Learning Phrasal Categories

William P. Headden III
Department of Computer Science
Brown University
Providence, RI 02912

Abstract

In this work we learn clusters of contextual annotations for non-terminals in the Penn Treebank. Perhaps the best way to think about this problem is to contrast our work with that of Klein and Manning (2003). That research used tree-transformations to create various grammars with different contextual annotations on the non-terminals. These grammars were then used in conjunction with a CKY parser. The authors explored the space of different annotation combinations by hand. Here we try to automate the process — to learn the "right" combination automatically. Our results are not quite as good as those carefully created by hand but they are close (84.8 vs 85.7).

1 Introduction and Previous Research

It is by now commonplace knowledge that accurate syntactic parsing is not possible given the only a context-free grammar with standard Penn Treebank (Marcus et al., 1993) labels (e.g., S, NP, etc.) (Charniak, 1996). Instead researchers condition parsing decisions on many other features, such as parent phrase-marker, and, famously, the lexical-head of the phrase (Magerman, 1995; Collins, 1996; Collins, 1997; Johnson, 1998; Charniak, 2000; Henderson, 2003; Klein and Manning, 2003; Matsuzaki et al., 2005) (and others).

One particularly perspicuous way to view the use of extra conditioning information is that of tree-transformation (Johnson, 1998; Klein and Manning, 2003). Rather than imagining the parser roaming around the tree to pick up the information it needs, we rather relabel the nodes to directly encode this information. Thus rather than have the parser "look" to find out that, say, the parent of some NP is an S, we simply relabel the NP as an $NP[S]$.

This viewpoint is even more compelling if one does not intend to smooth the probabilities. For example, consider $p(NP \rightarrow PRN \mid NP[S])$ If we have no intention of backing off this probability to $p(NP \rightarrow PRN \mid NP)$ we can treat $NP[S]$ as an uninterpreted phrasal category and run all of the standard PCFG algorithms without change. The result is a vastly simplified parser. This is exactly what is done in (Klein and Manning, 2003).

Thus the "phrasal categories" of our title refer to these new, hybrid categories, such as $NP[S]$. We hope to learn which of these categories work best given that they cannot be made too specific because that would create sparse data problems.

The parser from (Klein and Manning, 2003) is an unlexicalized PCFG with various carefully selected context annotations. Their model uses some parent annotations, and marks nodes which initiate or in certain cases conclude unary productions. They also propose several linguistically motivated annotations of several tags, including VP, IN, CC,NP and S. This results in a reasonably accurate unlexicalized PCFG parser.

The downside of this approach is that their features are very specific, applying different annotations to different treebank nonterminals. For instance, they mark right-recursive NPs and not VPs (e.g., an NP which is the right-most child of another NP). This is because data sparsity issues preclude annotating the nodes in the treebank too liberally. The goal of our work is to automate the process a bit, by annotating with more general features that apply broadly, and by learning clusters of these annotations.

Coming to this problem from the standpoint of tree transformation we naturally view our work as a descendent of (Johnson, 1998; Klein and Manning, 2003). In retrospect, however, there are perhaps even greater similarities to that of (Magerman, 1995; Henderson, 2003; Matsuzaki et al., 2005) Consider the approach of (Matsuzaki et al., 2005). They posit a series of latent annotations for each nonterminal, and learn a grammar using an EM algorithm similar to the inside-outside algorithm. Their approach, however, requires the number of annotations to be specified ahead of time, and assigns the same number of annotations to each treebank nonterminal. We would like to infer the number of annotations for each nonterminal automatically.

However, again in retrospect, it is in (Magerman, 1995) that we see the greatest similarity. Rather than taking about clustering nodes, as we do, Magerman creates a decision tree, but the differences between clustering and decision trees are small. Perhaps a more substantial difference is that by not casting his problem as one of learning phrasal categories Magerman looses all of the free PCFG technology that we can leverage, e.g., Magerman must use heuristic search to find his parses and incurs search errors because of it. We use an efficient CKY algorithms to do exhaustive search in reasonable time.

2 Background

A PCFG is a tuple $(V, M, \mu_0, R, q : R \to [0, 1])$, where V is a set of terminal symbols; $M = \{\mu_i\}$ is a set of nonterminal symbols; μ_0 is a start or root symbol; R is a set of productions of the form $\mu_i \to \rho$, where ρ is a sequence of terminals and nonterminals; and q is a family of probability distributions over rules conditioned on the rules' left-hand side.

As in (Johnson, 1998) and (Klein and Manning, 2003), we annotate the Penn treebank non-terminals with various context information. Suppose μ is a Treebank non-terminal. Let $\lambda = \mu[\alpha]$ denote the non-terminal category annotated with a vector of context features α. A PCFG is derived from the trees in the usual manner, with production rules taken directly from the annotated trees, and the probability of an annotated rule $q(\lambda \to \rho) = \frac{C(\lambda \to \rho)}{C(\lambda)}$ where $C(\lambda \to \rho)$ and $C(\lambda)$ are the number of observations of the production and its lefthand side, respectively.

We refer to the grammar resulting from extracting annotated productions directly out of the treebank as the base grammar.

Our goal is to partition the set of annotated non-terminals into clusters $\Phi = \{\phi_i\}$. Each possible clustering corresponds to a PCFG, with the set of non-terminals corresponding to the set of clusters. The probability of a production under this PCFG is

$$p(\phi_i \to \phi_j \phi_k) = \frac{C(\phi_i \to \phi_j \phi_k)}{C(\phi_i)}$$

where ϕs $\in$ Φ are clusters of annotated non-terminals and where:

$$C(\phi_i \to \phi_j \phi_k \ldots) = \\ \sum_{(\lambda_i, \lambda_j, \lambda_k \ldots) \in \phi_i \times \phi_j \times \phi_k \ldots} C(\lambda_i \to \lambda_j \lambda_k \ldots)$$

We refer to the PCFG of some clustering as the clustered grammar.

2.1 Features

Most of the features we use are fairly standard. These include the label of the parent and grandparent of a node, its lexical head, and the part of speech of the head.

Klein and Manning (Klein and Manning, 2003) find marking nonterminals which have unary rewrites to be helpful. They also find useful annotating two preterminals (DT,RB) with whether they are the product of a unary production. We generalize this via two width features: the first marking a node with the number of nonterminals to which it rewrites; the second marking each preterminal with the width of its parent. Another feature is the span of a nonterminal, or the number of terminals it dominates, which we normalize by dividing by the length of the sentence. Hence preterminals have normalized spans of 1/(length of the sentence), while the root has a normalized span of 1.

Extending on the notion of a Base NP, introduced by Collins (1996), we mark any nonterminal that dominates only preterminals as Base. Collins inserts a unary NP over any base NPs without NP parents. However, (Klein and Manning, 2003) find that this hurts performance relative to just marking the NPs, and so our Base feature does not insert.

We have two features describing a node's position in the expansion of its parent. The first, which we call the inside position, specifies whether the nonterminal's position relative to the heir of its parent's head, (to the left or right) or whether the nonterminal is the heir. (By "heir" we mean the constituent donates it's head. So the heir of an S is typically the VP under the S.) The second, outside position, specifies the nonterminal's position relative to the boundary of the constituent: it is the leftmost child, the rightmost child, or neither.

Related to this, we further noticed that several of Klein & Manning's features, such as marking NPs as right recursive or possesive have the property of annotating with the label of the rightmost child (when they are NP and POS respectively). We generalize this by marking all nodes both with their rightmost child and (an analogous feature) leftmost child.

We also mark whether or not a node borders the end of a sentence, save for ending punctuation. (For instance, in this sentence, all the constituents with the second ”marked” rightmost in their span would be marked).

Another (Klein and Manning, 2003) feature we try includes the temporal NP feature, where TMP markings in the treebank are retained, and propagated down the head inheritance path of the tree.

It is worth mentioning that all the features here come directly from the treebank. For instance, the part of speech of the head feature has values only from the raw treebank tagset. When a preterminal cluster is split, this assignment does not change the value of this feature.

3 Clustering

The input to the clusterer is a set of annotated grammar productions and counts. Our clustering algorithm is a divisive one reminiscent of (Martin et al., 1995). We start with a single cluster for each Treebank nonterminal and one additional for intermediate nodes, which are described in section 3.2.

The clustering method has two interleaved parts: one in which candidate splits are generated, and one in which we choose a candidate split to enact.

For each of the initial clusters, we generate a candidate split, and place that split in a priority queue. The priority queue is ordered by the Bayesian Information Criterion (BIC), e.g.(Hastie et al., 2003).

The BIC of a model M is defined as -2*(log likelihood of the data according to M) +d_M*(log number of observations). d_M is the number of degrees of freedom in the model, which for a PCFG is the number of productions minus the number of nonterminals. Thus in this context BIC can be thought of as optimizing the likelihood, but with a penalty against grammars with many rules.

While the queue is nonempty, we remove a candidate split to reevaluate. Reevaluation is necessary because, if there is a delay between when a split is proposed and when a split is enacted, the grammar used to score the split will have changed. However, we suppose that the old score is close enough to be a reasonable ordering measure for the priority queue. If the reevaluated candidate is no longer better than the second candidate on the queue, we reinsert it, and continue. However, if it is still the best on the queue, and it improves the model, we enact the split; otherwise it is discarded.

When a split is enacted, the old cluster is removed from the set of nonterminals, and is replaced with the two new nonterminals of the split. A candidate split for each of the two new clusters is generated, and placed on the priority queue.

This process of reevaluation, enacting splits, and generating new candidates continues until the priority queue is empty of potential splits.

We select a candidate split of a particular cluster as follows. For each context feature we generate a potential nominee split. First we partition randomly the values for the feature into two buckets. We then repeatedly try to move values from one bucket to the other. If doing so results in an improvement to the likelihood of the training data, we keep the change, otherwise we reject it. The swapping continues until moving no individual value results in an improvement in likelihood.

Suppose we have a grammar derived from a cor-

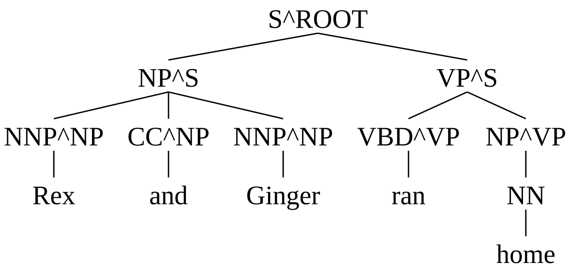

Figure 1: A Parent annotated tree.

pus of a single tree, whose nodes have been annotated with their parent as in Figure 1. The base productions for this corpus are:

$$\begin{array}{lll}
S[ROOT] \rightarrow & NP[S]\ VP[S] & 1/1 \\
VP[S] \rightarrow & VBD[VP]\ NP[VP] & 1/1 \\
NP[S] \rightarrow & NP[NP]\ CC[NP]\ NP[NP] & 1/1 \\
NP[VP] \rightarrow & NN[NP] & 1/1 \\
NP[NP] \rightarrow & NNP[NP] & 2/2
\end{array}$$

Suppose we are in the initial state, with a single cluster for each treebank nonterminal. Consider a potential split of the NP cluster on the parent feature, which in this example has three values: S, VP, and NP. If the S and VP values are grouped together in the left bucket, and the NP value is alone in the right bucket, we get cluster nonterminals $NP_L = \{NP[S], NP[VP]\}$ and $NP_R = \{NP[NP]\}$. The resulting grammar rules and their probabilities are:

$$\begin{array}{rcll}
S & \rightarrow & NP_L\, VP & 1/1 \\
VP & \rightarrow & VBD\ NP_L & 1/1 \\
NP_L & \rightarrow & NP_R\ CC\ NP_R & 1/2 \\
NP_L & \rightarrow & NN & 1/2 \\
NP_R & \rightarrow & NNP & 2/2
\end{array}$$

If however, VP is swapped to the right bucket with NP, the rules become:

$$\begin{array}{rcll}
S & \rightarrow & NP_L\, VP & 1/1 \\
VP & \rightarrow & VBD\ NP_R & 1/1 \\
NP_L & \rightarrow & NP_R\ CC\ NP_R & 1/1 \\
NP_R & \rightarrow & NN & 1/3 \\
NP_R & \rightarrow & NNP & 2/3
\end{array}$$

The likelihood of the tree in Figure 1 is $1/4$ under the first grammar, but only $4/27$ under the second. Hence in this case we would reject the swap of VP from the right to the left buckets.

The process of swapping continues until no improvement can be made by swapping a single value.

The likelihood of the training data according to the clustered grammar is

$$\prod_{r \in R} p(r)^{C(r)}$$

for R the set of observed productions $r = \phi_i \rightarrow \phi_j \ldots$ in the clustered grammar. Notice that when we are looking to split a cluster ϕ, only productions that contain the nonterminal ϕ will have probabilities that change. To evaluate whether a change increases the likelihood, we consider the ratio between the likelihood of the new model, and the likelihood of the old model.

Furthermore, when we move a value from one bucket to another, only a fraction of the rules will have their counts change. Suppose we are moving value x from the left bucket to the right when splitting ϕ_i. Let $\phi_x \subseteq \phi_i$ be the set of base nonterminals in ϕ_i that have value x for the feature being split upon. Only clustered rules that contain base grammar rules which use nonterminals in ϕ_x will have their probability change. These observations allow us to process only a relatively small number of base grammar rules.

Once we have generated a potential nominee split for each feature, we select the partitioning which leads to the greatest improvement in the BIC as the candidate split of this cluster. This candidate is placed on the priority queue.

One odd thing about the above is that in the local search phase of the clustering we use likelihood, while in the candidate selection phase we use BIC. We tried using both measures in each phase, but found that this hybrid measure outperformed using only one or the other.

3.1 Model Selection

Unfortunately the grammar that results at the end of the clustering process seems to overfit the training data. We resolve this by simply noting periodically the intermediate state of the grammar, and using this grammar to parse a small tuning set (we use the first 400 sentences of WSJ section 24, and parse this every 50 times we enact a split). At the conclusion of clustering, we select the grammar with the highest f-score on this tuning set as the final model.

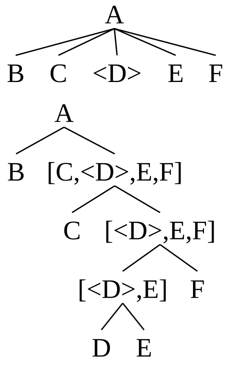

Figure 2:
Top: A production.
Bottom: The production, binarized.

3.2 Binarization

Since our experiments make use of a CKY (Kasami, 1965) parser (created by Mark Johnson and used for the research in (Johnson, 1998), available from his homepage) we must modify the treebank derived rules so that a nonterminal expands to at most two labels. We perform this in a manner similar to (Klein and Manning, 2003) and (Matsuzaki et al., 2005) through the creation of intermediate nodes, as in Figure 2. In this example, the nonterminal heir of A's head is D. The square brackets indicate an intermediate node, and the labels inside the brackets indicate that the node will eventually be expanded into those labels.

Klein & Manning employ Collins' (Collins, 1999) horizontal markovization to desparsify their intermediate nodes. This means that given an intermediate node such as $[C \langle D \rangle EF]$, we forget those labels which will not be expanded past a certain horizon. Klein & Manning use a horizon of two (or less, in some cases) which means only the next two labels to be expanded are retained. Thus $[C \langle D \rangle EF]$ is markovized to $[C \langle D \rangle \ldots F]$.

Our mechanism lays out the unmarkovized intermediate rules in the same way, but we mostly use our clustering scheme to reduce sparsity. We do so by aligning the labels contained in the intermediate nodes in the order in which they would be added when increasing the markovization horizon from zero to three. We also always keep the heir label as a feature. So for instance, $[C \langle D \rangle EF]$ would be represented as having Treebank label ”INTERMEDIATE”, and would have feature vector (D, C, F, E, D),while $[\langle D \rangle EF]$ would have feature vector $(D, F, E, D, -)$, where the first item is the heir of the parent's head. The clusterer would consider each of these five features as for a single possible split. We also incorporate our other features into the intermediate nodes in two ways. Some features, such as the parent or grandparent, will be the same for all the labels in the intermediate node, and hence only need to be included once. Others, such as the part of speech of the head, may be different for each label. These features we align with those of corresponding label in the markov ordering. In our running example, suppose each child node N has part of speech of its head P_N, and we have a parent feature. Our aligned intermediate feature vectors then become $(A, D, C, P_C, F, P_F, E, P_E, D, P_D)$ and $(A, D, F, P_F, E, P_E, D, P_D, -, -)$. As these are somewhat complicated, let us explain them by unpacking the first, the vector for $[C \langle D \rangle EF]$. Consulting Figure 2 we see that it's parent is A. We have chosen to put parents first in the vector, thus explaining $(A, ...)$. Next comes the heir of the constituent, D. This is followed by the first constituent that is to be unpacked from the binarized version, C, which in turn is followed by its head part-of-speech P_C, giving us $(A, D, C, P_C, ...)$. We follow with the next non-terminal to be unpacked from the binarized node and it's head part-of-speech, etc.

It might be fairly objected that this formulation of binarization loses the information of whether a label is to the left, right, or is the heir of the parent's head. This is solved by the inside position feature, described in Section 2.1 which contains exactly this information.

3.3 Smoothing

In order to make the PCFG we infer comparable to that of Klein & Manning, we follow their lead in smoothing no production probabilities save those going from preterminal to nonterminal. Our smoothing mechanism runs roughly along the lines of theirs.

Preterminal rules are smoothed as follows. We consider several classes of unknown words, based on capitalization, the presence of digits or hyphens, and the suffix. We estimate the probability of a tag T given a word (or unknown class) W, as $p(T \mid$

	LP	LR	F_1	CB	0CB
Klein & Manning	86.3	85.1	85.7	1.31	57.2
Matsuzaki et al.	86.1	86.0	86.1	1.39	58.3
This paper	84.8	84.8	84.8	1.47	57.1

Table 1: Parsing results

$W) = \frac{C(T,W)+hp(T|unk)}{C(W)+h}$, where $p(T \mid unk) = C(T, unk)/C(unk)$ is the probability of the tag given any unknown word class. In order to estimate probabilities given unknown classes, and the probability given an unknown word in general, we let the clusterer see every tree twice, once unmodified, and once with the unknown class replacing each word seen less than five times. The production probability $p(W \mid T)$ is then $p(T \mid W)p(W)/p(T)$ where $p(W)$ and $p(T)$ are the respective empirical distributions.

The clusterer does not use smoothed probabilities in allocating annotated preterminals to clusters, but simply the maximum likelihood estimates as it does elsewhere. Smoothing is only used in the parser.

4 Experiments

We trained our model on sections 2-21 of the Penn Wall Street Journal Treebank. We used the first 400 sentences of section 24 for model selection. Section 22 was used for testing during development, while section 23 was used for the final evaluation.

5 Discussion

Our results are shown in Table 5. Unfortunately, our model does not perform quite as well as that of Klein & Manning's or Matsuzaki's parsers. It is worth noting that Matsuzaki's grammar uses a different parse evaluation scheme than Klein & Manning or we do.

We select the parse with the highest probability according to the annotated grammar. Matsuzaki, on the other hand, argues that the proper thing to do is to find the most likely unannotated parse. The probability of this parse is the sum over the probabilities of all annotated parses that reduce to that unannotated parse. Since calculating the parse that maximizes this quantity is NP hard, they try several approximations. One is what Klein & Manning and we do. However, they have a better performing approximation which is used in their reported score. They do not report their score on section 23 using the most-probable-annotated-parse method. They do however compare the performance of different methods using development data, and find that their better approximation gives an absolute improvement in f-measure in the .5-1 percent range. Hence it is probable that even with their better method our grammar would not outperform theirs.

One interesting thing our method allows is for us to examine which features turn out to be useful in which contexts. We noted for each treebank nonterminal, and for each feature, how many times that nonterminal was split on that feature, for the grammar selected in the model selection stage. We ran the clustering with four different random seeds.

We find that in particular, the clusterer only found the head feature to be useful in very specific circumstances. It was used quite a bit to split preterminals; but for phrasals it was only used to split $ADJP$,$ADVP$,NP,PP,VP,QP, and $SBAR$. The part of speech of the head was only used to split NP and VP.

Furthermore, the grandparent tag appears to be of importance primarily for VP and PP nonterminals, though it is used once out of the four runs for NPs.

This indicates that lexical parsers might be able to make do by only using lexical head and grandparent information in very specific instances, thereby shrinking the sizes of their models, and speeding parsing.

6 Conclusion

We have presented a scheme for automatically discovering phrasal categories for parsing with a standard CKY parser. The parser achieves 84.8% precision-recall f-measure on the standard test-section of the Penn WSJ-Treebank (section 23). While this is not as accurate as the hand-tailored grammar of (Klein and Manning, 2003) it is close, and we believe there is room for improvement. For starters, the particular clustering scheme is only one of many. Our algorithm splits clusters along particular features (e.g., parent, head-part-of-speech, etc.). One alternative would be to cluster simultaneously on all the features. It is not obvious which scheme should be better, and they could be quite different. Decisions like this abound, and are worth exploring.

More radically, it is also possible to grow many decision trees, and thus many alternative grammars. We have been impressed by the success of random-forest methods in language modeling (Xu and Jelinek, 2004). In these methods many trees (the forest) are grown, each trying to predict the next word. The multiple trees together are much more powerful than any one individually. The same might be true for grammars.

References

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, Menlo Park. AAAI Press/MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. ACL*, pages 132–139.

M.J. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *The Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco. The Association for Computational Linguistics, Morgan Kaufmann.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, The University of Pennsylvania.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2003. *The Elements of Statistical Learning*. Springer, New York.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of HLT-NAACL 2003*, pages 25–31. The Association for Computational Linguistics.

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

T. Kasami. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *The Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, San Francisco. The Association for Computational Linguistics, Morgan Kaufman.

Michell 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.

S. Martin, J. Liermann, and H. Ney. 1995. Algorithms for bigram and trigram word clustering.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of the 2005 Meeting of the Association for Computational Linguistics*.

Peng Xu and Fred Jelinek. 2004. Random forests in language modeling. In *Proceedings of EMNLP'2004*.