice. How do we choose the annotation schemes to include in the larger theory-supporting scheme? In particular, how do we choose the source annotation scheme S? How do we define the mappings from source to target annotation schemes? How do we implement conversion algorithms, and how are these algorithms best used in actual treebank development?

When it comes to choosing the target annotation schemes, it seems that this must be decided by the (current) interests of the research community. Thus, in the case of the Swedish treebank discussed earlier, the three annotation schemes C, P and D together might satisfy the immediate needs of Swedish researchers.

If the target schemes are determined by the interests of the research community, the source annotation scheme S should be defined in relation to the given set T of target schemes. Ideally, S should be the minimal annotation scheme that supports accurate conversion to all the schemes $T_i \in T$, but the definition of S is also dependent on the way we define the mappings C_i.

We can make this formally more precise by adopting a logical perspective on corpus annotation (cf. Lager [8] and Simov [13]). From this perspective, an annotation scheme A can be seen as an interpreted formal language, and the annotation of a specific corpus c using A, which I will denote $A(c)$, is equivalent to a logical theory, typically consisting of a non-redundant set of atomic statements about c expressed in A.

A mapping (or conversion) C_i from a source annotation scheme S to a target annotation scheme T_i can be defined as another logical theory, typically consisting of universally quantified conditionals such that, for any corpus c, the annotation $A(c)$ together with C_i entails the annotation $T_i(c)$:

$$S(c) \cup C_i \vdash T_i(c)$$

Therefore, the first requirement on S must be that there exists a well-defined mapping C_i for every target scheme T_i:

$$\forall T_i \, \exists C_i \, \forall c \, : \, S(c) \cup C_i \vdash T_i(c)$$

However, this is clearly not sufficient to characterize the ideal source scheme, since this requirement could be satisfied by constructing S so that $S(c)$ is always the union of all $T_i(c)$, in which case we can let each C_i be the empty theory $\emptyset$. And since we want to minimize the effort needed to construct $S(c)$ for a given corpus c, we would rather put less information into $S(c)$ and more into the mappings.

Let us say that an annotation scheme S satisfying the above constraint is convertible with respect to the set T of target schemes, denoted $Conv_T(S)$. We can then propose that the source scheme S should satisfy the following requirement:

$$Conv_T(S) \wedge \forall S' \, \forall c : Conv_T(S') \Rightarrow S'(c) \vdash S(c)$$

That is, in addition to being convertible with respect to T, the ideal source scheme S should give us the weakest possible annotation of every corpus c. This theoretical ideal may not be attainable in practice, and it certainly does not give us a practical procedure for constructing S and C given T. Hence, in order to implement the idea of theory-supporting treebanks in practical projects, we need to organize the work in such a way that we can approximate this ideal as closely as possible. For example, we could adopt an iterative-incremental approach as follows:

1. Define the set T of target annotation schemes.

2. Define a tentative source annotation scheme S and a set C of conversion algorithms (with one algorithm C_i for every target scheme T_i).

3. Construct $S(c)$ for some (initially small) corpus c.

4. For every target scheme T_i, apply the corresponding conversion algorithm C_i to produce a converted annotation $C_i(S(c))$ and compare this to the (correct) target annotation $T_i(c)$:

   (a) If $C_i(S(c)) \neq T_i(c)$, try revising C_i and/or T_i until $C_i(S(c)) = T_i(c)$.

5. After evaluation and (possibly) revisions:

   (a) If there is some T_i such that $C_i(S(c)) \neq T_i(c)$, try revising S until $C_i(S(c)) = T_i(c)$ (while maintaining $C_j(S(c)) = T_j(c)$ for all previously accurate conversions).

   (b) If there is no T_i such that $C_i(S(c)) \neq T_i(c)$, consider the possibility of reducing S (weakening $S(c)$ for arbitrary c).

6. Repeat steps 3–5 with a successively larger corpus until no new conversion errors occur.

As far as I know, this approach to treebank development has never been tested in practice, and there are obviously many details that have to be worked out before it can really be put to use. One of the most serious difficulties probably resides in the construction of conversion algorithms. I have deliberately overloaded the notation C_i in this section, using it both to refer to a certain kind of logical theory, which together with a source annotation $S(c)$ entails a target annotation $T_i(c)$, and to an algorithm that actually computes a target annotation from a source annotation. But it is evident that the construction of the latter from the former is a non-trivial problem, if indeed it is a problem that can be solved at all in the general case.

In order to provide a proof of concept for the kind of conversion algorithms that are needed to create theory-supporting treebanks on a large scale, I will now present a specific conversion algorithm that has been developed in the context of the Swedish treebank project referred to above. Although this will not in itself be proof of either the possibility or the usefulness of theory-supporting treebanks as such, it may at least show that the idea is worth exploring further.

3 A Treebank Conversion Algorithm

3.1 Source: Phrase Structure Trees with Labeled Edges

The source annotation scheme defined in this section is based on the proposal by Volk and Gustafson-Capkova [15], which in turn is based on the annotation scheme adopted in the German TIGER project (Brants et al. [2]). The annotation consists of phrase structure trees, where nonterminal nodes are labeled with syntactic categories and edges are labeled with syntactic functions. Unlike in ordinary phrase structure trees, edges are allowed to cross in order to capture discontinuous constituents. Formally, we define such an annotation scheme as follows:

- Let $\mathcal{L}_C$ be a set of node labels (syntactic categories) and $\mathcal{L}_F$ a set of edge labels (syntactic functions).
- Given a sentence S consisting of k words, an annotation structure for S consists of:
    1. A set of nodes $V = \{1, \ldots, n\}$, where $T = \{1, \ldots, k\}$ is the set of terminal nodes corresponding to the words of S and $N = \{k+1, \ldots, n\}$ is the set of nonterminal nodes.
    2. A total function $C : N \to \mathcal{L}_N$ assigning labels to nonterminal nodes.
    3. A partial function $F : N \times V \to \mathcal{L}_F$, where $F(i, j) = l$ iff there is an edge from i to j labeled l. We assume that the labeled graph induced by F forms a rooted tree.

Terminal nodes are assumed to be labeled with word forms, possibly extended with parts-of-speech, and will be ignored in the following (since they remain constant across the two schemes). Figure 2 shows an example of a phrase structure tree with labeled edges for a simple Swedish sentence.

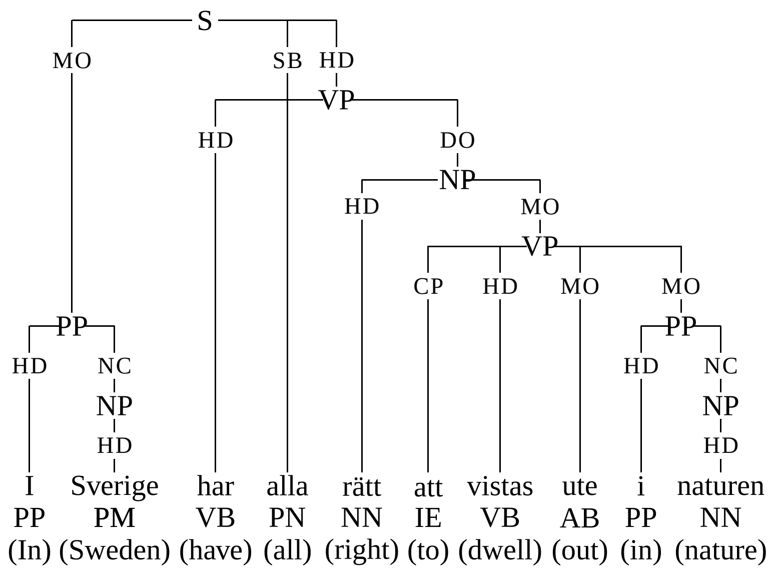

Figure 2: Phrase structure tree with labeled edges

3.2 Target: Dependency Trees

The target annotation scheme uses lexicalized dependency trees, where all nodes correspond to words of the sentence and edges are labeled with dependency relations. Formally, such a scheme can be defined in the following way:

- Let $\mathcal{L}_D$ be a set of edge labels (dependency relations).
- Given a sentence S consisting of k words, an annotation structure for S consists of:
    1. A set of nodes $T = \{1, \ldots, k\}$ corresponding to the words of S.
    2. A partial function $D : T^2 \rightarrow \mathcal{L}_D$, where $D(i, j) = l$ iff there is an edge from i to j labeled l. Again, we assume that the resulting graph is a rooted tree.

Figure 3 shows a dependency tree for the same sentence as in Figure 2.

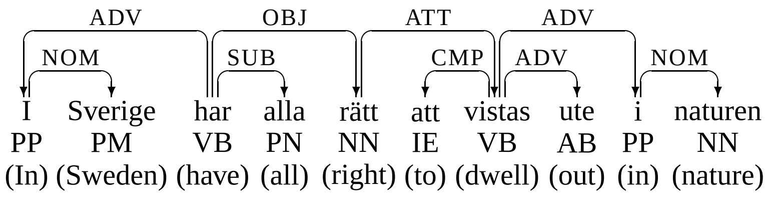

Figure 3: Dependency tree

3.3 Conversion Algorithm

It is well known that we can convert a phrase structure to a dependency structure provided that each nonterminal node has a head child (i.e. either a lexical head or a phrase containing the lexical head), which we can identify unambiguously. For every phrase in the original tree, we then make each non-head child depend on the lexical head of the head-child (Xia and Palmer [17]).

In practice, there are two problems with this approach. The first is that most annotation schemes based on phrase structure do not provide enough information to identify the head child for all kinds of phrases. Thus, neither the Penn treebank (Lin [9], Xia and Palmer [17]) nor the NEGRA treebank (Bohnet [1]) supports the identification of heads with perfect accuracy, which means that the conversion to dependency structures will only be approximately correct. In a theory-supporting annotation scheme, it should be possible to eliminate this problem altogether.

The second problem is more serious and resides in the fact that the notion of head that is needed for the construction of dependency structures may not be equivalent to the notion of head found in a phrase structure representation (if there is such a notion at all). For example, the latter scheme may allow phrases with multiple heads as well as phrases with no head.

Therefore, we will not require that every phrase in the source annotation has exactly one head, which coincides with the head in the dependency structure. Instead, we will only require that the source annotation contains enough information to identify the head needed for the dependency structure. More precisely, in order for the conversion to be possible, the following two conditions have to be satisfied:

- For every category label $c \in \mathcal{L}_C$, there is a partitioning of $\mathcal{L}_F$ into a set $H(c)$ (head functions) and a set $D(c)$ (dependent functions) such that every node labeled with c has exactly one outgoing edge labeled with an element of $H(c)$.

- For every category label $c \in \mathcal{L}_C$, there is a mapping $d : D(c) \rightarrow \mathcal{L}_D$ such that $d(l)$ is the dependency relation corresponding to syntactic function l.

Let S be a sentence of length k and let (T, N, C, F) be its phrase structure annotation. The algorithm in Figure 4 computes the corresponding dependency annotation (T, D), using two temporary data structures: an array HEAD of nodes (integers) of size n (with all elements initialized to zero), and an array LABEL of labels of size k. In addition, it uses the auxiliary function PARENT, which returns the parent of a node in the source tree.

```
for $i$ = 1 to $k$ do
    $c \leftarrow i$;
    while PARENT($c$) $\neq n$ do
        $p \leftarrow$ PARENT($c$);
        if $F(p, c) \in H(C(p))$ then
            HEAD[$p$] $\leftarrow i$;
        else
            HEAD[$i$] $\leftarrow p$;
            LABEL[$i$] $\leftarrow d(F(p, c))$;
            break;
        end if;
    end while;
end for;

$D \leftarrow \emptyset$
for $i$ = 1 to $k$ do
    if HEAD[i] $\neq$ 0 then
        $D \leftarrow D \cup \{\langle\langle$HEAD[HEAD[$i$]], $i\rangle$,LABEL[$i$]$\rangle\}$
    end if;
end for;
```

Figure 4: Conversion algorithm

The algorithm traces the path from each terminal i towards the root n of the source tree. As long as edges are labeled with head functions, it records that i is the head of each ancestor node p. When a dependent function is encountered, it stops the traversal after recording that the head of i is to be found in the current ancestor node p and that the dependency relation should be whatever the current edge label is mapped to in the target scheme. The final loop computes the function D by adding, for each terminal node i except the root, an edge from HEAD[HEAD[i]] to i with label LABEL[i]. As an example, consider the phrase structure in Figure 2, which can be converted into the dependency structure in Figure 3 given the information in Table 1.

c	$H(c)$	$D(c)$	$d(l)$
S	HD	MO	ADV
		SB	SUB
VP	HD	DO	OBJ
		CP	CMP
		MO	ADV
PP	HD	NC	NOM
NP	HD		

Table 1: Head and dependent functions; mappings for dependent functions

It is straightforward to prove that the algorithm always produces a well-formed dependency tree, given that the conditions stated above are satisfied. Moreover, the method of conversion allows considerable freedom in the way that a source analysis is mapped to a target analysis.

For example, it is not necessary that the two analysis schemes agree on what constitutes the head of a certain construction. Thus, if we want prepositions to depend on nouns in the dependency structure (instead of the other way around), we simply have to change the partitioning into head and dependent function so that NC $\in H(\mathrm{PP})$ and HD $\in D(\mathrm{PP})$ (cf. Table 1).

It remains to be seen if the method is general and flexible enough to provide adequate conversions for the whole range of linguistic phenomena that need to be dealt with in treebank annotation.

4 Conclusion

The idea of a theory-supporting treebank, as defined in the present paper, can be seen as an attempt to combine the advantages of theory-neutral and theory-specific treebank annotation. While the target annotation schemes are theory-specific, the source annotation scheme must at least be neutral in the sense that it supports conversion to all target schemes.

Having well-defined conversions from source to target schemes potentially has two advantages. First, it should allow us to produce a number of theory-specific treebanks at substantially lower cost than if each treebank had to be developed independently. Secondly, it should allow us to make systematic comparisons between analyses couched in different theoretical frameworks. In this way, the idea has connections with work on grammar conversion for evaluation purposes (see, e.g., Kinyon and Rambow [7]).

However, it still remains to be seen to what extent these potential advantages can be realized. Before anyone has implemented the idea in practice, it is very difficult to say how large the gain will be, practically as well as theoretically. The conversion algorithm described in section 3 should be seen as a first step in this direction. In the future, we hope to be able to take further steps along this route, using the methodology outlined in section 2 in order to develop a practical theory-supporting treebank annotation scheme for Swedish.

References

[1] Bohnet, B. (2003) Mapping Phrase Structures to Dependency Structures in the Case of Free Word Order Languages. *The First International Conference on Meaning-Text Theory*, Paris, 2003.

[2] Brants, S., Dipper, S., Hansen, S., Lezius, W. and Smith, G. (2002) The TIGER Treebank. In Hinrichs, E. and Simov, K. (eds) *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, 24–42.

[3] Cahill, A., McCarthy, M., Van Genabith, J. and Way, A. 2002. Evaluating Automatic F-Structure Annotation for the Penn-II Treebank. In Hinrichs, E. and Simov, K. (eds) *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, 43–60.

[4] Collins, M. (1997) Three Generative Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting and the 8th Meeting of the European Chapter of the Association for Computational Linguistics*, Morgan Kaufman, pp. 16–23.

[5] Collins, M. (1999) *Head-Driven Statistical Models for Natural Language Parsing*. PhD Thesis, University of Pennsylvania.

[6] Frank, A. (2001) Treebank Conversion. Converting the NEGRA treebank to an LTAG grammar. In *Proceedings of the Workshop on Multi-layer Corpus-based Analysis*, July 30, 2001, Iasi, Romania.

[7] Kinyon, A. and Rambow, O. (2003) The MetaGrammar: A Cross-Framework and Cross-Language Test-Suite Generation Tool. LINC-03. In Abeille A., Brants, T. and Uszkoreit, H. (eds.) *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*.

[8] Lager, T. (1995) *TagLog: A Logical Approach to Computational Corpus Linguistics*. PhD Thesis, Göteborg University.

[9] Lin, D. (1995) A Dependency-Based Method for Evaluating Broad-Coverage Parsers. In *Proceedings of IJCAI*, pp. 1420–1427.

[10] Mel'čuk, I. A. and Pertsov, N. (1987) *Surface-Syntax of English. A Formal Model in the Meaning-Text Theory*. Amsterdam/Philadelphia: Benjamins.

[11] Meyers, A., Grishman, R., Kosaka, M. and Zhao, S. (2001) Covering Treebanks with GLARF. In *Proceedings of the ACL 2001 Workshop on Sharing Tools and Resources*.

[12] Nivre, J. (2002) What kinds of trees grow in Swedish soil? A Comparison of Four Annotation Schemes for Swedish. In Hinrichs, E. and Simov, K. (eds) *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, 123–138.

[13] Simov, K. (2003) HPSG-Based Annotation Scheme for Corpora Development and Parsing Evaluation. In *Proceedings of RANLP 2003*, pp. 432–429.

[14] Swedish Treebank Symposium, 28–29 November 2002, Teleborgs Slott, Växjö University. URL: http://www.msi.vxu.se/∼rics/treebank/.

[15] Volk, M. and Gustafson-Capkova, S. (2003) Some Suggestions for a Swedish Treebank. University of Stockholm: Department of Linguistics.

[16] Xia, F. (2001) *Automatic Grammar Generation from Two Different Perspectives*. PhD Thesis, University of Pennsylvania.

[17] Xia, F. and Palmer, M. (2001) Converting Dependency Structures to Phrase Structures. In Allan, J. (ed.) *Proceedings of HLT 2001, First International Conference on Human Language Technology Research*, Morgan Kaufmann.