Unary Constraints for Efficient Context-Free Parsing

Nathan Bodenstab[†] **Kristy Hollingshead**[‡] **and Brian Roark**[†]
[†] Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR
[‡]University of Maryland Institute for Advanced Computer Studies, College Park, MD
{bodensta,roark}@cslu.ogi.edu hollingk@umiacs.umd.edu

Abstract

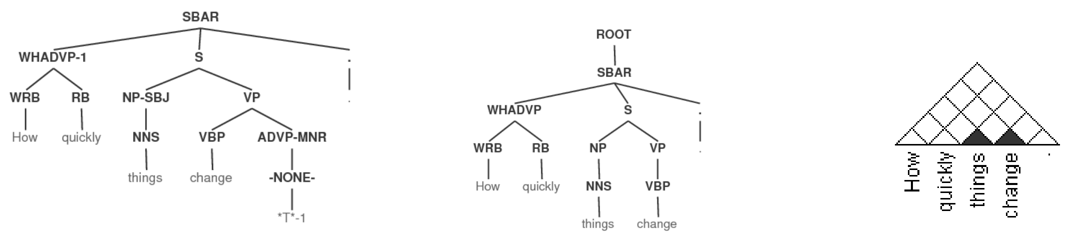
We present a novel pruning method for context-free parsing that increases efficiency by disallowing phrase-level unary productions in CKY chart cells spanning a single word. Our work is orthogonal to recent work on "closing" chart cells, which has focused on multi-word constituents, leaving span-1 chart cells unpruned. We show that a simple discriminative classifier can learn with high accuracy which span-1 chart cells to close to phrase-level unary productions. Eliminating these unary productions from the search can have a large impact on downstream processing, depending on implementation details of the search. We apply our method to four parsing architectures and demonstrate how it is complementary to the cell-closing paradigm, as well as other pruning methods such as coarse-to-fine, agenda, and beam-search pruning.

1 Introduction

While there have been great advances in the statistical modeling of hierarchical syntactic structure in the past 15 years, exact inference with such models remains very costly and most rich syntactic modeling approaches resort to heavy pruning, pipelining, or both. Graph-based pruning methods such as best-first and beam-search have both be used within context-free parsers to increase their efficiency. Pipeline systems make use of simpler models to reduce the search space of the full model. For example, the well-known Charniak parser (Charniak, 2000) uses a simple grammar to prune the search space for a richer model in a second pass.

Roark and Hollingshead (2008; 2009) have recently shown that using a finite-state tagger to close cells within the CKY chart can reduce the worst-case and average-case complexity of context-free parsing, without reducing accuracy. In their work, word positions are classified as beginning and/or ending multi-word constituents, and all chart cells not conforming to these constraints can be pruned. Zhang et al. (2010) and Bodenstab et al. (2011) both extend this approach by classifying chart cells with a finer granularity. Pruning based on constituent span is straightforwardly applicable to all parsing architectures, yet the methods mentioned above only consider spans of length two or greater. Lexical and unary productions spanning a single word are never pruned, and these can, in many cases, contribute significantly to the parsing effort.

In this paper, we investigate complementary methods to prune chart cells with finite-state preprocessing. Informally, we use a tagger to restrict the number of unary productions with non-terminals on the right-hand side that can be included in cells spanning a single word. We term these single word constituents (SWCs) (see Section 2 for a formal definition). Disallowing SWCs alters span-1 cell population from potentially containing all non-terminals to just pre-terminal part-of-speech (POS) non-terminals. In practice, this decreases the number of active states in span-1 chart cells by 70%, significantly reducing the number of allowable constituents in larger spans. Span-1 chart cells are also the most frequently queried cells in the CKY algorithm. The search over possible midpoints will always include two cells spanning a single word – one as the first left child and one as the last right child. It is therefore critical that the number of active states



(a) Original tree (b) Transformed tree (c) Dynamic programming chart

Figure 1: Example parse structure in (a) the original Penn treebank format and (b) after standard transformations have been applied. The black cells in (c) indicate CKY chart cells containing a single-word constituent from the transformed tree.

in these cells be minimized so that the number of grammar access requests is also minimized. Note, however, that some methods of grammar access – such as scanning through the rules of a grammar and looking for matches in the chart – achieve less of a speedup from diminished cell population than others, something we investigate in this paper.

Importantly, our method is orthogonal to prior work on tagging chart constraints and we expect efficiency gains to be additive. In what follows, we will demonstrate that a finite-state tagger can learn, with high accuracy, which span-1 chart cells can be closed to SWCs, and how such pruning can increase the efficiency of context-free parsing.

2 Grammar and Parsing Preliminaries

Given a probabilistic context-free grammar (PCFG) defined as the tuple $(V, T, S^\dagger, P, \rho)$ where V is the set of non-terminals, T is the set of terminals, $S^\dagger$ is a special start symbol, P is the set of grammar productions, and ρ is a mapping of grammar productions to probabilities, we divide the set of non-terminals V into two disjoint subsets V_{POS} and V_{PHR} such that V_{POS} contains all pre-terminal part-of-speech tags and V_{PHR} contains all phrase-level non-terminals. We define a **single word constituent** (SWC) unary production as any production $A \rightarrow B \in P$ such that $A \in V_{PHR}$ and A spans (derives) a single word. An example SWC unary production, VP → VBP, can be seen in Figure 1b. Note that ROOT → SBAR and RB → "quickly" in Figure 1b are also unary productions, but by definition they are not SWC unary productions.

One implementation detail necessary to leverage the benefits of sparsely populated chart cells is the grammar access method used by the inner loop of the CKY algorithm.[1] In bottom-up CKY parsing, to extend derivations of adjacent substrings into new constituents spanning the combined string, one can either iterate over all binary productions in the grammar and test if the new derivation is valid (grammar loop), or one can take the cross-product of active states in the cells spanning the substrings and poll the grammar for possible derivations (cross-product). With the cross-product approach, fewer active states in either child cell leads to fewer grammar access operations. Thus, pruning constituents in lower cells directly affects the overall efficiency of parsing. On the other hand, with the grammar loop method there is a constant number of grammar access operations (i.e., the number of grammar rules) and the number of active states in each child cell has no impact on efficiency. Therefore, with the grammar loop implementation of the CYK algorithm, pruning techniques such as unary constraints will have very little impact on the final run-time efficiency of the parser. We will report results in Section 5 with parsers using both approaches.

3 Treebank Unary Productions

In this section, we discuss the use of unary productions both in the Penn WSJ treebank (Marcus et al., 1999) and during parsing by analyzing their function and frequency. All statistics reported here are computed from sections 2-21 of the treebank.

A common pre-processing step in treebank parsing is to transform the original WSJ treebank before training and evaluation. There is some flex-

[1]Some familiarity with the CKY algorithm is assumed. For details on the algorithm, see Roark and Sproat (2007).

	Orig.	**Trans.**
Empty nodes	48,895	0
Multi-Word Const. unaries	1,225	36,608
SWC unaries	98,467	105,973
Lexical unaries	950,028	950,028
Pct words with SWC unary	10.4%	11.2%

Table 1: Unary production counts from sections 2-21 of the original and transformed WSJ treebank. All multisets are disjoint. Lexical unary count is identical to word count.

	Mk2	**Mk2+S**	**Latent**
$\lvert V_{POS}\rvert$	45	45	582
$\lvert V_{PHR}\rvert$	26	26	275
SWC grammar rules	159	1,170	91,858
Active V_{POS} states	2.5	45	75
Active V_{PHR} states	5.9	26	152

Table 2: Grammar statistics and averaged span-1 active state counts for exhaustive parsing of section 24 using a Markov order-2 (Mk2), a smoothed Markov order-2 (Mk2+S), and the Berkeley latent variable (Latent) grammars.

ibility in this process, but most pre-processing efforts include (1) affixing a ROOT unary production to the root symbol of the original tree, (2) removal of empty nodes, and (3) striping functional tags and cross-referencing annotations. See Figure 1 for an example. Additional transforms include (4) removing X → X unary productions for all non-terminals X, (5) collapsing unary chains to a single (possibly composite) unary production (Klein and Manning, 2001), (6) introducing new categories such as AUX (Charniak, 1997), and (7) collapsing of categories such as PRT and ADVP (Collins, 1997). For this paper we only apply transforms 1-3 and otherwise leave the treebank in its original form. We also note that ROOT unaries are a special case that do not affect search, and we choose to ignore them for the remainder of this paper.

These tree transformations have a large impact on the number and type of unary productions in the treebank. Table 1 displays the absolute counts of unaries in the treebank before and after processing. Multi-word constituent unary productions in the original treebank are rare and used primarily to mark quantifier phrases as noun phrases. But due to the removal of empty nodes, the transformed treebank contains many more unary productions that span multiple words, such as S → VP, where the noun phrase was left unspecified in the original clause.

The number of SWC unaries is relatively unchanged after processing the original treebank, but note that only 11.2% of words in the transformed treebank are covered by SWCs. This implies that we are unnecessarily adding SWC productions to almost 90% of span-1 chart cells during search. One may argue that an unsmoothed grammar will naturally disallow most SWC productions since they are never observed in the training data, for example VP → DT. This is true to some extent, but grammars induced from the WSJ treebank are notorious for over-generation. In addition, state-of-the-art accuracy in context-free parsing is often achieved by smoothing the grammar, so that rewrites from any one non-terminal to another are permissible, albeit with low probability.

To empirically evaluate the impact of SWCs on span-1 chart cells, we parse the development set (section 24) with three different grammars induced from sections 2-21. Table 2 lists averaged counts of active Viterbi states (derivations with probability greater than zero) from span-1 cells within the dynamic programming chart, as well as relevant grammar statistics. Note that these counts are extracted from exhaustive parsing – no pruning has been applied. We notice two points of interest. First, although $|V_{POS}| > |V_{PHR}|$, for the unsmoothed grammars more phrase-level states are active within the span-1 cells than states derived from POS tags. When parsing with the Markov order-2 grammar, 70% of active states are non-terminals from V_{PHR}, and with the latent-variable grammar, 67% (152 of 227). This is due to the highly generative nature of SWC productions. Second, although using a smoothed grammar maximizes the number of active states, the unsmoothed grammars still provide many possible derivations per word.

Given the infrequent use of SWCs in the treebank, and the search-space explosion incurred by including them in exhaustive search, it is clear that restricting SWCs in contexts where they are unlikely to occur has the potential for large efficiency gains. In the next section, we discuss how to learn such contexts via a finite-state tagger.

4 Tagging Unary Constraints

To automatically predict if word w_i from sentence $\mathbf{w}$ can be spanned by an SWC production, we train a binary classifier from supervised data using sections 2-21 of the Penn WSJ Treebank for training, section 00 as heldout, and section 24 as development. The class labels of all words in the training data are extracted from the treebank, where $w_i \in U$ if w_i is observed with a SWC production and $w_i \in \overline{U}$ otherwise. We train a log linear model with the averaged perceptron algorithm (Collins, 2002) using unigram word and POS-tag[2] features from a five word window. We also trained models with bi-gram and tri-gram features, but tagging accuracy did not improve.

Because the classifier output is imposing hard constraints on the search space of the parser, we may want to choose a tagger operating point that favors precision over recall to avoid over-constraining the downstream parser. To compare the tradeoff between possible precision/recall values, we apply the softmax activation function to the perceptron output to obtain the posterior probability of $w_i \in U$:

$$P(U|w_i, \theta) = (1 + \exp(-f(w_i) \cdot \theta))^{-1} \qquad (1)$$

where θ is a vector of model parameters and $f(\cdot)$ is a feature function. The threshold 0.5 simply chooses the most likely class, but to increase precision we can move this threshold to favor U over $\overline{U}$. To tune this value on a per-sentence basis, we follow methods similar to Roark & Hollingshead (2009) and rank each word position with respect to its posterior probability. If the total number of words w_i with $P(U|w_i, \theta) < 0.5$ is k, we decrease the threshold value from 0.5 until λk words have been moved from class $\overline{U}$ to U, where λ is a tuning parameter between 0 and 1. Although the threshold 0.5 produces tagging precision and recall of 98.7% and 99.4% respectively, we can adjust λ to increase precision as high as 99.7%, while recall drops to a tolerable 82.1%. Similar methods are used to replicate cell-closing constraints, which are combined with unary constraints in the next section.

[2]POS-tags were provided by a separately trained tagger.

5 Experiments and Results

To evaluate the effectiveness of unary constraints, we apply our technique to four parsers: an exhaustive CKY chart parser (Cocke and Schwartz, 1970); the Charniak parser (Charniak, 2000), which uses agenda-based two-level coarse-to-fine pruning; the Berkeley parser (Petrov and Klein, 2007a), a multi-level coarse-to-fine parser; and the BUBS parser (Bodenstab et al., 2011), a single-pass beam-search parser with a figure-of-merit constituent ranking function. The Berkeley and BUBS parsers both parse with the Berkeley latent-variable grammar (Petrov and Klein, 2007b), while the Charniak parser uses a lexicalized grammar, and the exhaustive CKY algorithm is run with a simple Markov order-2 grammar. All grammars are induced from the same data: sections 2-21 of the WSJ treebank.

Figure 2 contrasts the merit of unary constraints on the three high-accuracy parsers, and several interesting comparisons emerge. First, as recall is traded for precision within the tagger, each parser reacts quite differently to the imposed constraints. We apply constraints to the Berkeley parser during the initial coarse-pass search, which is simply an exhaustive CKY search with a coarse grammar. Applying unary and cell-closing constraints at this point in the coarse-to-fine pipeline speeds up the initial coarse-pass significantly, which accounted for almost half of the total parse time in the Berkeley parser. In addition, all subsequent fine-pass searches also benefit from additional pruning as their search is guided by the remaining constituents of the previous pass, which is the intersection of standard coarse-to-fine pruning and our imposed constraints.

We apply constraints to the Charniak parser during the first-pass agenda-based search. Because an agenda-based search operates at a constituent level instead of a cell/span level, applying unary constraints alters the search frontier instead of reducing the absolute number of constituents placed in the chart. We jointly tune lambda and the internal search parameters of the Charniak parser until accuracy degrades.

Application of constraints to the CKY and BUBS parsers is straightforward as they are both single pass parsers – any constituent violating the constraints is pruned. We also note that the CKY and

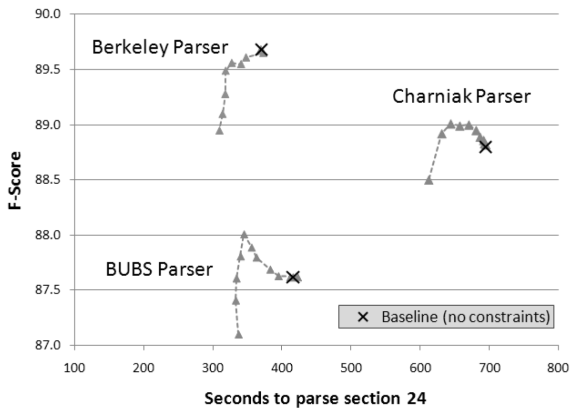


Figure 2: Development set results applying unary constraints at multiple values of λ to three parsers.

Parser	F-score	Seconds	Speedup
CKY	72.2	1,358	
+ UC (λ=0.2)	72.6	1,125	1.2x
+ CC	74.3	380	3.6x
+ CC + UC	74.6	249	5.5x
BUBS	88.4	586	
+ UC (λ=0.2)	88.5	486	1.2x
+ CC	88.7	349	1.7x
+ CC + UC	88.7	283	2.1x
Charniak	89.7	1,116	
+ UC (λ=0.2)	89.7	900	1.2x
+ CC	89.7	716	1.6x
+ CC + UC	89.6	679	1.6x
Berkeley	90.2	564	
+ UC (λ=0.4)	90.1	495	1.1x
+ CC	90.2	320	1.8x
+ CC + UC	90.2	289	2.0x

Table 3: Test set results applying unary constraints (UC) and cell-closing (CC) constraints (Roark and Hollingshead, 2008) to various parsers.

BUBS parsers both employ the cross-product grammar access method discussed in Section 2, while the Berkeley parser uses the grammar loop method. This grammar access difference dampens the benefit of unary constraints for the Berkeley parser.[3]

Referring back to Figure 2, we see that both speed and accuracy increase in all but the Berkeley parser. Although it is unusual that pruning leads to higher accuracy during search, it is not unexpected here as our finite-state tagger makes use of lexical relationships that the PCFG does not. By leveraging this new information to constrain the search space, we are indirectly improving the quality of the model.

Finally, there is an obvious operating point for each parser at which the unary constraints are too severe and accuracy deteriorates rapidly. For test conditions, we set the tuning parameter λ based on the development set results to prune as much of the search space as possible before reaching this degradation point.

Using lambda-values optimized for each parser, we parse the unseen section 23 test data and present results in Table 3. We see that in all cases, unary constraints improve the efficiency of parsing without significant accuracy loss. As one might expect, exhaustive CKY parsing benefits the most from unary constraints since no other pruning is applied. But even heavily pruned parsers using graph-based and pipelining techniques still see substantial speedups with the additional application of unary constraints. Furthermore, unary constraints consistently provide an additive efficiency gain when combined with cell-closing constraints.

[3]The Berkeley parser does maintain meta-information about where non-terminals have been placed in the chart, giving it some of the advantages of cross-product grammar access.

6 Conclusion

We have presented a new method to constrain context-free chart parsing and have shown it to be orthogonal to many forms of graph-based and pipeline pruning methods. In addition, our method parallels the cell closing paradigm and is an elegant complement to recent work, providing a finite-state tagging framework to potentially constrain all areas of the search space – both multi-word and single-word constituents.

Acknowledgments

We would like to thank Aaron Dunlop for his valuable discussions, as well as the anonymous reviewers who gave very helpful feedback. This research was supported in part by NSF Grants #IIS-0447214, #IIS-0811745 and DARPA grant #HR0011-09-1-0041. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

References

Nathan Bodenstab, Aaron Dunlop, Keith Hall, and Brian Roark. 2011. Beam-width prediction for efficient context-free parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon. Association for Computational Linguistics.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, Menlo Park, CA. AAAI Press/MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, Seattle, Washington. Morgan Kaufmann Publishers Inc.

John Cocke and Jacob T. Schwartz. 1970. Programming languages and their compilers. Technical report Preliminary notes, Courant Institute of Mathematical Sciences, NYU.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, page 1623, Morristown, NJ, USA. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing*, volume 10, pages 1–8, Philadelphia, July. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn treebank. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 338–345, Toulouse, France, July. Association for Computational Linguistics.

Mitchell P Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia.

Slav Petrov and Dan Klein. 2007a. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007b. Learning and inference for hierarchically split PCFGs. In *AAAI 2007 (Nectar Track)*.

Brian Roark and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In Donia Scott and Hans Uszkoreit, editors, *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 745–752, Manchester, UK, August. Association for Computational Linguistics.

Brian Roark and Kristy Hollingshead. 2009. Linear complexity context-free parsing pipelines via chart constraints. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 647–655, Boulder, Colorado, June. Association for Computational Linguistics.

Brian Roark and Richard W Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford University Press, New York.

Yue Zhang, Byung gyu Ahn, Stephen Clark, Curt Van Wyk, James R. Curran, and Laura Rimell. 2010. Chart pruning for fast lexicalised-grammar parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1472–1479, Beijing, China, June.