Generative and Discriminative Latent Variable Grammars

Slav Petrov
Google Research
New York, NY 10011
slav@google.com
slav@google.com

Abstract

Latent variable grammars take an observed (coarse) treebank and induce more fine-grained grammar categories, that are better suited for modeling the syntax of natural languages. Estimation can be done in a generative or a discriminative framework, and results in the best published parsing accuracies over a wide range of syntactically divergent languages and domains. In this paper we highlight the commonalities and the differences between the two learning paradigms.

1 Introduction

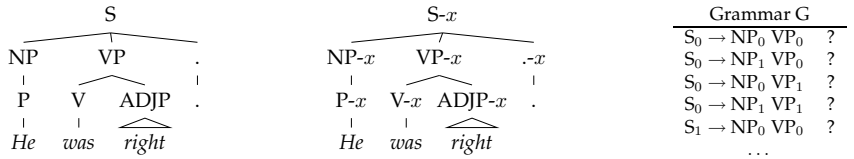
Latent variable grammars for parsing [1, 2] model an observed treebank of coarse *parse* trees with a model over more refined, but unobserved, *derivation* trees. Given sentences as input, the parse trees represent the desired output of the system, while the derivation trees represent the typically much more complex underlying syntactic processes. For example, the single treebank category NP (noun phrase) may be better modeled by several finer categories representing subject NPs, object NPs, and so on. Rather than attempting to manually specify these fine-grained subcategories, we automatically split each category into k subcategories (e.g. NP_1 - NP_k) and induce them from data.

Learning latent variable grammars therefore encompasses two tasks: determining the number of subcategories for each observed grammar category (some categories are more complex than others and we therefore expect them to have more subcategories), and learning the parameters of the grammar (i.e. the probabilities of the different grammar productions). In this paper, we review our past work on both generative [2, 3], as well as, discriminative [4, 5] latent variable grammars. While the final parsing accuracies are comparable, we highlight the unique challenges involved in estimating the models under the different paradigms. In our generative approach [2], we constrain model complexity using a split-merge heuristic and use the EM algorithm for parameter estimation. Alternatively, we can also accomplish both tasks together by including a prior in our objective function. In [3], we take a Bayesian standpoint and use a sparse Dirichlet prior and variational EM for learning. In our discriminative work [5], we include an L_1-regularization term in our objective function, which we maximize using numerical optimization (L-BFGS).

In addition to reviewing the technical differences of learning generative and discriminative latent variable grammars, we also compare the resulting grammars with each other. We show that even though the final accuracies of the two approaches are usually comparable, the underlying models are quite different and make different types of errors.

2 Latent Variable Grammars

In latent variable parsing, we view the training trees as a coarse trace of the true underlying processes. By augmenting the trees with a latent variable at each node (see Figure 1), we can model



(a) Parse tree T, Sentence w (b) Derivations $t : T$ (c) Parameters θ

Figure 1: The observed parse trees T (a) are split into derivations t (b). Learning latent variable grammars involves determining the set of grammar productions/features and their parameters (c).

this more refined process. Our log-linear grammars are parametrized by a vector θ which is indexed by productions $X \to \gamma$. The conditional probability of a derivation t given a sentence w is then:

$$\mathrm{P}_\theta(t|w) = \frac{1}{Z(\theta, w)} \prod_{X \to \gamma \in t} e^{\theta_{X \to \gamma}} = \frac{1}{Z(\theta, w)} e^{\theta^\mathsf{T} f(t)}, \tag{1}$$

where $Z(\theta, w)$ is the partition function and $f(t)$ is a vector indicating how many times each production occurs in the derivation t. The inside/outside algorithm gives us an efficient way of summing over this exponential set of derivations. We will consider generative grammars, where the parameters θ are set to maximize the joint likelihood of the training sentences and their parse trees, and discriminative grammars, where the parameters θ are set to maximize the likelihood of the correct parse tree (vs. all possible trees) given a sentence. Note that this is merely a comparison of different estimators, as probabilistic and weighted CFGs are equivalent [6].

2.1 Generative Grammars

Generative latent variables grammars can be seen as tree structured hidden Markov models. A simple EM algorithm [1] allows us to learn parameters for generative grammars which maximize the log joint likelihood of the training sentences w and parse trees T:

$$\mathcal{L}_{joint}(\theta) = \log \prod_i \mathrm{P}_\theta(w_i, T_i) = \log \prod_i \sum_{t:T_i} \mathrm{P}_\theta(w_i, t), \tag{2}$$

where t are derivations (over split categories) corresponding to the observed parse tree (over unsplit categories). In the E-Step we compute inside/outside scores over the set of derivations corresponding to the observed gold tree (in linear time), which allows us to compute expectations over the grammar productions.These expectations are then normalized in the M-Step to update the production probabilities $\phi_{X \to \gamma} = e^{\theta_{X \to \gamma}}$ to their maximum likelihood estimates:

$$\phi_{X \to \gamma} = \frac{\sum_T \mathbb{E}_\theta[f_{X \to \gamma}(t)|T]}{\sum_{\gamma'} \sum_T \mathbb{E}_\theta[f_{X \to \gamma'}(t)|T]} \tag{3}$$

Here, $\mathbb{E}_\theta\left[f_{X \to \gamma}(t)|T\right]$ denotes the expected count of the production (or feature) $X \to \gamma$ with respect to P_θ in the set of derivations t, which are consistent with the observed parse tree T. Similarly, we will write $\mathbb{E}_\theta\left[f_{X \to \gamma}(t)|w\right]$ for the expectation over all derivations of the sentence w.

To learn grammar complexity, we use a simple, yet powerful split-merge approach [2]. We iteratively refine the grammars in a hierarchical way. In each stage, all symbols are split in two, e.g. NP becoming NP_1 and NP_2, and then fit the model using the EM algorithm described above. After a splitting stage, half of the splits are rolled back based on (an approximation) to their likelihood gain. Empirically the gains level off after six split-merge rounds, and learning a generative grammar takes about 20 hours on a single CPU.

2.2 Discriminative Grammars

Discriminative latent variables grammars can be seen as conditional random fields [7] over trees. For discriminative grammars, we maximize the log conditional likelihood:

$$\mathcal{L}_{cond}(\theta) = \log \prod_i \mathrm{P}_\theta(T_i|w_i) = \log \prod_i \sum_{t:T_i} \frac{e^{\theta^\mathsf{T} f(t)}}{Z(\theta, w_i)} \tag{4}$$

Parser	≤ 40 words F1	EX	all F1	EX
ENGLISH				
Single-Scale Generative Latent Variable Grammars [1]	86.8	32.8	86.3	30.3
Single-Scale Discriminative Latent Variable Grammars [4]	88.8	35.7	88.3	33.1
Multi-Scale Discriminative Latent Variable Grammars [5]	90.0	**40.1**	89.4	**37.7**
Split-Merge Generative Latent Variable Grammars [10]	**90.6**	39.1	**90.1**	37.1
Lexicalized Generative Grammars [11]	90.3	39.6	89.7	37.2
Discriminative Reranking after Lexicalized Grammars [12]	**92.3**	**46.2**	**91.7**	**43.5**
GERMAN				
Split-Merge Generative Latent Variable Grammars [10]	80.8	40.8	80.1	39.1
Multi-Scale Discriminative Latent Variable Grammars [5]	**81.5**	**45.2**	**80.7**	**43.9**
Lexicalized Generative Grammars [13]	F_1 76.3		-	

Table 1: Test set parsing accuracies for latent variable grammars and other state-of-the-art methods.

We directly optimize this non-convex objective function using a numerical gradient based method (LBFGS [8] in our implementation). Fitting the log-linear model involves the following derivatives:

$$\frac{\partial \mathcal{L}_{cond}(\theta)}{\partial \theta_{X\to\gamma}} = \sum_i \Big(\mathbb{E}_\theta \left[f_{X\to\gamma}(t) | T_i \right] - \mathbb{E}_\theta [f_{X\to\gamma}(t) | w_i] \Big), \tag{5}$$

where the first term is the expected count of a production in derivations corresponding to the correct parse tree and the second term is the expected count of the production in all parses.

One of the main challenges in estimating discriminative grammars is that the computation of the derivatives requires repeatedly taking expectations over all parses of all sentences in the training set. To make this computation practical on large data sets, we extend the idea of coarse-to-fine parsing [9] to handle the repeated parsing of the same sentences. We cache computations of similar quantities between training iterations, allowing the efficient approximation of feature count expectations. Even with these approximations, training takes about three days on an 8-core CPU.

In practice, we add an L_1-regularization term to the objective function in Eq. 4 in order to control the complexity of the grammars. Multi-scale grammars [5] then take advantage of the sparsity produced by L_1-regularization by allowing some productions to reference fine categories, while others to reference coarse categories. As a result, a category such as NP can be complex in some regions of the grammar while remaining simpler in other regions, giving extremely compact grammars.

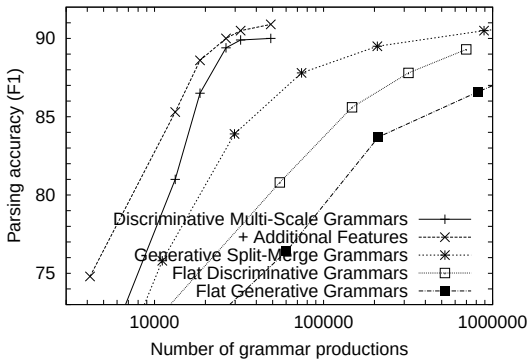
3 Experiments

To compare generative and discriminative latent variable grammars, we ran experiments on a broad range of languages. Due to space reasons Table 1 contains only an excerpt of our empirical results, and we refer the interested reader to [14] for a complete overview.

Figure 2 shows how parsing performance improves with the addition of more latent categories. In general, discriminatively trained grammars give better performance than their generative cousins, even when using an order of magnitude fewer parameters. However, the final accuracies of the generative models are slightly higher in terms of F1 because the split-merge approach seems to allocate the complexity in more meaningful way. Generative grammars are also significantly (20 times) faster to train, but the discriminative grammars allow for a convenient integration of additional (overlapping) features.

It is also interesting to examine how the complexity is allocated. While most categories have similar complexity in the two cases, the complexity of the two most rened phrasal categories are ipped. Generative grammars split NPs most highly, discriminative grammars split the VP. This distinction seems to be because the complexity of VPs is more syntactic (e.g. complex verb subcategorization), while that of NPs is more lexical (noun choice is generally higher entropy than verb choice).

We also examined the most likely parse trees produced by the different grammars and observed only a small overlap between the generative and discriminative grammars, despite their comparable F1 scores. Even in their top 50 lists, the overlap was less than 30%, suggesting that the grammars are modeling very different, and potentially complementary, linguistic phenomena.



	NP	VP	PP	S	SBAR	ADJP	ADVP	QP	PRN
Generative subcategories	32	24	20	12	12	12	8	7	5
Discriminative production parameters	19	32	20	14	14	8	7	9	6

Note that subcategories are compared to production parameters, indicating that the number of parameters grows cubicly in the number of subcategories for generative grammars, while growing linearly for multi-scale discriminative grammars.

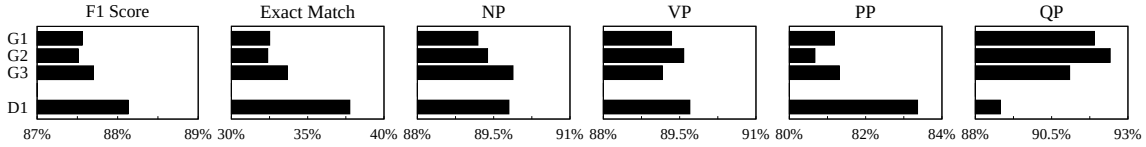


Figure 2: Top Left: Parsing accuracy vs. number of grammar productions. Top Right: Grammar complexity allocation. Bottom Row: Breakdown of different accuracy measures for three generative grammars (G_1-G_3), and one discriminative grammar (D_1).

Figure 2 also shows a more detailed error analysis for three generative and one discriminative grammar after three split-merge rounds. Not so surprisingly, there is a large, and maybe systematic, difference in the errors made by the generative and discriminative models. Interestingly, there is also a large difference between the different generative models that differ only in the random seed used for initialization. We leave the investigation of ensemble methods that can combine the strengths of the different grammars for future work.

References

[1] T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic CFG with latent annotations. In *ACL*, 2005.

[2] S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*, 2006.

[3] P. Liang, S. Petrov, M. I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP '07*, 2007.

[4] S. Petrov and D. Klein. Discriminative log-linear grammars with latent variables. In *NIPS '08*, 2008.

[5] S. Petrov and D. Klein. Sparse multi-scale grammars for discriminative latent variable parsing. In *EMNLP '08*, 2008.

[6] N. A. Smith and M. Johnson. Weighted and probabilistic context-free grammars are equally expressive. *Computational Lingusitics*, 2007.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01*, 2001.

[8] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[9] E. Charniak, S. Goldwater, and M. Johnson. Edge-based best-first chart parsing. 6^{th} *Workshop on Very Large Corpora*, 1998.

[10] S. Petrov and D. Klein. Improved inference for unlexicalized parsing. In *NAACL*, 2007.

[11] E. Charniak and M. Johnson. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. In *ACL'05*, 2005.

[12] L. Huang. Forest reranking: Discriminative parsing with non-local features. In *ACL '08*, 2008.

[13] A. Dubey. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *ACL '05*, 2005.

[14] S. Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, UC Berkeley, 2009.