Learning and Inference for Hierarchically Split PCFGs

Slav Petrov and Dan Klein
{petrov,klein}@cs.berkeley.edu
University of California, Berkeley,
Berkeley, CA, 94720

Abstract

Treebank parsing can be seen as the search for an optimally refined grammar consistent with a coarse training treebank. We describe a method in which a minimal grammar is hierarchically refined using EM to give accurate, compact grammars. The resulting grammars are extremely compact compared to other high-performance parsers, yet the parser gives the best published accuracies on several languages, as well as the best generative parsing numbers in English. In addition, we give an associated coarse-to-fine inference scheme which vastly improves inference time with no loss in test set accuracy.

Introduction

We present a general method for inducing structured models. Given labeled training data we extract a minimal model and show how to induce additional latent structure by iteratively refining the model. We then present an efficient inference procedure that takes advantage of the hierarchical structure of our model. While we illustrate our method on parsing natural language, the technique is applicable to other domains such as speech recognition and machine translation.

Parsing is the task of uncovering the syntactic structure of language and is often viewed as an important prerequisite for building systems capable of understanding language (see Lease *et al.* (2006) for an overview of parsing and its applications). Parsers are typically trained on a collection of hand parsed sentences (treebank). Because the constituents of the treebank imply unrealistic context-freedom assumptions, they are not well suited for modeling language. Therefore, a variety of techniques have been developed to both enrich and generalize the naive grammar by manually introducing annotations (Collins 1999; Klein & Manning 2003). In contrast, we induce latent structures without any additional human input, resulting in state-of-the art parsing performance on a variety of languages.

In the following we will focus on two problems: *learning*, in which we must select a model given a treebank, and *inference*, in which we must select a parse for a sentence given the learned model.

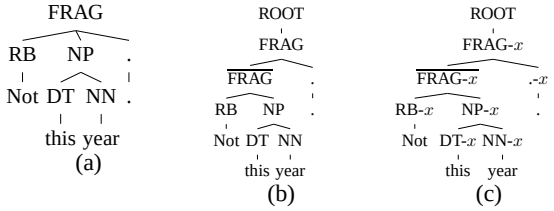

Figure 1: (a) The original tree. (b) The (binarized) X-bar tree. (c) The annotated tree.

Learning

To obtain a grammar from the training trees, we learn a set of rule probabilities on latent annotations that maximizes the likelihood of the training trees. The Expectation-Maximization (EM) algorithm allows us to do that, despite the fact that the original trees lack the latent annotations.

Initialization

We consider PCFG grammars which are derived from a raw treebank according to the method of Petrov *et al.* (2006), as follows: A simple X-bar grammar is created by binarizing the treebank trees; for each local tree rooted at an evaluation nonterminal X, we introduce a cascade of new nodes labeled $\overline{X}$ so that each has two children (Figure 1). Since we will evaluate our grammar on its ability to recover the Penn Treebank nonterminals, we must include them in our grammar. Therefore, this initialization is the absolute minimum starting grammar that includes the evaluation nonterminals (and maintains separate grammar symbols for each of them). It is a very compact grammar: 98 symbols,[1] 236 unary rules, and 3840 binary rules. However, it also has a very low parsing performance: 63.4% F_1[2] on the development set.

EM-Algorithm

Given a sentence w and its unannotated tree T, consider a nonterminal A spanning (r,t) and its children B and C spanning (r,s) and (s,t). Let A_x be a subsymbol of A, B_y of B, and C_z of C. Then the inside and outside probabilities $\mathrm{P_{IN}}(r,t,A_x) \stackrel{\text{def}}{=} P(w_{r:t}|A_x)$ and $\mathrm{P_{OUT}}(r,t,A_x) \stackrel{\text{def}}{=}$

[1] 45 part of speech tags, 27 phrasal categories and the 26 intermediate symbols which were added during binarization

[2] The harmonic mean of precision P and recall R: $\frac{2PR}{P+R}$.

$P(w_{1:r}A_x w_{t:n})$ can be computed recursively using the set of rule probabilities β (Matsuzaki, Miyao, & Tsujii 2005):

$$\begin{aligned}
\mathrm{P}_{\text{IN}}(r,t,A_x) &= \sum_{y,z} \beta(A_x \to B_y C_z) \\
&\quad \times \mathrm{P}_{\text{IN}}(r,s,B_y)\mathrm{P}_{\text{IN}}(s,t,C_z) \\
\mathrm{P}_{\text{OUT}}(r,s,B_y) &= \sum_{x,z} \beta(A_x \to B_y C_z) \\
&\quad \times \mathrm{P}_{\text{OUT}}(r,t,A_x)\mathrm{P}_{\text{IN}}(s,t,C_z) \\
\mathrm{P}_{\text{OUT}}(s,t,C_z) &= \sum_{x,y} \beta(A_x \to B_y C_z) \\
&\quad \times \mathrm{P}_{\text{OUT}}(r,t,A_x)\mathrm{P}_{\text{IN}}(r,s,B_y)
\end{aligned}$$

Although we show only the binary component here, of course there are both binary and unary productions that are included. In the Expectation step, one computes the posterior probability of each annotated rule and position in each training set tree T:

$$\begin{aligned}
P((r,s,t,A_x \to B_y C_z)|w,T) &\propto \mathrm{P}_{\text{OUT}}(r,t,A_x) \\
&\quad \times \beta(A_x \to B_y C_z)\mathrm{P}_{\text{IN}}(r,s,B_y)\mathrm{P}_{\text{IN}}(s,t,C_z) \qquad (1)
\end{aligned}$$

In the Maximization step, one uses the above probabilities as weighted observations to update the rule probabilities:

$$\beta(A_x \to B_y C_z) := \frac{\#\{A_x \to B_y C_z\}}{\sum_{y',z'} \#\{A_x \to B_{y'} C_{z'}\}}$$

Note that, because there is no uncertainty about the location of the brackets, this formulation of the inside-outside algorithm is linear in the length of the sentence rather than cubic (Pereira & Schabes 1992).

Splitting

EM is only guaranteed to find a local maximum of the likelihood, and, indeed, in practice it often gets stuck in a suboptimal configuration. If the search space is very large, even restarting may not be sufficient to alleviate this problem. We therefore repeatedly split and re-train the grammar. In each iteration we initialize EM with the results of the smaller grammar, splitting every previous annotation symbol in two and adding a small amount of randomness (1%) to break the symmetry. Hierarchical splitting leads to better parameter estimates over directly estimating a grammar with 2^k subsymbols per symbol. It is interesting to note that the induced splits are linguistically interpretable. As an example some of the learnt categories for the determiner part of speech are shown in Figure 2.

Merging

Creating more latent annotations results in a tighter fit to the training data but at the same time can lead to overfitting. Therefore, it would be to our advantage to split the latent annotations only where needed, rather than splitting them all. In addition, if all symbols are split equally often, one quickly (4 split cycles) reaches the limits of what is computationally feasible in terms of training time and memory usage. To prevent oversplitting, we could measure the utility of splitting each latent annotation individually and then split the best ones first. However, not only is this impractical, requiring an entire training phase for each new split, but it assumes the contributions of multiple splits are independent. In fact, extra subsymbols may need to be added to several nonterminals before they can cooperate to pass information along the parse tree. Therefore, we go in the opposite direction; that is, we split every symbol in two, train, and then measure for each annotation the loss in likelihood incurred when removing it. If this loss is small, the new annotation does not carry enough useful information and can be removed. What is more, contrary to the gain in likelihood for splitting, the loss in likelihood for merging can be efficiently approximated.

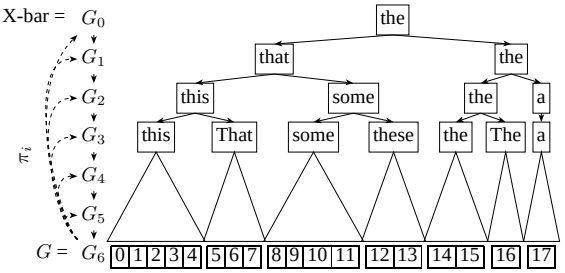

Figure 2: Hierarchical refinement proceeds top-down while projection recovers coarser grammars. The top word for the first refinements of the determiner tag is shown on the right.

Let T be a training tree generating a sentence w. Consider a node n of T spanning (r,t) with the label A; that is, the subtree rooted at n generates $w_{r:t}$ and has the label A. In the latent model, its label A is split up into several latent labels, A_x. The likelihood of the data can be recovered from the inside and outside probabilities at n:

$$\mathrm{P}(w,T) = \sum_x \mathrm{P}_{\text{IN}}(r,t,A_x)\mathrm{P}_{\text{OUT}}(r,t,A_x) \qquad (2)$$

Consider merging, at n only, two annotations A_1 and A_2. Since A now combines the statistics of A_1 and A_2, its production probabilities are the sum of those of A_1 and A_2, weighted by their relative frequency p_1 and p_2 in the training data. Therefore the inside score of A is:

$$\mathrm{P}_{\text{IN}}(r,t,A) = p_1\mathrm{P}_{\text{IN}}(r,t,A_1) + p_2\mathrm{P}_{\text{IN}}(r,t,A_2)$$

Since A can be produced as A_1 or A_2 by its parents, its outside score is:

$$\mathrm{P}_{\text{OUT}}(r,t,A) = \mathrm{P}_{\text{OUT}}(r,t,A_1) + \mathrm{P}_{\text{OUT}}(r,t,A_2)$$

Replacing these quantities in (2) gives us the likelihood $\mathrm{P}^n(w,T)$ where these two annotations and their corresponding rules have been merged, around only node n.

We approximate the overall loss in data likelihood due to merging A_1 and A_2 everywhere in all sentences w^i by the product of this loss for each local change:

$$\Delta_{\text{ANNOTATION}}(A_1,A_2) = \prod_i \prod_{n \in T_i} \frac{\mathrm{P}^n(w^i,T_i)}{\mathrm{P}(w^i,T_i)}$$

This expression is an approximation because it neglects interactions between instances of a symbol at multiple places in the same tree. These instances, however, are often far apart and are likely to interact only weakly, and this simplification avoids the prohibitive cost of running an inference algorithm for each tree and annotation. We refer to the operation of splitting annotations and re-merging some of them based on likelihood loss as a split-merge (SM) cycle. SM cycles allow us to progressively increase the complexity of our model, giving priority to the most useful extensions.

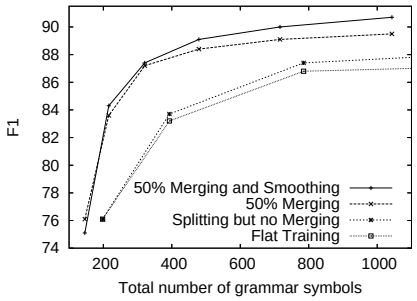

Figure 3: Hierarchical training leads to better parameter estimates. Merging reduces the grammar size significantly, while preserving the accuracy and enabling us to do more SM cycles. Parameter smoothing leads to even better accuracy.

Smoothing

Splitting nonterminals leads to a better fit to the data by allowing each annotation to specialize in representing only a fraction of the data. The smaller this fraction, the higher the risk of overfitting. Merging, by allowing only the most beneficial annotations, helps mitigate this risk, but it is not the only way. We can further minimize overfitting by forcing the production probabilities from annotations of the same nonterminal to be similar. For example, a noun phrase in subject position certainly has a distinct distribution, but may benefit from being smoothed with counts from other noun phrases. Smoothing the productions of each subsymbol by shrinking them towards their common base symbol gives a more reliable estimate, allowing them to share statistical strength.

We perform smoothing in a linear way. The estimated probability of a production $p_x = \mathrm{P}(A_x \rightarrow B_y\, C_z)$ is interpolated with the average over all subsymbols of A.

$$p'_x = (1-\alpha)p_x + \alpha\bar{p} \quad \text{where} \quad \bar{p} = \frac{1}{n}\sum_x p_x$$

Here, α is a small constant: we found 0.01 to be a good value, but the actual quantity was surprisingly unimportant.

Inference

Once we have learned a refined PCFG we can use it to do inference to predict the syntactic structure of a given sentence.

Coarse-to-Fine Approaches

When working with large grammars, it is standard to prune the search space in some way. A commonly adopted strategy is to use a *pre-parse* phase in which a sentence is rapidly parsed with a very coarse, treebank grammar. Any item $X{:}[i,j]$ with sufficiently low posterior probability in the preparse triggers the pruning of its refined variants in a subsequent full parse. In Petrov & Klein (2007), we proposed a novel multi-stage coarse-to-fine method which is particularly natural for our hierarchically split grammar, but which is, in principle, applicable to any grammar. We construct a sequence of increasingly refined grammars, reparsing with each refinement. The contributions of our method are that we derive sequences of refinements in a new way, we consider refinements which are themselves complex, and, because our full grammar is not impossible to parse with, we automatically tune the pruning thresholds on held-out data.

Projection

In our method, which we call *hierarchical coarse-to-fine* parsing, we consider a sequence of PCFGs $G_0, G_1, \ldots G_n = G$, where each G_i is a refinement of the preceding grammar G_{i-1} and G is the full grammar of interest. Each grammar G_i is related to $G = G_n$ by a *projection* $\pi_{n\rightarrow i}$ or π_i for brevity. A projection is a map from the nonterminal (including pre-terminal) symbols of G onto a reduced domain. A projection of grammar symbols induces a projection of rules and therefore entire non-weighted grammars (see Figure 2).

In our case, we also require the projections to be sequentially compatible, so that $\pi_{i\rightarrow j} = \pi_{k\rightarrow j} \circ \pi_{i\rightarrow k}$. That is, each projection is itself a coarsening of the previous projections. In particular, the projection $\pi_{i\rightarrow j}$ is the map that collapses split symbols in round i to their earlier identities in round j.

It is straightforward to take a projection π and map a CFG G to its induced projection $\pi(G)$. What is less obvious is how the probabilities associated with the rules of G should be mapped. In the case where $\pi(G)$ is coarser than the treebank originally used to train G, and when that treebank is available, it is easy to project the treebank and directly estimate, say, the maximum-likelihood parameters for $\pi(G)$.However, treebank estimation has several limitations. First, the treebank used to train G may not be available. Second, if the grammar G is heavily smoothed or otherwise regularized, its own distribution over trees may be far from that of the treebank. Third, and most importantly, we may wish to project grammars for which treebank estimation is problematic, for example, grammars which are more refined than the observed treebank grammars.

Estimating Projected Grammars

There is a well worked-out notion of estimating a grammar from an infinite distribution over trees (Corazza & Satta 2006). In particular, we can estimate parameters for a projected grammar $\pi(G)$ from the tree distribution induced by G (which can itself be estimated in any manner).

The generalization of maximum likelihood estimation is to find the estimates for $\pi(G)$ with minimum KL divergence from the tree distribution induced by G. Since $\pi(G)$ is a grammar over coarser symbols, we fit $\pi(G)$ to the distribution G induces over π-projected trees: $P(\pi(T)|G)$. The proofs of the general case are given in Corazza & Satta (2006), but the resulting procedure is quite intuitive. Given a (fully observed) treebank, the maximum-likelihood estimate for the probability of a rule $X \rightarrow YZ$ would simply be the ratio of the count of X to the count of the configuration $X \rightarrow YZ$. If we wish to find the estimate which has minimum divergence to an infinite distribution $P(T)$, we use the same formula, but the counts become expected counts:

$$P(X \rightarrow YZ) = \frac{E_{P(T)}[X \rightarrow YZ]}{E_{P(T)}[X]}$$

with unaries estimated similarly. In our specific case, X, Y, and Z are symbols in $\pi(G)$, and the expectations are taken over G's distribution of π-projected trees, $P(\pi(T)|G)$.

	G_0	G_2	G_4	G_6
Nonterminals	98	217	485	1090
Rules	3,700	19,600	126,100	531,200
No Pruning	52 min	99 min	288 min	1612 min
X-Bar Pruning	8 min	14 min	30 min	111 min
Coarse to Fine	6 min	10 min	12 min	15 min
F_1 for above	64.8	85.2	89.7	91.2

Table 1: Grammar sizes, parsing times and accuracies for hierarchically split PCFGs with and without hierarchical coarse-to-fine parsing on our development set.

Calculating Projected Expectations

Concretely, we can estimate the minimum divergence parameters of $\pi(G)$ for any projection π and PCFG G if we can calculate the expectations of the projected symbols and rules according to $P(\pi(T)|G)$.We can exploit the structure of our projections to obtain the desired expectations in a simple and efficient way.

First, consider the problem of calculating the expected counts of a symbol X in a tree distribution given by a grammar G, ignoring the issue of projection. These expected counts obey the following one-step equations (assuming a unique *root* symbol):

$$c(root) = 1$$
$$c(X) = \sum_{Y \to \alpha X \beta} P(\alpha X \beta | Y) c(Y)$$

Here, α, β, or both can be empty, and a rule $X \to \gamma$ appears in the sum once for each X it contains.

In principle, this linear system can be solved in any way.[3] In our experiments, we solve this system iteratively, with the following recurrences:

$$c_0(X) \leftarrow \begin{cases} 1 & \text{if } X = root \\ 0 & \text{otherwise} \end{cases}$$
$$c_{i+1}(X) \leftarrow \sum_{Y \to \alpha X \beta} P(\alpha X \beta | Y) c_i(Y)$$

Note that, as in many other iterative fixpoint methods, such as policy evaluation for Markov decision processes, the quantities $c_k(X)$ have a useful interpretation as the expected counts ignoring nodes deeper than depth k (i.e. the roots are all the root symbol, so $c_0(root) = 1$). This iteration may of course diverge if G is improper, but, in our experiments, it converged within around 25 iterations; this is unsurprising, since the treebank contains few nodes deeper than 25 and our base grammar G seems to have captured this property.

Hierarchical Coarse-to-Fine Parsing

For a final grammar $G = G_n$, we compute estimates for the n projections $G_{n-1}, \ldots, G_0$ =X-Bar, where $G_i = \pi_i(G)$ as described in the previous section. Additionally we project to a grammar G_{-1} in which all nonterminals, except for the preterminals, have been collapsed. During parsing, we start of by exhaustively computing the inside/outside scores with G_{-1}. At each stage, chart items with low posterior probability are removed from the chart, and we proceed to compute inside/outside scores with the next, more refined grammar, using the projections $\pi_{i \to i-1}$ to map between symbols in G_i and G_{i-1}. In each pass, we skip chart items whose projection into the previous stage had a probability below a stage-specific threshold, until we reach $G = G_n$ (after seven passes in our case). For G, we do not prune but instead return the minimum risk tree.

The pruning thresholds were empirically determined on a held out set. We found our projected grammar estimates to be significantly better suited for pruning than the original grammars, which were learned during training.

[3]Whether or not the system has solutions depends on the parameters of the grammar. In particular, G may be improper, though the results of Chi (1999) imply that G will be proper if it is the maximum-likelihood estimate of a finite treebank.

Experimental Results

Table 1 shows the tremendous reduction in parsing time (all times are cumulative) and gives an overview over grammar sizes and parsing accuracies. In particular, in our Java implementation on a 3GHz processor, it is possible to parse 1600 sentences in less than 900 sec. with an F_1 of 91.2%. This compares favorably to the previously best generative lexicalized parser for English (Charniak & Johnson (2005): 90.7% in 1300 sec.). For German and Chinese our learnt grammars outperform the previously best parsers by an even larger margin (see Petrov & Klein (2007) for details).

Conclusions

The approach we have presented gives an extremely accurate and compact grammar, learned in a fully automated fashion. In addition, the split structure admits an extremely efficient coarse-to-fine inference scheme. This approach is applicable more broadly, to other problems where we have observed training structure which is coarser than the true underlying process in a similar way. The final parser along with grammars for a variety of languages is available for download at `http://nlp.cs.bekerley.edu`.

References

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

Chi, Z. 1999. Statistical properties of probabilistic context-free grammars. In *Computational Linguistics*.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. Dissertation, U. of Penn.

Corazza, A., and Satta, G. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *HLT-NAACL '06*.

Klein, D., and Manning, C. 2003. Accurate unlexicalized parsing. In *ACL '03*, 423–430.

Lease, M.; Charniak, E.; Johnson, M.; and McClosky, D. 2006. A look at parsing and its applications. In *AAAI '06*.

Matsuzaki, T.; Miyao, Y.; and Tsujii, J. 2005. Probabilistic CFG with latent annotations. In *ACL '05*, 75–82.

Pereira, F., and Schabes, Y. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL '92*.

Petrov, S., and Klein, D. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*.

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