Prototype-Driven Grammar Induction

Aria Haghighi
Computer Science Division
University of California Berkeley
aria42@cs.berkeley.edu

Dan Klein
Computer Science Division
University of California Berkeley
klein@cs.berkeley.edu

Abstract

We investigate *prototype-driven* learning for primarily unsupervised grammar induction. Prior knowledge is specified declaratively, by providing a few canonical examples of each target phrase type. This sparse prototype information is then propagated across a corpus using distributional similarity features, which augment an otherwise standard PCFG model. We show that distributional features are effective at distinguishing bracket *labels*, but not determining bracket *locations*. To improve the quality of the induced trees, we combine our PCFG induction with the CCM model of Klein and Manning (2002), which has complementary stengths: it identifies brackets but does not label them. Using only a handful of prototypes, we show substantial improvements over naive PCFG induction for English and Chinese grammar induction.

1 Introduction

There has been a great deal of work on unsupervised grammar induction, with motivations ranging from scientific interest in language acquisition to engineering interest in parser construction (Carroll and Charniak, 1992; Clark, 2001). Recent work has successfully induced unlabeled grammatical structure, but has not successfully learned labeled tree structure (Klein and Manning, 2002; Klein and Manning, 2004; Smith and Eisner, 2004) .

In this paper, our goal is to build a system capable of producing labeled parses in a target grammar with as little total effort as possible. We investigate a *prototype-driven* approach to grammar induction, in which one supplies canonical examples of each target concept. For example, we might specify that we are interested in trees which use the symbol NP and then list several examples of prototypical NPs (determiner noun, pronouns, etc., see figure 1 for a sample prototype list). This prototype information is similar to specifying an *annotation scheme*, which even human annotators must be provided before they can begin the construction of a treebank. In principle, prototype-driven learning is just a kind of semi-supervised learning. However, in practice, the information we provide is on the order of dozens of total seed instances, instead of a handful of fully parsed trees, and is of a different nature.

The prototype-driven approach has three strengths. First, since we provide a set of target symbols, we can evaluate induced trees using standard labeled parsing metrics, rather than the far more forgiving unlabeled metrics described in, for example, Klein and Manning (2004). Second, knowledge is declaratively specified in an interpretable way (see figure 1). If a user of the system is unhappy with its systematic behavior, they can alter it by altering the prototype information (see section 7.1 for examples). Third, and related to the first two, one does not confuse the ability of the system to learn a consistent grammar with its ability to learn the grammar a user has in mind.

In this paper, we present a series of experiments in the induction of labeled context-free trees using a combination of unlabeled data and sparse prototypes. We first affirm the well-known result that simple, unconstrained PCFG induction produces grammars of poor quality as measured against treebank structures. We then augment a PCFG with prototype features, and show that these features, when propagated to non-prototype sequences using distributional similarity, are effective at learning bracket labels on fixed unlabeled trees, but are still not enough to learn good tree structures without bracketing information. Finally, we intersect the feature-augmented PCFG with the CCM model of Klein and Manning (2002), a high-quality bracketing model. The intersected model is able to learn trees with higher unlabeled F_1 than those in Klein and Manning (2004). More impor-

tantly, its trees are labeled and can be evaluated according to labeled metrics. Against the English Penn Treebank, our final trees achieve a labeled F_1 of 65.1 on short sentences, a 51.7% error reduction over naive PCFG induction.

2 Experimental Setup

The majority of our experiments induced tree structures from the WSJ section of the English Penn treebank (Marcus et al., 1994), though see section 7.4 for an experiment on Chinese. To facilitate comparison with previous work, we extracted WSJ-10, the 7,422 sentences which contain 10 or fewer words after the removal of punctuation and null elements according to the scheme detailed in Klein (2005). We learned models on all or part of this data and compared their predictions to the manually annotated treebank trees for the sentences on which the model was trained. As in previous work, we begin with the part-of-speech (POS) tag sequences for each sentence rather than lexical sequences (Carroll and Charniak, 1992; Klein and Manning, 2002).

Following Klein and Manning (2004), we report unlabeled bracket precision, recall, and F_1. Note that according to their metric, brackets of size 1 are omitted from the evaluation. Unlike that work, all of our induction methods produce trees labeled with symbols which are identified with treebank categories. Therefore, we also report labeled precision, recall, and F_1, still ignoring brackets of size 1.[1]

3 Experiments in PCFG induction

As an initial experiment, we used the inside-outside algorithm to induce a PCFG in the straightforward way (Lari and Young, 1990; Manning and Schütze, 1999). For all the experiments in this paper, we considered binary PCFGs over the nonterminals and terminals occuring in WSJ-10. The PCFG rules were of the following forms:

- $X \rightarrow Y\ Z$, for nonterminal types X, Y, and Z, with $Y \neq X$ or $Z \neq X$
- $X \rightarrow t\ \ Y$, $X \rightarrow Y\ t$, for each terminal t
- $X \rightarrow t\ \ t'$, for terminals t and t'

For a given sentence S, our CFG generates labeled trees T over S.[2] Each tree consists of binary productions $X(i,j) \rightarrow \alpha$ over constituent spans (i,j), where α is a pair of non-terminal and/or terminal symbols in the grammar. The generative probability of a tree T for S is:

$$P_{CFG}(T,S) = \prod_{X(i,j)\rightarrow\alpha\in T} P(\alpha|X)$$

In the inside-outside algorithm, we iteratively compute posterior expectations over production occurences at each training span, then use those expectations to re-estimate production probabilities. This process is guaranteed to converge to a local extremum of the data likelihood, but initial production probability estimates greatly influence the final grammar (Carroll and Charniak, 1992). In particular, uniform initial estimates are an (unstable) fixed point. The classic approach is to add a small amount of random noise to the initial probabilities in order to break the symmetry between grammar symbols.

We randomly initialized 5 grammars using treebank non-terminals and trained each to convergence on the first 2000 sentences of WSJ-10. Viterbi parses were extracted for each of these 2000 sentences according to each grammar. Of course, the parses' symbols have nothing to anchor them to our intended treebank symbols. That is, an NP in one of these grammars may correspond to the target symbol VP, or may not correspond well to any target symbol. To evaluate these learned grammars, we must map the models' phrase types to target phrase types. For each grammar, we followed the common approach of greedily mapping model symbols to target symbols in the way which maximizes the labeled F_1. Note that this can, and does, result in mapping multiple model symbols to the most frequent target symbols. This experiment, labeled PCFG × NONE in figure 4, resulted in an average labeled F_1 of 26.3 and an unlabeled F_1 of 45.7. The unlabeled F_1 is better than randomly choosing a tree (34.7), but not better than always choosing a right branching structure (61.7).

Klein and Manning (2002) suggest that the task of labeling constituents is significantly easier than identifying them. Perhaps it is too much to ask a PCFG induction algorithm to perform both of these tasks simultaneously. Along the lines of Pereira and Schabes (1992), we reran the inside-outside algorithm, but this time placed zero mass on all trees which did not respect the bracketing of the gold trees. This constraint does not fully

[1] In cases where multiple gold labels exist in the gold trees, precision and recall were calculated as in Collins (1999).

[2] Restricting our CFG to a binary branching grammar results in an upper bound of 88.1% on unlabeled F_1.

Phrase	Prototypes	Phrase	Prototypes
NP	DT NN	VP	VBN IN NN
	JJ NNS		VBD DT NN
	NNP NNP		MD VB CD
S	PRP VBD DT NN	QP	CD CD
	DT NN VBD IN DT NN		RB CD
	DT VBZ DT JJ NN		DT CD CD
PP	IN NN	ADJP	RB JJ
	TO CD CD		JJ
	IN PRP		JJ CC JJ
ADVP	RB RB		
	RB CD		
	RB CC RB		
VP-INF	VB NN	NP-INF	NN POS

Figure 1: English phrase type prototype list manually specified (The entire supervision for our system). The second part of the table is additional prototypes discussed in section 7.1.

eliminate the structural uncertainty since we are inducing binary trees and the gold trees are flatter than binary in many cases. This approach of course achieved the upper bound on unlabeled F_1, because of the gold bracket constraints. However, it only resulted in an average labeled F_1 of 52.6 (experiment PCFG $\times$ GOLD in figure 4). While this labeled score is an improvement over the PCFG $\times$ NONE experiment, it is still relatively disappointing.

3.1 Encoding Prior Knowledge with Prototypes

Clearly, we need to do something more than adding structural bias (e.g. bracketing information) if we are to learn a PCFG in which the symbols have the meaning and behaviour we intend. How might we encode information about our prior knowledge or intentions?

Providing labeled trees is clearly an option. This approach tells the learner how symbols should recursively relate to each other. Another option is to provide fully linearized yields as prototypes. We take this approach here, manually creating a list of POS sequences typical of the 7 most frequent categories in the Penn Treebank (see figure 1).[3] Our grammar is limited to these 7 phrase types plus an additional type which has no prototypes and is unconstrained.[4] This list grounds each symbol in terms of an observable portion of the data, rather than attempting to relate unknown symbols to other unknown symbols.

Broadly, we would like to learn a grammar which explains the observed data (EM's objective) but also meets our prior expectations or requirements of the target grammar. How might we use such a list to constrain the learning of a PCFG with the inside-outside algorithm? We might require that all occurences of a prototype sequence, say DT NN, be constituents of the corresponding type (NP). However, human-elicited prototypes are not likely to have the property that, when they occur, they are (nearly) always constituents. For example, DT NN is a perfectly reasonable example of a noun phrase, but is not a constituent when it is part of a longer DT NN NN constituent. Therefore, when summing over trees with the inside-outside algorithm, we could require a weaker property: whenever a prototype sequence is a constituent it must be given the label specified in the prototype file.[5] This constraint is enough to break the symmetry between the model labels, and therefore requires neither random initialization for training, nor post-hoc mapping of labels for evaluation. Adding prototypes in this way and keeping the gold bracket constraint gave 59.9 labeled F_1. The labeled F_1 measure is again an improvement over naive PCFG induction, but is perhaps less than we might expect given that the model has been given bracketing information and has prototypes as a form of supervision to direct it.

In response to a prototype, however, we may wish to conclude something stronger than a constraint on that *particular* POS sequence. We might hope that sequences which are similar to a prototype in some sense are generally given the same label as that prototype. For example, DT NN is a noun phrase prototype, the sequence DT JJ NN is another good candidate for being a noun phrase. This kind of propagation of constraints requires that we have a good way of defining and detecting similarity between POS sequences.

3.2 Phrasal Distributional Similarity

A central linguistic argument for constituent types is *substitutability*: phrases of the same type appear

[3] A possible objection to this approach is the introduction of improper reasearcher bias via specifying prototypes. See section 7.3 for an experiment utilizing an automatically generated prototype list with comparable results.

[4] In our experiments we found that adding prototypes for more categories did not improve performance and took more time. We note that we still evaluate against all phrase types regardless of whether or not they are modeled by our grammar.

[5] Even this property is likely too strong: prototypes may have multiple possible labels, for example DT NN may also be a QP in the English treebank.

Yield	Prototype	Skew KL	Phrase Type	Skew KL
DT JJ NN	DT NN	0.10	NP	0.39
IN DT VBG NN	IN NN	0.24	PP	0.45
DT NN MD VB DT NNS	PRP VBD DT NN	0.54	S	0.58
CC NN	IN NN	0.43	PP	0.71
MD NNS	PRP VBD DT NN	1.43	NONE	-

Figure 2: Yields along with most similar prototypes and phrase types, guessed according to (3).

in similar contexts and are mutually substitutable (Harris, 1954; Radford, 1988). For instance, DT JJ NN and DT NN occur in similar contexts, and are indeed both common NPs. This idea has been repeatedly and successfully operationalized using various kinds of *distributional clustering*, where we define a similarity measure between two items on the basis of their immediate left and right contexts (Schütze, 1995; Clark, 2000; Klein and Manning, 2002).

As in Clark (2001), we characterize the distribution of a sequence by the distribution of POS tags occurring to the left and right of that sequence in a corpus. Each occurence of a POS sequence α falls in a context $x\ \alpha\ y$, where x and y are the adjacent tags. The distribution over contexts $x - y$ for a given α is called its *signature*, and is denoted by $\sigma(\alpha)$. Note that $\sigma(\alpha)$ is composed of context counts from all occurences, constituent and distituent, of α. Let $\sigma_c(\alpha)$ denote the context distribution for α where the context counts are taken only from constituent occurences of α. For each phrase type in our grammar, X, define $\sigma_c(X)$ to be the context distribution obtained from the counts of all constituent occurences of type X:

$$\sigma_c(X) = \mathbb{E}_{p(\alpha|X)}\ \sigma_c(\alpha) \quad (1)$$

where $p(\alpha|X)$ is the distribution of yield types for phrase type X. We compare context distributions using the skewed KL divergence:

$$D_{SKL}(p, q) = D_{KL}(p \| \gamma p + (1 - \gamma)q)$$

where γ controls how much of the source distributions is mixed in with the target distribution.

A reasonable baseline rule for classifying the phrase type of a POS yield is to assign it to the phrase from which it has minimal divergence:

$$type(\alpha) = \arg\min_X D_{SKL}(\sigma_c(\alpha), \sigma_c(X)) \quad (2)$$

However, this rule is not always accurate, and, moreover, we do not have access to $\sigma_c(\alpha)$ or $\sigma_c(X)$. We chose to approximate $\sigma_c(X)$ using the prototype yields for X as samples from $p(\alpha|X)$. Letting $proto(X)$ denote the (few) prototype yields for phrase type X, we define $\tilde{\sigma}(X)$:

$$\tilde{\sigma}(X) = \frac{1}{|proto(X)|} \sum_{\alpha \in proto(X)} \sigma(\alpha)$$

Note $\tilde{\sigma}(X)$ is an approximation to (1) in several ways. We have replaced an expectation over $p(\alpha|X)$ with a uniform weighting of $proto(X)$, and we have replaced $\sigma_c(\alpha)$ with $\sigma(\alpha)$ for each term in that expectation. Because of this, we will rely only on high confidence guesses, and allow yields to be given a NONE type if their divergence from each $\tilde{\sigma}(X)$ exceeds a fixed threshold t. This gives the following alternative to (2):

$$type(\alpha) = \quad (3)$$
$$\begin{cases} \text{NONE, if } \min_X D_{SKL}(\sigma(\alpha), \tilde{\sigma}(X)) < t \\ \arg\min_X D_{SKL}(\sigma(\alpha), \tilde{\sigma}(X)), \text{ otherwise} \end{cases}$$

We built a distributional model implementing the rule in (3) by constructing $\sigma(\alpha)$ from context counts in the WSJ portion of the Penn Treebank as well as the BLIPP corpus. Each $\tilde{\sigma}(X)$ was approximated by a uniform mixture of $\sigma(\alpha)$ for each of X's prototypes α listed in figure 1.

This method of classifying constituents is very precise if the threshold is chosen conservatively enough. For instance, using a threshold of $t = 0.75$ and $\gamma = 0.1$, this rule correctly classifies the majority label of a constituent-type with 83% precision, and has a recall of 23% over constituent types. Figure 2 illustrates some sample yields, the prototype sequence to which it is least divergent, and the output of rule (3).

We incorporated this distributional information into our PCFG induction scheme by adding a *prototype feature* over each span (i, j) indicating the output of (3) for the yield α in that span. Associated with each sentence S is a feature map F specifying, for each (i, j), a prototype feature p_{ij}. These features are generated using an augmented CFG model, CFG_+, given by:[6]

$$P_{CFG_+}(T, F) = \prod_{X(i,j) \to \alpha \in T} P(p_{ij}|X) P(\alpha|X)$$
$$= \prod_{X(i,j) \to \alpha \in T} \phi_{CFG_+}(X \to \alpha, p_{ij})$$

[6]Technically, all features in F must be generated for each assignment to T, which means that there should be terms in this equation for the prototype features on distituent spans. However, we fixed the prototype distribution to be uniform for distituent spans so that the equation is correct up to a constant depending on F.

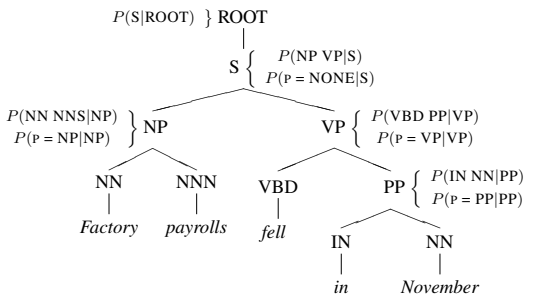

Figure 3: Illustration of PCFG augmented with prototype similarity features.

where $\phi_{CFG_+}(X \to \alpha, p_{ij})$ is the local factor for placing $X \to \alpha$ on a span with prototype feature p_{ij}. An example is given in figure 3.

For our experiments, we fixed $P(p_{ij}|X)$ to be:

$$P(p_{ij}|X) = \begin{cases} 0.60, \text{ if } p_{ij} = X \\ \text{uniform, otherwise} \end{cases}$$

Modifying the model in this way, and keeping the gold bracketing information, gave 71.1 labeled F_1 (see experiment PROTO $\times$ GOLD in figure 4), a 40.3% error reduction over naive PCFG induction in the presence of gold bracketing information. We note that the our labeled F_1 is upper-bounded by 86.0 due to unary chains and more-than-binary configurations in the treebank that cannot be obtained from our binary grammar.

We conclude that in the presence of gold bracket information, we can achieve high labeled accuracy by using a CFG augmented with distributional prototype features.

4 Constituent Context Model

So far, we have shown that, given perfect perfect bracketing information, distributional prototype features allow us to learn tree structures with fairly accurate labels. However, such bracketing information is not available in the unsupervised case.

Perhaps we don't actually need bracketing constraints in the presence of prototypes and distributional similarity features. However this experiment, labeled PROTO $\times$ NONE in figure 4, gave only 53.1 labeled F_1 (61.1 unlabeled), suggesting that some amount of bracketing constraint is necessary to achieve high performance.

Fortunately, there are unsupervised systems which can induce unlabeled bracketings with reasonably high accuracy. One such model is the constituent-context model (CCM) of Klein and Manning (2002), a generative distributional model. For a given sentence S, the CCM generates a bracket matrix, B, which for each span (i, j), indicates whether or not it is a constituent ($B_{ij} = c$) or a distituent ($B_{ij} = d$). In addition, it generates a feature map F', which for each span (i, j) in S specifies a pair of features, $F'_{ij} = (y_{ij}, c_{ij})$, where y_{ij} is the POS yield of the span, and c_{ij} is the context of the span, i.e identity of the conjoined left and right POS tags:

$$P_{CCM}(B, F') = P(B) \prod_{(i,j)} P(y_{ij}|B_{ij}) P(c_{ij}|B_{ij})$$

The distribution $P(B)$ only places mass on bracketings which correspond to binary trees. We can efficiently compute $P_{CCM}(B, F')$ (up to a constant) depending on F' using local factors $\phi_{CCM}(y_{ij}, c_{ij})$ which decomposes over constituent spans:[7]

$$\begin{aligned} P_{CCM}(B, F') &\propto \prod_{(i,j):B_{ij}=c} \frac{P(y_{ij}|c)P(c_{ij}|c)}{P(y_{ij}|d)P(c_{ij}|d)} \\ &= \prod_{(i,j):B_{ij}=c} \phi_{CCM}(y_{ij}, c_{ij}) \end{aligned}$$

The CCM by itself yields an unlabeled F_1 of 71.9 on WSJ-10, which is reasonably high, but does not produce labeled trees.

5 Intersecting CCM and PCFG

The CCM and PCFG models provide complementary views of syntactic structure. The CCM explicitly learns the non-recursive contextual and yield properties of constituents and distituents. The PCFG model, on the other hand, does not explicitly model properties of distituents but instead focuses on modeling the hierarchical and recursive properties of natural language syntax. One would hope that modeling both of these aspects simultaneously would improve the overall quality of our induced grammar.

We therefore combine the CCM with our feature-augmented PCFG, denoted by PROTO in experiment names. When we run EM on either of the models alone, at each iteration and for each training example, we calculate posteriors over that

[7] Klein (2005) gives a full presentation.

model's latent variables. For CCM, the latent variable is a bracketing matrix B (equivalent to an unlabeled binary tree), while for the CFG$_+$ the latent variable is a labeled tree T. While these latent variables aren't exactly the same, there is a close relationship between them. A bracketing matrix constrains possible labeled trees, and a given labeled tree determines a bracketing matrix. One way to combine these models is to encourage both models to prefer latent variables which are compatible with each other.

Similar to the approach of Klein and Manning (2004) on a different model pair, we intersect CCM and CFG$_+$ by multiplying their scores for any labeled tree. For each possible labeled tree over a sentence S, our generative model for a labeled tree T is given as follows:

$$P(T, F, F') = P_{CFG_+}(T, F) P_{CCM}(B(T), F') \quad (4)$$

where $B(T)$ corresponds to the bracketing matrix determined by T. The EM algorithm for the product model will maximize:

$$P(S,F,F') = \sum_{T \in \mathcal{T}(S)} P_{CCM}(B, F') P_{CFG_+}(T, F)$$
$$= \sum_{B} P_{CCM}(B, F') \sum_{T \in \mathcal{T}(B,S)} P_{CFG_+}(T, F)$$

where $\mathcal{T}(S)$ is the set of labeled trees consistent with the sentence S and $\mathcal{T}(B, S)$ is the set of labeled trees consistent with the bracketing matrix B and the sentence S. Notice that this quantity increases as the CCM and CFG$_+$ models place probability mass on compatible latent structures, giving an intuitive justification for the success of this approach.

We can compute posterior expectations over (B, T) in the combined model (4) using a variant of the inside-outside algorithm. The local factor for a binary rule $r = X \rightarrow YZ$, over span (i, j), with CCM features $F'_{ij} = (y_{ij}, c_{ij})$ and prototype feature p_{ij}, is given by the product of local factors for the CCM and CFG$_+$ models:

$$\phi(r, (i, j)) = \phi_{CCM}(y_{ij}, c_{ij}) \phi_{CFG_+}(r, p_{ij})$$

From these local factors, the inside-outside algorithm produces expected counts for each binary rule, r, over each span (i, j) and split point k, denoted by $P(r, (i, j), k|S, F, F')$. These posteriors are sufficient to re-estimate all of our model parameters.

	Labeled			Unlabeled		
Setting	Prec.	Rec.	F_1	Prec.	Rec.	F_1
No Brackets						
PCFG × NONE	23.9	29.1	26.3	40.7	52.1	45.7
PROTO × NONE	51.8	62.9	56.8	59.6	76.2	66.9
Gold Brackets						
PCFG × GOLD	47.0	57.2	51.6	78.8	100.0	88.1
PROTO × GOLD	64.8	78.7	71.1	78.8	100.0	88.1
CCM Brackets						
CCM	-	-	-	64.2	81.6	71.9
PCFG × CCM	32.3	38.9	35.3	64.1	81.4	71.8
PROTO × CCM	56.9	68.5	62.2	68.4	86.9	76.5
BEST	59.4	72.1	65.1	69.7	89.1	78.2
UBOUND	78.8	94.7	86.0	78.8	100.0	88.1

Figure 4: English grammar induction results. The upper bound on labeled recall is due to unary chains.

6 CCM as a Bracketer

We tested the product model described in section 5 on WSJ-10 under the same conditions as in section 3. Our initial experiment utilizes no protoype information, random initialization, and greedy remapping of its labels. This experiment, PCFG × CCM in figure 4, gave 35.3 labeled F_1, compared to the 51.6 labeled F_1 with gold bracketing information (PCFG × GOLD in figure 4).

Next we added the manually specified prototypes in figure 1, and constrained the model to give these yields their labels if chosen as constituents. This experiment gave 48.9 labeled F_1 (73.3 unlabeled). The error reduction is 21.0% labeled (5.3% unlabeled) over PCFG × CCM.

We then experimented with adding distributional prototype features as discussed in section 3.2 using a threshold of 0.75 and $\gamma = 0.1$. This experiment, PROTO × CCM in figure 4, gave 62.2 labeled F_1 (76.5 unlabeled). The error reduction is 26.0% labeled (12.0% unlabeled) over the experiment using prototypes without the similarity features. The overall error reduction from PCFG × CCM is 41.6% (16.7%) in labeled (unlabeled) F_1.

7 Error Analysis

The most common type of error by our PROTO × CCM system was due to the binary grammar restriction. For instance common NPs, such as DT JJ NN, analyzed as [$_{\text{NP}}$ DT [$_{\text{NP}}$ JJ NN]], which proposes additional $\overline{\text{N}}$ constituents compared to the flatter treebank analysis. This discrepancy greatly, and perhaps unfairly, damages NP precision (see figure 6). However, this is error is unavoidable

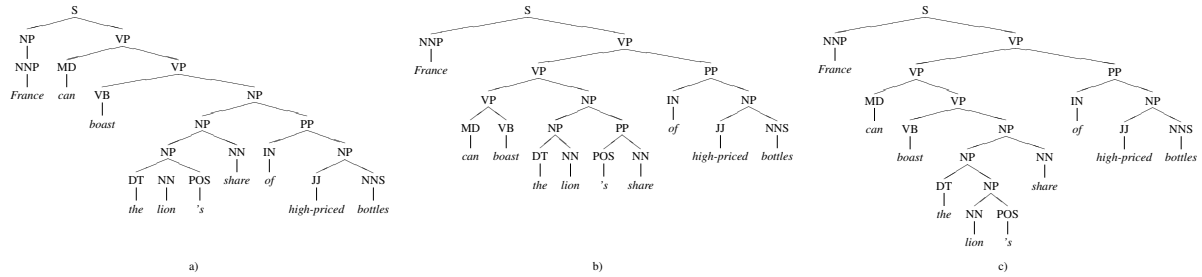

Figure 5: Examples of corrections from adding VP-INF and NP-POS prototype categories. The tree in (a) is the Treebank parse, (b) is the parse with PROTO × CCM model, and c) is the parse with the BEST model (added prototype categories), which fixes the possesive NP and infinitival VP problems, but not the PP attachment.

given our grammar restriction.

Figure 5(b) demonstrates three other errors. Possessive NPs are analyzed as [$_{\text{NP}}$ NN [$_{\text{PP}}$ POS NN]], with the POS element treated as a preposition and the possessed NP as its complement. While labeling the POS NN as a PP is clearly incorrect, placing a constituent over these elements is not unreasonable and in fact has been proposed by some linguists (Abney, 1987). Another type of error also reported by Klein and Manning (2002) is MD VB groupings in infinitival VPs also sometimes argued by linguists (Halliday, 2004). More seriously, prepositional phrases are almost always attached "high" to the verb for longer NPs.

7.1 Augmenting Prototypes

One of the advantages of the prototype driven approach, over a fully unsupervised approach, is the ability to refine or add to the annotation specification if we are not happy with the output of our system. We demonstrate this flexibility by augmenting the prototypes in figure 1 with two new categories NP-POS and VP-INF, meant to model possessive noun phrases and infinitival verb phrases, which tend to have slightly different distributional properties from normal NPs and VPs. These new sub-categories are used during training and then stripped in post-processing. This prototype list gave 65.1 labeled F_1 (78.2 unlabeled). This experiment is labeled BEST in figure 4. Looking at the CFG-learned rules in figure 7, we see that the basic structure of the treebank grammar is captured.

7.2 Parsing with only the PCFG

In order to judge how well the PCFG component of our model did in isolation, we experimented with training our BEST model with the CCM component, but dropping it at test time. This experiment gave 65.1 labeled F_1 (76.8 unlabeled). This demonstrates that while our PCFG performance degrades without the CCM, it can be used on its own with reasonable accuracy.

Label	Prec.	Rec.	F_1
S	79.3	80.0	79.7
NP	49.0	74.4	59.1
VP	80.4	73.3	76.7
PP	45.6	78.6	57.8
QP	36.2	78.8	49.6
ADJP	29.4	33.3	31.2
ADVP	25.0	12.2	16.4

Figure 6: Precision, recall, and F_1 for individual phrase types in the BEST model

Rule	Probability	Rule	Probability
S → NP VP	0.51	VP → VBZ NP	0.20
S → PRP VP	0.13	VP → VBD NP	0.15
S → NNP VP	0.06	VP → VBP NP	0.09
S → NNS VP	0.05	VP → VB NP	0.08
NP → DT NN	0.12	ROOT → S	0.95
NP → NP PP	0.09	ROOT → NP	0.05
NP → NNP NNP	0.09		
NP → JJ NN	0.07		
PP → IN NP	0.37	QP → CD CD	0.35
PP → CC NP	0.06	QP → CD NN	0.30
PP → TO VP	0.05	QP → QP PP	0.10
PP → TO QP	0.04	QP → QP NNS	0.05
ADJP → RB VBN	0.37	ADVP → RB RB	0.25
ADJP → RB JJ	0.31	ADVP → ADJP PRP	0.15
ADJP → RBR JJ	0.09	ADVP → RB CD	0.10

Figure 7: Top PCFG Rules learned by BEST model

7.3 Automatically Generated Prototypes

There are two types of bias which enter into the creation of prototypes lists. One of them is the bias to choose examples which reflect the annotation semantics we wish our model to have. The second is the iterative change of prototypes in order to maximize F_1. Whereas the first is appro-

priate, indeed the point, the latter is not. In order to guard against the second type of bias, we experimented with automatically extracted generated prototype lists which would not be possible without labeled data. For each phrase type category, we extracted the three most common yield associated with that category that differed in either first or last POS tag. Repeating our PROTO × CCM experiment with this list yielded 60.9 labeled F_1 (76.5 unlabeled), comparable to the performance of our manual prototype list.

7.4 Chinese Grammar Induction

In order to demonstrate that our system is somewhat language independent, we tested our model on CTB-10, the 2,437 sentences of the Chinese Treebank (Ircs, 2002) of length at most 10 after punctuation is stripped. Since the authors have no expertise in Chinese, we automatically extracted prototypes in the same way described in section 7.3. Since we did not have access to a large auxiliary POS tagged Chinese corpus, our distributional model was built only from the treebank text, and the distributional similarities are presumably degraded relative to the English. Our PCFG × CCM experiment gave 18.0 labeled F_1 (43.4 unlabeled). The PROTO × CCM model gave 39.0 labeled F_1 (53.2 unlabeled). Presumably with access to more POS tagged data, and the expertise of a Chinese speaker, our system would see increased performance. It is worth noting that our unlabeled F_1 of 53.2 is the best reported from a primarily unsupervised system, with the next highest figure being 46.7 reported by Klein and Manning (2004).

8 Conclusion

We have shown that distributional prototype features can allow one to specify a target labeling scheme in a compact and declarative way. These features give substantial error reduction in labeled F_1 measure for English and Chinese grammar induction. They also achieve the best reported unlabeled F_1 measure.[8] Another positive property of this approach is that it tries to reconcile the success of distributional clustering approaches to grammar induction (Clark, 2001; Klein and Manning, 2002), with the CFG tree models in the supervised literature (Collins, 1999). Most importantly, this is the first work, to the authors' knowledge, which has learned CFGs in an unsupervised or semi-supervised setting and can parse natural language language text with any reasonable accuracy.

[8]The next highest results being 77.1 and 46.7 for English and Chinese respectively from Klein and Manning (2004).

Acknowledgments We would like to thank the anonymous reviewers for their comments. This work is supported by a Microsoft / CITRIS grant and by an equipment donation from Intel.

References

Stephen P. Abney. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, MIT.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical Report CS-92-16.

Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL*, pages 91–94, Lisbon, Portugal.

Alexander Clark. 2001. The unsupervised induction of stochastic context-free grammars using distributional clustering. In *CoNLL*.

Michael Collins. 1999. *The Unsupervised learning of Natural Language Structure*. Ph.D. thesis, University of Rochester.

M.A.K Halliday. 2004. An introduction to functional grammar. Edward Arnold, 2nd edition.

Zellig Harris. 1954. *Distributional Structure*. University of Chicago Press, Chicago.

Nianwen Xue Ircs. 2002. Building a large-scale annotated chinese corpus.

Dan Klein and Christopher Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

Dan Klein. 2005. *The unsupervised learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

Karim Lari and Steve Young. 1990. The estimation of stochastic context-free grammars using the insideoutside algorithm. *Computer Speech and Language*, 2(4):35–56.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Fernando C. N. Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Meeting of the Association for Computational Linguistics*, pages 128–135.

Andrew Radford. 1988. *Transformational Grammar*. Cambridge University Press, Cambridge.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL*.

Noah A. Smith and Jason Eisner. 2004. Guiding unsupervised grammar induction using contrastive estimation. In *Working notes of the IJCAI workshop on Grammatical Inference Applications*.