Enforcing Structural Diversity in Cube-pruned Dependency Parsing

Hao Zhang   **Ryan McDonald**
Google, Inc.
{haozhang,ryanmcd}@google.com

Abstract

In this paper we extend the cube-pruned dependency parsing framework of Zhang et al. (2012; 2013) by forcing inference to maintain both label and structural ambiguity. The resulting parser achieves state-of-the-art accuracies, in particular on datasets with a large set of dependency labels.

1 Introduction

Dependency parsers assign a syntactic dependency tree to an input sentence (Kübler et al., 2009), as exemplified in Figure 1. Graph-based dependency parsers parameterize models directly over substructures of the tree, including single arcs (McDonald et al., 2005), sibling or grandparent arcs (McDonald and Pereira, 2006; Carreras, 2007) or higher-order substructures (Koo and Collins, 2010; Ma and Zhao, 2012). As the scope of each feature function increases so does parsing complexity, e.g., $o(n^5)$ for fourth-order dependency parsing (Ma and Zhao, 2012). This has led to work on approximate inference, typically via pruning (Bergsma and Cherry, 2010; Rush and Petrov, 2012; He et al., 2013)

Recently, it has been shown that cube-pruning (Chiang, 2007) can efficiently introduce higher-order dependencies in graph-based parsing (Zhang and McDonald, 2012). Cube-pruned dependency parsing runs standard bottom-up chart parsing using the lower-order algorithms. Similar to k-best inference, each chart cell maintains a beam of k-best partial dependency structures. Higher-order features are scored when combining beams during inference. Cube-pruning is an approximation, as the highest scoring tree may fall out of the beam before being fully scored with higher-order features. However, Zhang et al. (2013) observe state-of-the-art results when training accounts for errors that arise due to such approximations.

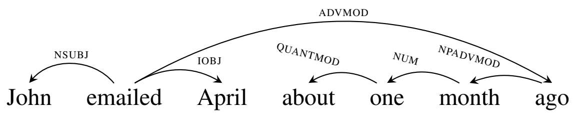

Figure 1: A sample dependency parse.

In this work we extend the cube-pruning framework of Zhang et al. by observing that dependency parsing has two fundamental sources of ambiguity. The first, structural ambiguity, pertains to confusions about the unlabeled structure of the tree, e.g., the classic prepositional phrase attachment problem. The second, label ambiguity, pertains to simple label confusions, e.g., whether a verbal object is direct or indirect.

Distinctions between arc labels are frequently fine-grained and easily confused by parsing models. For example, in the Stanford dependency label set (De Marneffe et al., 2006), the labels TMOD (temporal modifier), NPADVMOD (noun-phrase adverbial modifier), IOBJ (indirect object) and DOBJ (direct object) can all be noun phrases that modify verbs to their right. In the context of cube-pruning, during inference, the system opts to maintain a large amount of label ambiguity at the expense of structural ambiguity. Frequently, the beam stores only label ambiguities and the resulting set of trees have identical unlabeled structure. For example, in Figure 1, the aforementioned label ambiguity around noun objects to the right of the verb (DOBJ vs. IOBJ vs. TMP) could lead one or more of the structural ambiguities falling out of the beam, especially if the beam is small.

To combat this, we introduce a secondary beam for each unique unlabeled structure. That is, we partition the primary (entire) beam into disjoint groups according to the identity of unlabeled structure. By limiting the size of the secondary beam, we restrict label ambiguity and enforce structural diversity within the primary beam. The resulting parser consistently improves on the state-of-the-art parser of Zhang et al. (2013). In

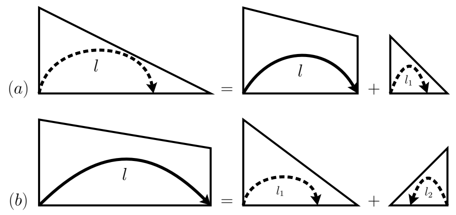

Figure 2: Structures and rules for parsing with the (Eisner, 1996) algorithm. Solid lines show only the construction of right-pointing first-order dependencies. l is the predicted arc label. Dashed lines are the additional sibling modifier signatures in a generalized algorithm, specifically the previous modifier in complete chart items.

particular, data sets with large label sets (and thus a large number of label confusions) typically see the largest jumps in accuracy. Finally, we show that the same result cannot be achieved by simply increasing the size of the beam, but requires explicit enforcing of beam diversity.

2 Structural Diversity in Cube-Pruning

Our starting point is the cube-pruned dependency parsing model of Zhang and McDonald (2012). In that work, as here, inference is simply the Eisner first-order parsing model (Eisner, 1996) shown in Figure 2. In order to score higher-order features, each chart item maintains a list of signatures, which represent subtrees consistent with the chart item. The stored signatures are the relevant portions of the subtrees that will be part of higher-order feature calculations. For example, to score features over adjacent arcs, we might maintain additional signatures, again shown in Figure 2.

The scope of the signature adds asymptotic complexity to parsing. Even for second-order siblings, there will now be $O(n)$ possible signatures per chart item. The result is that parsing complexity increases from $O(n^3)$ to $O(n^5)$. Instead of storing all signatures, Zhang and McDonald (2012) store the current k-best in a beam. This results in approximate inference, as some signatures may fall out of the beam before higher-order features can be scored. This general trick is known as *cube-pruning* and is a common approach to dealing with large hypergraph search spaces in machine translation (Chiang, 2007).

Cube-pruned parsing is analogous to k-best parsing algorithmically. But there is a fundamental difference. In k-best parsing, if two subtrees t_a and t_b belong to the same chart item, with t_a

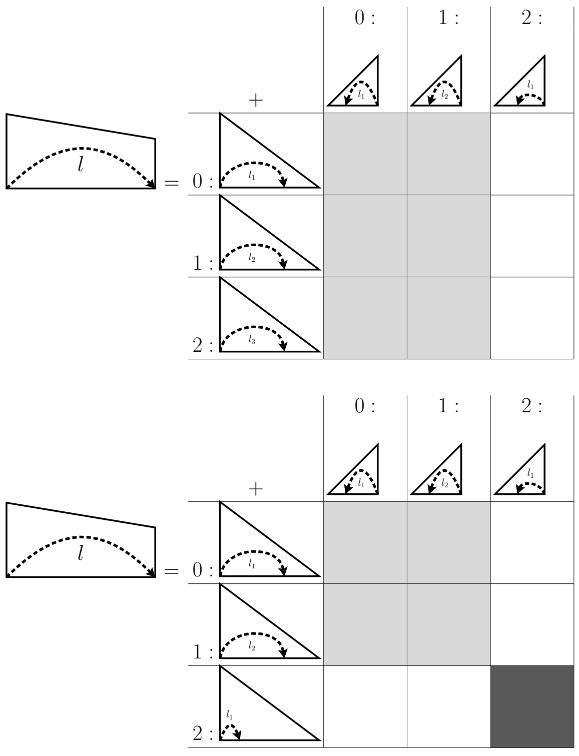

Figure 3: Merging procedure in cube pruning. The bottom shows that enforcing diversity in the k-best lists can give chance to a good structure at $(2, 2)$.

ranking higher than t_b, then an extension of t_a through combing with a subtree t_c from another chart item must also score higher than that of t_b. This property is called the *monotonicity property*. Based on it, k-best parsing merges k-best subtrees in the following way: given two chart items with k-best lists to be combined, it proceeds on the two sorted lists monotonically from beginning to end to generate combinations. Cube pruning follows the merging procedure despite the loss of monotonicity due to the addition of higher-order feature functions over the signatures of the subtrees. The underlying assumption of cube pruning is that the true k-best results are likely in the cross-product space of top-ranked component subtrees. Figure 3 shows that the space is the top-left corner of the grid in the binary branching cases.

As mentioned earlier, the elements in chart item k-best lists are feature signatures of subtrees. We make a distinction between *labeled signatures* and *unlabeled signatures*. As feature functions are defined on sub-graphs of the dependency trees, a feature signature is labeled if and only if feature functions draw information from both the arcs in the sub-graph and the labels on the arcs. Every labeled signature projects to an unlabeled signature that ignores the arc labels.

The motivation for introducing unlabeled signatures for labeled parsing is to enforce structural diversity. Figure 3 illustrates the idea. In the top diagram, there is only one unlabeled signature in one of the two lists. This is likely to happen when there is label ambiguity so that all three labels have similar scores. In such cases, alternative tree structures further down in the list that have the potential to be scored higher when incorporating higher-order features, lose this opportunity due to pruning. By contrast, if we introduce structural diversity by limiting the number of label variants, such alternative structures can come out on top.

More formally, when the feature signatures of the subtrees include arc labels, the cardinality of the set of all possible signatures grows by a polynomial of the size of the label set. This factor has a diluting effect on the diversity of unlabeled signatures within the beam. The larger the label set is, the greater the chance label ambiguity will dominate the beam. Therefore, we introduce a second level of beam specifically for labeled signatures. We call it the *secondary beam*, relative to the *primary beam*, i.e., the entire beam. The secondary beam limits the number of labeled signatures for each unlabeled signature, a projection of labeled signature, while the primary beam limits the total number of labeled signatures. To illustrate this, consider an original primary beam of length b and a secondary beam length of s_b. Let t_i^j represent the i^{th} highest scoring labeled variant of unlabeled structure j. The table below shows a specific example of beam configurations for $b = 4$ for all possible values of s_b. The original beam is the pathological case where all signatures have the same unlabeled projection. When $s_b = 1$, all signatures in the beam now have a different unlabeled projection. When $s_b = 4$, the beam reverts to the original without any structural diversity. Values between balance structural and label diversity.

beam rank	original b=4	$b = 4$ $s_b = 1$	$b = 4$ $s_b = 2$	$b = 4$ $s_b = 3$	$b = 4$ $s_b = 4$
1	t_1^1	t_1^1	t_1^1	t_1^1	t_1^1
2	t_2^1	t_1^2	t_2^1	t_2^1	t_2^1
3	t_3^1	t_1^3	t_1^2	t_3^1	t_3^1
4	t_4^1	t_1^4	t_1^3	t_1^2	t_4^1
· · · · · · · · · · beam cut-off · · · · · · · · · ·					
5	t_1^2	...	...	...	...
6	t_1^3	...	...	...	...
7	t_2^2	...	...	...	...
8	t_2^3	...	...	...	...
9	t_1^4	...	...	...	...

To achieve this in cube pruning, deeper exploration in the merging procedure becomes necessary. In this example, originally the merging procedure stops when t_4^1 has been explored. When $s_b = 1$, the exploration needs to go further from rank 4 to 9. When $s_b = 2$, it needs to go from 4 to 6. When $s_b = 3$, only one more step to rank 5 is necessary. The amount of additional computation depends on the value of s_b, the composition of the incoming k-best lists, and the feature functions which determine feature signatures. To account for this we also compare to baselines systems that simply increase the size of the beam to a comparable run-time.

In our experiments we found that $s_b = b/2$ is typically a good choice. As in most parsing systems, beams are applied consistently during learning and testing because feature weights will be adjusted according to the diversity of the beam.

3 Experiments

We use the cube-pruned dependency parser of Zhang et al. (2013) as our baseline system. To make an apples-to-apples comparison, we use the same online learning algorithm and the same feature templates. The feature templates include first-to-third-order labeled features and valency features. More details of these features are described in Zhang and McDonald (2012). For online learning, we apply the same violation-fixing strategy (so-called single-node max-violation) on MIRA and run 8 epochs of training for all experiments.

For English, we conduct experiments on the commonly-used constituency-to-dependency-converted Penn Treebank data sets. The first one, Penn-YM, was created by the Penn2Malt[1] software. The second one, Penn-S-2.0.5, used the Stanford dependency framework (De Marneffe et al., 2006) by applying version 2.0.5 of the Stanford parser. The third one, Penn-S-3.3.0 was converted by version 3.3.0 of the Stanford parser. The train/dev/test split was standard: sections 2-21 for training; 22 for validation; and 23 for evaluation. Automatic POS tags for Penn-YM and Penn-S-2.0.5 are provided by TurboTagger (Martins et al., 2013) with an accuracy of 97.3% on section 23. For Chinese, we use the CTB-5 dependency treebank which was converted from the original constituent treebank by Zhang and Nivre (2011) and use gold-standard POS tags as is standard.

[1]http://stp.lingfil.uu.se/∼nivre/research/Penn2Malt.html

	Berkeley Parser		*TurboParser*		*Cube-pruned w/o diversity*		*Cube-pruned w/ diversity*	
	UAS	*LAS*	*UAS*	*LAS*	*UAS*	*LAS*	*UAS*	*LAS*
Penn-YM	-	-	93.07	-	93.50	92.41	**93.57**	**92.48**
Penn-S-2.0.5	-	-	92.82	-	93.59	91.17	**93.71**	**91.37**
Penn-S-3.3.0	93.31	91.01	92.20	89.67	92.91	90.52	**93.01**	**90.64**
Penn-S-3.3.0-gold	93.65	92.05	93.56	91.99	94.32	92.90	**94.40**	**93.02**
CTB-5	-	-	-	-	87.78	86.13	**87.96**	**86.34**

Table 1: English and Chinese results for cube pruning dependency parsing with the enforcement of structural diversity. Penn-S and CTB-5 are significant at $p < 0.05$. Penn-S-2.0.5 TurboParser result is from Martins et al. (2013). Following Kong and Smith (2014), we trained our models on Penn-S-3.3.0 with gold POS tags and evaluated with both non-gold (Stanford tagger) and gold tags.

	w/o diversity		*w/ diversity*	
Language(labels)	*UAS*	*LAS*	*UAS*	*LAS*
Czech(82)	**88.36**	**82.16**	88.36	82.02
Swedish(64)	91.62	85.08	**91.85**	**85.26**
Portuguese(55)	92.07	88.30	**92.23**	**88.50**
Danish(53)	**91.88**	**86.95**	91.78	86.93
Hungarian(49)	85.85	81.02	**86.55**	**81.79**
Greek(46)	86.14	78.20	**86.21**	**78.45**
German(46)	**92.03**	89.44	92.01	**89.52**
Catalan(42)	94.58	89.05	**94.91**	**89.54**
Basque(35)	79.59	71.52	**80.14**	**71.94**
Arabic(27)	80.48	69.68	**80.56**	**69.98**
Turkish(26)	76.94	66.80	**77.14**	**67.00**
Slovene(26)	86.01	77.14	**86.27**	**77.44**
Dutch(26)	**83.57**	**80.29**	83.39	80.19
Italian(22)	**87.57**	**83.22**	87.38	82.95
Spanish(21)	87.96	**84.95**	**87.98**	84.79
Bulgarian(19)	**94.02**	**89.87**	93.88	89.63
Japanese(8)	**93.26**	**91.67**	93.16	91.51
Avg	87.76	82.08	87.87	82.20

Table 2: Results for languages from CoNLL 2006/2007 shared tasks. When a language is in both years, the 2006 set is used. Languages are sorted by the number of unique arc labels.

Table 1 shows the main results of the paper. Both the baseline and the new system keep a beam of size 6 for each chart cell. The difference is that the new system enforces structural diversity with the introduction of a secondary beam for label variants. We choose the secondary beam that yields the highest LAS on the development data sets for Penn-YM, Penn-S-2.0.5 and CTB-5. Indeed we observe larger improvements for the data sets with larger label sets. Penn-S-2.0.5 has 49 labels and observes a 0.2% absolute improvement in LAS. Although CTB-5 has a small label set (18), we do see similar improvements for both UAS and LAS. There is a slight improvement for Penn-YM despite the fact that Penn-YM has the most compact label set (12). These results are the highest known in the literature. For the Penn-S-3.3.0 results we can see that our model outperforms TurboPaser and is competitive with the Berkeley constituency parser (Petrov et al., 2006). In particular, if gold tags are assumed, cube-pruning significantly outperforms Berkeley. This suggests that joint tagging and parsing should improve performance further in the non-gold tag setting, as that is a differentiating characteristic of constituency parsers. Table 2 shows the results on the CoNLL 2006/2007 data sets (Buchholz and Marsi, 2006; Nivre et al., 2007). For simplicity, we set the secondary beam to 3 for all. We can see that overall there is an improvement in accuracy and this is highly correlated with the size of the label set.

In order to examine the importance of balancing structural diversity and labeled diversity, we let the size of the secondary beam vary from one to the size of the primary beam. In Table 3, we show the results of all combinations of beam settings of primary beam sizes 4 and 6 for three data sets: Penn-YM, Penn-S-2.0.5, and CTB-5 respectively. In the table, we highlight the best results for each beam size and data set on the development data. For 5 of the total of 6 comparison groups – three languages times two primary beams – the best result is obtained by choosing a secondary beam size that is close to one half the size of the primary beam. Contrasting Table 1 and Table 3, the accuracy improvements are consistent across the development set and the test set for all three data sets.

A reasonable question is whether such improvements could be obtained by simply enlarging the beam in the baseline parser. The bottom row of Table 3 shows the parsing results for the three data sets when the beam is enlarged to 16. On Penn-S-2.0.5, the baseline with beam 16 is at roughly the same speed as the highlighted best system with primary beam 6 and secondary beam 3. On CTB-5, the beam 16 baseline is 30% slower. Table 3 indicates that simply enlarging the beam – relative to parsing speed – does not recover the wins of structural diversity on Penn-S-2.0.5 and CTB-5, though it does reduce the gap on Penn-S-2.0.5. On Penn-YM, the beam 16 baseline is slightly better than the new system, but 90% slower.

primary beam	*secondary beam*	PENN-YM *UAS*	PENN-YM *LAS*	PENN-S-2.0.5 *UAS*	PENN-S-2.0.5 *LAS*	CTB-5 *UAS*	CTB-5 *LAS*
4	1	93.67	92.64	93.65	91.04	87.53	85.85
	2	**93.79**	**92.68**	**93.77**	**91.30**	87.62	85.96
	3	93.80	92.66	93.69	91.23	87.48	85.91
	4	93.75	92.63	93.62	91.11	**87.68**	**86.08**
6	1	93.65	92.46	93.76	91.15	87.72	86.05
	2	93.80	92.69	93.80	91.35	87.61	85.96
	3	93.75	92.64	**93.99**	**91.55**	87.80	86.18
	4	**93.82**	**92.74**	93.84	91.40	**87.91**	**86.28**
	5	93.82	92.71	93.71	91.26	87.75	86.12
	6	93.74	92.61	93.70	91.21	87.66	86.05
16	16	93.87	92.75	93.77	91.35	87.59	85.86

Table 3: Varying the degree of diversity by adjusting the secondary beam for labeled variants, with different primary beams. When the size of the secondary beam is equal to the primary beam, the parser degenerates to not enforcing structural diversity. In the opposite, when the secondary beam is smaller, there is more structural diversity and less label diversity. Results are on development sets.

To better understand the behaviour of structural diversity pruning relative to increasing the beam, we looked at the unlabeled attachment F-score per dependency label in the Penn-S-2.0.5 development set[2]. Table 4 shows the 10 labels with the largest increase in attachment scores for structural diversity pruning relative to standard pruning. Importantly, the biggest wins are primarily for labels in which unlabeled attachment is lower than average (93.99, 8 out of 10). Thus, diversity pruning gets most of its wins on difficult attachment decisions. Indeed, many of the relations represent clausal dependencies that are frequently structurally ambiguous. There are also cases of relatively short dependencies that can be difficult to attach. For instance, *quantmod* dependencies are typically adverbs occurring after verbs that modify quantities to their right. But these can be confused as adverbial modifiers of the verb to the left. These results support our hypothesis that label ambiguity is causing hard attachment decisions to be pruned and that structural diversity can ameliorate this.

Label	*w/o diversity large beam*	*w/ diversity small beam*	*diff*
quantmod	86.65	88.06	1.41
partmod	83.63	85.02	1.39
xcomp	87.76	88.74	0.98
tmod	89.75	90.72	0.97
appos	88.89	89.84	0.95
nsubjpass	92.53	93.31	0.78
complm	94.50	95.15	0.64
advcl	81.10	81.74	0.63
ccomp	82.64	83.17	0.54
number	96.86	97.39	0.53

Table 4: Unlabeled attachment F-score per dependency relation. The top 10 score increases for structural diversity pruning (beam 6 and label beam of 3) over basic pruning (beam 16) are shown. Only labels with more than 100 instances in the development data are considered.

4 Discussion

Keeping multiple beams in approximate search has been explored in the past. In machine translation, multiple beams are used to prune translation hypotheses at different levels of granularity (Zens and Ney, 2008). However, the focus is improving the speed of translation decoder rather than improving translation quality through enforcement of hypothesis diversity. In parsing, Bohnet and Nivre (2012) and Bohnet et al. (2013) propose a model for joint morphological analysis, part-of-speech tagging and dependency parsing using a left-to-right beam. With a single beam, token level ambiguities (morphology and tags) dominate and dependency level ambiguity is suppressed. This is addressed by essentially keeping two beams. The first forces every tree to be different at the dependency level and the second stores the remaining highest scoring options, which can include outputs that differ only at the token level.

The present work looks at beam diversity in graph-based dependency parsing, in particular label versus structural diversity. It was shown that by keeping a diverse beam significant improvements could be achieved on standard benchmarks, in particular with respect to difficult attachment decisions. It is worth pointing out that other dependency parsing frameworks (e.g., transition-based parsing (Zhang and Clark, 2008; Zhang and Nivre, 2011)) could also benefit from modeling structural diversity in search.

[2]Using eval.pl from Buchholz and Marsi (2006).

References

S. Bergsma and C. Cherry. 2010. Fast and accurate arc filtering for dependency parsing. In *Proc. of COLING*.

B. Bohnet and J. Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP/CoNLL*.

B. Bohnet, J. Nivre, I. Boguslavsky, F. Ginter, Richárd F., and J. Hajic. 2013. Joint morphological and syntactic analysis for richly inflected languages. *TACL*, 1.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL*.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

M. De Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proc. of COLING*.

H. He, H. Daumé III, and J. Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proc. of EMNLP*.

L. Kong and N. A. Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. In *ArXiv:1404.4314*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency parsing*. Morgan & Claypool Publishers.

X. Ma and H. Zhao. 2012. Fourth-order dependency parsing. In *Proc. of COLING*.

A. F. T. Martins, M. B. Almeida, and N. A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*.

A. Rush and S. Petrov. 2012. Efficient multi-pass dependency pruning with vine parsing. In *Proc. of NAACL*.

R. Zens and H. Ney. 2008. Improvements in dynamic programming beam search for phrase-based statistical machine translation. In *Proc. IWSLT*.

Y. Zhang and S. Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proc. of EMNLP*.

H. Zhang and R. McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proc. of EMNLP*.

Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proc. of ACL-HLT*, volume 2.

H. Zhang, L. Huang, K.Zhao, and R. McDonald. 2013. Online learning for inexact hypergraph search. In *Proc. of EMNLP*.