Latent-Variable PCFGs: Background and Applications

Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
scohen@inf.ed.ac.uk

Abstract

Latent-variable probabilistic context-free grammars are latent-variable models that are based on context-free grammars. Nonterminals are associated with latent states that provide contextual information during the top-down rewriting process of the grammar. We survey a few of the techniques used to estimate such grammars and to parse text with them. We also give an overview of what the latent states represent for English Penn treebank parsing, and provide an overview of extensions and related models to these grammars.

1 Introduction

Probabilistic grammars have been one of the most important modeling tools available in the natural language processing toolkit. They are often humanly interpretable because of their symbolic backbone, while their probabilistic component helps with reasoning under uncertainty. Probabilistic grammars have mostly been used for syntactic analysis in NLP (Charniak, 1997; Collins, 2003; Hockenmaier and Steedman, 2002), but they are also useful for other problems both in and outside of NLP (Sakakibara et al., 1994; Guerra and Aloimonos, 2005; Lin et al., 2009).

Latent-variable models, on the other hand, are also a modeling tool of great importance in natural language processing. They have been used for many applications, including machine translation, natural language generation, question answering and semantics. Latent-variable models are centered around learning from incomplete data. This means that the underlying statistical model is defined over latent random variables that are not observed in the data used for learning. The latent variables explain correlations between observed random variables.

As such, it is not surprising that latent-variable models were combined with probabilistic grammars to train strong models that detect unobserved patterns in data, while retaining the interpretability and symbolic backbone contained within grammars. Latent-variable grammars have been mostly used for syntactic parsing, most prominently through the use of *latent-variable probabilistic context-free grammars* (L-PCFGs) – PCFGs that are augmented with latent states.

In this paper, we survey the use of L-PCFGs for syntactic parsing and other applications. We also survey the two main families of algorithms used for learning L-PCFGs: expectation-maximization algorithms and spectral algorithms. We analyze the latent state representations that one learns with L-PCFGs, and also describe extensions of L-PCFGs and related models (such as those that appear in deep learning).

2 Latent-Variable PCFGs

Latent-variable PCFGs (L-PCFGs) are PCFGs with additional latent states that decorate each nonterminal in each rule. While the backbone of an L-PCFG is simply a context-free grammar (because the decoration of the nonterminal with a latent state together with the nonterminal itself can be thought of as a new composite nonterminal), the use of L-PCFGs also implies a specific process of learning them from data: the decoration of the nonterminals with latent states is assumed to be absent from the sampled data from which we learn the model.

More formally, a latent-variable probabilistic context-free grammar (L-PCFG; in Chomsky normal form) is 5-tuple $(\mathcal{N}, \mathcal{R}, m, n, p)$ where:

- $\mathcal{N}$ is the set of nonterminal symbols in the

grammar.

- $[m]$ is the set of possible hidden states where $[m]$ is defined as $\{1, \ldots, m\}$.
- $[n]$ is the set of possible words.
- $\mathcal{R}$ is a set of context-free rules in the form of $a \rightarrow b\,c$ or $a \rightarrow x$, where $a, b, c \in \mathcal{N}$ are nonterminals and $x \in [n]$ is a word.
- For all $a \rightarrow b \quad c \in \mathcal{R}$, $h_1, h_2, h_3 \in [m]$, we have a context-free rule $a(h_1) \rightarrow b(h_2) \quad c(h_3)$ and a parameter $p(a(h_1) \rightarrow b(h_2) \;\; c(h_3) \mid a(h_1))$.
- For all $a \rightarrow x \in \mathcal{R}$, $h \in [m]$, we have a context-free rule $a(h) \rightarrow x$ and a parameter $p(a(h) \rightarrow x \mid a(h))$.
- For all $a \in \mathcal{N}$ and $h \in [m]$, we have a parameter $p(a(h))$ which is the probability of nonterminal a paired with hidden variable h being at the root of the tree.

The parameters satisfy the following normalization constraints:

$$\sum_{a,h} p(a(h)) = 1,$$

and for all $a \in \mathcal{N}$ and $h \in [m]$:

$$\sum_{a(h) \rightarrow b(h_2) \;\; c(h_3)} p(a(h) \rightarrow b(h_2) \;\; c(h_3)) + \sum_{a(h) \rightarrow x} p(a(h) \rightarrow x) = 1$$

Note that for simplicity, we consider the case where every nonterminal symbol has the same number m of hidden state values. It is simple to generalize the method to allow different numbers of hidden states for each nonterminal. In both cases, though, the space of latent states for each nonterminal is separate. This means that latent state 2 for example for an NP has no relationship to latent state 2 for VP.

The generative story that such an L-PCFG model induces is similar to one of PCFGs. We begin with the top node of the derivation tree with its latent state by drawing a nonterminal and a latent state from $p(a(h))$, and then recursively draw rules in the form of $a(h) \rightarrow x$ or $a(h_1) \rightarrow b(h_2) \quad c(h_3)$ conditioned on the parent node, until all nodes at the bottom of the tree are terminal nodes from $[n]$. Figure 1 provides an example of such a derivation.

We refer to a rule of the form $a \rightarrow b \quad c$ as a "skeletal" rule from the "skeletal grammar." We note that while we provided the formulation of L-PCFGs in Chomsky normal form, there is a natural extension to arbitrary PCFGs, where rules of the form $a \rightarrow \alpha$ for $\alpha \in \mathcal{N}^*$ would be decorated with $|\alpha| + 1$ latent states, a state per nonterminal that appears in the rule.

The usual independence assumption made by a PCFG is that "inside" and "outside" trees (shown in Figure 2) are conditionally independent from each other if we know the node that connects them. More formally, if we denote a tree τ and a node β in that tree as a decomposition $\tau = (\tau_0, \tau_1, \beta)$ where τ_0 is the outside tree at node β and τ_1 is the inside tree at node β, then it holds that

$$p(\tau_1 \mid \tau_0, \beta) = p(\tau_1 \mid \beta).$$

This independence assumption of PCFGs can be quite restrictive in modeling syntax of language or in general. Essentially, no local context is modeled for syntactic categories, context which should be carried from different parts of the tree.

Latent-variable PCFGs weaken these independence assumption by introducing a latent state at every node in the tree. Now an inside and an outside tree are conditionally independent of each other given both the nonterminal node in the tree that connects these two subtrees *and* the latent state that is associated with that node.

In the rest of the paper, we denote a skeletal tree by τ and a full derivation with latent state assignments $h = (h_1, \ldots, h_N)$ by $\tau(h)$ where N is the number of nodes in τ.

3 Evolution of Latent-Variable PCFGs

The idea of decorating nonterminals with additional information, and breaking the statistical independence assumptions that PCFGs typically make, has a long history in natural language processing. Johnson (1998) introduced a variety of tree transformations on the Penn treebank, with an aim to improve the parsing accuracy of a PCFG extracted from that treebank. One of the transformations introduced was that of "parent annotation" where each nonterminal is annotated with its parent symbol.

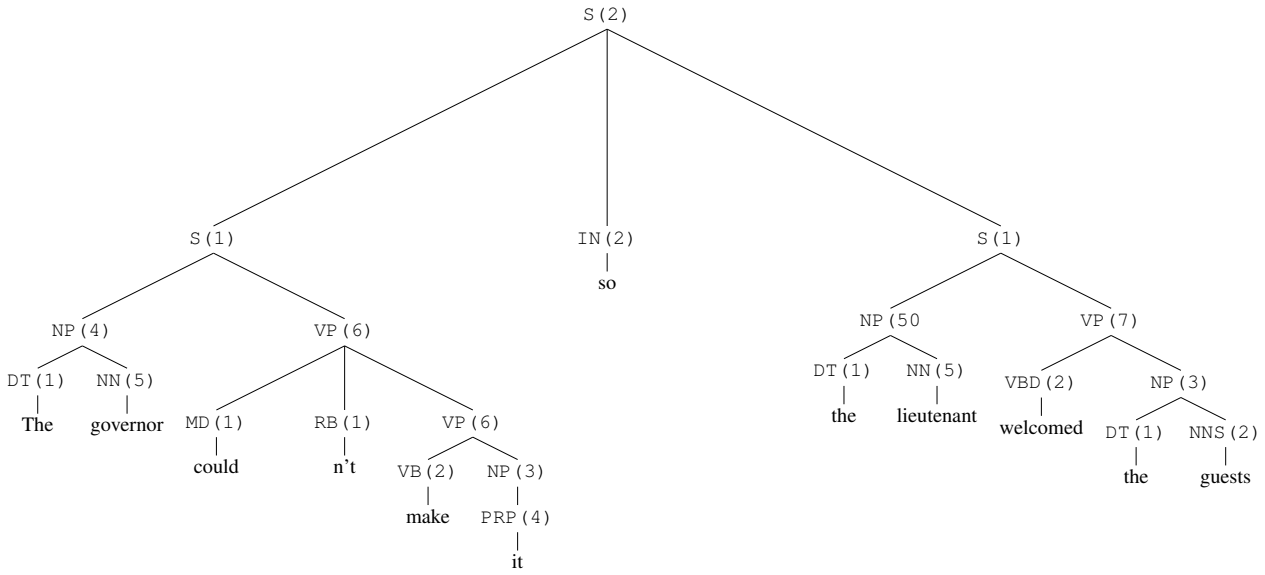

Figure 1: An example of a phrase-structure tree in English inspired by the Penn treebank (Marcus et al., 1993), potentially generated from an L-PCFG model. The indices next to each nonterminal in the tree denote the latent states associated with that node in the derivation. (Punctuation omitted.)

This idea is also strongly related to *lexicalized grammars*,[1] in which nonterminals are decorated with a head word propagated from the bottom of the tree. Most often, the head word is propagated using *head rules*, which decide which child of a given node is the head node based on linguistically motivated rules.

The context-free grammar formalism that corresponds to head lexicalization is bilexical grammar, which was introduced by Eisner and Satta (1999). Head lexicalization was used by Charniak (1997) and Collins (2003) to achieve state-of-the-art parsing results for English. Head lexicalization of grammars served as the basis for much of the subsequent parsing work.

Klein and Manning (2003) further built on the idea of tree transformations, and created linguistically motivated nonterminal refinements to parse the English treebank. Their work avoided the use of head lexicalization, but still produced a relatively high level of accuracy (though not state of the art) for parsing the Penn treebank. Some of the refinements they proposed generalize parent annotation (to higher order "vertical" Markovization) in a rather generic manner, but other refinements rely heavily on linguistic knowledge of English, and as such they do not generalize to treebanks in other languages.

With all of this previous work, nonterminal refinement is central to the underlying parsing formalism. However, these decorations are extracted from the treebank by means of transformations on trees. It was not until the work by Matsuzaki et al. (2005) and Prescher (2005) that the decoration became a "latent annotation." At that point, L-PCFGs were performing close to state of the art in syntactic parsing. Dreyer and Eisner (2006) suggested a more complex training algorithm for L-PCFGs to improve their accuracy. Then, Petrov et al. (2006) further improved the parsing results of L-PCFGs to match state of the art and also suggested a coarse-to-fine approach that made parsing much more efficient (the asymptotic computational complexity of parsing with L-PCFGs, in their vanilla form, grows cubically with the number of latent states). It was at this time that many other researchers started to make use of L-PCFGs for a variety of syntax parsers in different languages, some of which are described in the rest of the paper.

4 Learning of L-PCFGs

Given that we assume with L-PCFGs that the latent states decorating the nonterminals are not ob-

[1] This is different than the lexicalization of grammars where all derivation rules are put into the lexicon, such as with combinatory categorial grammars (Steedman, 2000) or head-driven phrase structure grammars (Pollard and Sag, 1994).

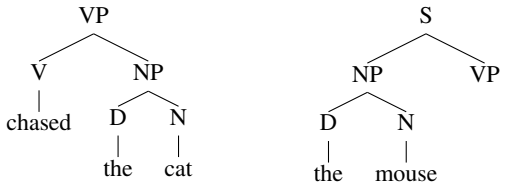

Figure 2: The inside tree (left) and outside tree (right) for the nonterminal `VP` in the parse tree `(S (NP (D the) (N mouse)) (VP (V chased) (NP (D the) (N cat))))` for the sentence *"the mouse chased the cat."*

served as part of the data, the learning problem posed by L-PCFGs is challenging and non-trivial. It requires learning the rule probabilities of an L-PCFG – such rules have latent states attached to them, as described in §2 – without knowing any of the latent states that are attached to the trees sampled from the underlying L-PCFG distribution.

Formally, we are given M training examples in the form of skeletal trees, $\tau^{(1)}, \ldots, \tau^{(M)}$ and our goal is to assign probabilities to the grammar rules with latent states. Implicitly, we assume that the skeletal grammar is given, and indeed it can be read off a treebank by extracting parents together with their immediate children (possibly after a "binarization" process).

4.1 Expectation-Maximization Learning

The first attempt at learning the rule probabilities of an L-PCFG from a treebank was carried out by Matsuzaki et al. (2005), who used the expectation-maximization (EM) algorithm. The EM algorithm is tailored to this specific problem of estimating L-PCFGs. It iterates between two steps, the E-step and the M-step. In the E-step it collects statistics about the latent state distributions for all nodes in all trees in the treebank (using a dynamic programming algorithm akin to CKY) and in the M-step it re-estimates the model based on these collected statistics. Before the first E-step is executed, the L-PCFG rule probabilities are initialized randomly.

While the EM algorithm is a highly influential algorithm that changed the way we reason about learning from incomplete data, it has some weaknesses, both practical and theoretical. The major practical weakness is that it requires running an "inference" algorithm multiple times over the data to collect the statistics in each E-step, which can be computationally expensive. This inference, as mentioned above, comes in the form of running a dynamic programming algorithm that computes a marginal probability for each node in each tree of the form $\mu(a, h_k, i, j)$ where i and j are indices in the string the dynamic programming is run on, a is a nonterminal and h_k is a latent state for node k in a tree.

This marginal probability corresponds to:

$$\mu(a, h_k, i, j) = \sum_{\tau(h):\ (a,h_k,i,j) \in \tau(h)} p(\tau(h))$$

i.e. the sum of all probabilities of full tree derivations (that include latent states) such that $a(h_k)$ spans the substring between index i and j. It is important to note that while the dynamic programming algorithm is akin to parsing algorithms such as CKY, its complexity is not cubic, but linear. This is true because during the E-step, the skeletal tree is fixed. We are not actually parsing a string, but instead marginalize the latent states in the fixed skeletal tree. When parsing a string with latent-variable PCFG (see §5), the complexity indeed becomes cubic because the skeletal tree needs to be inferred.

By estimating the parameters of an L-PCFG, the EM algorithm finds a maximum for the following objective function:

$$L(\tau^{(1)}, \ldots, \tau^{(M)}) = \sum_{i=1}^{M} \log \left(\sum_{h} p(\tau^{(i)}(h)) \right). \quad (1)$$

This is the *log-likelihood* of the observed data, marginalizing out the latent state from the underlying L-PCFG distribution. This log-likelihood function can be thought of as a measure of how well a specific set of parameters fit the data. The higher the log-likelihood is for a specific set of parameters, the more "likely" these parameters make the observed data.

Maximizing the log-likelihood function has deep roots in frequentist statistical theory, and in many cases, it can be shown that finding the global maximum of the log-likelihood will lead to "consistent" parameter estimates – meaning, as the amount of data increases, the parameters will get closer to the parameters from which the data was sampled.

The problem with the EM algorithm is that it only identifies a *local* maximum of this log-likelihood function, as the function has a "bumpy"

surface that includes several maxima, some better than others. Finding the global one (i.e. the local maximum that gives the highest value to the log-likelihood) is a computationally difficult problem for L-PCFGs, and there are no known solutions that guarantee such identification in an efficient manner. As a result, much of the theory of maximum likelihood estimation that is mentioned above does not apply to the EM algorithm with L-PCFGs.

It is important to note that this issue with the EM algorithm and L-PCFGs is more of a theoretical concern than a practical concern. In practice, if the EM algorithm is initialized in the way specified by Matsuzaki et al. (2005), it converges to a local maximum that provides a relatively high parsing accuracy for syntactic parsing of English and other languages. In addition, the log-likelihood is not fully correlated with parsing performance (such as that measured by the PARSEVAL metric; Black et al., 1991) and therefore identifying the global maximum of the log-likelihood function does not guarantee optimal parsing performance.

Coarse-to-Fine Techniques Building on the expectation-maximization algorithm, Petrov et al. (2006) introduced a coarse-to-fine technique (Charniak and Johnson, 2005) for estimating L-PCFGs. This technique uses the EM algorithm as a subroutine. It first starts by running the EM algorithm with a small number of latent states for each nonterminal. It then works by successively "splitting" and "merging" nonterminals. In a split step, more latent states are added to the nonterminals (usually by multiplying the number of latent states associated with them by two). To avoid overfitting the model to the training data, this is accompanied by a merge step, in which latent states are merged together to decrease the number of latent states associated with each nonterminal.

After each split and merge step (which are done in "split-merge cycles"), EM is initialized using the last model and re-run again to obtain a new set of parameters for the split/merge grammar. As such, this coarse-to-fine technique aims to locally maximize the marginal log-likelihood as given in Eq. 1 while also controlling for model size (the number of latent variables associated with each nonterminal).

4.2 Spectral Learning

At their core, spectral algorithms exploit the conditional independence that L-PCFGs makes to extract the parameters with the latent states (Cohen et al., 2013, 2014). More specifically, L-PCFGs assume that an "inside" tree and an "outside" tree, shown in Figure 2 are conditionally independent of each other given the nonterminal and latent state that attaches them to each other. As such, the correlation between patterns in the inside tree and outside tree distributions dictate the identity of the latent states and their distribution. To identify such a correlation, one can extract the latent state parameters by building a co-occurrence matrix (or a cross-covariance matrix) of inside and outside trees (in skeletal form; these are represented by feature vectors over such trees; see below), and then apply singular value decomposition (SVD; Strang et al., 1993) on this matrix. This approach was originally introduced for hidden Markov models (Hsu et al., 2012) and has been used for other types of grammars and parsing formalisms as well (Bailly et al., 2010; Luque et al., 2012; Dhillon et al., 2012).

As mentioned above, the inside and outside trees are represented by *feature vectors* in the co-occurrence matrix. This means that the inside and outside trees are mapped to real vectors. This is a common way to reduce a structured object into a manageable mathematical object that can be statistically processed. In the case of spectral algorithms for parsing, the feature functions indicate local neighborhood surrounding the top node (for inside trees) or footer node (for outside trees). As such, these methods distill information that was previously used by approaches such as parent annotation, annotation with linguistic features or Markovization (see §3) into latent states. The EM algorithm, on the other hand, does not make any use of feature functions in the process of learning.

Another advantage of spectral algorithms over the expectation-maximization algorithm is that they provide a natural way to select the number of latent states for each nonterminal. The singular values of the inside-outside co-occurrence matrix offer a criterion to do that. Each singular value is associated with a latent state, and to retain a good approximation of this matrix with SVD, one needs to select only the largest singular values. The smaller ones can be removed from the SVD approximation.

That being said, Narayan and Cohen (2016) showed that the number of latent states can be further optimized with spectral algorithms by using coarse-to-fine techniques such as Petrov et al. (2006) used. This means that while the criterion above with top singular values is natural and easy to implement, further refinements can be made to improve it.

It is important to note that unlike spectral algorithms, the EM algorithm has an interpretation that is valid even when the data it is applied on is not generated from an L-PCFG in the family we are estimating from. It can be viewed as minimizing Kullback-Leibler (KL) divergence, a measure for distributional divergence, between the empirical distribution and the family of possible L-PCFGs from which a model is selected. To date, the theoretical guarantees of L-PCFGs with spectral algorithms require the assumption that the data is generated from an L-PCFG distribution. Still, the EM algorithm and spectral algorithms yield similar results on a variety of benchmarks for multilingual parsing, even for data that are clearly not sampled from an L-PCFG (as one might argue is true for most natural language data).

4.3 Other Learning Algorithms

Other scenarios and algorithms for learning L-PCFGs have been proposed. One such example is that of the work on self-training (McClosky et al., 2006) of L-PCFGs by Huang et al. (2010) and Huang and Harper (2009). With self-training, a parser is trained from seed annotated data (such as the Penn treebank), and then the parser learned is used to parse a large amount of unlabeled data. After that step, a new parser is learned from both the annotated data and the parsed data, as if the latter are gold-standard data.

Petrov (2010) exploited the fact that the EM algorithm is sensitive to its initialization point (and returns a different model in each execution) and estimated an L-PCFG multiple times from annotated data using a coarse-to-fine EM technique. Once all grammars were learned, they were combined in a product-of-experts style, and then they were used to parse unseen data. Similar experiments were done later with spectral algorithms, and showed that essentially using multiple grammars has the effect of regularization.

Petrov and Klein (2008) extended L-PCFGs to log-linear latent grammars – this means that the rule weights learned are no longer constrained to be probabilities. Instead, the model has an additional normalization constant that ensures that it defines a probability of derivation trees even though the rule weights are no longer probabilities. They used *discriminative training* to estimate such L-PCFGs. Instead of finding a local maximum of the marginal log-likelihood as EM does, they find a local maximum for the *conditional* marginal log-likelihood.

One of the earlier training algorithms of L-PCFGs selectively refines nonterminals by using EM with annealing (Dreyer and Eisner, 2006). The authors slightly modify the L-PCFG model to pass features between nodes in the parse tree, motivated by prior linguistic work on feature passing. Finally, Stanojević and Sima'an (2015) used a refinement model for the induction of a Reordering Grammar for machine translation. They used the EM algorithm for estimating their model.

5 L-PCFGs for Multilingual Parsing

In this section, we discuss the use of L-PCFGs for syntactic parsing.

5.1 Parsing with L-PCFGs

Parsing with L-PCFGs usually entails finding a skeletal tree for a given string. While the latent variables assist in the modeling by adding contextual information to the derivation, they are not necessarily a target for prediction, and therefore we are interested in marginalizing them out during parsing.

Given a string w, this means that we are interested in finding the following tree:

$$\tau^* = \arg\max_{\tau} \sum_{h} p(\tau(h) \mid w).$$

The maximization of a sum of products in this form (the product originates in the probability $p(\tau(h) \mid w)$, which is proportional to a product of rule probabilities) is computationally intractable. As such, other approaches to parsing with L-PCFGs have been developed. The most common one used is based on *minimum Bayes risk decoding* (MBR; Goodman, 1996). With MBR, the maximization problem turns into maximizing the sum of the marginal probabilities of each node that appear in the tree. It is motivated by maximizing the recall of correct constituents in the predicted tree.

MBR can be quite expensive to run with a large number of latent states, as the dynamic programming algorithm that performs the parsing scales cubically as a function of the number of latent states. This is where coarse-to-fine techniques have an advantage – one can parse incrementally with coarser models until reaching the most refined model, at each point pruning the parsing chart with low probability items from the coarser model. Tensor decomposition can also be used to speed up L-PCFG parsing (Cohen and Collins, 2012).

The main application for L-PCFGs is multilingual syntactic parsing. In their early version (Matsuzaki et al., 2005), L-PCFGs were used to parse the English Penn treebank with some success – the results were not state of the art, but close to it. In subsequent work, Petrov et al. (2006) used coarse-to-fine techniques to further improve EM estimation of L-PCFGs. This led to the development of the Berkeley parser, which has given state-of-the-art results for English and other languages. Spectral algorithms also yield results which are close to state of the art in a multilingual setting.

Since their inception, L-PCFGs have been used for syntactic parsing in multiple studies for a variety of languages such as English, French, German, Chinese, Arabic and other morphologically rich languages (Candito et al., 2010; Attia et al., 2010; Green and Manning, 2010; Tounsi and Van Genabith, 2010; Goldberg and Elhadad, 2011; Dehdari et al., 2011; Björkelund et al., 2014; Zeng et al., 2014; Sun et al., 2014; Huang et al., 2014; Narayan and Cohen, 2016)

5.2 Interpretation for Latent States

One of the advantages of using latent variable models is that often the latent variables can be assigned an interpretation once they are inferred. This post-hoc interpretation can be revealing about linguistic patterns that are present in the data and are learned automatically.

To do this kind of analysis, we computed the marginals for each node in each tree in the Penn treebank after training a model with the spectral algorithm of Narayan and Cohen (2015). The results are available in the LPCFGVIEWER tool.[2]

[2]Available in http://cohort.inf.ed.ac.uk/lpcfgviewer. The online tool includes analysis of other languages, including French, German, Hebrew, Hungarian, Korean, Polish, Swedish and Basque.

State	Frequent words
IN (preposition)	
0	of ×323
1	about ×248
2	than ×661, as ×648, because ×209
3	from ×313, at ×324
4	into ×178
5	over ×122
6	Under ×127
DT (determiners)	
0	These ×105
1	Some ×204
2	that ×190
3	both ×102
4	any ×613
5	the ×574
6	those ×247, all ×242
7	all ×105
8	another ×276, no ×211
CD (numbers)	
0	8 ×132
1	million ×451, billion ×248
RB (adverb)	
0	up ×175
1	as ×271
2	not ×490, n't ×2695
3	not ×236
4	only ×159
5	well ×129
CC (conjunction)	
0	But ×255
1	and ×101
2	and ×218
3	But ×196
4	or ×162
5	and ×478

Table 1: A list of part-of-speech tags and most frequent words associated with latent states that were learned for them using the algorithm of Narayan and Cohen (2015). The numbers next to each word indicate a strength (the number of times that word appeared with that POS tag and latent state in the Penn treebank).

There are a few observations that can be concluded when inspecting these results:

- **Lexicalization of closed-word tags:** For the closed-word part-of-speech (POS) tags, both the EM algorithm and the spectral algorithm asso-

ciate a latent state with mostly a single word. For example, for determiners, there would be a latent state for "the" and for "a." Often words with different casing or contracted form joined in the same cluster. Table 1, for example, shows that latent state 2 for RB (adverbs) is associated with "not" and "n't."

- **Semantic clustering of phrases:** Consider the noun phrases in Table 2, which are the most frequently ones associated with each latent state in the learned L-PCFG model. We see that there is in certain cases semantic clustering of such noun phrases, in cases where such semantic clustering can be directed by syntactic information in the parse trees. For example, the first latent state of the noun phrases is mostly associated with dates in the form of month, day, year. The seventh latent state is mostly associated with dollar amount.
- **Dependence on domain:** It is clear from Table 2 and similar statistics for other nonterminals that the association of latent states with nonterminal phrases is highly dependent on the domain on which the L-PCFG was trained. The L-PCFG in Table 2 was trained on the Penn treebank, and as such, latent states are associated with financial terms that have some similarity (such as shares and currency).

When inspecting the phrases associated with specific latent states and nonterminals (in the LPCFGVIEWER tool), one might argue that there is a weak notion of substitutability that exists with L-PCFGs. By this, we are referring to the idea that phrases associated with an identical latent state with high probability are more likely to be substitutable in different contexts, not just syntactically, but also semantically. The latent state associated with dates (in noun phrases) in Table 1 is one example of that. This hypothesis remains to be proven empirically in a methodical way.

6 Extensions of L-PCFGs and Related Models

Extensions Natural generalizations of PCFGs, such as probabilistic linear-context free rewriting systems (LCFRS; Kallmeyer and Maier, 2010) and synchronous grammars can also be turned into probabilistic grammars with latent states. As long as the backbone structure of the skeletal grammar is of the form $a \rightarrow \alpha$ where α includes nonterminals in one form or the other, the nonterminals can be decorated with additional latent state information.

Work about using other grammar formalisms with latent states includes the work of Fowler and Penn (2010) who introduced latent states into a combinatory categorial grammar (CCG) for syntactic parsing, the work of Saluja et al. (2014), who generalized L-PCFGs to synchronous L-PCFGs and proposed to estimate them using both a spectral algorithm and EM for machine translation and the work of Louis and Cohen (2015) who modeled online forum topic structure by using LCFRS with latent states (the latent states corresponded to topics that need to be inferred from data). Models similar to L-PCFGs have been used for parsing with discontinuous elements (Nederhof and Yli-Jyrä, 2017). They have also been used to describe transition-based systems for dependency formalisms (Nederhof, 2016).

Related Models As mentioned above, the traditional L-PCFG parsing algorithm requires computing marginals for each node in the tree. They are computed using an inside-outside algorithm – an inside pass that works bottom up in the tree, similarly to the CKY algorithm, and an outside pass that works top down.

When computing marginals with the skeletal tree being fixed, the bottom up inside algorithm can be viewed as an algorithm that *propagates* vector representations of each node up in the tree by using tensor contraction. The general form of that algorithm is:

$$v_{\text{parent}} = F(v_{\text{lc}}, v_{\text{rc}}). \tag{2}$$

where $v_{\text{parent}}, v_{\text{lc}}, v_{\text{rc}} \in \mathbb{R}^m$ denoting vectors associated with a parent node, its left child and its right child, and $F\colon \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ takes as input two children node vectors and returns the output node vector. In the case of L-PCFGs, F is a function that is associated with a rule $a \rightarrow b \;\; c$, $T^{a \rightarrow b \;\; c}$, where b is the nonterminal for the left child, c is the nonterminal for the right child and a is the parent nonterminal. $T^{a \rightarrow b \;\; c}$ is then defined as:

$$[T^{a \rightarrow b \;\; c}(v_{\text{lc}}, v_{\text{rc}})]_{h_1} = \sum_{h_2, h_3} p(a(h_1) \rightarrow b(h_2) \;\; c(h_3) \mid a(h_1))[v_{\text{lc}}]_{h_2}[v_{\text{rc}}]_{h_3}.$$

0	”Aug. 30 , 1988”, ”Aug. 31 , 1987”, ”Dec. 31 , 1988”, ”Oct. 16 , 1996”, ”Oct. 1 , 1999”, ”Oct. 1 , 2019”, ”Nov. 8 , 1996”, ”Oct. 15 , 1999”, ”April 30 , 1988”, ”Nov. 8 , 1994”
1	”12,000 miles”, ”About 20,000 years”, “this year”, “A year”, “a year” $\times 7$
2	“FROG-7B missiles , the bomber version of the An-12 , MiG-23BN high-altitude aircraft , MiG-29s , which can outfly Pakistan ’s U.S.-built F16s ,”, “AMERICAN BUILDING MAINTENANCE INDUSTRIES Inc. , San Francisco , provider of maintenance services , annual revenue of $ 582 million , NYSE ,”, “DIASONICS INC. , South San Francisco , maker of magnetic resonance imaging equipment , annual sales of $ 281 million , Amex ,”, “EVEREX SYSTEMS INC. , Fremont , maker of personal computers and peripherals , annual sales of $ 377 million , OTC ,”, “ANTHEM ELECTRONICS INC. , San Jose , distributor of electronic parts , annual sales of about $ 300 million , NYSE ,”, “APPLIED MATERIALS INC. , Santa Clara , maker of computer-chip machine systems , annual sales of $ 490 million , OTC ,”
3	“James McCall , vice president , materials , at Battelle , a technology and management-research giant based in Columbus , Ohio”, “Frank Kline Jr. , partner in Lambda Funds , a Beverly Hills , Calif. , venture capital concern”, “Allen Hadhazy , senior analyst at the Institute for Econometric Research , Fort Lauderdale , Fla. , which publishes the New Issues newsletter on IPOs”, “a group of investment banks headed by First Boston Corp. and co-managed by Goldman , Sachs & Co. , Merrill Lynch Capital Markets , Morgan Stanley & Co. , and Salomon Brothers Inc”, “Charles J. O’Connell , deputy district director in Los Angeles of the California Department of Transportation , nicknamed Caltrans”, “Francis J. McNeil , who , as deputy assistant secretary of state for inter-American affairs , first ran across reports about Mr. Noriega in 1977”
4	“TREASURY BILLS : Results of the Monday , October 16 , 1989 , auction of short-term U.S. government bills , sold at a discount from face value in units of $ 10,000 to $ 1 million : 7.37 % 13 weeks ; 7.42 % 26 weeks .”, “California Health Facilities Financing Authority – $ 144.35 million of revenue bonds for Kaiser Permanente , due 19931999 , 2004 , 2008 , 2018 and 2019 , tentatively priced by a PaineWebber Inc. group to yield from 6.25 % in 1993 to 7.227 % in 2018 .”, “TREASURY BILLS : Results of the Monday , October 16 , 1989 , auction of short-term U.S. government bills , sold at a discount from face value in units of $ 10,000 to $ 1 million : 7.37 % 13 weeks ; 7.42 % 26 weeks .”, “Health Care Property Investors Inc. , offering of 2,250,000 shares of common stock , via Merrill Lynch Capital Markets , Alex . Brown & Sons Inc. and Dean Witter Reynolds Inc .”, ”SUN MICROSYSTEMS INC. , Mountain View , maker of desktop computers , annual sales $ 1.77 billion , OTC , no injured employees and very little damage to buildings .”
5	“$ 615,000 face amount”, “a share” $\times$ 7, “a revival”
6	“bonds due Nov. 2 , 1993 , with equity-purchase warrants”, “bonds due Nov. 8 , 1994 , with equity-purchase warrants”, “20,000 to 30,000 Soviet Central Asian KGB Border Guards”, “bonds due Nov. 1 , 1993 , with equity-purchase warrants”, “a van Gogh , a Monet , other paintings , furniture”, “3,111,000 common shares”, “offering of 2,250,000 shares of common stock”, “a Newark , N.J. , textile businessman”
7	“$ 1,150,000”, “166,900,000 shares”, “$ 124,732”, “$ 45,000”, “$ 1,500”, “$ 20,000”, “$ 342,122”, “$ 1,000”, “$ 3,000”, “S$ 500,000”
8	“$ 20,000 a year”, “$ 342,122 last year”, “as many as 60,000 additional tourists a day”, “$ 80,000 a year”, “1,000 flights a day”, “26,000 units next year”, “200,000 cars a year”, “$ 200 a share” $\times 3$

Table 2: Examples of most likely phrases for the noun phrase category (NP) for a latent-variable PCFG extracted using the algorithm of Narayan and Cohen (2015) from the Penn treebank. Numbers next to the phrases indicate that the phrase appeared multiple times in the list.

This general formulation as in Eq. 2 gives rise to generalizations of other formulations of latent representations that are propagated in a (parse) tree structure. Perhaps the most related one to L-PCFGs is the recursive neural network of Socher et al. (2010). This recursive neural network propagates word vectors (Turian et al., 2010) from the bottom of an unlabeled tree all the way to its top using the function:

$$F(v_{\text{lc}}, v_{\text{rc}}) = \tanh(W[v_{\text{lc}}, v_{\text{rc}}] + b),$$

where $W \in \mathbb{R}^{m \times 2m}$ and $b \in \mathbb{R}^m$ are weight matrices and biases that are learned when training the neural network, $[u_1, u_2]$ denotes the concatenation of two vectors u_1 and u_2, and $\tanh\colon \mathbb{R}^m \to \mathbb{R}^m$ is a function that applies the hyperbolic tangent function coordinate-wise. Even more closely related to L-PCFGs is the further refinement of these recursive neural networks by Socher et al. (2013) where the weights in the neural network are parametrized by labels in the tree, corresponding to syntactic categories.

Similarly to a formulation of the inside tree as a vector propagation procedure (in Eq. 2), there is also a formulation for the outside algorithm (Cohen et al., 2014). Le and Zuidema (2014) also extended the recursive neural networks mentioned above to make use of the outside tree information.

Finally, it is also important to note that L-PCFGs are related to probabilistic regular tree grammars (PRTGs; Knight and Graehl, 2005) where the righthand side trees of the PRTG rules are of depth 1. With general PRTGs, the righthand side can be of arbitrary depth, where the leaf nodes of these trees correspond to latent states in the L-PCFG formulation above and the internal nodes of these trees correspond to interminal symbols in the L-PCFG formulation.

7 Conclusion

Latent-variable PCFGs are a flexible model for modeling syntax and other problems in NLP. Their backbone is a symbolic grammar, and as such they can be easily interpreted, while they are augmented with probabilities and latent states, allowing the modeler to reason under uncertainty.

We gave an overview of the parsing and learning algorithms used with L-PCFGs and described some natural extensions and related models.

Acknowledgments

I would like to thank Shashi Narayan, Nikos Papasarantopoulos and Giorgio Satta for useful feedback and comments. This work was supported by an EU H2020 grant (688139/H2020-ICT-2015; SUMMA) and a grant from Bloomberg.

References

M. Attia, J. Foster, D. Hogan, J. Le Roux, L. Tounsi, and J. Van Genabith. 2010. Handling unknown words in statistical latent-variable parsing models for Arabic, English and French. In *Proceedings of the NAACL-HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*.

R. Bailly, A. Habrard, and F. Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*.

A. Björkelund, Ö. Çetinoğlu, A. Faleńska, R. Farkas, T. Müller, W. Seeker, and Z. Szántó. 2014. Introducing the IMS-Wrocław-Szeged-CIS entry at the SPMRL 2014 shared task: Reranking and morphosyntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of DARPA Workshop on Speech and Natural Language*.

M. Candito, J. Nivre, P. Denis, and E. H. Anguiano. 2010. Benchmarking of statistical dependency parsers for french. In *Proceedings of COLING*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

S. B. Cohen and M. Collins. 2012. Tensor decomposition for fast parsing with latent-variable PCFGs. In *Proceedings of NIPS*.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *Journal of Machine Learning Research* .

M. Collins. 2003. Head-driven statistical models for natural language processing. *Computational Linguistics* 29:589–637.

J. Dehdari, L. Tounsi, and J. van Genabith. 2011. Morphological features for parsing morphologically-rich languages: A case of Arabic. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*.

P. S. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. 2012. Spectral dependency parsing with latent variables. In *Proceedings of CoNLL-EMNLP*.

M. Dreyer and J. Eisner. 2006. Better informed training of latent syntactic features. In *Proceedings of EMNLP*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*.

T. AD Fowler and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of ACL*.

Y. Goldberg and M. Elhadad. 2011. Joint Hebrew segmentation and parsing using a PCFG-LA lattice parser. In *Proceedings of ACL (short papers)*.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*.

S. Green and C. D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of COLING*.

G. Guerra and Y. Aloimonos. 2005. Discovering a language for human activity. In *Proceedings of AAAI Workshop on Anticipation in Cognitive Systems*.

J. Hockenmaier and M. Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of ACL*.

D. Hsu, S. M. Kakade, and T. Zhang. 2012. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences* 78(5):1460–1480.

Q. Huang, L. He, D. F. Wong, and L. S. Chao. 2014. Chinese unknown word recognition for PCFG-LA parsing. *The Scientific World Journal* 2014.

Z. Huang and M. Harper. 2009. Self-training pcfg grammars with latent annotations across languages. In *Proceedings of EMNLP*.

Z. Huang, M. Harper, and S. Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of EMNLP*.

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

L. Kallmeyer and W. Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proceedings of COLING*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Computational linguistics and intelligent text processing*, Springer, volume 3406 of *Lecture Notes in Computer Science*, pages 1–24.

P. Le and W. Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of EMNLP*.

L. Lin, T. Wu, J. Porway, and Z. Xu. 2009. A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition* 8.

A. Louis and S. B. Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *Proceedings of EMNLP*.

F. M. Luque, A. Quattoni, B. Balle, and X. Carreras. 2012. Spectral learning for non-deterministic dependency parsing. In *Proceedings of EACL*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19:313–330.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL*.

S. Narayan and S. B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

S. Narayan and S. B. Cohen. 2016. Optimizing spectral learning for parsing. In *Proceedings of ACL*.

M.-J. Nederhof. 2016. Transition-based dependency parsing as latent-variable constituent parsing. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*.

M.-J. Nederhof and A. Yli-Jyrä. 2017. A derivational model of discontinuous parsing. In *International Conference on Language and Automata Theory and Applications*. Springer, pages 299–310.

S. Petrov. 2010. Products of random latent variable grammars. In *Proceedings of NAACL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

S. Petrov and D. Klein. 2008. Discriminative log-linear grammars with latent variables. In *Proceedings of NIPS*.

C. Pollard and I. A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

D. Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *Proceedings of ECML*.

Y. Sakakibara, M. Brown, R. Hughey, S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. 1994. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research* 22.

A. Saluja, C. Dyer, and S. B. Cohen. 2014. Latent-variable synchronous CFGs for hierarchical translation. In *Proceedings of EMNLP*.

R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*.

R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS Deep Learning and Unsupervised Feature Learning Workshop*.

M. Stanojević and K. Sima'an. 2015. Reordering grammar induction. In *Proceedings of EMNLP*.

M. Steedman. 2000. *The syntactic process*. MIT Press.

G. Strang. 1993. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA.

L. Sun, J. Mielens, and J. Baldridge. 2014. Parsing low-resource languages using Gibbs sampling for PCFGs with latent annotations. In *Proceedings of EMNLP*.

L. Tounsi and J. Van Genabith. 2010. Arabic parsing using grammar transforms .

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*.

X. Zeng, D. F. Wong, L. S. Chao, I. Trancoso, L. He, and Q. Huang. 2014. Lexicon expansion for latent variable grammars. *Pattern Recognition Letters* 42:47–55.