Exponential Decay Pruning for Bottom-Up Beam-Search Parsing

Nathan Bodenstab, Aaron Dunlop, Brian Roark
Oregon Health & Science University
Portland, Oregon, USA
{bodenstab|dunlopa|roark}@cslu.ogi.edu

Keith Hall
Google, Inc.
Zurich, Switzerland
kbhall@google.com

Abstract

We describe and motivate bottom-up beam-search parsing, a probabilistic constituent parsing architecture that combines the advantages of best-first agenda parsing and bottom-up chart parsing into a unified framework. We also present Exponential Decay Pruning (EDP), a novel beam-width pruning technique that is simple and effective, increasing parsing speed up to 43% with no loss in accuracy. Using these techniques together, our system requires no knowledge of the underlying grammar, has a small memory footprint, and parses at state-of-the-art speed and accuracy levels. We demonstrate the flexibility of our approach by parsing with five different grammars and validate its effectiveness by comparing it to both the best-first and coarse-to-fine parsing algorithms.

1 Introduction

The syntactic parsing community has seen significant advances in accuracy over the past ten years, primarily due to grammar engineering. Probabilistic context-free grammars have increased from a few thousand productions to several million via linguistically inspired non-terminal annotation (Johnson, 1998; Klein and Manning, 2003a) as well as automatic grammar induction techniques (Matsuzaki et al, 2005; Petrov et al, 2006). The resulting distribution over these grammars is much more sensitive to contextual factors, but at the cost of dramatically increasing the solution space.

Unfortunately, the exhaustive CYK chart parsing algorithm becomes ineffective with these large grammars. Its computational complexity is $O(n^3|G|)$ where n is the length of the sentence and $|G|$ is the number of grammar productions (a non-negligible constant). Searching over this entire space, even with the help of dynamic programming, is simply too slow.

Efficient search through the space created by these large grammars is a difficult problem, one that is often approached in a grammar-specific way. We believe parsing algorithms that can efficiently search through any grammar space without customizations have a distinct advantage over other methods in that they decouple decoding from grammar induction and provide the freedom to pursue improvements in each domain separately.

Towards this goal, we propose Exponential Decay Pruning (EDP) for bottom-up beam-search parsing, an adaptive beam thresholding technique that adjusts the beam size according to the confidence we have in a constituent's likelihood of participating in a complete parse tree. Beam thresholding is a well-known technique to prune the search space within a statistical parser. During bottom-up chart parsing, beam thresholding compares constituents covering the same span (i.e., in the same chart cell) and prunes candidates that fall outside the beam width; instead of retaining all constituents allowed by the grammar, only the most promising subset is retained. Eliminating constituents that cover short spans has a cascading effect that eliminates future constituents of longer spans and can decrease parsing time significantly. We describe and motivate bottom-up beam-search in Section 3, and discuss how Exponential Decay Pruning fits into this framework in Section 5.

In addition, we observe that many figure-of-merit functions have been proposed to rank the quality of individual constituents (Caraballo and Charniak, 1997; Klein and Manning, 2003b), but the application of these metrics have primarily been used in a best-first parsing architecture (also called agenda-based parsing). The utility of these figure-of-merit functions to prune at a local

level has not been fully explored and many state-of-the-art parsing systems only use simple heuristics when pruning at this level (Goodman, 1997; Collins, 1999; Charniak, 2000). Furthermore, when sophisticated techniques have been used within the beam-search framework (Xiong, et al., 2005) it is unclear how they perform relative to other parsing architectures or if performance gains are tied to a particular grammar.

The key contributions of this paper are as follows: (1) demonstration that the simple combination of beam-search pruning with an appropriate figure-of-merit function is sufficient to match the speed and accuracy of other state-of-the-art syntactic parsers; (2) the introduction of Exponential Decay Pruning, which decreases parsing time by 40% relative to a constant beam-width on a variety of grammars, without accuracy loss; (3) a novel decomposition of the Caraballo and Charniak (1997) figure-of-merit such that each figure-of-merit computation during decoding is constant-time instead of linear with the length of the input string.

2 Pruning the Search Space

Exhaustive search of the solution space is prohibitive for many natural language processing problems and pruning the search space based on local or global heuristics is a popular alternative to achieve efficient inference. The classic tradeoff of speed vs. accuracy comes into play when deciding how to prune. Exhaustive search is at the "accuracy" end of the spectrum and finds the globally optimal solution according to the model, but does so very slowly. At the other extreme, a majority of the search space is not even considered; finding a solution may be fast, but the quality of the solution will often be sub-optimal. The art of pruning is to find the best operating point between these two extremes such that searching the solution space is both efficient and accurate.

Many pruning techniques have been applied to constituent-based probabilistic chart parsing and we categorized them into three bins: local cell pruning, global cell pruning, and coarse-to-fine pruning.

2.1 Local Cell Pruning

Local cell pruning compares constituents that cover the same span and prunes constituents that are relatively unlikely. Beam-search at the span level is a type of local cell pruning and is used by many parsers, for instance Collins (1999) and Charniak (2000). Constituents can fall below the beam and be pruned in two ways: first, have a probability sufficiently below the best local constituent, or second, have a low rank when all local constituents are ordered.

2.2 Global Cell Pruning

Global cell pruning is an orthogonal method of decreasing the search space and uses non-local information to predict the structure of the final parse tree. If there is high confidence that a particular chart cell will not contain any constituents of the final parse tree, the entire cell is pruned and all constituents contained in that cell are eliminated from consideration during decoding. Examples include Roark & Hollingshead's Cell Constraints (2008) and Collins' comma raising (1999).

2.3 Coarse-to-Fine Pruning

Coarse-to-fine pruning, also known as multi-pass parsing, (Goodman, 1997; Charniak and Johnson, 2005; Petrov and Klein, 2007) is a refined version of global cell pruning. The sentence is first parsed with a simplified version of the final grammar, which runs relatively quickly. The posterior probabilities from the resulting "coarse" parse trees are then used to prune their corresponding constituents in the "fine" grammar space. Instead of eliminating all constituents in a given chart cell, as is done by global cell pruning, the coarse-to-fine algorithm prunes individual non-terminals per chart cell and can be more selective with this additional level of granularity.

A mapping of fine non-terminals to coarse non-terminals is necessary to prune the search space in this fashion. For example, Goodman 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. The coarse non-terminal *NP* would map to all *NP-X* non-terminals in the fine grammar where *X* represents the set of possible head-word labels. Obviously, the coarse grammar here is determined out of convenience and it is unlikely that this would be the optimal grammar for pruning the fine grammar space. Petrov and Klein (2007) derive coarse grammars in a more principled way, although the technique is closely tied to their latent variable grammar representation.

The pruning strategies above have all been shown to eliminate significant portions of the

search space effectively. In the next section we introduce bottom-up beam-search parsing, an algorithm that accommodates both local and global cell pruning. Together, these techniques provide a competitive alternative to coarse-to-fine pruning with the important advantage that they operate exclusively in the target grammar space, eliminating the need to derive or store multiple grammars as well as learn (or heuristically assign) non-terminal mappings between the grammar levels.

3 Bottom-Up Beam-Search Parsing

Bottom-up beam-search parsing combines the advantages of bottom-up chart parsing and best-first agenda parsing into a single framework. The CYK algorithm is the standard approach to bottom-up chart parsing and makes exhaustive parsing tractable through the use of dynamic programming. Additionally, only a single constituent is stored for each non-terminal in each cell, resulting in a maximum of $n^2|NT|$ entries, where n is the length of the sentence and $|NT|$ is the size of the non-terminal set. The downfall of the CYK algorithm is that it exhaustively traverses the entire search space, no matter how unlikely. As probabilistic grammars grow larger and more flexible (especially smoothed grammars), the CKY algorithm simply takes too long to be of any practical use.

Best-first agenda based parsers, in contrast, maintain a global agenda of all currently-buildable constituents and place the best constituent into the chart at each iteration. Depending on the ranking function of the global agenda (called the figure-of-merit function), best-first parsing may traverse only a fraction of the search space to find a complete syntactic parse of the input sentence. But this efficient search comes at a cost: in the worst case, it is required to keep all $n^3|NT|^3$ constituents sorted in the global agenda. Maintaining such an agenda guarantees that the algorithm will explore a smaller search space than the CYK algorithm, but it does not guarantee that the time to parse a sentence will be less due to the added overhead of ordering constituents in the global agenda. Improvements in best-first parse speed are a direct consequence of improvements in the figure-of-merit ranking function, which we discuss in more detail in Section 4.

Bottom-up beam-search parsing situates itself in between these two methods. Given a figure-of-merit (FOM) function that scores chart constituents, we traverse the chart cells in bottom-up order. At each cell, we build a *local* agenda and rank constituents by their FOM scores, pruning all entries that fall below a given rank or score difference from the best local constituent. Pseudo code for this algorithm can be seen in Figure 1.

```
Input: c: dynamic programming chart initialized with input
       sentence and POS tags
    n: length of input sentence
    b1: absolute beam count threshold
    b2: relative beam percentage threshold
Output: c: dynamic programming chart filled with constituent entries
Functions: Agenda(): creates a new agenda
    GenConstituents(c, beg, end): return a list of all buildable constituents that span the substring from beg to end according to the CYK algorithm
    AddConstituent(c, beg, end, x): add constituent x to the chart c in the chart cell that spans the substring beg to end
    Push(x, A): pushes x onto agenda A
    Pop(A): removes maximum value from agenda A
    Peek(A): returns maximum values from agenda A without removing the element

procedure BeamSearchParser(c, n, b1, b2)
 for span ← 1 to n do
  for beg ← 0 to n – span do
   end ← beg + span
   A ← Agenda()
   for each x in GenConstituents(c, beg, end) do
    Push(x, A)
   end
   minScore ← Peek(A) * b2
   i ← 0
   while i < b1 and Peek(A) > minScore begin
    x ← Pop(A)
    AddConstituent(c, beg, end, x)
    i ← i + 1
   end
  end
 end
```

Figure 1: Pseudo code for the bottom-up beam-search parsing algorithm

When using the same FOM function, the best-first algorithm will often add fewer constituents to the chart than the beam-search algorithm because the merit of each constituent is compared at a global level instead of a local level. For example, if all constituents in a cell are globally very poor (i.e., they are unlikely to participate in a final parse) the beam-search algorithm will still place the best of these constituents into the chart since they are locally optimal, whereas the best-first algorithm would not. Nevertheless, the bot-

tom-up beam-search algorithm is superior for three reasons:

1) **Memory** – constituents in beam-search parsing are only analyzed on a per-cell basis meaning there are at most $n\sigma^2|NT|$ entries temporarily stored in the local agenda at any time, where σ is the beam width[1]. Once entries for a cell have been processed, the local agenda can be discarded and at most σ entries per cell are permanently stored, leading to an upper bound of $n^2\sigma$ constituents in the entire chart. The best-first global agenda, on the other hand, must keep all buildable constituents for all chart cells in the global agenda until a complete parse is found, a maximum of $n^3|NT|^3$ entries[2]. In practice, the difference in memory consumption is as much as an order of magnitude on long sentences.

2) **Speed** – maintaining a sorted global agenda is not only costly in terms of memory, but also speed. Restricting comparisons to a local level as bottom-up beam-search does is more efficient, as demonstrated in Section 6.

3) **Comparability** – as Blaheta and Charniak (1999) point out, it is not intuitive to compare the merit of constituents at a global level. Is a *NP* covering the first three words of a sentence better or worse than a *VP* covering a different substring in the same sentence? Instead of side-stepping this problem with heuristic normalization factors and tuning parameters, bottom-up beam-search parsing only compares constituents covering the same words. In terms of comparability, this is the correct thing to do.

4 Boundary Figure-of-Merit

When choosing which constituents to place in a chart cell, the existence of a good ranking function is critical. But finding a ranking function that is both effective and efficient is a difficult problem. Since the inside probability, $\beta(\cdot)$, of each constituent is readily available in bottom-up parsing, it is often used as the comparison metric. But as Caraballo and Charniak (C&C) (1997) note, this metric answers the incorrect question "Which derivation best covers this span?" Instead, we should ask "Which derivation will best participate in a complete parse of the entire sentence?" To answer the second question, C&C propose a collection of methods to estimate the outside probability, $\alpha(\cdot)$, of an incomplete parse tree. Together, the inside and outside probabilities can be combined into a single FOM score, which is used to rank a constituent's relative merit of inclusion in the maximum likelihood parse tree.

We follow C&C's Boundary Trigram Estimate FOM with two modifications. First, C&C normalize the inside portion of their estimate by the trigram probability of the covered span. This relieves the bias of short spans over long spans in their global agenda, but because bottom-up beam-search parsing only compares spans within a single chart cell, all of which cover the same substring, we can ignore this term. Second, C&C assume gold part-of-speech tags, an assumption that simplifies the calculation of their estimate. We borrow from Hall and Johnson (2004) and incorporate the forward-backward part-of-speech probabilities of the outside tags, which are necessary when the part-of-speech tags are ambiguous.

Considering these two modifications, we compute the Boundary FOM as follows: given a sentence of length n: $w_0, w_1, \ldots, w_n$, with ambiguous part-of-speech tags $T_i^0, T_i^1, \ldots T_i^{p_i}$ for each word at index i, we compute the figure-of-merit for non-terminal N^x spanning words $w_j, w_{j+1}, \ldots, w_{k-1}, w_k$ as:

$$FOM(N_{j,k}^x) = \alpha_{left}(N_j^x)\beta(N_{j,k}^x)\alpha_{right}(N_k^x) \quad (1)$$

$$\alpha_{left}(N_j^x) = \max_q \left[fwd\left(T_{j-1}^q\right)P(N^x|T^q)\right]$$

$$\alpha_{right}(N_k^x) = \max_r [P(T^r|N^x)bkwd(T_{k+1}^r)]$$

The $fwd(\cdot)$ score computes the probability of the tag sequence from the beginning of the sentences to the designated part-of-speech tag; the $bkwd(\cdot)$ function does this in reverse, starting at the end of the sentence. The semiring used to compute these scores must be the same as that used by the inside probability $\beta(\cdot)$. In all our experiments, we use the Tropical semiring.

Note that using the Boundary FOM adds an additional step to the initialization of the CYK algorithm. After populating the base chart cells with the part-of-speech tags for each word, the forward-backward algorithm is run over these tags to generate $\alpha_{left}(\cdot)$ and $\alpha_{right}(\cdot)$ scores for all possible word-index/non-terminal pairs. The runtime of this pre-computation scales linearly

[1] Maximum local agenda entries: $\sigma^2|NT|$ grammar rules (σ left/right children and $|NT|$ possible parents) for n possible midpoints

[2] Maximum global agenda entries: All grammar rules (maximum of $|NT|^3$ entries) for n^2 chart cells and n possible midpoints

with the length of the sentence and is dwarfed by the cubic complexity of the subsequent parsing algorithm. Computing the conditional probabilities of constituents occurring before or after specific part-of-speech tags does not affect decoding time since these probabilities are computed offline and can be retrieved in constant-time.

Because we decouple the outside estimate in Equation 1 into left and right scores, the figure-of-merit for each constituent can also be computed in constant-time. To our knowledge, this is the first time this derivation has been presented and it contributes significantly to our final improvement in parsing speed since there can be tens of thousands of FOM calculations per sentence.

5 Exponential Decay Pruning

As discussed in Section 2, two thresholds exist in beam-search parsing to prune competing constituents in the same span: a minimum probability difference from the best constituent, and a maximum number of ranked entries. Traditionally, these two thresholds are assigned a constant value that is used during decoding for all sentences. Exponential Decay Pruning (EDP), on the other hand, is an adaptive thresholding technique that reduces the maximum number of entries relative to the span size and sentence length. As we parse larger spans, fewer constituents are allowed in each chart cell.

The motivation behind EDP stems from the observation that the figure-of-merit for larger spans is more accurate than for shorter spans. This is true for the Boundary figure-of-merit as well as the simple inside probability. When spans are smaller, the figure-of-merit is less reliable and we need to be more prudent when eliminating partial derivations from consideration. But once the inside probability can account for much of the final parse structure, our confidence in the figure-of-merit increases and we can prune more aggressively.

Equation 2 gives the EDP formula for computing the maximum number of constituents in a chart cell, given the initial baseline maximum number of constituents N_0, sentence length n, span width s, and tuning parameter λ.

$$N_{n,s} = N_0 \exp(-\lambda n^2 \frac{s}{n}) \quad (2)$$

$$= N_0 \exp(-\lambda n s) \quad (3)$$

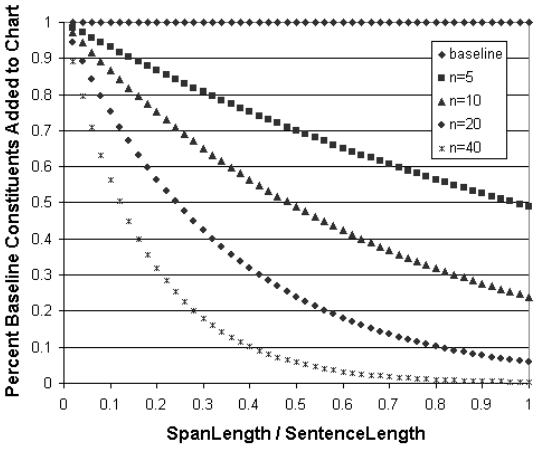

Figure 2: Maximum number of constituent entries for exponential decay pruning with λ=0.1. Four example sentence lengths are plotted along with the constant baseline value.

This formula has two goals. First, gradually decrease the number of constituents as the span width increases. The relative increase in span size is represented by the ratio $\frac{s}{n}$ in Equation 2 and corresponds to the time element in the exponential decay function. Second, we prune shorter sentences less aggressively since the gains of heavy pruning on these candidates are not as beneficial. This objective is realized by the n^2 element in Equation 2. Finally, the tuning parameter λ increases the steepness of decay for all sentence lengths and determines the overall degree of pruning.

We optimize λ on Section 24 of the WSJ Penn Treebank corpus and plot the maximum ranked entries allowed by EDP vs. the span's relative height in Figure 2 for four sentence lengths: 5, 10, 20, and 40. As the figure shows, a large number of constituents are pruned relative to the constant baseline value, even for short sentences. We also set a minimum number of constituents that must be added to each cell regardless of the EDP prediction (two for all our experiments).

6 Results

We evaluate the utility of bottom-up beam-search parsing and exponential decay pruning and put these results in context by comparing the accuracy and runtime of these techniques to the best-first and coarse-to-fine algorithms. We also report results across a wide range of grammars to demonstrate the generalizability of our methods.

	Binary	Unary	Lexical	Non-terms	POS tags
Right-factored, Markov order 0	3809	250	31460	100	48
Right-factored, Markov order 2	13210	250	31460	2915	48
Right-factored, Markov order 2, parent annotation	24570	772	31460	6711	48
Left-factored, Markov order 2, parent annotation	25229	772	31460	6971	48
Berkeley SM6 grammar	1320984	116959	2586996	1110	639

Table 1: Attributes of the grammars used in evaluation.

	Seconds	Edges	F-Score
Right-factored, Markov order 0			
Exhaustive CYK	0.11	56796	63.0
Best-First, Boundary	0.15	3891	63.4
Beam, Boundary, Constant	0.02	1175	63.2
Beam, Boundary, Decay	0.01	744	63.8
Right-factored, Markov order 2			
Exhaustive CYK	0.47	470925	71.7
Best-First, Boundary	0.22	2960	71.7
Beam, Boundary, Constant	0.07	2658	71.9
Beam, Boundary, Decay	0.04	1301	72.1
Right-factored, Markov order 2, parent annotation			
Exhaustive CYK	1.03	945855	76.0
Best-First, Boundary	0.15	748	75.4
Beam, Boundary, Constant	0.11	3752	76.0
Beam, Boundary, Decay	0.09	2750	76.1
Left-factored, Markov order 2, parent annotation			
Exhaustive CYK	1.16	685559	76.0
Best-First, Boundary	0.24	1237	74.8
Beam, Boundary, Constant	0.11	4097	76.4
Beam, Boundary, Decay	0.08	1813	76.3
Berkeley SM6 grammar			
Exhaustive CYK	94.1	163537	87.2
Best-First, Inside	138.0	152472	87.2
Beam, Inside, Constant	5.68	35501	87.2
Beam, Inside, Decay	3.10	20002	87.0
Best-First, Boundary	1.43	349	85.2
Beam, Boundary, Constant	0.62	7548	87.0
Beam, Boundary, Decay	0.37	5145	87.1

Table 2: Parsing results on development data: Section 24 of the WSJ Penn Treebank corpus

Table 1 lists important characteristics of each grammar used in evaluation. It is not the purpose of this paper to detail how each factor affects parsing nor the induction techniques involved in producing these grammars, but we will dwell for a moment one important point: the Berkeley SM6 grammar (Petrov and Klein, 2007) is a latent variable grammar where the latent variables are automatically derived to optimize parsing accuracy. Although the number of non-terminals is relatively small, the number of grammar productions is nearly two orders of magnitude larger than either of the parent annotated grammars. As seen in Table 2, this directly corresponds to an increase in average CYK parse time from 1.03 to 94.1 seconds per sentence – a similar two-orders-of-magnitude difference. Information on Markov grammar smoothing or parent annotation can be found in Manning and Schütze (2000) or Johnson (1998), respectively.

All timing tests were run on an Intel 3.00GHz processor with 6MB of cache and 16GB of memory. All parsing implementations listed are written in Java. Tuning parameters for beam thresholding and Exponential Decay Pruning were jointly estimated on Section 24 of the WSJ Penn Treebank corpus.

Table 2 compares parsing performance on the development data. The three metrics we report are average seconds per sentence (Seconds), average constituents added to the chart (Edges), and total F-Score (F-Score), where F-Score is the harmonic mean between the precision and recall on a per-constituent basis.

There are many interesting comparisons in Table 2. First, CYK performance is heavily dependent on the grammar in terms of accuracy and speed. Although parsing with the Markov order 0 grammar is fast, parse trees with an F-score of 63.0 have limited utility for most NLP applications. Likewise, parsing with the Berkeley SM6 grammar is accurate, but disappointingly slow.

Second, when using the Boundary figure-of-merit, both the best-first and beam-search algorithms prune a large portion of the search space and perform substantially better than CYK. The benefit from these parsing strategies increases with the size of the grammar and the largest gains are a 100-fold decrease in parsing time for the Berkeley grammar. The one exception to this

trend is best-first parsing with the Markov order 0 grammar; because the grammar is so impoverished, the figure-of-merit scores between "good" and "bad" constituents are relatively small and the best-first algorithm must search a large portion of the solution space. Consequently, the overhead of maintain the global best-first agenda nullifies the potential gains of pruning the search space.

Third, in all trials beam-search with a constant beam-width outperforms best-first search. Furthermore, using Exponential Decay Pruning ("Decay" in Table 2) within the beam-search framework is always superior to a constant beam-width, by as much as much as 40%.

Fourth, EDP is effective across different FOM functions. We simplify the FOM to only use the inside probability $\beta(\cdot)$ and run comparisons with the Berkeley SM6 grammar. Note that the best-first algorithm actually runs slower than the exhaustive CYK algorithm since it must maintain a global agenda of nearly all possible constituents. Beam-search, on the other hand, is 95% faster than CYK and using EDP decrease the parse time by an additional 45%.

We also observe that best-first F-score accuracies are consistently worse when parsing with the Berkeley or parent annotated grammars. This is because the outside estimate of the Boundary FOM may overestimate the true outside probability of a constituent, leading the best-first algorithm to find a complete parse tree that maximizes the FOM function instead of the grammar. One technique used to remedy this is called overparsing (Charniak et al, 1998) which continues to search for additional results with a higher grammar score *after* the first completed parse tree is found. This works well for increasing F-Score, but will also always increase the time to parse.

Next, we compare our method to the coarse-to-fine algorithm under equal conditions by applying our optimal configuration on the development set to the unseen test data, Section 23 of the WSJ. Both parsers use the same Berkeley SM6 grammar and both are written in Java. Table 3 displays the results.

We first point out that the trends we saw on the development set carry over: beam-search continues to outperform best-first, and exponential decay pruning is 38% faster than using a constant beam-width. In addition, we explained in Section 5 that EDP is a local pruning strategy that compares constituents covering the same span. Global pruning strategies, such as Roark and Hollingshead's Cell Constraints (2008) are orthogonal to EDP and can be combined to reduce the total search space even further.

Berkeley SM6 grammar	**Seconds**	**F-Score**
Exhaustive CYK	76.63	88.0
Best-First, Boundary	1.25	86.1
Beam, Boundary, Constant	0.45	87.9
Beam, Boundary, Decay	0.28	88.0
Beam, Boundary, Decay, Global	0.16	88.3
Berkeley Coarse-to-Fine Viterbi	0.21	88.3

Table 3: Parsing results on test data: Section 23 of the WSJ Penn Treebank corpus

We apply Cell Constraints (Global) to our optimal parsing setup and report results in Table 3. The combination of EDP with Cell Constraints is remarkably complementary, as seen by the additional 43% reduction in parse time. F-Score also improves from 88.0 to 88.3. When compared to the 88.0 maximum likelihood F-Score, it suggests that we are correcting model errors in the grammar using additional information provided by these constraints.

Additional methods to improve accuracy, such as grammar smoothing (Jelinek, 1997), n-best extraction and re-ranking (Charniak and Johnson, 2005), Max-Rule decoding (Petrov and Klein, 2007), or averaging results over multiple grammars (Petrov, 2010) can all be applied to bottom-up best-first parsing. This is because the dynamic programming chart contains many high-probability constituents other than those in the 1-best parse tree. Adjusting the strength of EDP is an ideal tactic to increase the number of high-probability parse trees embedded in the chart.

7 Conclusion and Future Work

We have presented Exponential Decay Pruning, a new beam-search pruning technique for statistical parsing, and shown that it decreases parsing time by up to 40% when compared to a traditional constant beam threshold. In addition, EDP prunes portions of the search space orthogonal to other methods, and when combined, leads to a competitive parsing architecture that is as fast and as accurate as other state-of-the art parsers.

We have also shown how these pruning techniques fit into the larger framework of bottom-up beam-search parsing, a simple extension of CYK parsing that can find accurate parse trees efficiently. The beam-search parsing algorithm is superior to the best-first algorithm in terms of memory consumption, decoding speed, and con-

stituent comparability. It also has the advantage over coarse-to-fine pruning of operating exclusively in the target grammar space, eliminating the need to derive or store multiple grammars or assign non-terminal mappings between the grammar levels.

We see Exponential Decay Pruning as an important step towards the more general goal of learning which areas of the search space are most promising and how to explore those areas most effectively. In future work we plan to explore new figures-of-merit, which have the potential to both decrease the time to parse and generate more accurate parse trees.

References

Don Blaheta and Eugene Charniak. 1999. *Automatic compensation for parser figure-of-merit flaws*. In Proceedings of ACL.

Sharon A. Caraballo and Eugene Charniak. 1997. *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 ACL.

Eugene Charniak. 2000. *A Maximum-Entropy-inspired parser*. In Proceedings of NAACL.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. *Edge-Based Best-First Chart Parsing*. In 6th Annual Workshop for Very Large Corpora, pages 127-133.

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

Joshua Goodman. 1997. *Global Thresholding and Multiple-Pass Parsing*. In Proceedings of EMNLP.

Keith B. Hall and Mark Johnson. 2004. *Attention Shifting for Parsing Speech*. In Proceedings of ACL.

Frederick Jelinek. 1997. Statistical Methods for Speech Recognition. The MIT Press, Cambridge, Massachusetts.

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

Dan Klein and Christopher D. Manning. 2003a. *Accurate unlexicalized parsing*. In Proceedings of ACL.

Dan Klein and Christopher D. Manning. 2003b. *A* Parsing: Fast Exact Viterbi Parse Selection*. In Proceedings of NAACL.

Christopher Manning and Hinrich Schütze. 2000. Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. *Probabilistic CFG with latent annotations*. In Proceedings of ACL.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. *Learning Accurate, Compact, and Interpretable Tree Annotation*. In Proceedings of ACL.

Slav Petrov and Dan Klein. 2007. *Improved inference for unlexicalized parsing*. In Proceedings of HLT-NAACL.

Slav Petrov. 2010. *Products of Random Latent Variable Grammars*. In Proceedings of NAACL.

Brian Roark and Kristy Hollingshead. 2008. *Classifying chart cells for quadratic complexity context-free inference*. In Proceedings of COLING.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2005. *Lexicalized Beam Thresholding Parsing with Prior and Boundary Estimates*. In Proceedings of CICLing.