Exploring Latent Variable Models for Syntactic Co-occurrence Data

John Blitzer and Fernando Pereira
Computer and Information Science Department
University of Pennsylvania
blitzer@cis.upenn.edu, pereira@cis.upenn.edu

Abstract

We propose two latent variable models for lexical and non-lexical data which co-occur syntactically. Such data occur most prominently in the context of statistical parsing, where lexicalized nonterminals have created an explosion in the number of parameters to be estimated. As such, modern head-lexicalized statistical parsers are confronted with a severe data sparseness problem. All parsers alleviate this problem by smoothing the maximum likelihood distributions over lexicalized nonterminals, but to our knowledge no systematic study of smoothing techniques has been done. In this paper, we compare the backoff scheme of Collins [5] with two latent variable models designed to smooth maximum likelihood conditional distributions. The first model is a variant of the aggregate Markov model of Saul and Pereira [12], which we extend to complex, multivariate histories. The second model is a more flexible distributed latent variable model which scales more elegantly than an aggregate Markov model and is designed specifically with multivariate histories in mind. We give a GEM algorithm for estimating the maximum likelihood parameters of this model, and for both models we give perplexity results showing improvement over the smoothing scheme of Collins.

1 Introduction

The best statistical CFG parsers all share the common representation of lexicalized nonterminals; each CFG nonterminal is augmented with its lexical head [5]. This addition of lexical items to the nonterminal space constitutes a powerful parsing model and leads to a significant increase in performance over unlexicalized models. Unfortunately, it also explodes the number of possible nonterminals, and the syntactic cooccurrences that are used to estimate the parameters of these parsers are consequently quite sparse. Thus a parser's technique for smoothing the maximum likelihood distributions over the space of lexicalized nonterminals is essential to its performance. To our knowledge, however, there have been no thorough investigations of different kinds of smoothing schemes for natural language parsing.

There have been some investigations of the backoff smothing scheme in the generative PCFG model of Collins [5]. In particular, Gildea [6] and Bikel [2] investigate the use of maximum likelihood distributions of different specificity and show that maximum likelihood bilexical distributions contribute very little to parser performance. These investigations raise the question of whether or not the complex histories employed by statistical parsers are useful at all. If they are, what model can improve over a backoff scheme and take advantage of their usefulness? In this work we compare systematically the smoothing scheme of Collins model 1 [5] with two latent variable models which are designed to induce compact representations for complex, multivariate histories such as those found in probabilistic context free parsers.

The first latent variable model is an extension of the aggregate Markov model of Saul and Pereira [12]. We give a version of this model which is designed explicitly to give a representation intermediate to two multivariate histories, such as the type encountered in statistical parsing. The second model is a model with multiple latent variables that we call the distributed model. It is similar in spirit to the neural network language model of Bengio et. al. [1], but it is most closely related to sigmoid belief networks [10]. The distributed model is designed explicitly to model multivariate histories, and the number of parameters scales more gracefully than that of the aggregate Markov model. Training is more complicated than for the aggregate model; in particular, the M-step does not have a closed-form solution. Nevertheless, we found that a GEM algorithm achieves reasonable convergence.

The latent variable models we introduce in this paper can be seen as a type of clustering, and it is worth mentioning that the idea of clustering histories to improve parsing is not novel. Magerman [9] used a mutual information clustering technique to encode parse derivation histories as bit vectors, resulting in a history representation that can be thought of as a heuristic hard-clustering precursor of our distributed model. Charniak [3] also attempted to generalize across histories by clustering, and the parser of Henderson [7] uses a neural network to induce low-dimensional representations for complex histories. However, none of those studies provided a conclusive comparison of clustered and unclustered history representations. Charniak reported no empirical improvement from clustering, but the details of his clustering method and its use in history representation are not available.

To facilitate accurate model comparison, we use exactly the same conditioning events in our models as in the Collins model. We show that our latent variable models give improvement in perplexity over Collins model 1. Lower perplexity may not always yield better accuracy, as is shown by the long struggle to demonstrate the benefits of lower-perplexity language models in speech recognition [4]. However, perplexity results help us gain insight into the relative strengths and weaknesses of different smoothing schemes. In section 2 we describe the Collins smoothing model. Section 3 describes our aggregate Markov models. Section 4 gives perplexity results for these models. Section 5 introduces the distributed model and gives results comparing it with aggregate Markov models and with Collins model 1. We conclude in section 6.

2 The Collins Smoothing Scheme for Syntactic Cooccurrences

For his parser, Collins used a generative probabilistic context-free grammar (PCFG) model that decomposes the joint probability of a sentence and its parse as the product of the probabilities of the rewrite rules used to derive the parse. Each rewrite rule probability is then again decomposed: the right-hand side of a rule is generated by first generating the head child and then generating the modifier children independently of each other, conditioning on the head child and the parent. These independence assumptions result in distributions over lexicalized modifier children, conditioned on lexicalized parents and head children. The most sparse distribution that the Collins parser must estimate is the distribution over modifier lexical heads. This is the distribution on which we report our results. Collins smooths it by interpolating three maximum likelihood estimates:

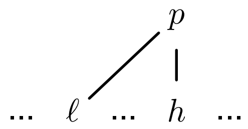

Figure 1: Modifier-head-parent relation

$$\begin{aligned}
e_1 &= P_{\mathrm{ML}}(w(l)|c(l), t(l), c(p), c(h), w(h), t(h), \Delta) \\
e_2 &= P_{\mathrm{ML}}(w(l)|c(l), t(l), c(p), c(h), t(h), \Delta) \\
e_3 &= P_{\mathrm{ML}}(w(l)|t(l))
\end{aligned}$$

Here, p is the parent phrase, h is the head phrase, and l the modifier phrase being generated, as shown schematically on Figure 1. The functions $c(\cdot)$, $t(\cdot)$ and $w(\cdot)$ denote the non-terminal symbol, head part-of-speech tag, and head word of the given phrase, respectively. Δ represents the distance between the parent head and the modifier head quantized into a small number of bins. When we do need to specify explicitly the conditioning variables, we use the generic $\mathcal{H}$ (the history) to represent the whole conditioning event. The interpolated probability estimate is then

$$P(w(l)|c(l), t(l), c(p), c(h), w(h), t(h), \Delta) = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2) e_3)$$

The interpolation coefficients λ_i, which depend on the history, are computed using a modified form of the Witten-Bell smoothing technique

3 Aggregate Markov Models

3.1 Model Description and EM Algorithm

The aggregate Markov models (AMMs) used by Saul and Pereira [12] are simple latent variable language models. Similar models were later studied by Hofmann and Puzicha as special cases of what they call "aspect" or "separable mixture" models [8]. In our application, the aggregate model smooths a conditional distribution such as $P(w|\mathcal{H})$, where w is a modifier lexical head, by marginalizing a latent variable model:

$$P(w|\mathcal{H}) = \sum_z p(z|\mathcal{H}) \cdot q(w|z)$$

Figure 2 represents the model graphically. The latent variable Z ranges over hidden states, and the histories $\mathcal{H}$ and words w are conditionally independent given Z. When the Z has a cardinality much smaller than the vocabulary size, the model has the effect of aggregating pairs $\langle \mathcal{H}, w \rangle$ and will assign probabilities to pairs which are unseen in training.

The parameters of the AMM are $p(z|\mathcal{H})$ and $q(w|z)$, which can be determined using the EM algorithm. In the E-step, we estimate the posterior distribution over the hidden variable Z given the observed data:

$$\rho^{(t)}(Z = z|\mathcal{H}, w) = \frac{p^{(t)}(z|\mathcal{H}) \cdot q^{(t)}(w|z)}{\sum_{z'} p^{(t)}(z'|\mathcal{H}) \cdot q^{(t)}(w|z')}$$

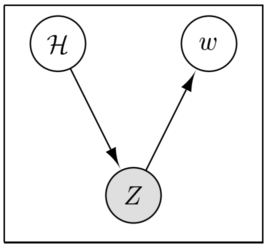

Figure 2: Graphical Model Representation for Aggregate Markov Models

In the M-step, we update the model parameters p and q to maximize the complete data log-likelihood function. The updates are:

$$
\begin{aligned}
p^{(t+1)}(z|\mathcal{H}) &= \frac{\sum_i c(\mathcal{H}, w_i)\rho^{(t)}(z|\mathcal{H}, w_i)}{\sum_i c(\mathcal{H}, w_i)\sum_{z'}\rho^{(t)}(z'|\mathcal{H}, w_i)} \\
q^{(t+1)}(w|z) &= \frac{\sum_i c(\mathcal{H}_i, w)\rho^{(t)}(z|\mathcal{H}_i, w)}{\sum_i \sum_{w'} c(\mathcal{H}_i, w')\rho^{(t)}(z|\mathcal{H}_i, w')}
\end{aligned}
$$

Here $c(\cdot,\cdot)$ is the empirical count of the given joint event, and i indexes over training data instances.

3.2 Nonterminal Information and Regularization

The AMMs of section 3.1 are intermediate models which are less specific than a maximum likelihood distribution conditioned on some history but more specific than an unconditioned marginal distribution, which discards all of that history. In practice we wish to use AMMs as models between a maximum likelihood distribution and a backoff distribution which discards only part of the history. Consider a history $\mathcal{H} = \langle \mathcal{H}_1, \mathcal{H}_2 \rangle$. An AMM that represents an intermediate distribution between these two models gives the following conditional distribution:

$$P_{\text{AMM}}(w|\mathcal{H}) = \sum_z p(z|\mathcal{H}_1) \cdot q(w|z, \mathcal{H}_2)$$

We can then interpolate the AMM with the original models.

$$
\begin{aligned}
P(w|\mathcal{H}) = \;& \lambda_1(\text{bin}(\mathcal{H}))P(w|\mathcal{H}) + \lambda_2(\text{bin}(\mathcal{H}))P_{\text{AMM}}(w|\mathcal{H}) \\
& +\lambda_3(\text{bin}(\mathcal{H}))P(w|\mathcal{H}_2)
\end{aligned} \tag{1}
$$

Here $\text{bin}(\mathcal{H})$ is a mapping from histories to bins, the number of which is chosen empirically. The interpolation weights for each bin are constrained to be positive and sum to unity. In practice, we use AMMs by inserting them in this way between e_1 and e_2 and between e_2 and e_3.

AMMs are designed to alleviate the overfitting caused by limited training data. Unfortunately, and especially in the case of AMMs where the cardinality of the latent variable Z grows large, AMMs themselves can overfit the training data. Tempered EM is an application of deterministic annealing to the posterior distribution over hidden states estimated in the E-step [8]. Tempering gives a free parameter which parameterizes a new posterior. Depending on the value of that parameter, the new posterior varies between the original posterior of the EM algorithm and the uniform distribution over hidden states. A more uniform distribution is more robust in scenarios with sparse data. We optimize this free parameter on a heldout data set.

4 AMM Results and Shortcomings

To demonstrate the smoothing power of AMMs, we give perplexity results first on lexical head bigrams (verb and object, verb and preposition, etc). In table 1, "Collins" refers to the interpolation between bigram and unigram distributions, with the weights chosen according to the Witten Bell scheme of Collins [5]. "Linear Interpolation" refers to an interpolation in which the weights are chosen to maximize the likelihood of heldout data. Rows labeled "AMM" indicate that an aggregate Markov model of the corresponding size was interpolated together with the bigram and unigram distributions. The resulting probability distribution has the form of equation 1, where $\mathcal{H} = w(h)$ and $\mathcal{H}_2$ is the empty history. Once again the weights were chosen to maximize heldout data likelihood. All of the aggregate models are trained using tempered EM.

Model	Perplexity
Collins	581
Linear Interpolation	573
AMM-8-Tempered	537
AMM-32-Tempered	516
AMM-128-Tempered	**510**

Table 1: AMM results on lexical bigram data

In table 2, "Collins" refers to the full Collins backoff scheme of section 2. "Linear Interpolation" describes interpolating among e_1, e_2, and e_3 with the weights trained to maximize likelihood. AMM-e_1/e_2 refers to adding to the linear interpolation the AMM that falls between e_1 and e_2 (the one which aggregates lexical head information). Similarly AMM-e_2/e_3 refers to the AMM that falls between e_2 and e_3.

Model	Perplexity
Collins	80.13
Linear Interpolation	80.40
AMM-e_1/e_2	80.12
AMM-e_2/e_3	80.13
AMM-Both	79.3

Table 2: AMM results on lexical bigram data

While the best aggregate models do improve perplexity over the Collins model, the improvement is only by about 1%, compared to the 12% improvement in the case of the lexical bigrams. This could be due to two closly-related shortcoming of AMMs. The first is the number of parameters. An AMM has roughly a number of parameters that scales linearly with the cardinality of the latent variable. This is reasonable for lexical bigrams and a 15000 word vocabulary, where the total number of parameters for a 128 class model is roughly 3.8 million. But for models such as AMM-e_1/e_2, this number is already 1.9 billion (There are roughly 1000 unique non-terminal-only histories). The second shortcoming is a lack of flexibility. Ideally we would want a scheme which gives separate latent variable representations for separate components of the history and then uses these components together to predict the modifier's lexical head.

5 A Distributed Latent Variable Model

5.1 Model Description and GEM Algorithm

The distributed model is designed to surmount the shortcomings of AMMs. It provides a flexible way of smoothing $P(w|\mathcal{H})$ by first giving a probabilistic encoding of $\mathcal{H}$ as a vector $\mathbf{b} = b_1 \cdots b_m$ of m latent bits. Different components of a complex history can be tied to different bits. The bits are Bernoulli random variables which are conditionally independent given $\mathcal{H}$. This representation gives a number of parameters linear in m, which is logarithmic in the number of bit vector configurations. Then, to avoid exponential blow-up when predicting w, we parameterize the conditional distribution over w using a maximum entropy model with features on bit-word pairs. The distribution induced by this model is specified as follows:

$$
\begin{aligned}
P(w|\mathcal{H}) &= \sum_{\mathbf{b}} p(\mathbf{b}|\mathcal{H}) \cdot q(w|\mathbf{b}) \\
&= \sum_{\mathbf{b}} \Bigg[\left(\prod_{j=1}^{m} p_j^{b_j} (1-p_j)^{(1-b_j)} \right) \cdot \\
&\qquad \frac{1}{Z(\mathbf{b})} \exp \sum_f \lambda_f f(\mathbf{b}, w) \Bigg] \qquad (2)
\end{aligned}
$$

Here b_j refers to the jth bit, p_j is the parameter of the conditional Bernoulli distribution $p(b_j = 1|\mathcal{H})$, the f are binary feature functions to be specified, and the λ_f are the corresponding parameters. The feature functions we currently use are

$$
\begin{aligned}
g_v(\mathbf{b}, w) &= [\![v = w]\!] \\
h_{j,v}(\mathbf{b}, w) &= [\![b_j = 1]\!][\![v = w]\!]
\end{aligned}
$$

The model can also be viewed graphically in Figure 3, where the conditional probability of w given $\mathbf{b}$ is parameterized as in (2).

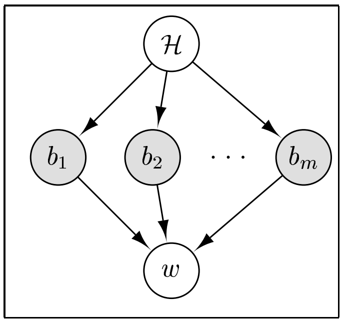

Figure 3: Graphical Representation for the Distributed Model

The parameters of the distributed model are the Bernoulli parameters p_j and the weights $\lambda_{j,v}$ of the maximum entropy model. As with the aggregate model, we can give an EM algorithm to estimate them. The E-step is derived in a similar manner to the E-step of an AMM, and again the normalization requires a summation over all bit configurations. The M-step for the conditional Bernoulli parameters p_j is as follows:

$$p_j^{(t+1)}(b_j|\mathcal{H}) = \frac{\sum_w c(\mathcal{H},w)\sum_{\mathbf{b}:b_j=1}\rho^{(t)}(\mathbf{b}|\mathcal{H},w)}{c(\mathcal{H})}$$

Again, ρ is the posterior probability over a single bit vector configuration and $c(\mathcal{H})$ refers to the empirical count of the history. The M-step for the maximum entropy parameters is not closed-form, but we give a GEM algorithm in which we increase the expected complete data log-likelihood function by fixing the posterior and using generalized iterative scaling (GIS). The GIS update for the parameter λ_f is:

$$\begin{aligned}\lambda_f^{(t+1)} \quad &= \quad \lambda_f^{(t)} + \frac{1}{\#}\Bigg\{ \log\left(\sum_{\mathcal{H},w} c(\mathcal{H},w)\sum_{\mathbf{b}}\rho(\mathbf{b}|\mathcal{H},w)f(\mathbf{b},w)\right) - \\ &\qquad \log\left(\sum_{\mathbf{b}}\Big[\sum_{\mathcal{H},w} c(\mathcal{H},w)\rho(\mathbf{b}|\mathcal{H},w)\Big]\Big[\sum_{v} q(v|\mathbf{b},\Lambda^{(t)})f(\mathbf{b},v)\Big]\right)\Bigg\}\end{aligned}$$

As in standard GIS, $\#$ is a correction constant representing the maximum feature count for any instance. Here, however, the "empirical" feature counts are the expected counts with respect to the posterior. Similarly, the "training instance counts" in the model-expected feature counts term are the marginalized counts for each bit vector, again using the posterior.

In practice, GIS converges slowly, especially in the case where the number of features grows large. Because of this, we sped up the GEM algorithm by taking only a few GIS steps per EM step and using an adaptive, overrelaxed update [11]. For large models, this modified update gives convergence in under 50 iterations, whereas the standard GEM update does not. As with the aggregate model, we can temper the GEM algorithm for the distributed model to avoid overfitting.

5.2 Lexical Head Bigram Results

Model	**Perplexity**
Collins	581
Linear Interpolation	573
AMM-Tempered	510
Distributed-7-Tempered	514
Distributed-9-Tempered	504
Distributed-11-Tempered	**496**

Table 3: Distributed Model results on lexical bigram data

Table 3 presents results on lexical bigrams. The data are identical to those of section 4. Distributed-n refers to an n-bit model (with 2^n possible bit configurations). The lexical bigram results show a 3% improvement over the best AMM and a 15% improvement over the Collins model.

6 Conclusions and Further Work

We presented two latent variable models for alleviating data sparseness in syntactic cooccurrences. We extended the AMM of Saul and Pereira to complex histories and introduced a distributed model designed to alleviate the scaling and flexibilty problems inherent in AMMs. Both of these models are probabilistic latent variable models. They offer an advantage over neural networks in that they can be trained with EM, which has no inherent free parameters. Furthermore, using EM has allowed us to take advantage of recent work in regularization [8] and in speeding up the convergence of bound optimizers [11].

We gave results showing an improvement in perplexity over the model of Collins. We recognize that the ultimate objective of these models is to improve parser performance and that this objective is not always directly related to perplexity. One important near-term goal is to run experiments which verify that our models will improve parser performance. Another related goal is to improve the speed of the distributed model. While it does offer a solution to the parameter scaling problem inherent in the aggregate model, exact inference in the distributed model still requires time linear in the number of bit configurations. When modelling the full history used by Collins model 1, this inference becomes prohibitively expensive for large numbers of bits. We do believe, however, that an efficient approximation scheme exists for this model, and we are currently investigating Gibbs sampling and variational inference techniques.

References

[1] BENGIO, Y., DUCHARME, R., VINCENT, P., AND JANVIN, C. A neural probabilistic language model. *Journal of Machine Learning Research 3* (2003), 1137–1155.

[2] BIKEL, D. Intricacies of collins' parsing model. Tech. Rep. MS-CIS-03-11, 2003.

[3] CHARNIAK, E. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI* (1997), pp. 598–603.

[4] CHELBA, C., AND JELINEK, F. Recognition performance of a structured language model. In *Proceedings of Eurospeech* (Budapest, Hungary, 1999).

[5] COLLINS, M. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[6] GILDEA, D. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2001).

[7] HENDERSON, J. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Meeting of the North American Association for Computational Linguistics and the Human Language Technologies Conference* (2003).

[8] HOFMANN, T., AND PUZICHA, J. Statistical models for co-occurrence data. Tech. Rep. AIM-1625, 1998.

[9] MAGERMAN, D. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.

[10] NEAL, R. Connectionist learning of belief networks. *Artificial Intelligence 56* (1992), 71–113.

[11] SALAKHUTDINOV, R., AND ROWEIS, S. Adaptive overrelaxed bound optimization methods, 2003.

[12] SAUL, L., AND PEREIRA, F. Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. 1997.