Extended constituent-to-dependency conversion for English

Johansson, Richard; Nugues, Pierre

Published in:
NODALIDA 2007 Proceedings

2007

Link to publication

Citation for published version (APA):
Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In J. Nivre, H.-J. Kalep, K. Muischnek, & M. Koit (Eds.), *NODALIDA 2007 Proceedings* (pp. 105-112). University of Tartu. http://dspace.utlib.ee/dspace/bitstream/10062/2560/1/reg-Johansson-10.pdf

Total number of authors:
2

Extended Constituent-to-Dependency Conversion for English

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

Abstract

We describe a new method to convert English constituent trees using the Penn Treebank annotation style into dependency trees. The new format was inspired by annotation practices used in other dependency treebanks with the intention to produce a better interface to further semantic processing than existing methods. In particular, we used a richer set of edge labels and introduced links to handle long-distance phenomena such as *wh*-movement and topicalization.

The resulting trees generally have a more complex dependency structure. For example, 6% of the trees contain at least one non-projective link, which is difficult for many parsing algorithms. As can be expected, the more complex structure and the enriched set of edge labels make the trees more difficult to predict, and we observed a decrease in parsing accuracy when applying two dependency parsers to the new corpus. However, the richer information contained in the new trees resulted in a 23% error reduction in a baseline FrameNet semantic role labeler that relied on dependency arc labels only.

1 Introduction

Labeled dependency parsing has become increasingly popular during the last few years. Dependency syntax offers a number of advantages from a practical perspective such as the availability of efficient parsing algorithms that analyze sentences in linear time while still achieving state-of-the-art results. It is arguably easier to understand and to teach to people without a linguistic background, which may be of use when annotating domain-specific data such as in medicine. Finally, some linguists argued that dependency grammar is universal whereas constituents would be more English-centric (Mel'čuk, 1988).

From a theoretical perspective, dependency syntax is arguably more intuitive than constituent syntax when explaining *linking*, i.e. the realization of the semantic arguments of predicates as syntactic units. This may also have practical implications for "semantic parsers", although this still remains to be seen in practice.

As statistical parsing is becoming the norm, syntactically annotated data, and hence the annotation style they adopt, plays a central role. For English, no significant dependency treebank exists, although there have been some preliminary efforts to create one (Rambow et al., 2002). Instead, the constituent-based Penn Treebank (Marcus et al., 1993), which is the largest treebank for English and the most common training resource for constituent parsing of this language, has been used to train most of the data-driven dependency parsers reported in the literature. However, since it based on constituent structures, a conversion method must be applied that transforms its constituent trees into dependency graphs.

The dependency trees produced by existing conversion methods (Magerman, 1994; Collins, 1999; Yamada and Matsumoto, 2003), which have been used by all recent papers on English dependency parsing, have been somewhat simplistic in view of original dependency treebanks such as the Danish Dependency Treebank (Trautner Kromann, 2003), in particular with respect to the set of edge labels and the treatment of complex long-distance linguistic relations such as *wh*-movement, topicalization, *it*-clefts, expletives, and gapping. However, this information *is* available in the Penn Treebank from version II when its syntactic representation was extended from bare bracketing to a much richer structure (Marcus et al., 1994), but with a few exceptions this has not yet been reflected by automatic parsers, neither constituent-based nor dependency-based.

This article describes a new constituent-to-dependency conversion procedure that makes better use of the existing information in the Treebank. The idea of the new conversion method is to make use of the extended structure of the recent versions of the Penn Treebank to derive a more "semantically useful" representation. The first section of the article presents previous approaches to converting constituent trees into dependency trees. We then describe the modifications we brought to the previous methods. The last section describes a small experiment in which we study the impact of the new format on the performance of two statistical dependency parsers. Finally, we examine how the new representation affects semantic role classification.

2 Previous Constituent-to-Dependency Conversion Methods

The current conversion procedures are based on the idea of assigning each constituent in the parse tree a unique *head* selected amongst the constituent's children (Magerman, 1994). For example, the toy grammar below would select the noun as the head of an NP, the verb as the head of a VP, and VP as the head of an S consisting of a noun phrase and and a verb phrase:

```
NP --> DT NN*
VP --> VBD* NP
S  --> NP VP*
```

By following the child-parent links from the token level up to the root of the tree, we can label every constituent with a *head token*. The heads can then be used to create dependency trees: to determine the parent of a token in the dependency tree, we locate the highest constituent that it is the head of and select the head of its parent constituent.

Magerman (1994) produced a *head percolation table*, a set of priority lists, to find heads of constituents. Collins (1999) modified Magerman's rules and used them in his parser, which is constituent-based but uses dependency structures as an intermediate representation. Yamada and Matsumoto (2003) modified the table further and their procedure has become the most popular one to date. Penn2Malt (Nivre, 2006) is a reimplementation of Yamada and Matsumoto's method, and also defines a set of heuristics to infer arc labels in the dependency tree. Figure 1 shows the constituent tree of the sentence *Why, they wonder, should it belong to the EC?* from the Penn Treebank and Figure 2, the corresponding dependency tree produced by Penn2Malt.

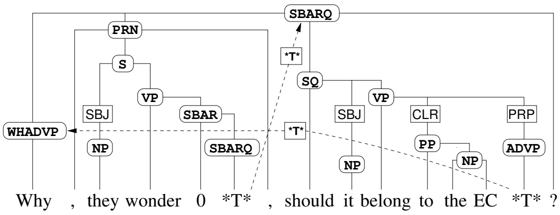

Figure 1: A constituent tree from the Penn Treebank.

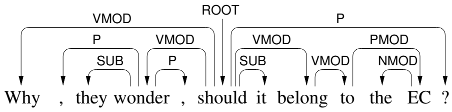

Figure 2: Dependency tree by Penn2Malt.

3 The New Conversion Procedure

As can be seen from the figures, the dependency tree that is created by Penn2Malt discards deep information such as the fact that the word *Why* refers to the purpose of the verb *belong*. It thus misses the direct relation between this question and a possible answer *It should belong to the EC because...* This relation is nevertheless present in the Penn Treebank II and is encoded in the form of a PRP link (purpose or reason) from the verb phrase to an empty node that is linked via a secondary edge to *Why* (Figure 1). In the new method, we link *wh*-words and topicalized phrases to their semantic heads, which we believe makes more sense in a dependency grammar.

In addition to the modification of dependency links, the new method uses a much richer set of dependency arc labels than Penn2Malt. The Penn annotation guidelines define a fairly large set of edge labels (referring to grammatical functions or properties of phrases), and most of these are retained in the new format. Penn2Malt only used SBJ, subject, and PRD, predicative complement. In addition, the number of inferred labels (i.e. the labels on the edges that carry no label in the Penn Treebank) has been extended.

Figure 3 shows the dependency tree that is produced by the new procedure. The benefit of retaining the deeper information should be obvious for applications that need to carry out some semantic processing, for example in question answering systems.

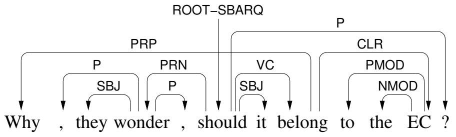

Figure 3: Dependency tree by the new procedure.

The next subsections detail the modifications of the previous methods.

3.1 Heuristically Deepening Noun Phrases

As a preprocessing step, the conversion method uses a few heuristic rules to add internal structure to some noun phrases. This is because a large number of noun phrases with a complex internal structure are annotated using a completely flat structure in the Penn Treebank. An extreme example is *other small apparel makers, button suppliers, trucking firms and fabric houses*. The main reasons for this are probably practical; it saves annotation time, and the internal structure may not be entirely clear to the manual annotators unless they are domain experts. However, the flat structure is very unappealing when the phrase is converted to a dependency structure, since this makes all words in the noun phrase direct dependents of the head word.

We used the following heuristics:

- Certain adverbs (such as *quite* or *too*) are joined with a consecutive adjective into an ADJP.
- Some common words in coordinated NPs (such as *& Co* and *and Sons*) provide a clue to how to bracket these coordinations.
- If there are two words with identical part-of-speech tags around a conjunction, they are assumed to be coordinated, such as in *a small and venomous snake*.

3.2 Head Rule Modifications

The fundamental task in a constituent-to-dependency conversion system is to find the head of each phrase, which is needed in order to create the dependency links. For the most part, we followed the earlier approach by using a set of head percolation rules based on the phrase type, but our rules also made use of the context of the phrases and of grammatical functions. Table 1 shows the complete set of rules. In the table, NP−ε means NP with no function tag, ⋆⋆ means any phrase, and ⋆−PRD means any phrase with a PRD function tag. The following subsections list the modifications of the rules used by Yamada and Matsumoto (2003).

ADJP	←	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	→	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	→	CC RB IN
FRAG	→	(NN⋆ | NP) W⋆ SBAR (PP | IN) (ADJP | JJ) ADVP RB
INTJ	←	⋆⋆
LST	→	LS :
NAC	←	NN⋆ NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
NP, NX	←	(NN⋆ | NX) JJR CD JJ JJS RB QP NP−ε NP
PP, WHPP	→	(first non-punctuation after preposition)
PRN	→	(first non-punctuation)
PRT	→	RP
QP	←	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	→	VP NP ADVP ADJP PP
S	←	VP ⋆−PRD S SBAR ADJP UCP NP
SBAR	←	S SQ SINV SBAR FRAG IN DT
SBARQ	←	SQ S SINV SBARQ FRAG
SINV	←	VBZ VBD VBP VB MD VP ⋆−PRD S SINV ADJP NP
SQ	←	VBZ VBD VBP VB MD ⋆−PRD VP SQ
UCP	→	⋆⋆
VP	→	VBD VBN MD VBZ VB VBG VBP VP ⋆−PRD ADJP NN NNS NP
WHADJP	←	CC WRB JJ ADJP
WHADVP	→	CC WRB
WHNP	←	NN⋆ WDT WP WP$ WHADJP WHPP WHNP
X	→	⋆⋆

Table 1: Head percolation rules.

Coordinated Phrases. The method of Yamada and Matsumoto (2003) analyzed coordinations inconsistently, although Collins (1999) had special rules for such constructions. In the new procedure, the leftmost conjunct is consistently regarded as the head of a coordinated structure, and all other conjuncts and conjunctions as children of the first conjunct. There is a considerable amount of literature on how to represent coordinations in dependency grammars. Treating the leftmost conjunct as the head introduces ambiguities when modifiers attach to the left. To have an unambiguous representation, the coordination should be represented using the conjunction as the head, but this is usually not preferred since it makes parsing more difficult.

PPs, Subordinate and Relative Clauses. In prepositional phrases, including *wh*-phrases such as *in which*, the preposition itself is regarded as a case marker and treated as a dependent. The same is true for other "linking words" such as subordinating conjunctions and relative pronouns.

Noun Phrases. For noun phrases, NX phrases (incomplete NPs) are moved to the highest priority. Similarly to the treatment of PPs above, possessive markers are regarded as dependents of the preceding noun. When trying to set a child NP as the head of an NP, the new conversion procedure skips NPs having a function tag (for instance, to avoid setting *tomorrow* as the head of *the meeting tomorrow*). In WHNP phrases (such as *what cat*), the noun instead of the *wh*-word is considered head.

Main Clauses (S, SQ, and SINV). In some rare cases, a main clause may lack a verb or a verb phrase. In those cases, we look for a constituent with a PRD edge label.

3.3 Modification of Arc Labeling Rules

3.3.1 Grammatical Functions from Penn

In addition to phrase labels such as NP and VP, Penn Treebank II uses a set of 21 property labels such as subject, SBJ, location, LOC, or manner, MNR. The properties may be combined, such as LOC-PRD-TPC. Of these labels, all were used to label dependency relations except four which reflect a structural property rather than a grammatical function: HLN (headline), TTL (title), NOM (non-NP acting as a nominal), and TPC (topicalization). The final one, topicalization, represents a property of a phrase that is arguably more semantically relevant than the three others, e.g. when analyzing the rhetorical structure. However, we think that this property is independent from grammatical functions – an object is an object whether fronted or not – and it is probably not relevant to a dependency grammar. For the treatment of the CLF (cleft) tag, which is also a structural property, see Sect 3.3.3.

Regarding a few of the function tags from Penn, we introduced minor modifications. The adverbial tag, ADV, was extended to all unmarked ADVP and PP nodes in verb phrases. According to Penn annotation conventions, ADV is implicit in these cases. The logical subject in passive clause tag, LGS, was moved to the edge between the verb phrase and *by*, rather than the edge between *by* and the noun phrase.

3.3.2 Inferred Labels

Most of the edges in the Treebank have no label. For these edges, we used heuristics to infer a suitable function tag. These rules are to a large extent based on corresponding rules in PENN2MALT.

The treatment of objects is somewhat different from previous approaches: we included clause complements (SBAR and S) into this category, whereas PENN2MALT includes NPs only. To filter out some frequent annotation errors (SBARs which should carry an edge label), we excluded SBARs starting with *as*, *for*, *since*, or *with*. Arguably, the clause complements should not use the same label as noun phrase objects. On the other hand, it is quite intuitive that the same label is used in *I told him that...* as in *I told him a message*.

In addition, we used a distinction between direct objects (OBJ) and indirect objects (IOBJ). Adding the IOBJ labels is not problematic if there is more than one object, in which case the IOBJ label is assigned to the first of them. However, if we make a distinction between direct and indirect object, it is not clear that there won't occur cases where there is only a single object, but that object should have an IOBJ function tag (such as in *Tell me!*). To have an idea of the number of such cases, we inspected a large set of instances of the verbs *give*, *tell*, and *provide*. Fortunately, the Treebank annotates most of those cases with an empty node to denote a missing object, although there are a few annotation errors that make the rule fail.

The function tag on the root token was used to express the type of sentence. We used four root labels: ROOT-S when the root constituent was S or SINV, ROOT-SBARQ and ROOT-SQ for SBARQ and SQ respectively, and ROOT-FRAG for everything else.

Algorithm 1 shows the complete set of rules that were used to assign labels to the edges that were not labeled by the Penn annotators.

Algorithm 1 Rules to label unlabeled arcs

```
let $c$ be a token, $C$ the highest phrase that $c$ is the head of,
and $P$ the parent of $C$
returns The label on the dependency arc from $c$ to its parent

if $C$ is the root node
    if $C$ is S or SINV return ROOT-S
    if $C$ is SQ return ROOT-SQ
    if $C$ is SBARQ return ROOT-SBARQ
    else return ROOT-FRAG
else
    if $C$ is the first of more than one object return IOBJ
    if $C$ is an object return OBJ
    if $C$ is PRN return PRN
    if $c$ is punctuation return P
    if $C$ is coordinated with $P$ return COORD
    if $C$ is PP, ADVP, or SBAR and $P$ is VP return ADV
    if $C$ is PRT and $P$ is VP return PRT
    if $C$ is VP and $P$ is VP, SQ, or SINV return VC
    if $P$ is VP, S, SBAR, SBARQ, SINV, or SQ return VMOD
    if $P$ is NP, NX, NAC, or WHNP return NMOD
    if $P$ is ADJP, ADVP, WHADJP, or WHADVP return AMOD
    if $P$ is PP or WHPP return PMOD
    else return DEP
end if
```

3.3.3 Structural Labels

Although it is preferable that the dependency relations reflect function rather than structure, structural labels were still needed for a proper representation of a small set of complex constructions. We used three such labels: EXP (expletive), CLF (cleft), and GAP (gapping).

Expletive constructions and cleft sentences are rhetorical transformations that usually result in a fronted *it*. Although superficially similar, expletives and clefts are handled rather differently in the Penn conventions. In an expletive construction, the referent S node is linked via a secondary edge to the preceding *it*, while for clefts the main clause carries the function tag CLF and the referent is unlabeled. In the converted format, these constructions were treated similarly: we attached the referent to the main verb and put the CLF or EXP label on that link. Figures 4 and 5 show examples of an expletive and a cleft, respectively, and their corresponding representations as dependency trees.

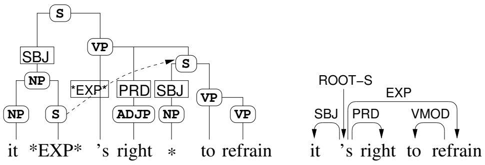

Figure 4: An expletive construction and its dependency representation.

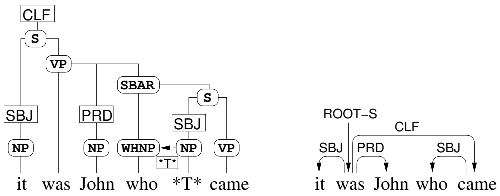

Figure 5: A cleft sentence and its dependency representation.

The phenomenon of gapping, i.e. when some part of a coordinated structure is ellipsed, is difficult to handle for any grammatical formalism, and a number of idiosyncratic solutions have been proposed. The approach used in Penn Treebank II is based on "templates." A coordinated structure with ellipsed constituents is assumed to be structurally identical to the first, and secondary edges (=) are used to identify corresponding constituents. In the dependency representation, we used the secondary edges as dependency links. Figure 6 shows an example of a constituent tree with gapping, and Figure 7 its corresponding dependency tree.

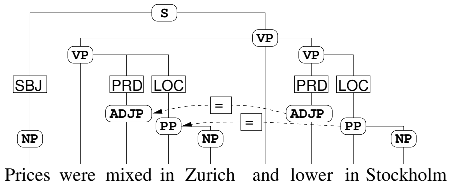

Figure 6: Example of gapping in the Penn Treebank.

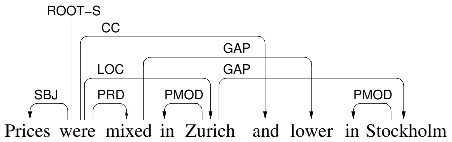

Figure 7: Dependency representation of gapping.

3.4 Relinking of Secondary Edges

Penn Treebank II defines seven kinds of secondary edges, which are listed in Table 2 along with their frequencies in WSJ sections 2–21 in the Treebank.

In many cases, the secondary edge represents a "deep governor", and is thus more useful as a dependency arc than the constituent attachment. In those cases, we relinked the heads of the constituents

Type	Description	#
T	Trace of *wh* and topicalization	15943
*	Other trace	18398
ICH	Discontinuous constituent	1000
RNR	Right node raising	345
=	Gapping	599
EXP	Expletive	557
PPA	Permanent predictable ambiguity	20

Table 2: Secondary edges in the Penn Treebank.

pointed to by the secondary edges. This was done for all *T* and *ICH* edges, unless the relinking causes the dependency graph to become cyclic (such as the link between the empty node and the root node in Figure 1). For right node raising, *RNR*, as for instance in *a U.S. and a Soviet naval vessel*, there are usually two secondary edges, of which only the first one is used. The treatment of the *EXP* and = links was described previously in Sect. 3.3.3.

The constituents pointed to by the "other trace" edges, of which traces of object movement in passive clauses seem to be the most frequent, could not be relinked since their original constituent attachments in most cases seem to be more meaningful as the dependency relation. For instance, we think the subject of a passive clause should not be relinked as an object of the passive verb. However, if the formalism were extended to allow for multiple heads, it could be useful to include those links as well.

The *PPA* (permanent predictable ambiguity) edges refer to cases where there is a structural ambiguity that cannot be resolved by the annotator, such as in *I saw a man with a telescope*. These links were not used in the conversion.

The relinking of constituents makes some trees *nonprojective*, i.e. the dependency tree cannot be drawn without crossing links. An example of this can be seen in Figure 3. In WSJ sections 2–21, the number of resulting nonprojective sentences was 2459 out of 39832, that is 6.17% of the sentences.

4 Experiments

4.1 Impact on Parsing Performance

The new format introduces more complexity in the dependency trees and and a practical issue is to determine how "parsable" they are. For instance, nonprojective trees makes parsing more complicated for some dependency parsers. To quantify this, we trained and evaluated two statistical dependency parsers on the new treebank.

MALTPARSER (Nivre et al., 2006) is based on a greedy parsing procedure that builds a parse tree incrementally while proceeding through the sentence one token at a time. By using a greedy strategy, a rich history-based feature set for the SVM classifier that selects the actions can be used. The parser produces projective trees only, but can handle nonprojectivity if a preprocessing step is used before training and a postprocessing step after parsing ("pseudo-projective parsing").

MSTPARSER (McDonald and Pereira, 2006) predicts a parse tree by maximizing a scoring function over the space of all possible parse trees. The scoring function is a weighted sum of features of single links or, if the "second-order" feature set is used, pairs of adjacent links. The parser can handle nonprojectivity, although the search then becomes NP-hard and has to be approximated.

Following convention, we trained the parsers on sections 2–21 of the WSJ part of the treebank. The training step took a few hours for MALTPARSER using a 64-bit AMD processor running at 2.2 GHz and roughly two days for MSTPARSER using a 32-bit Intel processor at 3.0 GHz.

To test the parsers, we ran the parser on Section 23 of the treebank and measured the labeled and unlabeled accuracy excluding punctuation. The gold-standard part-of-speech tags were used. Table 3 shows the results of the evaluation. For the new format, the relative increase in the number of errors is shown in brackets.

As can be expected, the new format is more difficult for parsers. For the labeled accuracy, this can partly be attributed to the richer set of function tags. For instance, PENN2MALT does not distinguish between temporal and locative adjuncts, but labels them all as verb modifiers. The difference in unlabeled accuracy is probably partly due to the fact that links can now be nonprojective, although this does not explain the whole difference. In addition, the feature sets used by the parsers may be suboptimal for the new way to represent some constructions. For instance, the large decrease in labeled accuracy by MSTPARSER can probably be explained by the fact that "linking words" such as prepositions and subordinating conjunctions do not attach to the verb (see Sect. 3.2). Since the feature set of MST-

	MaltParser		MSTParser	
	Labeled	Unlabeled	Labeled	Unlabeled
Penn2Malt	90.30%	91.36%	92.04%	93.06%
New conversion	87.63% (28%)	90.54% (9%)	86.92% (64%)	91.64% (20%)

Table 3: Parsing accuracy. Relative error increase in brackets.

Parser cannot use features of grandchildren (because of independence assumptions needed to make search tractable), the lexical information about attachment behavior is lost in those cases. This is especially clear for the LGS label, which is assigned by MSTParser to many PPs not starting with *by*. MaltParser, on the other hand, can use this lexical information and performs better for those cases.

Function	R (MST)	P (MST)	R (Malt)	P (Malt)
CLF	0	0	0	0
CLR	50%	46%	70%	51%
COORD	69%	78%	82%	84%
EXP	45%	52%	35%	45%
GAP	16%	50%	20%	45%
IOBJ	54%	89%	63%	87%
LGS	64%	67%	90%	93%
OBJ	91%	78%	90%	90%
PRN	57%	72%	66%	40%
TMP	77%	80%	81%	86%

Table 4: Precision and recall results for a subset of the relations.

Table 4 shows the precision and recall results for the two parsers for some of the dependency relation types added in this conversion. The structural links (cleft, expletive, and gap) are difficult, which is hardly surprising since these phenomena result in long-distance dependencies and are comparatively rare in the Treebank.

4.2 Impact on Semantic Role Classification

To assess the semantic usefulness of the new dependency representation, we created a baseline semantic role labeler that we applied to the FrameNet example corpus (Baker et al., 1998), version 1.3, and compared its accuracy using the old and the new dependency treebanks. All sentences having a verb as target word were used and we tagged them using the MXPOST tagger (Ratnaparkhi, 1996). We then ran MaltParser using the statistical models obtained from both dependency treebanks. As input, the labeler received sentences where the semantic arguments were segmented but not labeled. For each argument that was not null-instantiated, we located the dependency node that was closest to the target in terms of the dependency tree. For most cases, this node was a direct dependent of the target verb.

The baseline role classifier considered the grammatical function of the argument node and assigned the semantic role label that was most frequently associated with this grammatical function for each verb in each frame. For instance, for the verb *tell* in the frame Telling, we mapped the subject to the semantic role Speaker, the direct object to Message, and, for the new format, the indirect object to Addressee.

Method	Accuracy
Penn2Malt	64.3%
New conversion	72.5% (23%)

Table 5: Semantic role classification results.

Table 5 shows the accuracy of this baseline classifier when using the Penn2Malt and the new conversion, respectively. The new format gives a 23% error reduction for classification. Clearly, the improved performance is a result of the increased granularity of the set of edge labels that is gained by using Penn's edge labels and by distinguishing between direct and indirect objects. Table 6 shows an example of this: for the verb *receive* in the frame Receiving, the grammatical functions can express twice as many semantic roles. For this frame, the error reduction was 37.5%.

FN Role	Penn2Malt	New conversion
Countertransfer	∅	∅
Depictive	∅	∅
Donor	VMOD	CLR, DIR
Manner	∅	∅
Means	∅	∅
Mode_of_transfer	∅	MNR
Path	∅	∅
Place	∅	LOC
Purpose_of_donor	∅	∅
Purpose_of_theme	∅	PRP
Recipient	SUB	LGS, SBJ, VMOD
Role	∅	∅
Theme	OBJ	ADV, OBJ
Time	∅	TMP

Table 6: FrameNet semantic roles and their corresponding grammatical functions for the verb *receive* in the frame Receiving.

5 Conclusion and Future Work

This paper presented a new method to convert English constituent structures in the Penn Treebank format into dependency trees. The aim was that the resulting trees should make more sense semantically than those produced by previous approaches. The new procedure relied on the extended representation that is available in the recent versions of the Treebank. The set of arc labels used by previous methods was enriched by using Penn's own set of labels and by creating a set of rules to infer some other.

The new format is structurally more complex; for instance, some sentences now have nonprojective links. This is reflected in the performance of two statistical parsers: the error rate increased by 28% for the best system. It would be interesting to examine in detail which constructions are problematic for the parser, and how complex phenomena such as coordination should be represented for best parsing performance. Possibly, better parsing results could be achieved by first predicting a parse tree in the PENN2MALT style or some other surface-oriented format, and then applying a (possibly statistically trained) transformation to arrive at the richer dependency structure.

A further step could be to extend the dependency structures to allow multiple-headed graphs, for which a practical parsing algorithm was recently proposed (McDonald and Pereira, 2006). This work was restricted to conventional single-headed dependency trees, which might be inadequate in some cases, such as right node raising and verbs of control and raising. Multiple-headed dependency parsing is also relevant for semantic interpretation of parse trees; ideally, all semantic arguments of a predicate verb would be direct dependents of that verb.

Finally, the motivation for this research is that we believe that a semantically oriented dependency structure will make automatic semantic analyses, such as FrameNet-based predicate argument structure analysis, more robust and easier to implement. While we see a large gain in semantic role classification accuracy with a baseline technique using only grammatical functions, it remains to be seen which impact the new formalism has on semantic role labeling in general. A well-designed dependency structure would ideally allow us to get rid of the very sparse and brittle *Path* feature that has been used in most contituent-based semantic role labelers to date.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL'98*.

Michael J. Collins. 1999. Head-driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia.

David M. Magerman. 1994. Natural language parsing as statistical pattern recognition. Ph.D. thesis, Stanford University.

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.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings EACL-2006*.

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 LREC2006*.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer Verlag.

Owen Rambow, Cassandre Creswell, Rachel Szekely, Harriet Tauber, and Marilyn Walker. 2002. A dependency treebank for English. In *Proceedings of LREC2002*.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP-1996*.

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)*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies*.