The Effect of Higher-Order Dependency Features in Discriminative Phrase-Structure Parsing

Gregory F. Coppola and Mark Steedman
School of Informatics
The University of Edinburgh
g.f.coppola@sms.ed.ac.uk
steedman@inf.ed.ac.uk

Abstract

Higher-order dependency features are known to improve dependency parser accuracy. We investigate the incorporation of such features into a cube decoding *phrase-structure* parser. We find considerable gains in accuracy on the range of standard metrics. What is especially interesting is that we find strong, statistically significant gains on dependency recovery on *out*-of-domain tests (Brown vs. WSJ). This suggests that higher-order dependency features are not simply overfitting the training material.

1 Introduction

Higher-order dependency features encode more complex sub-parts of a dependency tree structure than first-order, bigram head-modifier relationships.[1] The clear trend in dependency parsing has been that the addition of such higher-order features improves parse accuracy (McDonald & Pereira, 2006; Carreras, 2007; Koo & Collins, 2010; Zhang & Nivre, 2011; Zhang & McDonald, 2012). This finding suggests that the same benefits might be observed in phrase-structure parsing. But, this is not necessarily implied. Phrase-structure parsers are generally stronger than dependency parsers (Petrov et al., 2010; Petrov & McDonald, 2012), and make use of more kinds of information. So, it might be that the information modelled by higher-order dependency features adds less of a benefit in the phrase-structure case.

To investigate this issue, we experiment using Huang's (2008) *cube decoding* algorithm. This algorithm allows structured prediction with *non-local features*, as discussed in §2. Collins's (1997) strategy of expanding the phrase-structure parser's dynamic program to incorporate head-modifier dependency information would not scale to the complex kinds of dependencies we will consider. Using Huang's algorithm, we can indeed incorporate arbitrary types of dependency feature, using a single, simple dynamic program.

Compared to the baseline, non-local feature set of Collins (2000) and Charniak & Johnson (2005), we find that higher-order dependencies do in fact tend to improve performance significantly on both dependency and constituency accuracy metrics. Our most interesting finding, though, is that higher-order dependency features show a consistent and unambiguous contribution to the dependency accuracy, both labelled and unlabelled, of our phrase-structure parsers on *out*-of-domain tests (which means, here, trained on WSJ, but tested on BROWN). In fact, the gains are even stronger on out-of-domain tests than on in-domain tests. One might have thought that higher-order dependencies, being rather specific by nature, would tend to pick out only very rare events, and so only serve to over-fit the training material, but this is not what we find. We speculate as to what this might mean in §5.2.

The cube decoding paradigm requires a first-stage parser to prune the output space. For this, we use the generative parser of Petrov et al. (2006). We can use this parser's model score as a feature in our discriminative model at no additional cost. However, doing so conflates the contribution to accuracy of the generative model, on the one hand, and the discriminatively trained, hand-

[1] Examples of first-order and higher-order dependency features are given in §3.2.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 610–616,
Sofia, Bulgaria, August 4-9 2013.

written, features, on the other. Future systems might use the same or a similar feature set to ours, but in an architecture that does not include any generative parser. On the other hand, some systems might indeed incorporate this generative model's score. So, we need to know exactly what the generative model is contributing to the accuracy of a generative-discriminative model combination. Thus, we conduct experiments in sets: in some cases the generative model score is used, and in others it is not used.

Compared to the faster and more psychologically plausible shift-reduce parsers (Zhang & Nivre, 2011; Zhang & Clark, 2011), cube decoding is a computationally expensive method. But, cube decoding provides a relatively exact environment with which to compare different feature sets, has close connections with modern phrase-based machine translation methods (Huang & Chiang, 2007), and produces very accurate parsers. In some cases, one might want to use a slower, but more accurate, parser during the training stage of a semi-supervised parser training strategy. For example, Petrov et al. (2010) have shown that a fast parser (Nivre et al., 2007) can be profitably trained from the output of a slower but more accurate one (Petrov et al., 2006), in a strategy they call *uptraining*.

We make the source code for these experiments available.[2]

2 Phrase-Structure Parsing with Non-Local Features

2.1 Non-Local Features

To decode using exact dynamic programming (i.e., CKY), one must restrict oneself to the use of only *local* features. Local features are those that factor according to the individual rule productions of the parse. For example, a feature indicating the presence of the rule S $\rightarrow$ NP VP is local.[3] But, a feature that indicates that the head word of this S is, e.g., *joined*, is non-local, because the head word of a phrase cannot be determined by looking at a single rule production. To find a phrase's head word (or tag), we must recursively find the head phrase of each local rule production, until we reach a terminal node (or tag node). This recursion would not be allowed in standard CKY. Many discriminative parsers have used only local features (Taskar et al., 2004; Turian et al., 2007; Finkel et al., 2008). However, Huang (2008) shows that the use of non-local features does in fact contribute substantially to parser performance. And, our desire to make heavy use of head-word dependency relations necessitates the use of non-local features.

2.2 Cube Decoding

While the use of non-local features destroys the ability to do exact search, we can still do inexact search using Huang's (2008) *cube decoding* algorithm.[4] A tractable first-stage parser prunes the space of possible parses, and outputs a *forest*, which is a set of rule production instances that can be used to make a parse for the given sentence, and which is significantly pruned compared to the entire space allowed by the grammar. The size of this forest is at most cubic in the length of the sentence (Billot & Lang, 1989), but implicitly represents exponentially many parses. To decode, we fix an beam width of k (an integer). Then, when parsing, we visit each node n in the same bottom-up order we would use for Viterbi decoding, and compute a list of the top k parses to n, according to a global linear model (Collins, 2002), using the trees that have survived the beam at earlier nodes.

2.3 The First-Stage Parser

As noted, we require a first-stage parser to prune the search space.[5] As a by-product of this pruning procedure, we are able to use the model score of the first-stage parser as a feature in our ultimate model at no additional cost. As a first-stage parser, we use Huang et al.'s (2010) implementation of the LA-PCFG parser of Petrov et al. (2006), which uses a generative, latent-variable model.

3 Features

3.1 Phrase-Structure Features

Our phrase-structure feature set is taken from Collins (2000), Charniak & Johnson (2005), and

[2] See `http://gfcoppola.net/code.php`. This software is available for free for non-profit research uses.

[3] A feature indicating that, e.g., the first word dominated by S is *Pierre* is also local, since the words of the sentence are constant across hypothesized parses, and words can be referred to by their position with respect to a given rule production. See Huang (2008) for more details.

[4] This algorithm is closely related to the algorithm for phrase-based machine translation using a language model (Huang & Chiang, 2007).

[5] All work in this paradigm has used a generative parser as the first-stage parser. But, this is arguably a historical accident. We could just as well use a discriminative parser with only local features, like Petrov & Klein (2007a).

Huang (2008). Some features are omitted, with choices made based on the ablation studies of Johnson & Ural (2010). This feature set, which we call Φ_{phrase}, contains the following, mostly non-local, features, which are described and depicted in Charniak & Johnson (2005), Huang (2008), and Johnson & Ural (2010):

- **CoPar** The depth (number of levels) of parallelism between adjacent conjuncts
- **CoParLen** The difference in length between adjacent conjuncts
- **Edges** The words or (part-of-speech) tags on the outside and inside edges of a given XP[6]
- **NGrams** Sub-parts of a given rule production
- **NGramTree** An n-gram of the input sentence, or the tags, along with the minimal tree containing that n-gram
- **HeadTree** A sub-tree containing the path from a word to its maximal projection, along with all siblings of all nodes in that path
- **Heads** Head-modifier bigrams
- **Rule** A single rule production
- **Tag** The tag of a given word
- **Word** The tag of and first XP above a word
- **WProj** The tag of and maximal projection of a word

Heads is a first-order dependency feature.

3.2 Dependency Parsing Features

McDonald et al. (2005) showed that chart-based dependency parsing, based on Eisner's (1996) algorithm, could be successfully approached in a discriminative framework. In this earliest work, each feature function could only refer to a single, bigram head-modifier relationship, e.g., **Modifier**, below. Subsequent work (McDonald & Pereira, 2006; Carreras, 2007; Koo & Collins, 2010) looked at allowing features to access more complex, *higher-order* relationships, including trigram and 4-gram relationships, e.g., all features apart from **Modifier**, below. With the ability to incorporate non-local phrase-structure parse features (Huang, 2008), we can recognize dependency features of arbitrary order (cf. Zhang & McDonald (2012)). Our dependency feature set, which we call Φ_{deps}, contains:

- **Modifier** head and modifier
- **Sibling** head, modifier m, and m's nearest inner sibling
- **Grandchild** head, modifier m, and one of m's modifiers
- **Sibling+Grandchild** head, modifier m, m's nearest inner sibling, and one of m's modifiers
- **Grandchild+Grandsibling** head, modifier m, one of m's modifiers g, and g's inner sibling

These features are insensitive to arc labels in the present experiments, but future work will incorporate arc labels. Each feature class contains more and less lexicalized versions.

3.3 Generative Model Score Feature

Finally, we have a feature set, Φ_{gen}, containing only one feature function. This feature maps a parse to the logarithm of the MAX-RULE-PRODUCT score of that parse according to the LA-PCFG parsing model, which is trained separately. This score has the character of a conditional likelihood for the parse (see Petrov & Klein (2007b)).

4 Training

We have two feature sets Φ_{phrase} and Φ_{deps}, for which we fix weights using parallel stochastic optimization of a structured SVM objective (Collins, 2002; Taskar et al., 2004; Crammer et al., 2006; Martins et al., 2010; McDonald et al., 2010). To the single feature in the set Φ_{gen} (i.e. the generative model score), we give the weight 1.

The combined models, $\Phi_{\text{phrase+deps}}$, $\Phi_{\text{phrase+gen}}$, and $\Phi_{\text{phrase+deps+gen}}$, are then *model combinations* of the first three. The combination weights for these combinations are obtained using Och's (2003) Minimum Error-Rate Training (MERT). The MERT stage helps to avoid feature undertraining (Sutton et al., 2005), and avoids the problem of scaling involved in a model that contains mostly boolean features, but one, real-valued, log-scale feature. Training is conducted in three stages (SVM, MERT, SVM), so that there is no influence of any data outside the given training set (WSJ2-21) on the combination weights.

5 Experiments

5.1 Methods

All models are trained on WSJ2-21, with WSJ22 used to pick the stopping iteration for online

[6]The tags outside of a given XP are approximated using the marginally most likely tags given the parse.

		Test Set					
		WSJ			BROWN		
Type	Model	F_1	UAS	LAS	F_1	UAS	LAS
G	LA-PCFG	90.3	93.7	91.5	85.1	88.7	85.0
D	phrase	91.2	93.9	91.0	86.1	89.4	85.1
	deps	—	93.3	—	—	89.3	—
	phrase+deps	**91.7**	**94.4**	**91.5**	**86.4**	**90.1**	**85.9**
G+D	phrase+gen	92.1	94.7	92.6	87.0	90.0	86.5
	phrase+deps+gen	**92.4**	**94.9**	**92.8**	**87.4**	**90.7**	**87.1**

Table 1: Performance of the various models in cube decoding experiments, on the WSJ test set (in-domain) and the BROWN test set (out-of-domain). G abbreviates *generative*, D abbreviates *discriminative*, and G+D a combination. Some cells are empty because Φ_{deps} features are only sensitive to unlabelled dependencies. Best results in D and G+D conditions appear in bold face.

		Test Set					
Hypothesis		WSJ			BROWN		
Greater	Lesser	F_1	UAS	LAS	F_1	UAS	LAS
phrase+deps	phrase	**.042**	**.029**	**.018**	.140	**.022**	**.009**
phrase+deps	deps	—	**<.001**	—	—	**.012**	—
phrase+gen	phrase	**.013**	**.003**	**<.001**	**.016**	.090	**<.001**
phrase+deps+gen	phrase+gen	**.030**	.122	.151	.059	**.008**	**.020**
phrase+deps+gen	phrase+deps	**.019**	**.020**	**<.001**	**.008**	**.040**	**<.001**

Table 2: Results of statistical significance evaluations of hypotheses of the form *X's accuracy is greater than Y's* on the various test sets and metrics. Bold face indicates $p < .05$.

optimization, as is standard. The test sets are WSJ23 (in-domain test set), and BROWN9 (out-of-domain test set) from the Penn Treebank (Marcus et al., 1993).[7] We evaluate using harmonic mean between labelled bracket recall and precision (EVALB F_1), unlabelled dependency accuracy (UAS), and labelled dependency accuracy (LAS). Dependencies are extracted from full output trees using the algorithm of de Marneffe & Manning (2008). We chose this dependency extractor, firstly, because it is natively meant to be run on the output of phrase-structure parsers, rather than on gold trees with function tags and traces still present, as is, e.g., the *Penn-Converter* of Johansson & Nugues (2007). Also, this is the extractor that was used in a recent shared task (Petrov & McDonald, 2012). We use EVALB and *eval.pl* to calculate scores.

For hypothesis testing, we used the paired bootstrap test recently empirically evaluated in the context of NLP by Berg-Kirkpatrick et al. (2012). This involves drawing b subsamples of size n with replacement from the test set in question, and checking relative performance of the models on the subsample (see the reference). We use $b = 10^6$ and $n = 500$ in all tests.

[7] Following Gildea (2001), the BROWN test set is usually divided into 10 parts. If we start indexing at 0, then the last (test) section has index 9. We received the BROWN data splits from David McClosky, p.c.

5.2 Results

The performance of the models is shown in Table 1, and Table 2 depicts the results of significance tests of differences between key model pairs.

We find that adding in the higher-order dependency feature set, Φ_{deps}, makes a statistically significant improvement in accuracy on most metrics, in most conditions. On the in-domain WSJ test set, we find that $\Phi_{\text{phrase+deps}}$ is significantly better than either of its component parts on all metrics. But, $\Phi_{\text{phrase+deps+gen}}$ is significantly better than $\Phi_{\text{phrase+gen}}$ only on F_1, but not on UAS or LAS. However, on the *out*-of-domain BROWN tests, we find that adding Φ_{deps} always adds considerably, and in a statistically significant way, to both LAS and UAS. That is, not only is $\Phi_{\text{phrase+deps}}$ better at dependency recovery than its component parts, but $\Phi_{\text{phrase+deps+gen}}$ is also considerably bet-

ter on dependency recovery than $\Phi_{\text{phrase+gen}}$, which represents the previous state-of-the-art in this vein of research (Huang, 2008). This result is perhaps counter-intuitive, in the sense that one might have supposed that higher-order dependency features, being highly specific by nature, might only have only served to over-fit the training material. However, this result shows otherwise. Note that the dependency features include various levels of lexicalization. It might be that the more unlexicalized features capture something about the structure of correct parses, that transfers well out-of-domain. Future work should investigate this. And, it of course remains to be seen how this result will transfer to other train-test domain pairs.

To our knowledge, this is the first work to specifically separate the role of the generative model feature from the other features of Collins (2000) and Charniak & Johnson (2005). We note that, even without the Φ_{gen} feature, the discriminative parsing models are very strong, but adding Φ_{gen} nevertheless yields considerable gains. Thus, while a fully discriminative model, perhaps implemented using a shift-reduce algorithm, can be expected to do very well, if the best accuracy is necessary (e.g., in a semi-supervised training strategy), it still seems to pay to use the generative-discriminative model combination. Note that the LAS scores of our models without Φ_{gen} are relatively weak. This is presumably largely because our dependency features are, at present, not sensitive to arc labels, so our results probably underestimate the capability of our general framework with respect to labelled dependency recovery.

Table 3 compares our work with Huang's (2008). Note that our model $\Phi_{\text{phrase+gen}}$ uses essentially the same features as Huang (2008), so the fact that our $\Phi_{\text{phrase+gen}}$ is noticeably more accurate on F_1 is presumably due to the benefits in reduced feature under-training achieved by the MERT combination strategy. Also, our $\Phi_{\text{phrase+deps}}$ model is as accurate as Huang's, without even using the generative model score feature. Table 4 compares our work to McClosky et al.'s (2006) domain adaptation work with the Charniak & Johnson (2005) parser. Their three models shown have been trained on: i) the WSJ (supervised, out-of-domain), ii) the WSJ plus 2.5 million sentences of automatically labelled NANC newswire text (semi-supervised, out-of-domain), and iii) the BROWN corpus (supervised, in-domain). We test on BROWN. We see that our best (WSJ-trained) model is over 2% more accurate (absolute F_1 difference) than the Charniak & Johnson (2005) parser trained on the same data. In fact, our best model is nearly as good as McClosky et al.'s (2006) self-trained, semi-supervised model. Of course, the self-training strategy is orthogonal to the improvements we have made.

Type	Model	WSJ
G+D	Huang (2008)	91.7
D	phrase+deps	91.7
G+D	phrase+gen	92.1
G+D	phrase+deps+gen	**92.4**

Table 3: Comparison of constituency parsing results in the cube decoding framework, on the WSJ test set. On G+D, D, see Table 1.

Parser	Training Data	BROWN F_1
CJ	WSJ	85.2
CJ	WSJ+NANC	87.8
CJ	BROWN	**88.4**
Our Best	WSJ	<u>87.4</u>

Table 4: Comparison of our best model, $\Phi_{\text{phrase+deps+gen}}$, on BROWN, with the Charniak & Johnson (2005) parser, denoted CJ, as reported in McClosky et al. (2006). Underline indicates best trained on WSJ, bold face indicates best overall.

6 Conclusion

We have shown that the addition of higher-order dependency features into a cube decoding phase-structure parser leads to statistically significant gains in accuracy. The most interesting finding is that these gains are clearly observed on out-of-domain tests. This seems to imply that higher-order dependency features do not merely over-fit the training material. Future work should look at other train-test domain pairs, as well as look at exactly which higher-order dependency features are most important to out-of-domain accuracy.

Acknowledgments

This work was supported by the Scottish Informatics and Computer Science Alliance, The University of Edinburgh's School of Informatics, and ERC Advanced Fellowship 249520 GRAMPLUS. We thank Zhongqiang Huang for his extensive help in getting started with his LA-PCFG parser.

References

Berg-Kirkpatrick, T., Burkett, D., & Klein, D. (2012). An empirical investigation of statistical significance in NLP. In *EMNLP*, 995–1005.

Billot, S., & Lang, B. (1989). The structure of shared forests in ambiguous parsing. In *ACL*, 143–151.

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, 957–961.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*, 173–180.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *ACL*, 16–23.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *ICML*, 175–182.

Collins, M. (2002). Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *EMNLP*, 1–8.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *JMLR*, 7, 551–585.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *COLING*, 340–345.

Finkel, J. R., Kleeman, A., & Manning, C. D. (2008). Efficient, feature-based, conditional random field parsing. In *ACL*, 959–967.

Gildea, D. (2001). Corpus variation and parser performance. In *EMNLP*, 167–202.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *ACL*, 586–594.

Huang, L., & Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *ACL*.

Huang, Z., Harper, M., & Petrov, S. (2010). Self-training with products of latent variable grammars. In *EMNLP*, 12–22.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, 105–112.

Johnson, M., & Ural, A. E. (2010). Reranking the Berkeley and Brown parsers. In *HLT-NAACL*, 665–668.

Koo, T., & Collins, M. (2010). Efficient third-order dependency parsers. In *ACL*, 1–11.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 313–330.

de Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 1–8.

Martins, A. F., Gimpel, K., Smith, N. A., Xing, E. P., Figueiredo, M. A., & Aguiar, P. M. (2010). Learning structured classifiers with dual coordinate ascent. Technical report, DTIC Document.

McClosky, D., Charniak, E., & Johnson, M. (2006). Reranking and self-training for parser adaptation. In *ACL*, 337–344.

McDonald, R., & Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *EACL*, 81–88.

McDonald, R. T., Crammer, K., & Pereira, F. C. N. (2005). Online large-margin training of dependency parsers. In *ACL*, 91–98.

McDonald, R. T., Hall, K., & Mann, G. (2010). Distributed training strategies for the structured perceptron. In *HLT-NAACL*, 456–464.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., & Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2), 95–135.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *ACL*, 160–167.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *ACL*, 433–440.

Petrov, S., Chang, P.-C., Ringgaard, M., & Alshawi, H. (2010). Uptraining for accurate deterministic question parsing. In *EMNLP*, 705–713.

Petrov, S., & Klein, D. (2007a). Discriminative log-linear grammars with latent variables. In *NIPS*.

Petrov, S., & Klein, D. (2007b). Improved inference for unlexicalized parsing. In *HLT-NAACL*, 404–411.

Petrov, S., & McDonald, R. (2012). Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Sutton, C., Sindelar, M., & McCallum, A. (2005). Feature bagging: Preventing weight undertraining in structured discriminative learning. In *HLT-NAACL*.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. D. (2004). Max-margin parsing. In *EMNLP*, 1–8.

Turian, J., Wellington, B., & Melamed, I. D. (2007). Scalable discriminative learning for natural language parsing and translation. In *NIPS*, 1409–1416.

Zhang, H., & McDonald, R. (2012). Generalized higher-order dependency parsing with cube pruning. In *EMNLP*, 238–242.

Zhang, Y., & Clark, S. (2011). Shift-reduce CCG parsing. In *ACL*, 683–692.

Zhang, Y., & Nivre, J. (2011). Transition-based dependency parsing with rich non-local features. In *ACL*, 188–293.