Effective Inference for Generative Neural Parsing

Mitchell Stern **Daniel Fried** **Dan Klein**
Computer Science Division
University of California, Berkeley
{mitchell,dfried,klein}@cs.berkeley.edu

Abstract

Generative neural models have recently achieved state-of-the-art results for constituency parsing. However, without a feasible search procedure, their use has so far been limited to reranking the output of external parsers in which decoding is more tractable. We describe an alternative to the conventional action-level beam search used for discriminative neural models that enables us to decode directly in these generative models. We then show that by improving our basic candidate selection strategy and using a coarse pruning function, we can improve accuracy while exploring significantly less of the search space. Applied to the model of Choe and Charniak (2016), our inference procedure obtains 92.56 F1 on section 23 of the Penn Treebank, surpassing prior state-of-the-art results for single-model systems.

1 Introduction

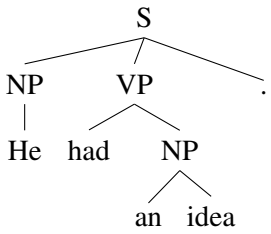
A recent line of work has demonstrated the success of generative neural models for constituency parsing (Dyer et al., 2016; Choe and Charniak, 2016). As with discriminative neural parsers, these models lack a dynamic program for exact inference due to their modeling of unbounded dependencies. However, while discriminative neural parsers are able to obtain strong results using greedy search (Dyer et al., 2016) or beam search with a small beam (Vinyals et al., 2015), we find that a simple action-level approach fails outright in the generative setting. Perhaps because of this, the application of generative neural models has so far been restricted to reranking the output of external parsers.

Intuitively, because a generative parser defines a joint distribution over sentences and parse trees, probability mass will be allocated unevenly between a small number of common structural actions and a large vocabulary of lexical items. This imbalance is a primary cause of failure for search procedures in which these two types of actions compete directly. A notion of equal competition among hypotheses is then desirable, an idea that has previously been explored in generative models for constituency parsing (Henderson, 2003) and dependency parsing (Titov and Henderson, 2010; Buys and Blunsom, 2015), among other tasks. We describe a related state-augmented beam search for neural generative constituency parsers in which lexical actions compete only with each other rather than with structural actions. Applying this inference procedure to the generative model of Choe and Charniak (2016), we find that it yields a self-contained generative parser that achieves high performance.

Beyond this, we propose an enhanced candidate selection strategy that yields significant improvements for all beam sizes. Additionally, motivated by the look-ahead heuristic used in the top-down parsers of Roark (2001) and Charniak (2010), we also experiment with a simple coarse pruning function that allows us to reduce the number of states expanded per candidate by several times without compromising accuracy. Using our final search procedure, we surpass prior state-of-the-art results among single-model parsers on the Penn Treebank, obtaining an F1 score of 92.56.

2 Common Framework

The generative neural parsers of Dyer et al. (2016) and Choe and Charniak (2016) can be unified under a common shift-reduce framework. Both systems build parse trees in left-to-right depth-first order by executing a sequence of actions, as illustrated in Figure 1. These actions can be grouped



(S (NP He NP) (VP had (NP an idea NP) VP) . S)

Figure 1: A parse tree and the action sequence that produced it, corresponding to the sentence "He had an idea." The tree is constructed in left-to-right depth-first order. The tree contains only nonterminals and words; part-of-speech tags are not included. OPEN(X) and CLOSE(X) are rendered as "$(X$" and "$X)$" for brevity.

into three major types: OPEN(X) and CLOSE(X), which open and close a constituent with nonterminal X,[1] respectively, and SHIFT(x), which adds the word x to the current constituent. The probability of an action sequence $(a_1, \ldots, a_T)$ is

$$P(a_1, \ldots, a_T) = \prod_{t=1}^{T} P(a_t \mid a_1, \ldots, a_{t-1})$$
$$= \prod_{t=1}^{T} [\text{softmax}(\mathbf{W}\mathbf{u}_t + \mathbf{b})]_{a_t},$$

where $\mathbf{u}_t$ is a continuous representation of the parser's state at time t, and $[\mathbf{v}]_j$ denotes the jth component of a vector $\mathbf{v}$. We refer readers to the respective authors' papers for the parameterization of $\mathbf{u}_t$ in each model.

In both cases, the decoding process reduces to a search for the most probable action sequence that represents a valid tree over the input sentence. For a given hypothesis, this requirement implies several constraints on the successor set (Dyer et al., 2016); e.g., SHIFT(x) can only be executed if the next word in the sentence is x, and CLOSE(X) cannot be executed directly after OPEN(X).

3 Model and Training Setup

We reimplemented the generative model described in Choe and Charniak (2016) and trained it on the Penn Treebank (Marcus et al., 1993) using

[1]The model described in Dyer et al. (2016) has only a single CLOSE action, whereas the model described in Choe and Charniak (2016) annotates CLOSE(X) actions with their nonterminals. We present the more general version here.

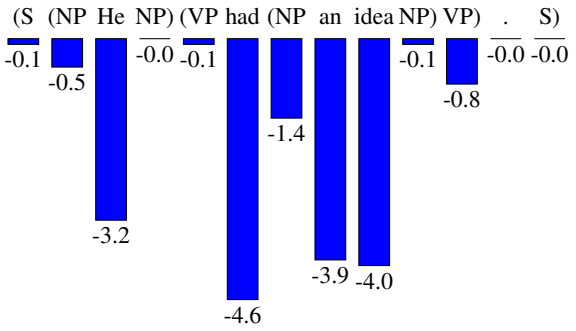


Figure 2: A plot of the action log probabilities $\log P(a_t \mid a_1, \ldots, a_{t-1})$ for the example in Figure 1 under our main model. We observe that OPEN and CLOSE actions have much higher probability than SHIFT actions. This imbalance is responsible for the failure of standard action-level beam search.

their published hyperparameters and preprocessing. However, rather than selecting the final model based on reranking performance, we instead perform early stopping based on development set perplexity. We use sections 2-21 of the Penn Treebank for training, section 22 for development, and section 23 for testing. The model's action space consists of 26 matching pairs of OPEN and CLOSE actions, one for each nonterminal, and 6,870 SHIFT actions, one for each preprocessed word type. While we use this particular model for our experiments, we note that our subsequent discussion of inference techniques is equally applicable to any generative parser that adheres to the framework described above in Section 2.

4 Action-Level Search

Given that ordinary action-level search has been applied successfully to discriminative neural parsers (Vinyals et al., 2015; Dyer et al., 2016), it offers a sensible starting point for decoding in generative models. However, even for large beam sizes, the following pathological behavior is encountered for generative decoding, preventing reasonable parses from being found. Regardless of the sequence of actions taken so far, the generative model tends to assign much higher probabilities to structural OPEN and CLOSE actions than it does lexical SHIFT actions, as shown in Figure 2. The model therefore prefers to continually open new constituents until a hard limit is reached, as the alternative at each step is to take the low-probability action of shifting the next word. The resulting

Beam Size k	200	400	600	800	1000	2000
$k_w = k$	87.47	89.86	90.98	91.62	91.97	92.74
$k_w = k/10$	89.25	91.16	91.83	92.12	92.38	92.93

Table 1: Development F1 scores using word-level search with various beam sizes k and two choices of word beam size k_w.

sequence typically has much lower overall probability than a plausible parse, but the model's myopic comparison between structural and lexical actions prevents reasonable candidates from staying on the beam. Action-level beam search with beam size 1000 obtains an F1 score of just 52.97 on the development set.

5 Word-Level Search

The imbalance between the probabilities of structural and lexical actions suggests that the two kinds of actions should not compete against each other within a beam. This leads us to consider an augmented state space in which they are kept separate by design, as was done by Fried et al. (2017). In conventional action-level beam search, hypotheses are grouped by the length of their action history $|A|$. Letting A_i denote the set of actions taken since the ith shift action, we instead group hypotheses by the pair $(i, |A_i|)$, where i ranges between 0 and the length of the sentence.

Let k denote the target beam size. The search process begins with the empty hypothesis in the $(0, 0)$ bucket. Word-level steps are then taken according to the following procedure for $i = 0, 1, \ldots$, up to the length of the sentence (inclusive). Beginning with the $(i, 0)$ bucket, the successors of each hypothesis are pooled together, sorted by score, and filtered down to the top k. Of those that remain, successors obtained by taking an OPEN or CLOSE action advance to the $(i, 1)$ bucket, whereas successors obtained from a SHIFT action are placed in the $(i+1, 0)$ bucket if i is less than the sentence length, or the completed list if i is equal to the sentence length. This process is repeated for the $(i, 1)$ bucket, the $(i, 2)$ bucket, and so forth, until the $(i+1, 0)$ bucket contains at least k hypotheses. If desired, a separate word beam size $k_w < k$ can be used at word boundaries, in which case each word-level step terminates when the $(i + 1, 0)$ bucket has k_w candidates instead of k. This introduces a bottleneck that can help to promote beam diversity.

Development set results for word-level search with a variety of beam sizes and with $k_w = k$ or $k_w = k/10$ are given in Table 1. We observe that performance in both cases increases steadily with beam size. Word-level search with $k_w = k/10$ consistently outperforms search without a bottleneck at all beam sizes, indicating the utility of this simple diversity-inducing modification. The top result of 92.93 F1 is already quite strong compared to other single-model systems.

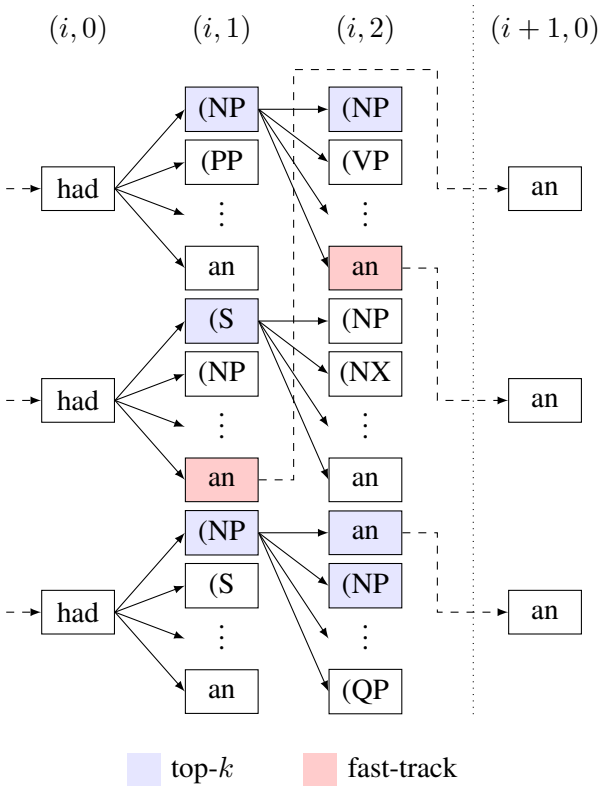


Figure 3: One step of word-level search with fast-track candidate selection (Sections 5 and 6) for the example in Figure 1. Grouping candidates by the current word i ensures that low-probability lexical actions are kept separate from high-probability structural actions at the beam level. Fast-track selection mitigates competition between the two types of actions within a single pool of successors.

6 Fast-Track Candidate Selection

The word-level beam search described in Section 5 goes one step toward ameliorating the issue that causes action-level beam search to fail, namely the direct competition between common structural actions with high probabilities and low-frequency shift actions with low probabilities. However, the issue is still present to some extent, in that successors of both types from a given bucket are pooled

Beam Size k	200	400	600	800	1000	2000
$k_w = k$	91.33	92.17	92.51	92.73	92.89	93.05
$k_w = k/10$	91.41	92.34	92.70	92.94	93.09	93.18

Table 2: Development F1 scores using the settings from Table 1, together with the fast-track selection strategy from Section 6 with $k_s = k/100$.

together and filtered down as a single collection before being routed to their respective destinations. We therefore propose a more direct solution to the problem, in which a small number $k_s \ll k$ of SHIFT successors are fast-tracked to the next word-level bucket *before* any filtering takes place. These fast-tracked candidates completely bypass competition with potentially high-scoring OPEN or CLOSE successors, allowing for higher-quality results in practice with minimal overhead. See Figure 3 for an illustration.

We repeat the experiments from Section 5 with $k_s = k/100$ and report the results in Table 2. Note that the use of fast-tracked candidates offers significant gains under all settings. The top result improves from 92.93 to 93.18 with the use of fast-tracked candidates, surpassing prior single-model systems on the development set.

7 OPEN Action Pruning

At any point during the trajectory of a hypothesis, either 0 or all 26 of the OPEN actions will be available, compared with at most 1 CLOSE action and at most 1 SHIFT action. Hence, when available, OPEN actions comprise most or all of a candidate's successor actions. To help cut down on this portion of the search space, it is natural to consider whether some of these actions could be ruled out using a coarse model for pruning.

7.1 Coarse Model

We consider a class of simple pruning models that condition on the $c \geq 0$ most recent actions and the next word in the sentence, and predict a probability distribution over the next action. In the interest of efficiency, we collapse all SHIFT actions into a single unlexicalized SHIFT action, significantly reducing the size of the output vocabulary.

The input $\mathbf{v}_t$ to the pruning model at time t is the concatenation of a vector embedding for each action in the context $(a_{t-c}, a_{t-c+1}, \ldots, a_{t-1})$ and a vector embedding for the next word w:

$$\mathbf{v}_t = [\mathbf{e}_{a_{t-c}}; \mathbf{e}_{a_{t-c+1}}; \ldots; \mathbf{e}_{a_{t-1}}; \mathbf{e}_w],$$

where each $\mathbf{e}_j$ is a learned vector embedding. The pruning model itself is implemented by feeding the input vector through a one-layer feedforward network with a ReLU non-linearity, then applying a softmax layer on top:

$$\begin{aligned} &P(a_t = a \mid a_1, \ldots, a_{t-1}, \text{next-word} = w) \\ &= P(a_t = a \mid a_{t-c}, \ldots, a_{t-1}, \text{next-word} = w) \\ &= [\text{softmax}(\mathbf{W}_2 \max(\mathbf{W}_1 \mathbf{v}_t + \mathbf{b}_1, 0) + \mathbf{b}_2)]_a. \end{aligned}$$

The pruning model is trained separately from the main parsing model on gold action sequences derived from the training corpus, with log-likelihood as the objective function and a cross entropy loss.

7.2 Strategy and Empirical Lower Bound

Once equipped with a coarse model, we use it for search reduction in the following manner. As mentioned above, when a hypothesis is eligible to open a new constituent, most of its successors will be obtained through OPEN actions. Accordingly, we use the coarse model to restrict the set of OPEN actions to be explored. When evaluating the pool of successors for a given collection of hypotheses during beam search, we run the coarse model on each hypothesis to obtain a distribution over its next possible actions, and gather together all the coarse scores of the would-be OPEN successors. We then discard the OPEN successors whose coarse scores lie below the top $1 - p$ quantile for a fixed $0 < p < 1$, guaranteeing that no more than a p-fraction of OPEN successors are considered for evaluation. Taking $p = 1$ corresponds to the unpruned setting.

This strategy gives us a tunable hyperparameter p that allows us to trade off between the amount of search we perform and the quality of our results. Before testing our procedure, however, we would first like to investigate whether there is a principled bound on how low we can expect to set p without a large drop in performance. A simple estimate arises from noting that the pruning fraction p should be set to a value for which most or all of the outputs encountered in the training set are retained. Otherwise, the pruning model would prevent the main model from even recreating the training data, let alone producing good parses for new sentences.

To this end, we collect training corpus statistics on the occurrences of inputs to the pruning function and their corresponding outputs. We then

c	1	2	3	4	5	6	7	8	9	10
0	20.0	58.4	82.4	91.0	94.9	96.8	97.9	98.6	98.9	99.2
1	54.9	80.5	91.1	95.9	97.7	98.8	99.5	99.8	99.9	100.0
2	61.2	85.0	93.8	97.4	98.6	99.5	99.8	99.9	100.0	100.0

Table 3: Cumulative distributions of the number of unique OPEN outputs per input for an order-c pruning function, computed over pruning inputs with at least one OPEN output.

p	6/26	7/26	8/26	9/26	10/26	11/26	1
Dev F1	92.78	93.00	93.08	93.13	93.19	93.19	93.18

Table 4: Results when the best setting from Section 6 is rerun with OPEN action pruning with context size $c = 2$ and various pruning fractions p. Lower values of p indicate more aggressive pruning, while $p = 1$ means no pruning is performed.

compute the number of unique OPEN actions associated with inputs occurring at least 20 times, and restrict our attention to inputs with at least one OPEN output. The resulting cumulative distributions for context sizes $c = 0, 1, 2$ are given in Table 3. If we require that our pruning fraction p be large enough to recreate at least 99% of the training data, then since there are 26 total nonterminals, approximate[2] lower bounds for p are $10/26 \approx 0.385$ for $c = 0$, $7/26 \approx 0.269$ for $c = 1$, and $6/26 \approx 0.231$ for $c = 2$.

7.3 Pruning Results

We reran our best experiment from Section 6 with an order-2 pruning function and pruning fractions $p = 6/26, \ldots, 11/26$. The results are given in Table 4. We observe that performance is on par with the unpruned setup (at most 0.1 absolute difference in F1 score) for p as low as $8/26 \approx 0.308$. Setting p to $7/26 \approx 0.269$ results in a drop of 0.18, and setting p to $6/26 \approx 0.231$ results in a drop of 0.40. Hence, degradation begins to occur right around the empirically-motivated threshold of $6/26$ given above, but we can prune $1 - 8/26 \approx 69.2\%$ of OPEN successors with minimal changes in performance.

8 Final Results and Conclusion

We find that the best overall settings are a beam size of $k = 2000$, a word beam size of $k_w = 200$, and $k_s = 20$ fast-track candidates per step, as this setup achieves both the highest probabilities under the model and the highest development F1. We report our test results on section 23 of the Penn Treebank under these settings in Table 5 both with and without pruning, as well as a number of other recent results. We achieve F1 scores of 92.56 on the test set without pruning and 92.53 when $1 - 8/26 \approx 69.2\%$ of OPEN successors are pruned, obtaining performance well above the previous state-of-the-art scores for single-model parsers. This demonstrates that the model of Choe and Charniak (2016) works well as an accurate, self-contained system. The fact that we match the performance of their reranking parser using the same generative model confirms the efficacy of our approach. We believe that further refinements of our search procedure can continue to push the bar higher, such as the use of a learned heuristic function for forward score estimation, or a more sophisticated approximate decoding scheme making use of specific properties of the model. We look forward to exploring these directions in future work.

[2]These thresholds are not exact due to the fact that our pruning procedure operates on collections of multiple hypotheses' successors at inference time rather than the successors of an individual hypothesis.

Parser	LR	LP	F1
Vinyals et al. (2015)	–	–	88.3
Shindo et al. (2012)	–	–	91.1
Cross and Huang (2016)	90.5	92.1	91.3
Dyer et al. (2016)	–	–	91.7
Liu and Zhang (2017)	91.3	92.1	91.7
Stern et al. (2017)	90.63	92.98	91.79
Our Best Result	92.57	92.56	92.56
Our Best Result (with pruning)	92.52	92.54	92.53
Vinyals et al. (2015) (ensemble)	–	–	90.5
Shindo et al. (2012) (ensemble)	–	–	92.4
Choe and Charniak (2016) (rerank)	–	–	92.6
Dyer et al. (2016) (rerank)	–	–	93.3
Fried et al. (2017) (ensemble, rerank)	–	–	94.25

Table 5: Comparison of F1 scores on section 23 of the Penn Treebank. Here we only include models trained without external silver training data. Results in the first two sections are for single-model systems.

Acknowledgments

MS is supported by an NSF Graduate Research Fellowship. DF is supported by an NDSEG fellowship.

References

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the*

Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 863–869, Beijing, China. Association for Computational Linguistics.

Eugene Charniak. 2010. Top-down nearly-context-sensitive parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 674–683. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada. Association for Computational Linguistics.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 24–31. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational linguistics*, 27(2):249–276.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–448, Jeju Island, Korea. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2010. A latent variable model for generative dependency parsing. In *Trends in Parsing Technology*, pages 35–55. Springer.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.