Beam-Width Prediction for Efficient Context-Free Parsing

Nathan Bodenstab[†] **Aaron Dunlop**[†] **Keith Hall**[‡] **and Brian Roark**[†]
[†] Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR
[‡]Google, Inc., Zurich, Switzerland
{bodensta,dunlopa,roark}@cslu.ogi.edu kbhall@google.com

Abstract

Efficient decoding for syntactic parsing has become a necessary research area as statistical grammars grow in accuracy and size and as more NLP applications leverage syntactic analyses. We review prior methods for pruning and then present a new framework that unifies their strengths into a single approach. Using a log linear model, we learn the optimal beam-search pruning parameters for each CYK chart cell, effectively predicting the most promising areas of the model space to explore. We demonstrate that our method is faster than coarse-to-fine pruning, exemplified in both the Charniak and Berkeley parsers, by empirically comparing our parser to the Berkeley parser using the same grammar and under identical operating conditions.

1 Introduction

Statistical constituent parsers have gradually increased in accuracy over the past ten years. This accuracy increase has opened the door to automatically derived syntactic information within a number of NLP tasks. Prior work incorporating parse structure into machine translation (Chiang, 2010) and Semantic Role Labeling (Tsai et al., 2005; Punyakanok et al., 2008) indicate that such hierarchical structure can have great benefit over shallow labeling techniques like chunking and part-of-speech tagging.

Although syntax is becoming increasingly important for large-scale NLP applications, constituent parsing is slow — too slow to scale to the size of many potential consumer applications. The exhaustive CYK algorithm has computational complexity $O(n^3|G|)$ where n is the length of the sentence and $|G|$ is the number of grammar productions, a non-negligible constant. Increases in accuracy have primarily been accomplished through an increase in the size of the grammar, allowing individual grammar rules to be more sensitive to their surrounding context, at a considerable cost in efficiency. Grammar transformation techniques such as linguistically inspired non-terminal annotations (Johnson, 1998; Klein and Manning, 2003b) and latent variable grammars (Matsuzaki et al., 2005; Petrov et al., 2006) have increased the grammar size $|G|$ from a few thousand rules to several million in an explicitly enumerable grammar, or even more in an implicit grammar. Exhaustive search for the maximum likelihood parse tree with a state-of-the-art grammar can require over a minute of processing for a single sentence of 25 words, an unacceptable amount of time for real-time applications or when processing millions of sentences. Deterministic algorithms for dependency parsing exist that can extract syntactic dependency structure very quickly (Nivre, 2008), but this approach is often undesirable as constituent parsers are more accurate and more adaptable to new domains (Petrov et al., 2010).

The most accurate constituent parsers, e.g., Charniak (2000), Petrov and Klein (2007a), make use of approximate inference, limiting their search to a fraction of the total search space and achieving speeds of between one and four newspaper sentences per second. The paradigm for building state-of-the-art parsing models is to first design a model structure that can achieve high accuracy and then, after the model has been built, design effective approximate inference methods around that particular model; e.g., coarse-to-fine non-terminal hierarchies for a given model, or agenda-based methods

that are empirically tuned to achieve acceptable efficiency/accuracy operating points. While both of the above mentioned papers use the CYK dynamic programming algorithm to search through possible solutions, their particular methods of approximate inference are quite distinct.

In this paper, we examine a general approach to approximate inference in constituent parsing that learns cell-specific thresholds for arbitrary grammars. For each cell in the CYK chart, we sort all potential constituents in a local agenda, ordered by an estimate of their posterior probability. Given features extracted from the chart cell context – e.g., span width; POS-tags and words surrounding the boundary of the cell – we train a log linear model to predict how many constituents should be popped from the local agenda and added to the chart. As a special case of this approach, we simply predict whether the number to add should be zero or greater than zero, in which case the method can be seen as a cell-by-cell generalization of Roark and Hollingshead's (2008; 2009) tagger-derived Chart Constraints. More generally, instead of a binary classification decision, we can also use this method to predict the desired cell population directly and get cell closure for free when the classifier predicts a beam-width of zero. In addition, we use a non-symmetric loss function during optimization to account for the imbalance between over-predicting or under-predicting the beam-width.

A key feature of our approach is that it does not rely upon reference syntactic annotations when learning to search. Rather, the beam-width prediction model is trained to learn the rank of constituents in the maximum likelihood trees.[1] We will illustrate this by presenting results using a latent-variable grammar, for which there is no "true" reference latent variable parse. We simply parse sections 2-21 of the WSJ treebank and train our search models from the output of these trees, with no prior knowledge of the non-terminal set or other grammar characteristics to guide the process. Hence, this approach is broadly applicable to a wide range of scenarios, including tuning the search to new domains where domain mismatch may yield very different efficiency/accuracy operating points.

[1]Note that we do not call this method "unsupervised" because all grammars used in this paper are induced from supervised data, although our framework can also accommodate unsupervised grammars. We emphasize that we are learning to search using only maximum likelihood trees, not that we are doing unsupervised parsing.

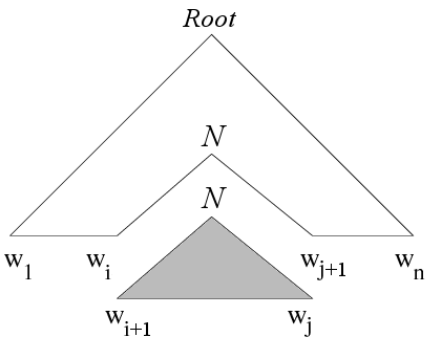


Figure 1: Inside (grey) and outside (white) representations of an example chart edge $N_{i,j}$.

In the next section, we present prior work on approximate inference in parsing, and discuss how our method to learn optimal beam-search parameters unite many of their strengths into a single framework. We then explore using our approach to open or close cells in the chart as an alternative to Roark and Hollingshead (2008; 2009). Finally, we present results which combine cell closure and adaptive beam-width prediction to achieve the most efficient parser.

2 Background

2.1 Preliminaries and notation

Let $S = w_1 \dots w_{|S|}$ represent an input string of $|S|$ words. Let $w_{i,j}$ denote the substring from word w_{i+1} to w_j; i.e., $S = w_{0,|S|}$. We use the term chart *edge* to refer to a non-terminal spanning a specific substring of the input sentence. Let $N_{i,j}$ denote the edge labeled with non-terminal N spanning $w_{i,j}$, for example $\text{NP}_{3,7}$. We define an edge's figure-of-merit (FOM) as an estimate of the product of its inside (β) and outside (α) scores, conceptually the relative merit the edge has to participate in the final parse tree (see Figure 1). More formally:

$$\begin{aligned} \alpha(N_{i,j}) &= P(w_{0,i}, N_{i,j}, w_{j,n}) \\ \beta(N_{i,j}) &= P(w_{i,j}|N) \\ \text{FOM}(N_{i,j}) &= \hat{\alpha}(N_{i,j})\hat{\beta}(N_{i,j}) \end{aligned}$$

With bottom-up parsing, the true inside probability is accumulated and $\beta(N_{i,j})$ does not need to be estimated, improving the FOMs ability to represent the true inside/outside distribution.

In this paper we use a modified version of the Caraballo and Charniak **Boundary FOM** (1998) for local edge comparison, which computes $\hat{\alpha}(N_{i,j})$ using POS forward-backward scores and POS-to-nonterminal constituent boundary transition probabilities. Implementation and modification details can be found in (Bodenstab et al., 2010).

We use the Boundary FOM scoring function to rank constituents in a local (per-span) agenda, but the application of any function to rank competitors is possible within our framework, including other best-first or A* priority functions. We demonstrate this by presenting results with only the inside probability and the Boundary FOM in Section 6.

2.2 Agenda-based parsing

Agenda-based parsers maintain a global agenda of edges, ranked by FOM score. At each iteration, the highest-scoring edge is popped off of the agenda, added to the chart, and combined with other edges already in the chart. The agenda-based approach includes best-first parsing (Bobrow, 1990) and A* parsing (Klein and Manning, 2003a), which differ in whether an admissible FOM estimate $\hat{\alpha}(N_{i,j})$ is required. A* uses an admissible FOM, and thus guarantees finding the maximum likelihood parse, whereas an inadmissible heuristic (best-first) may require less exploration of the search space. Much work has been pursued in both admissible and inadmissible heuristics for agenda parsing (Caraballo and Charniak, 1998; Klein and Manning, 2003a; Pauls et al., 2010).

In this paper, we also make use of agendas, but at a local rather than a global level. We maintain an agenda for each cell, which has two significant benefits: 1) Competing edges can be compared directly, avoiding the difficulty inherent in agenda-based approaches of comparing edges of radically different span lengths and characteristics; and 2) Since the agendas are very small, the overhead of agenda maintenance — a large component of agenda-based parse time — is minimal.

2.3 Beam-search parsing

CYK parsing with a beam-search is a local pruning strategy, comparing edges within the same chart cell. The beam-width can be defined in terms of a threshold in the number of edges allowed, or in terms of a threshold on the difference in probability relative to the highest scoring edge (Collins, 1999; Zhang et al., 2010). For the current paper, we use both kinds of thresholds, avoiding pathological cases that each individual criteria is prone to encounter. Further, unlike most beam-search approaches we will make use of a FOM estimate of the posterior probability of an edge, defined above, as our ranking function. Finally, we will learn log linear models to assign cell-specific thresholds, rather than relying on a single search parameter.

2.4 Coarse-to-Fine Parsing

Coarse-to-fine parsing, also known as multiple pass parsing (Goodman, 1997; Charniak, 2000; Charniak and Johnson, 2005), first parses the input sentence with a simplified (coarse) version of the target (fine) grammar in which multiple non-terminals are merged into a single state. Since the coarse grammar is quite small, parsing is much faster than with the fine grammar, and can quickly yield an estimate of the outside probability $\alpha(\cdot)$ for use in subsequent agenda or beam-search parsing with the fine grammar. This approach can also be used iteratively with grammars of increasing complexity (Petrov and Klein, 2007a).

Building a coarse grammar from a fine grammar is a non-trivial problem, and most often approached with detailed knowledge of the fine grammar being used. For example, Goodman (1997) suggests using a coarse grammar consisting of regular non-terminals, such as NP and VP, and then non-terminals augmented with head-word information for the more accurate second-pass grammar. Such an approach is followed by Charniak (2000) as well. Petrov and Klein (2007a) derive coarse grammars in a more statistically principled way, although the technique is closely tied to their latent variable grammar representation.

To the extent that our cell-specific threshold classifier predicts that a chart cell should contain zero edges or more than zero edges, it is making coarse

predictions about the unlabeled constituent structure of the target parse tree. This aspect of our work is can be viewed as a coarse-to-fine process, though without considering specific grammatical categories or rule productions.

2.5 Chart Constraints

Roark and Hollingshead (2008; 2009) introduced a pruning technique that ignores entire chart cells based on lexical and POS features of the input sentence. They train two finite-state binary taggers: one that allows multi-word constituents to start at a word, and one that allows constituents to end at a word. Given these tags, it is straightforward to completely skip many chart cells during processing.

In this paper, instead of tagging word positions to infer valid constituent spans, we classify chart cells directly. We further generalize this cell classification to predict the beam-width of the chart cell, where a beam-width of zero indicates that the cell is completely closed. We discuss this in detail in the next section.

3 Open/Closed Cell Classification

3.1 Constituent Closure

We first look at the binary classification of chart cells as either open or closed to full constituents, and predict this value from the input sentence alone. This is the same problem that Roark and Hollingshead (2008; 2009) solve with Chart Constraints; however, where they classify lexical items as either beginning or ending a constituent, we classify individual chart cells as open or closed, an approach we call *Constituent Closure*. Although the number of classifications scales quadratically with our approach, the total parse time is still dominated by the $O(n^3|G|)$ parsing complexity and we find that the added level of specificity reduces the search space significantly.

To learn to classify a chart cell spanning words $w_{i+1} \dots w_j$ of a sentence S as open or closed to full constituents, we first map cells in the training corpus to tuples:

$$\Phi(S, i, j) = (\boldsymbol{x}, y) \tag{1}$$

where $\boldsymbol{x}$ is a feature-vector representation of the chart cell and y is the target class 1 if the cell contains an edge from the maximum likelihood parse tree, 0 otherwise. The feature vector $\boldsymbol{x}$ is encoded with the chart cell's absolute and relative span width, as well as unigram and bigram lexical and part-of-speech tag items from $w_{i-1} \dots w_{j+2}$.

Given feature/target tuples $(\boldsymbol{x}, y)$ for every chart cell in every sentence of a training corpus τ, we train a weight vector θ using the averaged perceptron algorithm (Collins, 2002) to learn an open/closed binary decision boundary:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{(\boldsymbol{x},y)\in\Phi(\tau)} L_\lambda(H(\theta \cdot \boldsymbol{x}), y) \tag{2}$$

where $H(\cdot)$ is the unit step function: 1 if the inner product $\theta \cdot \boldsymbol{x} > 0$, and 0 otherwise; and $L_\lambda(\cdot, \cdot)$ is an asymmetric loss function, defined below.

When predicting cell closure, all misclassifications are not equal. If we leave open a cell which contains no edges in the maximum likelihood (ML) parse, we incur the cost of additional processing, but are still able to recover the ML tree. However, if we close a chart cell which contains an ML edge, search errors occur. To deal with this imbalance, we introduce an asymmetric loss function $L_\lambda(\cdot, \cdot)$ to penalize false-negatives more severely during training.

$$L_\lambda(h, y) = \begin{cases} 0 & \text{if } h = y \\ 1 & \text{if } h > y \\ \lambda & \text{if } h < y \end{cases} \tag{3}$$

We found the value $\lambda = 10^2$ to give the best performance on our development set, and we use this value in all of our experiments.

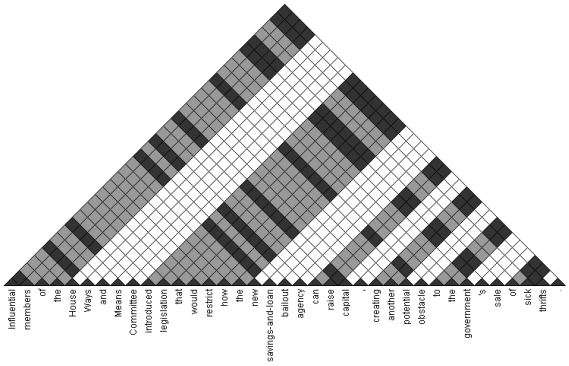
Figures 2a and 2b compare the pruned charts of Chart Constraints and Constituent Closure for a single sentence in the development set. Note that both of these methods are predicting where a complete constituent may be located in the chart, not partial constituents headed by factored nonterminals within a binarized grammar. Depending on the grammar factorization (right or left) we can infer chart cells that are restricted to only edges with a factored left-hand-side non-terminal. In Figure 2 these chart cells are colored gray. Note that Constituent Closure reduces the number of completely open cells considerably vs. Chart Constraints, and the number of cells open to factored categories somewhat.

3.2 Complete Closure

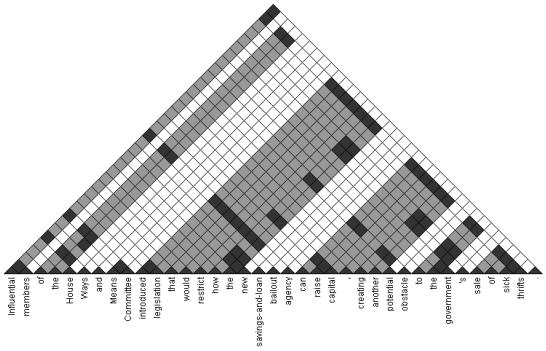
Alternatively, we can predict whether a chart cell contains any edge, either a partial or a full constituent, an approach we call *Complete Closure*. This is a more difficult classification problem as partial constituents occur in a variety of contexts. Nevertheless, learning this directly allows us to remove a large number of internal chart cells from consideration, since no additional cells need to be left open to partial constituents. The learning algorithm is identical to Equation 2, but training examples are now assigned a positive label if the chart cell contains any edge from the binarized maximum likelihood tree. Figure 2c gives a visual representation of Complete Closure for the same sentence; the number of completely open cells increases somewhat, but the total number of open cells (including those open to factored categories) is greatly reduced.

We compare the effectiveness of Constituent Closure, Complete Closure, and Chart Constraints, by decreasing the percentage of chart cells closed until accuracy over all sentences in our development set start to decline. For Constituent and Complete Closure, we also vary the loss function, adjusting the relative penalty between a false-negative (closing off a chart cell that contains a maximum likelihood edge) and a false-positive. Results show that using Chart Constrains as a baseline, we prune (skip) 33% of the total chart cells. Constituent Closure improves on this baseline only slightly (36%), but we see our biggest gains with Complete Closure, which prunes 56% of all chart cells in the development set.

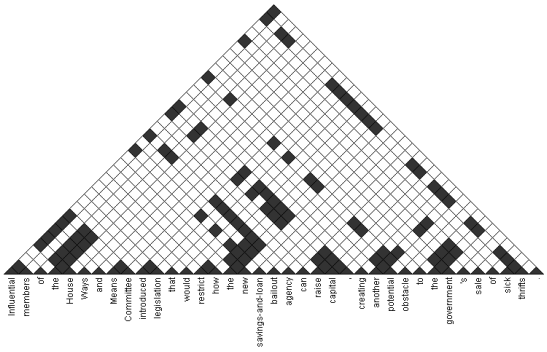
All of these open/closed cell classification methods can improve the efficiency of the exhaustive CYK algorithm, or any of the approximate inference methods mentioned in Section 2. We empirically evaluate them when applied to CYK parsing and beam-search parsing in Section 6.



(a) Chart Constraints (Roark and Hollingshead, 2009)



(b) Constituent Closure (this paper)



(c) Complete Closure (this paper)

Figure 2: Comparison of Chart Constraints (Roark and Hollingshead, 2009) to Constituent and Complete Closure for a single example sentence. Black cells are open to all edges while grey cells only allow factored edges (incomplete constituents).

4 Beam-Width Prediction

The cell-closing approaches discussed in Section 3 make binary decisions to either allow or completely block all edges in each cell. This all-on/all-off tactic ignores the characteristics of the local cell population, which, given a large statistical grammar, may contain hundred of edges, even if very improbable. Retaining all of these partial derivations forces the search in larger spans to continue down improbable paths, adversely affecting efficiency. We can further improve parsing speed in these open cells by leveraging local pruning methods, such as beam-search.

When parsing with a beam-search, finding the optimal beam-width threshold(s) to balance speed and accuracy is a necessary step. As mentioned in Sec-

tion 2.3, two variations of the beam-width are often considered: a fixed number of allowed edges, or a relative probability difference from the highest scoring local edge. For the remainder of this paper we fix the relative probability threshold for all experiments and focus on adapting the number of allowed edges per cell. We will refer to this number-of-allowed-edges value as the beam-width, notated by b, and leave adaptation of the relative probability difference to future work.

The standard way to tune the beam-width is a simple sweep over possible values until accuracy on a heldout data set starts to decline. The optimal point will necessarily be very conservative, allowing outliers (sentences or sub-phrases with above average ambiguity) to stay within the beam and produce valid parse trees. The majority of chart cells will require much fewer than b entries to find the maximum likelihood (ML) edge, yet, constrained by a constant beam-width, the cell will continue to be filled with unfruitful edges, exponentially increasing downstream computation.

For example, when parsing with the Berkeley latent-variable grammar and Boundary FOM, we find we can reduce the global beam-width b to 15 edges in each cell before accuracy starts to decline. However we find that 73% of the ML edges are ranked *first* in their cell and 96% are ranked in the top three. Thus, in 24 of every 25 cells, 80% of the edges are unnecessary (12 of the top 15). Clearly, it would be advantageous to adapt the beam-width such that it is restrictive when we are confident in the FOM ranking and more forgiving in ambiguous contexts.

To address this problem, we learn the optimal beam-width for each chart cell directly. We define $R_{i,j}$ as the rank of the ML edge in the chart cell spanning $w_{i+1} \dots w_j$. If no ML edge exists in the cell, then $R_{i,j} = 0$. Given a global maximum beam-width b, we train b different binary classifiers, each using separate mapping functions Φ_k, where the target value y produced by Φ_k is 1 if $R_{i,j} > k$ and 0 otherwise.

The same asymmetry noted in Section 3 applies in this task as well. When in doubt, we prefer to over-predict the beam-width and risk an increase in processing time opposed to under-predicting at the expense of accuracy. Thus we use the same loss function L_λ, this time training several classifiers:

$$\hat{\theta}_k = \operatorname*{argmin}_{\theta} \sum_{(\boldsymbol{x},y)\in\Phi_k(\tau)} L_\lambda(H(\theta \cdot \boldsymbol{x}), y) \qquad (4)$$

Note that in Equation 4 when $k = 0$, we recover the open/closed cell classification of Equation 2, since a beam width of 0 indicates that the chart cell is completely closed.

During decoding, we assign the beam-width for chart cell spanning $w_{i+1} \dots w_j$ given models $\theta_0, \theta_1, ...\theta_{b-1}$ by finding the lowest value k such that the binary classifier θ_k classifies $R_{i,j} \leq k$. If no such k exists, $\hat{R}_{i,j}$ is set to the maximum beam-width value b:

$$\hat{R}_{i,j} = \operatorname*{argmin}_{k} \theta_k \cdot x_i \leq 0 \qquad (5)$$

In Equation 5 we assume there are b unique classifiers, one for each possible beam-width value between 0 and $b-1$, but this level of granularity is not required. Choosing the number of classification bins to minimize total parsing time is dependent on the FOM function and how it ranks ML edges. With the Boundary FOM we use in this paper, 97.8% of ML edges have a local rank less than five and we find that the added cost of computing b decision boundaries for each cell is not worth the added specificity. We searched over possible classification bins and found that training four classifiers with beam-width decision boundaries at 0, 1, 2, and 4 is faster than 15 individual classifiers and more memory efficient, since each model θ_k has over 800,000 parameters. All beam-width prediction results reported in this paper use these settings.

Figure 3 is a visual representation of beam-width prediction on a single sentence of the development set using the Berkeley latent-variable grammar and Boundary FOM. In this figure, the gray scale represents the relative size of the beam-width, black being the maximum beam-width value, b, and the lightest gray being a beam-width of size one. We can see from this figure that very few chart cells are classified as needing the full 15 edges, apart from span-1 cells which we do not classify.

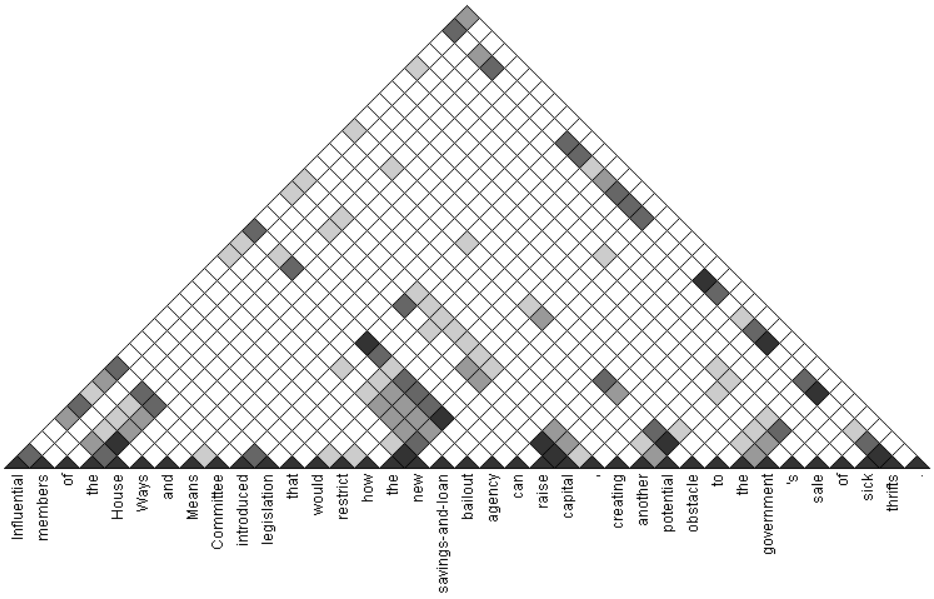


Figure 3: Visualization of Beam-Width Prediction for a single example sentence. The grey scale represents the size of the predicted beam-width: white is 0 (cell is skipped) and black is the maximum value b (b=15 in this example).

5 Experimental Setup

We run all experiments on the WSJ treebank (Marcus et al., 1999) using the standard splits: section 2-21 for training, section 22 for development, and section 23 for testing. We preprocess the treebank by removing empty nodes, temporal labels, and spurious unary productions ($X \rightarrow X$), as is standard in published works on syntactic parsing.

The pruning methods we present in this paper can be used to parse with any grammar. To achieve state-of-the-art accuracy levels, we parse with the Berkeley SM6 latent-variable grammar (Petrov and Klein, 2007b) where the original treebank non-terminals are automatically split into subclasses to optimize parsing accuracy. This is an explicit grammar consisting of 4.3 million productions, 2.4 million of which are lexical productions. Exhaustive CYK parsing with the grammar takes more than a minute per sentence.

Accuracy is computed from the 1-best Viterbi (max) tree extracted from the chart. Alternative decoding methods, such as marginalizing over the latent variables in the grammar or MaxRule decoding (Petrov and Klein, 2007a) are certainly possible in our framework, but it is unknown how effective these methods will be given the heavily pruned nature of the chart. We leave investigation of this to future work. We compute the precision and recall of constituents from the 1-best Viterbi trees using the standard EVALB script (Collins, 1997), which ignores punctuation and the root symbol. Accuracy results are reported as F-measure (F_1), the harmonic mean between precision and recall.

We ran all timing tests on an Intel 3.00GHz processor with 6MB of cache and 16GB of memory. Our parser is written in Java and publicly available at http://nlp.csee.ogi.edu.

6 Results

We empirically demonstrate the advantages of our pruning methods by comparing the total parse time of each system, including FOM initialization, chart cell classification, and beam-width prediction. The parse times reported for Chart Constraints do not include tagging times as we were provided with this pre-tagged data, but tagging all of Section 22 takes less than three seconds and we choose to ignore this contribution for simplicity.

Figure 4 contains a timing comparison of the three components of our final parser: Boundary FOM initialization (which includes the forward-backward algorithm over ambiguous part-of-speech tags), beam-

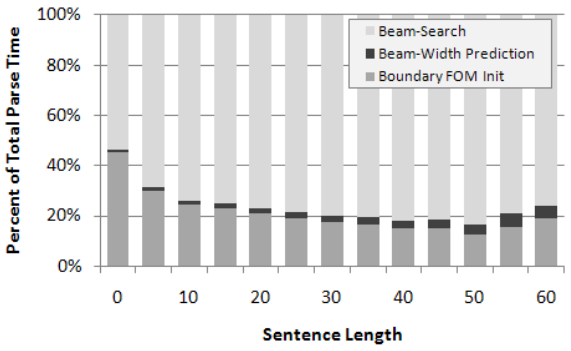


Figure 4: Timing breakdown by sentence length for major components of our parser.

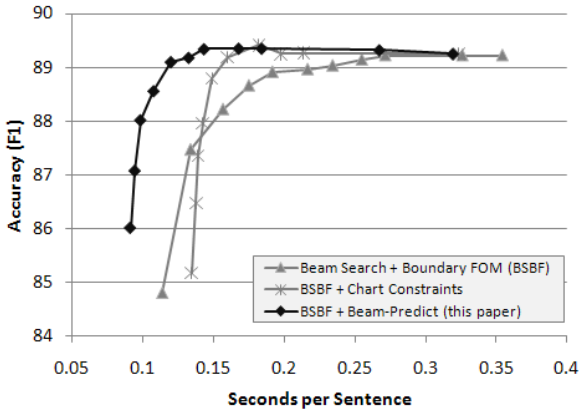


Figure 5: Time vs. accuracy curves comparing beam-width prediction (Beam-Predict) and Chart Constraints.

width prediction, and the final beam-search, including 1-best extraction. We bin these relative times with respect to sentence length to see how each component scales with the number of input words. As expected, the $O(n^3|G|)$ beam-search begins to dominate as the sentence length grows, but Boundary FOM initialization is not cheap, and absorbs, on average, 20% of the total parse time. Beam-width prediction, on the other hand, is almost negligible in terms of processing time even though it scales quadratically with the length of the sentence.

We compare the accuracy degradation of beam-width prediction and Chart Constraints in Figure 5 as we incrementally tighten their respective pruning parameters. We also include the baseline beam-search parser with Boundary FOM in this figure to demonstrate the accuracy/speed trade-off of adjusting a global beam-width alone. In this figure we see that the knee of the beam-width prediction curve (Beam-Predict) extends substantially further to the left before accuracy declines, indicating that our pruning method is intelligently removing a significant portion of the search space that remains unpruned with Chart Constraints.

In Table 1 we present the accuracy and parse time for three baseline parsers on the development set: exhaustive CYK parsing, beam-search parsing using only the inside score $\beta(\cdot)$, and beam-search parsing using the Boundary FOM. We then apply our two cell-closing methods, Constituent Closure and Complete Closure, to all three baselines. As expected, the relative speedup of these methods across the various baselines is similar since the open/closed cell classification does not change across parsers. We also see that Complete Closure is between 22% and 31% faster than Constituent Closure, indicating that the greater number of cells closed translates directly into a reduction in parse time. We can further apply beam-width prediction to the two beam-search baseline parsers in Table 1. Dynamically adjusting the beam-width for the remaining open cells decreases parse time by an additional 25% when using the Inside FOM, and 28% with the boundary FOM.

We apply our best model to the test set and report results in Table 2. Beam-width prediction, again, outperforms the baseline of a constant beam-width by 65% and the open/closed classification of Chart Constraints by 49%. We also compare beam-width prediction to the Berkeley Coarse-to-Fine parser. Both our parser and the Berkeley parser are written in Java, both are run with Viterbi decoding, and both parse with the same grammar, so a direct comparison of speed and accuracy is fair.[2]

7 Conclusion and Future Work

We have introduced three new pruning methods, the best of which unites figure-of-merit estimation from agenda-based parsing, local pruning from beam-search parsing, and unlabeled constituent structure

[2]We run the Berkeley parser with the default search parameterization to achieve the fastest possible parsing time. We note that 3 of 2416 sentences fail to parse under these settings. Using the '-accurate' option provides a valid parse for all sentences, but increases parsing time of section 23 to 0.293 seconds per sentence with no increase in F-score. We assume a back-off strategy for failed parses could be implemented to parse all sentences with a parsing time close to the default parameterization.

Parser	Sec/Sent	F_1
CYK	70.383	89.4
CYK + Constituent Closure	47.870	89.3
CYK + Complete Closure	32.619	89.3
Beam + Inside FOM (BI)	3.977	89.2
BI + Constituent Closure	2.033	89.2
BI + Complete Closure	1.575	89.3
BI + Beam-Predict	1.180	89.3
Beam + Boundary FOM (BB)	0.326	89.2
BB + Constituent Closure	0.279	89.2
BB + Complete Closure	0.199	89.3
BB + Beam-Predict	0.143	89.3

Table 1: Section 22 development set results for CYK and Beam-Search (Beam) parsing using the Berkeley latent-variable grammar.

Parser	Sec/Sent	F_1
CYK	64.610	88.7
Berkeley CTF MaxRule	0.213	90.2
Berkeley CTF Viterbi	0.208	88.8
Beam + Boundary FOM (BB)	0.334	88.6
BB + Chart Constraints	0.244	88.7
BB + Beam-Predict (this paper)	0.125	88.7

Table 2: Section 23 test set results for multiple parsers using the Berkeley latent-variable grammar.

prediction from coarse-to-fine parsing and Chart Constraints. Furthermore, our pruning method is trained using only maximum likelihood trees, allowing it to be tuned to specific domains without labeled data. Using this framework, we have shown that we can decrease parsing time by 65% over a standard beam-search without any loss in accuracy, and parse significantly faster than both the Berkeley parser and Chart Constraints.

We plan to explore a number of remaining questions in future work. First, we will try combining our approach with constituent-level Coarse-to-Fine pruning. The two methods prune the search space in very different ways and may prove to be complementary. On the other hand, our parser currently spends 20% of the total parse time initializing the FOM, and adding additional preprocessing costs, such as parsing with a coarse grammar, may not outweigh the benefits gained in the final search.

Second, as with Chart Constraints we do not prune lexical or unary edges in the span-1 chart cells (i.e., chart cells that span a single word). We expect pruning entries in these cells would notably reduce parse time since they cause exponentially many chart edges to be built in larger spans. Initial work constraining span-1 chart cells has promising results (Bodenstab et al., 2011) and we hope to investigate its interaction with beam-width prediction even further.

Finally, the size and structure of the grammar is the single largest contributor to parse efficiency. In contrast to the current paradigm, we plan to investigate new algorithms that jointly optimize accuracy and efficiency during grammar induction, leading to more efficient decoding.

Acknowledgments

We would like to thank Kristy Hollingshead for her 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

Robert J. Bobrow. 1990. Statistical agenda parsing. In *DARPA Speech and Language Workshop*, pages 222–224.

Nathan Bodenstab, Aaron Dunlop, Keith Hall, and Brian Roark. 2010. Exponential decay pruning for bottom-up beam-search parsing. In *Pacific Northwest Regional NLP Workshop (NW-NLP 2010)*.

Nathan Bodenstab, Kristy Hollingshead, and Brian Roark. 2011. Unary constraints for efficient context-free parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon.

Sharon A Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24:275–298.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on As-*

sociation for Computational Linguistics, pages 173–180, Ann Arbor, Michigan.

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

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48rd Annual Meeting on Association for Computational Linguistics*, pages 1443–1452.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD dissertation, University of Pennsylvania.

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.

Joshua Goodman. 1997. Global thresholding and Multiple-Pass parsing. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11–25.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2003a. A* parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, pages 40–47, Edmonton, Canada.

Dan Klein and Christopher D. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 423–430, Sapporo, Japan.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, pages 75–82, Ann Arbor, Michigan.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553.

Adam Pauls, Dan Klein, and Chris Quirk. 2010. Top-down k-best a* parsing. In *In proceedings of the Annual Meeting on Association for Computational Linguistics Short Papers*, ACLShort '10, pages 200–204, Morristown, NJ, USA.

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.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713, Cambridge, MA, October.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

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.

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

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of ME and SVM via integer linear programming. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 233–236, Morristown, NJ, USA.

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.