Robust Constituent-to-Dependency Conversion for English

Jinho D. Choi and Martha Palmer

Institute of Cognitive Science
University of Colorado at Boulder
E-mail: choijd@colorado.edu, mpalmer@colorado.edu

Abstract

This paper suggests a robust way of converting constituent-based trees in the Penn Treebank style into dependency trees for several different English corpora. For English, there already exist conversion tools. However, these tools are often customized enough for a specific corpus that they do not necessarily work as well when applied to different corpora involving newly introduced POS-tags or annotation schemes. The desire to improve conversion portability motivated us to build a new conversion tool that would produce more robust results across different corpora. In particular, we have modified the treatment of head-percolation rules, function tags, coordination, gapping, and empty category mappings. We compare our method with the LTH conversion tool used for the CoNLL'07-09 shared tasks. For our experiments, we use 6 different English corpora from OntoNotes release 4.0. To demonstrate the impact our approach has on parsing, we train and test two state-of-the-art dependency parsers, MaltParser and MSTParser, and our own parser, ClearParser, using converted output from both the LTH tool and our method. Our results show that our method removes certain unnecessary non-projective dependencies and generates fewer unclassified dependencies. All three parsers give higher parsing accuracies on average across these corpora using data generated by our method; especially on semantic dependencies.

1 Introduction

There has been growing interest in statistical dependency parsing. For those who need to parse, in addition to newswire, a large amount of less structured text (e.g., web-blogs or automatic translations), dependency parsing has advantages over constituent-based parsing because it is simple and fast, yet gives useful information (Shen et al. [19], Cui et al. [3]). Moreover, since dependency structure is not constrained by word-order, it is considered to be more domain or language independent than phrase structure.

Most current state-of-art dependency parsers use a supervised learning approach (McDonald et al. [13], Nivre et al. [16]), which usually requires a large amount of annotated data. For English, there are some manually annotated dependency Treebanks available (Rambow et al. [18], Čmejrek et al. [20]); nonetheless, constituent-based Treebanks such as the Penn Treebank (Marcus et al. [11]) are more dominant. It has been shown that these Penn Treebank style constituent-based trees can reliably be converted into dependency trees using heuristics (Johansson and Nugues [9]). The result of such a conversion is that statistical dependency parsers have access to larger amounts of annotated data in dependency structure.

There already exist tools that convert phrase structure to dependency structure. The most popular one is the LTH constituent-to-dependency conversion tool[1] that had been used for the CoNLL'07-09 shared tasks (Hajič et al. [6]) and gave useful results. The LTH tool makes several improvements over its predecessor, Penn2Malt[2]: it adds semantic dependencies extracted from function tags in the Penn Treebank (e.g,. `LOC`, `TMP`; Marcus et al. [10]) and remaps dependencies related to empty categories, producing non-projective dependencies. Although the tool works well in many ways, it is somewhat customized to the Penn Treebank (mainly for the Wall Street Journal corpus), so it does not necessarily work as well when applied to different corpora. We tested the LTH tool on the OntoNotes English data (Hovy et al. [7]). These corpora contain part-of-speech tags not introduced in the original Penn Treebank (e.g., `EDITED`, `META`) and show occasional departures from the original guidelines (e.g., inserting `NML` phrases, separating hyphenated words). Unfortunately, these new formats affect the LTH tool's ability to find correct dependency relations, motivating us to aim for a more resiliant approach.

In this paper we present a robust method for doing constituent-to-dependency conversion across different corpora. In particular, we show improvements due to modifications of head-percolation rules, function tags, coordination, gapping relations, and empty category mappings. For our experiments, we use 6 different English corpora from the latest release of the OntoNotes Treebank. To demonstrate the advantages of our approach for dependency parsing, we train and test two state-of-the-art dependency parsers, MaltParser and MSTParser, and our own parser, ClearParser, using converted output from both the LTH tool and our method. Our results show that our method removes certain unnecessary non-projective dependencies and generates fewer unclassified dependencies. Moreover, all three parsers give higher parsing accuracies on average across these corpora using data generated by our method; especially on semantic dependencies. The improvement in parsing accuracy is even more significant when parsing models are tested on corpora different from their training corpora, leading us to believe that our method also gives more robust results across different corpora.

[1]The LTH tool: `http://nlp.cs.lth.se/software/treebank_converter/`

[2]Penn2Malt: `http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html`

2 Improved constituent-to-dependency conversion

Our work was inspired by Johansson [8, Chap. 4], which gives descriptive explanations about how the LTH tool converts Penn Treebank style constituent-based trees to CoNLL style dependency trees. We carefully followed their steps and modified certain heuristics to generate more robust output. This section describes some of the key changes we made to Johansson's approach.

2.1 Head-percolation rules

ADJP	r	JJ*|VB*|NN*|ADJP;IN;RB|ADVP;CD|QP;FW|NP;*
ADVP	r	VB*;RB|JJ*;RB+;ADJP;ADVP;QP;IN;NN;CD;RP;NP;*
CONJP	l	CC;TO;IN;VB;*
EDITED	r	VB*|VP;NN*|PRP|NP;IN|PP;S*;*
FRAG	l	NN*|NP;W*;S;SBAR;IN|PP;JJ|ADJP;RB|ADVP;*
INTJ	l	VB;NN*;UH;INTJ;*
LST	l	LS;NN;CD;*
META	r	VP;NP;*
NAC	r	NN*;NP;S;SBAR;*
NML	r	NN*|NML;CD|NP|QP|JJ*|VB*;*
NP	r	NN*|NML;NX;PRP;FW;CD;NP|QP|JJ*|VB*;ADJP;S;SBAR;*
NX	r	NN*;NX;NP;*
PP	l	TO;IN;VBG|VBN;RP;PP;NN*;JJ;RB;*
PRN	r	*
PRT	l	RP;PRT;*
QP	l	JJR|RBR;JJS|RBS;CD;NN*;PDT|DT;ADVP;JJ;*
RRC	l	VBG|VBN;VP;NP|NN*;ADJP;ADVP;PP;*
S	r	TO;MD;VB*;VP;*-SBJ;*-TPC;*-PRD;S|SINV|S*Q;SBAR;NP;PP;*
SBAR	r	IN|TO;DT;MD;VB*;VP;*-PRD;S|SINV|S*Q;SBAR;*
SBARQ	r	MD;VB*;VP;S*Q;S|SINV;*
SINV	r	MD;VB*;VP;*-SBJ;*-TPC;*-PRD;S|SINV;NP;*
SQ	r	MD;VB*;VP;*-PRD;SQ;S;*
UCP	l	*
VP	l	TO;MD;VB*;VP;*-SBJ;*-TPC;*-PRD;NN;NNS;NP;QP;JJ*;ADJP;*
WHADJP	r	JJ*|VBN|VBG;ADJP;*
WHADVP	l	WRB;WHADVP;WDT;RB;*
WHNP	r	NN*|NML;CD;VBG;NP;JJ*;WP;QP;WHNP;WHADJP;*
WHPP	l	IN|TO;*
X	r	*

Table 1: New head-percolation rules. l/r implies to look for the leftmost/rightmost item. */+ implies 0/1 or more characters and *-TAG implies any POS tag with the function tag. | implies a logical OR and ; is a delimiter between POS tags. Each rule gives higher precedence to the left (e.g., TO takes the highest precedence in PP).

We use head-percolation rules (from now on, headrules) to find the head of each constituent in phrase structure. Although there are other headrules available (Yamada and Matsumoto [21]), we designed our own (Table 1) for two reasons. First, previous rules could not handle POS tags not included in the original Penn Treebank (e.g., EDITED, NML). Second, we found it useful to make more use of function

tags; Johansson used one function tag for his rules, PRD (predicative), whereas we used two additional tags, SBJ (subject) and TPC (topic) (e.g., Figure 1). Furthermore, we made minor modifications to some rules. For example, we changed the rule for ADJP (adjective phrase) such that adjectives now get higher priority than nouns.

2.2 Small clauses

Figure 1 shows a constituent-based tree (left) with a small clause, *us happy*, a dependency tree generated by the LTH tool (right-top), and one generated by our approach (right-bottom). According to our headrules, NP-SBJ becomes the head of ADJP-PRD in S-1. However, the LTH tool treats ADJP-PRD as an object predicative (OPRD) of VBP. Both approaches are valid; we take this approach because we find it easier to integrate this structure with a semantic corpus like PropBank (Palmer et al. [17]): *us happy* is annotated as a single argument of *made* in the PropBank, and by making *happy* a child of *us*, we can simply treat the subtree of *us* as an argument of *made* (ARG_1), whereas the treatment gets more complicated when *happy* is also a child of *made*.

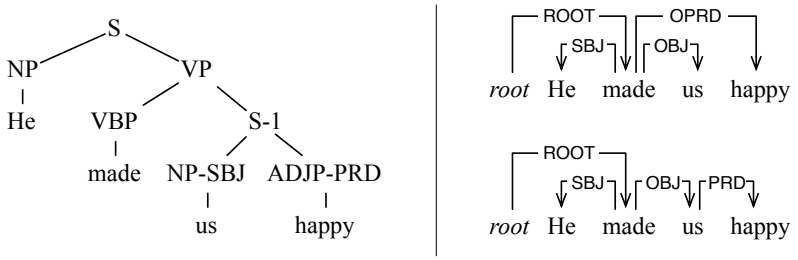

Figure 1: Small clause example.

2.3 Function tags

We use 14 function tags to generate dependency labels. Some of them are joined together (e.g., LOC-TMP); the LTH tool converts each joined function tag into a single dependency label. However, most statistical parsers do not often find joined tags correctly (cf. Table 9), so it may be better to select just one of the tags from the joined tag pair. For example, if LOC and TMP are joined (LOC-TMP), we keep only LOC as a dependency label.[3] We follow the precedence table described below when choosing which tag to keep. There are cases where that the choice becomes difficult; however, such cases are rare enough that they can be ignored.

```
DTV|EXT|LGS|SBJ > LOC > BNF|DIR|MNR|PRP|TMP > SEZ|VOC > PRD > ADV
IGNORE ::= CLF|CLR|ETC|HLN|IMP|NOM|PUT|TPC|TTL|UNF
```

Table 2: Function tag precedences.

[3]In a sentence, [ADVP There] [VP goes [PP *ICH*-1]] [NP all your payments] [PP-1 down in the toilet], [ADVP There] is marked with a joined tag, LOC-TMP.

Table 2 shows how we set the precedences for selecting a member of a joined function tag. For example, SBJ takes precedence over LOC, which again takes precedence over MNR and so on. IGNORE shows a list of function tags that we do not use as dependency labels. There are several reasons for this; mainly, we keep only tags with more semantic values, which usually appear as modifiers in the PropBank.

2.4 Coordination

We take a right branching approach for coordination (the left conjunct becomes the head of the conjunction, which becomes the head of the right conjunct). It sometimes gets hard to decide whether or not a phrase contains coordination. We consider a phrase contains coordination if it is tagged as an unlike coordinated phrase (UCP), if it contains a child annotated with a function tag ETC (et cetera), or if it contains at least one conjunction (CC) or a conjunction phrase (CONJP). Even if there is a conjunction, if either the left or the right conjunct does not appear within the same phrase, we do not consider there to be a coordination.

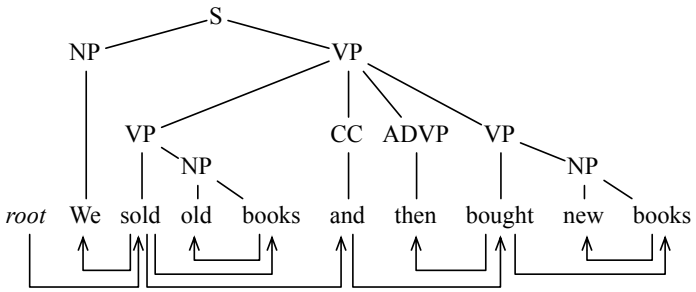

Figure 2: Coordination example.

Within a coordinated phrase, we use commas and semicolons as separators (similar to Johansson's). In addition to Johansson's approach, we apply the following heuristics to check if the left and right conjuncts have the same phrasal type. SKIP shows a list of POS tags that can be skipped to find the correct conjuncts (e.g., ADVP in Figure 2).

```
NounLike  ::= NN*|PRP|NML|NP|WHNP|*-NOM
AdjLike   ::= JJ*|ADJP
WhAdvLike ::= WHADVP|WRB|WHPP|IN
SKIP      ::= PRN|INTJ|EDITED|META|CODE|ADVP|SBAR

if ((parent.pos == UCP) || (left.pos == right.pos) || (left.tag == ETC) ||
    (left.pos == NounLike && right.pos == NounLike) ||
    (left.pos == AdjLike  && right.pos == AdjLike) ||
    (parent.pos == WHADVP && (left.pos == WhAdvLike && right.pos == WhAdvLike))
     return true;
else return false;
```

2.5 Gapping relations

Most statistical parsers perform poorly on gapping relations because it is hard to distinguish them from coordination, and they do not appear frequently enough to be trained on. Johansson's approach usually generates correct dependencies for gapping relations; however, it sometimes produces a very flat structure with many long-distance dependencies. The top tree in Figure 3 shows how gapping relations are handled by the LTH tool.

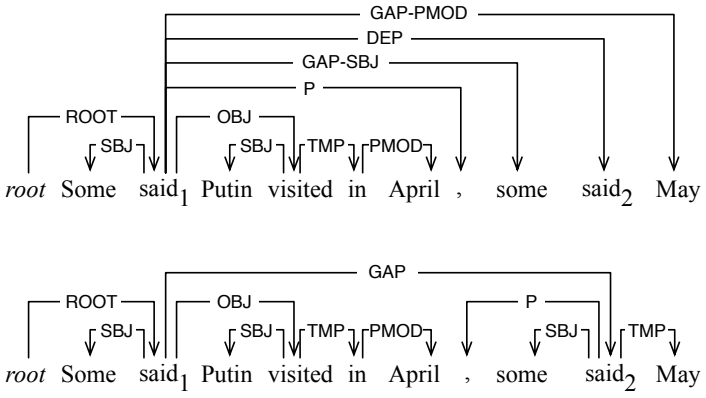

Figure 3: Gapping relation example.

As illustrated, *comma*, *some*, $said_2$, and *May* become children of $said_1$. Although this is an accurate representation, automatic parsers almost never find these dependencies correctly. The bottom tree in Figure 3 shows our approach of handling gapping relations. In our case, *comma*, *some*, and *May* become children of $said_2$, which then becomes a child of $said_1$. This way, parsers can easily learn the local information, yet still can recover the original representation by using the gapping relation (`GAP`).

2.6 Empty category mappings

Our approach to empty category mappings is similar to Johansson's, except for our handling of `*RNR*` (right node raising; see Figure 4). `*RNR*` appears in coordinated phrases such that there are always at least two `*RNR*` nodes mapped to the same antecedent. In this case, we map the antecedent to its closest `*RNR*` as illustrated in Figure 4. This way, we can eliminate many non-projective dependencies without sacrificing loss of semantic interpretation.

3 Experiments

We evaluate the changes we have made to the conversion process by training several parsers and testing them on various corpora.

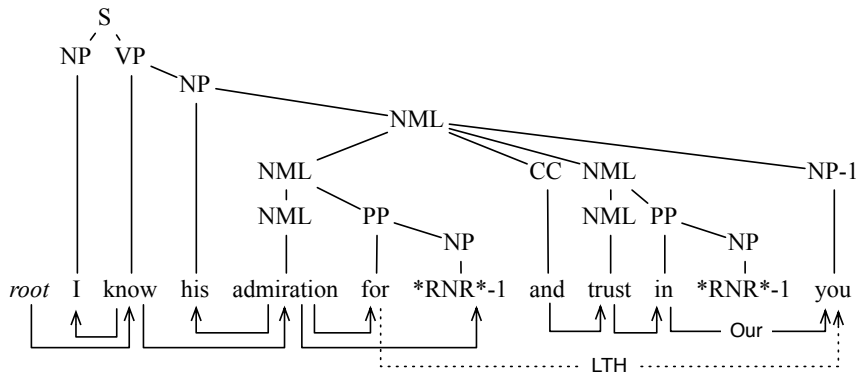

Figure 4: Right node raising (`*RNR*`) example. The solid dependency link labeled as 'Our' shows our way of handling `*RNR*`, and the dashed dependency labeled as 'LTH' shows Johansson's way of handling `*RNR*`. After the conversion, `*RNR*` nodes are dropped from the dependency tree.

3.1 Corpora

For our experiments, we use OntoNotes release 4.0 (Hovy et al. [7]).[4] We choose, from among all data, six different English corpora: GALE Broadcast Conversation (EBC), GALE Broadcast News (EBN), the Sinorama Chinese Treebank English Translation (SIN), the Xinhua Chinese Treebank English Translation (XIN), GALE Web Text (WEB), and the Wall Street Journal (WSJ). EBC and EBN contain broadcast conversations and news from various sources like CNN, NBC, etc. SIN contains English translations of the Sinorama magazine from Taiwan. XIN contains English translations of the Xinhua newswire from Beijing. WEB contains documents from web-blogs and newsgroups. Finally, WSJ contains non-financial news from the Wall Street Journal.[5]

	EBC	EBN	SIN	XIN	WEB	WSJ	ALL
Train	14,873	11,968	7,259	3,156	13,419	12,311	62,986
Eval.	1,291	1,339	1,066	1,296	1,172	1,381	7,545
Avg.	15.21	19.49	23.36	29.77	22.01	24.03	21.02

Table 3: Corpora distributions (in # of sentences). Avg.-row shows the average sentence length of each corpus (sentence length = # of words in sentence).

Table 3 shows how the corpora are divided into training and evaluation sets. For WSJ, we use Sections 0-21 for training, Section 24 for development, and Section 23 for evaluation. We do not separate development sets for the other corpora because

[4] At this moment, the OntoNotes release 4.0 is not available to the public, but it will soon to be available through the Linguistic Data Consortium (LDC).

[5] WSJ from OntoNotes contains only a subset of the entire Penn Treebank, extracting non-financial news only. Thus, the data is not bias from financial news.

all parsing models are optimized only for WSJ. ALL-column shows the total number of sentences when all corpora are combined.

3.2 Converting phrase structure to dependency structure

All corpora are annotated in Penn Treebank phrase structure style (Marcus et al. [11]). For each corpus, we generate two sets of dependency trees, one converted by the LTH tool, and the other converted by our approach. For the LTH tool, we mostly use their default settings, but exclude the GLARF labels (e.g., NAME, POSTHON, SUFFIX, TITLE; Meyers et al. [14]) and the APPO label for appositions. However, we include the QMOD label for quantifier phrases as we include this label in our approach. During conversion, we discard sentences of length 1 to avoid a bias to such a trivial case.

	EBC	EBN	SIN	XIN	WEB	WSJ	ALL
LTH- dep	1.44	0.81	0.70	0.29	0.95	0.51	0.82
Our - dep	1.29	0.73	0.69	0.21	0.83	0.46	**0.73**
LTH- sen	11.14	8.66	8.47	5.30	11.29	7.27	9.27
Our - sen	9.19	7.39	8.22	3.75	9.02	6.24	**7.78**

Table 4: Distributions of non-projective dependencies (in %). 'LTH' indicates output from the LTH tool, and 'Our' indicates output from our approach. The top two rows show % of non-projective dependencies among all dependencies. The bottom two rows show % of dependency trees containing at least one non-projective dependency. The numbers do not account for non-projective dependencies caused by punctuation.

Table 4 shows the distribution of non-projective dependencies in each corpus. Our approach generates fewer non-projective dependencies because of our new method of handling *RNR* nodes (cf. Section 2.6).

	EBC	EBN	SIN	XIN	WEB	WSJ	ALL
LTH	4.77	1.51	1.16	1.63	1.93	1.93	**2.20**
Our	0.86	0.57	0.33	0.44	1.03	0.25	**0.60**

Table 5: Distributions of unclassified dependencies (in %).

Table 5 shows the distribution of unclassified dependencies (labeled as DEP). Some of these dependencies may not be errors; they can be rather ambiguous in nature. However, reducing the number of unclassified dependencies as much as possible is preferred because they can appear as noise during training. The numbers show that our approach reduces the percentage of unclassified dependencies from 2.2% to 0.6%, a reduction of 72.7%.

3.3 State-of-art dependency parsers

To show the impact each conversion method has on dependency parsing, we train and test two state-of-art dependency parsers, MaltParser (Nivre et al. [16]) and

MSTParser (McDonald et al. [13]), and our own parser, ClearParser. First, we train all parsers on each corpus and test on the same corpus. Then, we train all parsers on WSJ, and test on the other corpora. In addition, we test all parsers on WSJ using models trained on EBN. No extensive optimization is made for any parser, so the performances of these parsers are expected to improve with further optimizations.

For MaltParser, we choose Nivre's *swap* algorithm for parsing (Nivre [15]), and LibLinear multi-class SVM for learning (Fan et al. [5]).[6] For MSTParser, we choose Chu-Liu-Edmonds' algorithm for parsing (McDonald et al. [13]), and the Margin Infused Relaxed algorithm (MIRA) for learning (Mcdonald and Pereira [12]). For ClearParser, we choose Choi-Nicolov's approach to Nivre's *list-based* algorithm for parsing (Choi and Nicolov [2]), and LibLinear L2-L1 SVM for learning.

3.4 Accuracy comparisons

3.4.1 Overall parsing accuracies

	EBC	EBN	SIN	XIN	WEB	WSJ	ALL
Malt - LTH	82.91	86.38	86.20	84.61	85.10	86.93	85.44
Malt - Our	83.20	86.40	86.03	84.85	85.45	87.40	**85.65**
MST - LTH	81.64	85.47	85.02	84.10	84.05	85.93	84.49
MST - Our	82.54	85.68	85.11	83.85	84.03	86.43	**84.69**
Clear - LTH	83.36	86.32	86.80	85.50	85.53	87.15	85.88
Clear - Our	84.06	86.77	86.55	85.41	85.70	87.58	**86.09**

Table 6: LAS (in %) when trained and tested on the same corpora.

	EBC	EBN	SIN	XIN	WEB	WSJ	ALL
Malt - LTH	74.80	82.40	81.74	79.39	80.42	80.59	80.01
Malt - Our	75.60	83.05	81.81	81.46	80.81	81.17	**80.85***
MST - LTH	76.65	82.45	82.29	80.46	80.64	80.02	80.49
MST - Our	77.20	83.06	82.52	80.88	80.82	81.04	**81.01***
Clear - LTH	76.37	83.16	83.53	81.29	81.83	81.29	81.36
Clear - Our	77.14	84.16	83.66	82.45	82.26	82.32	**82.16***

Table 7: LAS (in %) when trained and tested on different corpora.

Table 6 shows labeled attachment scores (LAS) of each parser when trained and tested on the same corpora. All parsers give higher parsing accuracies on average using data generated by our approach. It is not clear which significance tests are appropriate for our data; the data contains too many dependencies (over 165K) so even a 0.2% improvement becomes statistically significant using the Chi-square test (thus, all improvements made in ALL-column are significant with $p \leq 0.025$).

[6] MaltParser comes with a default feature template designed for LibSVM (Chang and Lin [1]), so we contacted the MaltParser team to get a different feature template for LibLinear.

If we use the Wilcoxon signed-ranks test, pretending that each corpus is an independent sample, these 0.2% improvements are not statistically significant.

Table 7 shows LAS of each parser when trained and tested on different corpora. All parsers use models trained on WSJ except they use models trained on EBN when testing on WSJ. All parsers make statistically significant improvements (by the Wilcoxon signed-ranks test, $p \leq 0.03$) on average using data generated by our approach. This suggests that our method might create more uniform structures across different corpora, improving domain adaptation.

3.4.2 Parsing accuracies on semantic dependencies

Table 8 shows LAS on semantic dependencies (BNF, DIR, EXT, LOC, MNR, PRD, PRP, TMP) using the ALL evaluation set. There are a total of 11,583 and 11,934 semantic dependencies in the LTH and our data, respectively. In terms of F1-score, all parsers again give consistently higher parsing accuracies using data generated by our approach.

	Precision		Recall		F1-score	
	LTH	Clear	LTH	Clear	LTH	Clear
Malt - Same	67.97	**68.68**	62.67	**63.00**	65.21	**65.72**
MST - Same	66.86	**67.67**	60.80	**60.82**	63.69	**64.06**
Clear - Same	69.56	**70.22**	65.62	**65.74**	67.53	**67.91**
Malt - Diff	57.33	**57.83**	55.99	**56.38**	56.65	**57.10**
MST - Diff	57.65	**58.53**	**54.64**	54.49	56.10	**56.44**
Clear - Diff	58.81	**60.22**	58.60	**59.69**	58.70	**59.95**

Table 8: LAS on semantic dependencies using the ALL evaluation set. Same/Diff-rows show results using the same/different corpora for training and testing.

Table 9 shows LAS on joined semantic dependencies (e.g., LOC-PRD) using the LTH-ALL evaluation set (our data does not include those dependencies; Section 2.3). There are a total of 240 of these dependencies. As shown, all parsers give below 50% accuracies on these dependencies.

	Same			Diff		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Malt	56.29	35.42	43.48	45.54	21.25	28.98
MST	58.82	33.33	42.55	60.49	20.42	30.53
Clear	66.43	39.58	49.60	51.49	28.75	36.90

Table 9: LAS on joined semantic dependencies using the LTH-ALL evaluation set.

4 Conclusion and future work

We present a more robust method of doing constituent-to-dependency conversion across different corpora. Our results show that parsers generally perform more accurately using data generated by our approach than by the LTH tool; especially for semantic dependencies. The improvements are even more significant when parsing models are tested on corpora different from their training corpora, leading us to believe that our method generates data that is more suitable for domain adaptation.

In the future, we are planning to apply automatic error detection techniques (Dickinson [4]) to make our data more consistent. We will also generate dependency trees with empty categories for dropped arguments. Finally, we will try to integrate semantic roles from PropBank directly into the dependency trees.

Acknowledgments

We gratefully acknowledge the support of the National Science Foundation Grants CISE-CRI-0551615, Towards a Comprehensive Linguistic Annotation and CISE-CRI 0709167, Collaborative: A Multi-Representational and Multi-Layered Treebank for Hindi/Urdu, and a grant from the Defense Advanced Research Projects Agency (DARPA/IPTO) under the GALE program, DARPA/CMO Contract No. HR0011-06-C-0022, subcontract from BBN, Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[2] Jinho D. Choi and Nicolas Nicolov. K-best, locally pruned, transition-based dependency parsing using robust risk minimization. In *Recent Advances in Natural Language Processing V*, pages 205–216. John Benjamins, 2009.

[3] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of ACM-SIGIR'05*, pages 400–407, 2005.

[4] Markus Dickinson. Correcting dependency annotation errors. In *Proceedings of EACL'09*, pages 193–201, 2009.

[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification journal of machine learning research. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09 Shared Task*, pages 1–18, 2009.

[7] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90 In *Proceedings of HLT-NAACL'06*, pages 57–60, 2006.

[8] Richard Johansson. *Dependency-based Semantic Analysis of Natural-language Text*. PhD thesis, Lund University, 2008.

[9] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA'07*, 2007.

[10] M. Marcus, G. Kim, M. Ann Marcinkiewicz, R. Macintyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*, pages 114–119, 1994.

[11] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19 (2):313–330, 1993.

[12] Ryan Mcdonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL'06*, pages 81–88, 2006.

[13] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP'05*, pages 523–530, 2005.

[14] Adam Meyers, Ralph Grishman, Kosaka Michiko, and Shubin Zhao. Covering treebanks with glarf. In *Proceedings of the EACL'01 Workshop on Sharing Tools and Resources for Research and Education*, 2001.

[15] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP'09*, pages 351–359, 2009.

[16] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC'06*, pages 2216–2219, 2006.

[17] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[18] Owen Rambow, Cassandre Creswell, Rachel Szekely, Harriet Taber, and Marilyn Walker. A dependency treebank for english. In *Proceedings of LREC'02*, 2002.

[19] Libin Shen, Jinxi Xu, and Ralph Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL:HLT'08*, pages 577–585, 2008.

[20] M. Čmejrek, J. Cuřín, and J. Havelka. Prague czech-english dependency treebank: Any hopes for a common annotation scheme? In *HLT-NAACL'04 workshop on Frontiers in Corpus Annotation*, pages 47–54, 2004.

[21] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machine. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT'03)*, pages 195–206, 2003.