Joint and conditional estimation of tagging and parsing models[*]

Mark Johnson
Brown University
`Mark_Johnson@Brown.edu`

Abstract

This paper compares two different ways of estimating statistical language models. Many statistical NLP tagging and parsing models are estimated by maximizing the (joint) likelihood of the fully-observed training data. However, since these applications only require the conditional probability distributions, these distributions can in principle be learnt by maximizing the conditional likelihood of the training data. Perhaps somewhat surprisingly, models estimated by maximizing the joint were superior to models estimated by maximizing the conditional, even though some of the latter models intuitively had access to "more information".

1 Introduction

Many statistical NLP applications, such as tagging and parsing, involve finding the value of some hidden variable Y (e.g., a tag or a parse tree) which maximizes a conditional probability distribution $\mathrm{P}_\theta(Y|X)$, where X is a given word string. The model parameters θ are typically estimated by maximum likelihood: i.e., maximizing the likelihood of the training data. Given a (fully observed) training corpus $D = ((y_1, x_1), \ldots, (y_n, x_n))$, the *maximum (joint) likelihood estimate* (MLE) of θ is:

$$\hat{\theta} \quad = \quad \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} \mathrm{P}_\theta(y_i, x_i). \tag{1}$$

However, it turns out there is another maximum likelihood estimation method which maximizes the conditional likelihood or "pseudo-likelihood" of the training data (Besag, 1975). Maximum conditional likelihood is consistent *for the conditional distribution*. Given a training corpus D, the *maximum conditional likelihood estimate* (MCLE) of the model parameters θ is:

$$\hat{\theta} \quad = \quad \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} \mathrm{P}_\theta(y_i|x_i). \tag{2}$$

Figure 1 graphically depicts the difference between the MLE and MCLE. Let Ω be the universe of all possible pairs (y, x) of hidden and visible values. Informally, the MLE selects the model parameter θ which make the training data pairs (y_i, x_i) as likely as possible relative to all other pairs (y', x') in Ω. The MCLE, on the other hand, selects the model parameter θ in order to make the training data pair (y_i, x_i) more likely than other pairs (y', x_i) in Ω, i.e., pairs with the same visible value x_i as the training datum.

In statistical computational linguistics, maximum conditional likelihood estimators have mostly been used with general exponential or "maximum entropy" models because standard maximum likelihood estimation is usually computationally intractable (Berger et al., 1996; Della Pietra et al., 1997; Jelinek, 1997). Well-known computational linguistic models such as

[*]I would like to thank Eugene Charniak and the other members of BLLIP for their comments and suggestions. Fernando Pereira was especially generous with comments and suggestions, as were the ACL reviewers; I apologize for not being able to follow up all of your good suggestions. This research was supported by NSF awards 9720368 and 9721276 and NIH award R01 MH60922-01A2.

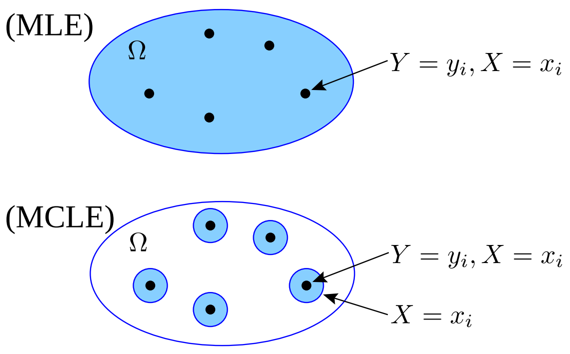

Figure 1: The MLE makes the training data (y_i, x_i) as likely as possible (relative to Ω), while the MCLE makes (y_i, x_i) as likely as possible relative to other pairs (y', x_i).

Maximum-Entropy Markov Models (McCallum et al., 2000) and Stochastic Unification-based Grammars (Johnson et al., 1999) are standardly estimated with conditional estimators, and it would be interesting to know whether conditional estimation affects the quality of the estimated model. It should be noted that in practice, the MCLE of a model with a large number of features with complex dependencies may yield far better performance than the MLE of the much smaller model that could be estimated with the same computational effort. Nevertheless, as this paper shows, conditional estimators can be used with other kinds of models besides MaxEnt models, and in any event it is interesting to ask whether the MLE differs from the MCLE in actual applications, and if so, how.

Because the MLE is consistent for the joint distribution $\mathrm{P}(Y, X)$ (e.g., in a tagging application, the distribution of word-tag sequences), it is also consistent for the conditional distribution $\mathrm{P}(Y|X)$ (e.g., the distribution of tag sequences given word sequences) and the marginal distribution $\mathrm{P}(X)$ (e.g., the distribution of word strings). On the other hand, the MCLE is consistent for the conditional distribution $\mathrm{P}(Y|X)$ alone, and provides no information about either the joint or the marginal distributions. Applications such as language modelling for speech recognition and EM procedures for estimating from hidden data either explicitly or implicitly require marginal distributions over the visible data (i.e., word strings), so it is not statistically sound to use MCLEs for such applications. On the other hand, applications which involve predicting the value of the hidden variable from the visible variable (such as tagging or parsing) usually only involve the conditional distribution, which the MCLE estimates directly.

Since both the MLE and MCLE are consistent for the conditional distribution, both converge *in the limit* to the "true" distribution *if the true distribution is in the model class*. However, given that we often have insufficient data in computational linguistics, and there are good reasons to believe that the true distribution of sentences or parses cannot be described by our models, there is no reason to expect these asymptotic results to hold in practice, and in the experiments reported below the MLE and MCLE behave differently experimentally.

A priori, one can advance plausible arguments in favour of both the MLE and the MCLE. Informally, the MLE and the MCLE differ in the following way. Since the MLE is obtained by maximizing $\prod_i \mathrm{P}_\theta(y_i|x_i)\mathrm{P}_\theta(x_i)$, the MLE exploits information about the distribution of word strings x_i in the training data that the MCLE does not. Thus one might expect the MLE to converge faster than the MCLE in situations where training data is not over-abundant, which is often the case in computational linguistics.

On the other hand, since the intended application requires a conditional distribution, it seems reasonable to directly estimate this conditional distribution from the training data as the MCLE does. Furthermore, suppose that the model class is wrong (as is surely true of all our current language models), i.e., the "true" model $\mathrm{P}(Y, X) \neq \mathrm{P}_\theta(Y, X)$ for all θ, and that our best models are particularly poor approximations to the true distribution of word strings $\mathrm{P}(X)$. Then ignoring the distribution of word strings in the training data as the MCLE does might indeed be a reasonable thing to do.

The rest of this paper is structured as follows. The next section formulates the MCLEs for HMMs and PCFGs as constrained optimization problems and describes an iterative dynamic-programming method for solving them. Because of the computational complexity of these problems, the method is only applied to a simple PCFG based on the ATIS corpus. For this example, the MCLE PCFG does perhaps produce slightly better parsing results than the standard MLE (relative-frequency) PCFG, although the result does not reach statistical significance.

It seems to be difficult to find model classes for which the MLE and MCLE are both easy to compute. However, often it is possible to find two closely related model classes, one of which has an easily computed MLE and the other which has an easily computed MCLE. Typically, the model classes which have an easily computed MLE define *joint* probability distributions over both the hidden and the visible data (e.g., over word-tag pair sequences for tagging), while the model classes which have an easily computed MCLE define *conditional* probability distributions over the hidden data given the visible data (e.g., over tag sequences given word sequences).

Section 3 investigates closely related joint and conditional tagging models (the latter can be regarded as a simplification of the Maximum Entropy Markov Models of McCallum et al. (2000)), and shows that MLEs outperform the MCLEs in this application. The final empirical section investigates two different kinds of stochastic shift-reduce parsers, and shows that the model estimated by the MLE outperforms the model estimated by the MCLE.

2 PCFG parsing

In this application, the pairs (y, x) consist of a parse tree y and its terminal string or yield x (it may be simpler to think of y containing all of the parse tree except for the string x). Recall that in a PCFG with production set R, each production $(A \to \alpha) \in R$ is associated with a parameter $\theta_{A\to\alpha}$. These parameters satisfy a normalization constraint for each nonterminal A:

$$\sum_{\alpha:(A\to\alpha)\in R} \theta_{A\to\alpha} \quad = \quad 1 \tag{3}$$

For each production $r \in R$, let $f_r(y)$ be the number of times r is used in the derivation of the tree y. Then the PCFG defines a probability distribution over trees:

$$\mathrm{P}_\theta(Y) \quad = \quad \prod_{(A\to\alpha)\in R} {\theta_{A\to\alpha}}^{f_{A\to\alpha}(Y)}$$

The MLE for θ is the well-known "relative-frequency" estimator:

$$\hat\theta_{A\to\alpha} \quad = \quad \frac{\sum_{i=1}^n f_{A\to\alpha}(y_i)}{\sum_{i=1}^n \sum_{\alpha':(A\to\alpha')\in R} f_{A\to\alpha'}(y_i)}.$$

Unfortunately the MCLE for a PCFG is more complicated. If x is a word string, then let $\tau(x)$ be the set of parse trees with terminal string or yield x generated by the PCFG. Then given a training corpus $D = ((y_1, x_1), \ldots, (y_n, x_n))$, where y_i is a parse tree for the string x_i, the log conditional likelihood of the training data $\log \mathrm{P}(\vec{y}|\vec{x})$ and its derivative are given by:

$$\begin{aligned}\log \mathrm{P}(\vec{y}|\vec{x}) \quad &= \quad \sum_{i=1}^n \left(\log \mathrm{P}_\theta(y_i) - \log \sum_{y\in\tau(x_i)} \mathrm{P}_\theta(y) \right) \\ \frac{\partial \log \mathrm{P}(\vec{y}|\vec{x})}{\partial \theta_{A\to\alpha}} \quad &= \quad \frac{1}{\theta_{A\to\alpha}} \sum_{i=1}^n (f_{A\to\alpha}(y_i) - \mathrm{E}_\theta(f_{A\to\alpha}|x_i))\end{aligned}$$

Here $\mathrm{E}_\theta(f|x)$ denotes the expectation of f with respect to P_θ conditioned on $Y \in \tau(x)$. There does not seem to be a closed-form solution for the θ that maximizes $\mathrm{P}(\vec{y}|\vec{x})$ subject to the constraints (3), so we used an iterative numerical gradient ascent method, with the constraints (3) imposed at each iteration using Lagrange multipliers. Note that $\sum_{i=1}^n \mathrm{E}_\theta(f_{A\to\alpha}|x_i)$ is a quantity calculated in the Inside-Outside algorithm (Lari and Young, 1990) and $\mathrm{P}(\vec{y}|\vec{x})$ is easily computed as a by-product of the same dynamic programming calculation.

Since the expected production counts $\mathrm{E}_\theta(f|x)$ depend on the production weights θ, the entire training corpus must be reparsed on each iteration (as is true of the Inside-Outside algorithm). This is computationally expensive with a large grammar and training corpus; for this reason the MCLE PCFG experiments described here were performed with the relatively small ATIS treebank corpus of air travel reservations distributed by LDC.

In this experiment, the PCFGs were always trained on the 1088 sentences of the ATIS1 corpus and evaluated on the 294 sentences of the ATIS2 corpus. Lexical items were ignored; the PCFGs generate preterminal strings. The iterative algorithm for the MCLE was initialized with the MLE parameters, i.e., the "standard" PCFG estimated from a treebank. Table 1 compares the MLE and MCLE PCFGs.

The data in table 1 shows that compared to the MLE PCFG, the MCLE PCFG assigns a higher conditional probability of the parses in the training data given their yields, at the expense of assigning a lower marginal probability to the yields themselves. The labelled precision and recall parsing results for the MCLE PCFG were slightly higher than those of the MLE PCFG. Because

	MLE	MCLE
$-\log \mathrm{P}(\vec{y})$	13857	13896
$-\log \mathrm{P}(\vec{y}\|\vec{x})$	1833	1769
$-\log \mathrm{P}(\vec{x})$	12025	12127
Labelled precision	0.815	0.817
Labelled recall	0.789	0.794

Table 1: The likelihood $\mathrm{P}(\vec{y})$ and conditional likelihood $\mathrm{P}(\vec{y}|\vec{x})$ of the ATIS1 training trees, and the marginal likelihood $\mathrm{P}(\vec{x})$ of the ATIS1 training strings, as well as the labelled precision and recall of the ATIS2 test trees, using the MLE and MCLE PCFGs.

both the test data set and the differences are so small, the significance of these results was estimated using a bootstrap method with the difference in F-score in precision and recall as the test statistic (Cohen, 1995). This test showed that the difference was not significant ($p \approx 0.1$). Thus the MCLE PCFG did not perform significantly better than the MLE PCFG in terms of precision and recall.

3 HMM tagging

As noted in the previous section, maximizing the conditional likelihood of a PCFG or a HMM can be computationally intensive. This section and the next pursues an alternative strategy for comparing MLEs and MCLEs: we compare similiar (but not identical) model classes, one of which has an easily computed MLE, and the other of which has an easily computed MCLE. The application considered in this section is bitag POS tagging, but the techniques extend straight-forwardly to n-tag tagging. In this application, the data pairs (y, x) consist of a tag sequence $y = t_1 \ldots t_m$ and a word sequence $x = w_1 \ldots w_m$, where t_j is the tag for word w_j (to simplify the formulae, w_0, t_0, w_{m+1} and t_{m+1} are always taken to be end-markers). Standard HMM tagging models define a *joint* distribution over word-tag sequence pairs; these are most straight-forwardly estimated by maximizing the likelihood of the joint training distribution. However, it is straight-forward to devise closely related HMM tagging models which define a *conditional* distribution over tag sequences given word sequences, and which are most straight-forwardly estimated by maximizing the conditional likelihood of the distribution of tag sequences given word sequences in the training data.

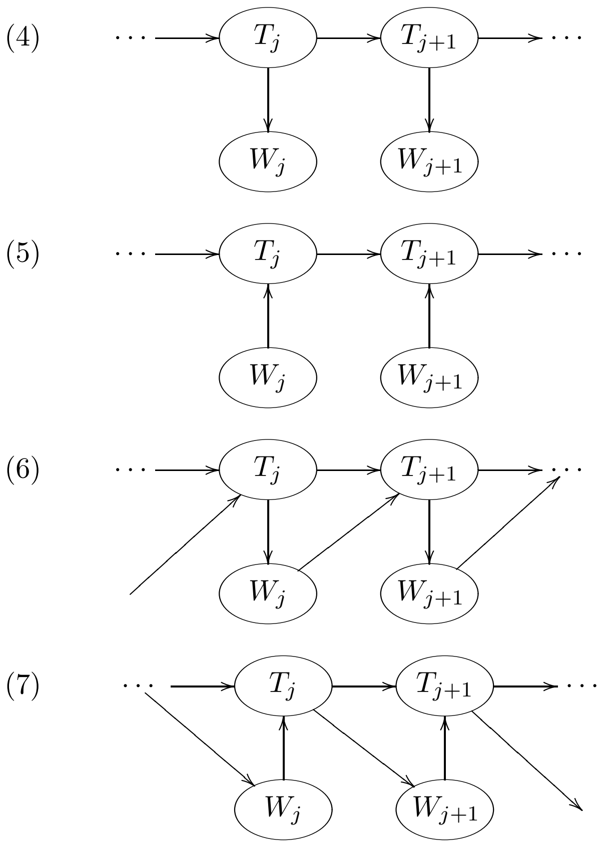

Figure 2: The HMMs depicted as "Bayes net" graphical models.

All of the HMM models investigated in this section are instances of a certain kind of graphical model that Pearl (1988) calls "Bayes nets"; Figure 2 sketches the networks that correspond to all of the models discussed here. (In such a graph, the set of incoming arcs to a node depicting a variable indicate the set of variables on which this variable is conditioned).

Recall the standard bitag HMM model, which defines a joint distribution over word and tag sequences:

$$\mathrm{P}(Y, X) = \prod_{j=1}^{m+1} \hat{\mathrm{P}}(T_j|T_{j-1})\hat{\mathrm{P}}(W_j|T_j) \quad (4)$$

As is well-known, the MLE for (4) sets $\hat{\mathrm{P}}$ to the empirical distributions on the training data.

Now consider the following *conditional model* of the conditional distribution of tags given words (this is a simplified form of the model described in McCallum et al. (2000)):

$$\mathrm{P}(Y|X) = \prod_{j=1}^{m+1} \mathrm{P}_0(T_j|W_j, T_{j-1}) \quad (5)$$

The MCLE of (5) is easily calculated: P_0 should be set the empirical distribution of the training data. However, to minimize sparse data problems we estimated $\mathrm{P}_0(T_j|W_j, T_{j-1})$ as a mixture of $\hat{\mathrm{P}}(T_j|W_j)$, $\hat{\mathrm{P}}(T_j|T_{j-1})$ and $\hat{\mathrm{P}}(T_j|W_j, T_{j-1})$, where the $\hat{\mathrm{P}}$ are empirical probabilities and the (bucketted) mixing parameters are determined using deleted interpolation from heldout data (Jelinek, 1997).

These models were trained on sections 2-21 of the Penn tree-bank corpus. Section 22 was used as heldout data to evaluate the interpolation parameters λ. The tagging accuracy of the models was evaluated on section 23 of the tree-bank corpus (in both cases, the tag t_j assigned to word w_j is the one which maximizes the marginal $\mathrm{P}(t_j|w_1 \ldots w_m)$, since this minimizes the expected loss on a tag-by-tag basis).

The conditional model (5) has the worst performance of any of the tagging models investigated in this section: its tagging accuracy is 94.4%. The joint model (4) has a considerably lower error rate: its tagging accuracy is 95.5%.

One possible explanation for this result is that the way in which the interpolated estimate of P_0 is calculated, rather than conditional likelihood estimation per se, is lowering tagger accuracy somehow. To investigate this possibility, two additional joint models were estimated and tested, based on the formulae below.

$$\mathrm{P}(Y, X) = \prod_{j=1}^{m+1} \hat{\mathrm{P}}(W_j|T_j)\mathrm{P}_1(T_j|W_{j-1}, T_{j-1}) \quad (6)$$

$$\mathrm{P}(Y, X) = \prod_{j=1}^{m+1} \mathrm{P}_0(T_j|W_j, T_{j-1})\hat{\mathrm{P}}(W_j|T_{j-1}) \quad (7)$$

The MLEs for both (6) and (7) are easy to calculate. (6) contains a conditional distribution P_1 which would seem to be of roughly equal complexity to P_0, and it was estimated using deleted interpolation in exactly the same way as P_0, so if the poor performance of the conditional model was due to some artifact of the interpolation procedure, we would expect the model based on (6) to perform poorly. Yet the tagger based on (6) performs the best of all the taggers investigated in this section: its tagging accuracy is 96.2%.

(7) is admitted a rather strange model, since the right hand term in effect predicts the *following* word from the current word's tag. However, note that (7) differs from (5) only via the presence of this rather unusual term, which effectively converts (5) from a conditional model to a joint model. Yet adding this term improves tagging accuracy considerably, to 95.3%. Thus for bitag tagging at least, the conditional model has a considerably higher error rate than any of the joint models examined here. (While a test of significance was not conducted here, previous experience with this test set shows that performance differences of this magnitude are extremely significant statistically).

4 Shift-reduce parsing

The previous section compared similiar joint and conditional tagging models. This section compares a pair of joint and conditional parsing models. The models are both stochastic shift-reduce parsers; they differ only in how the distribution over possible next moves are calculated. These parsers are direct simplifications of the Structured Language Model (Jelinek, 2000). Because the parsers' moves are determined solely by the top two category labels on the stack and possibly the look-ahead symbol, they are much simpler than stochastic LR parsers (Briscoe and Carroll, 1993; Inui et al., 1997). The distribution over trees generated by the joint model is a probabilistic context-free language (Abney et al., 1999). As with the PCFG models discussed earlier, these parsers are not lexicalized; lexical items are ignored, and the POS tags are used as the terminals.

These two parsers only produce trees with unary or binary nodes, so we binarized the training data before training the parser, and debinarize the trees the parsers produce before evaluating them with respect to the test data (Johnson, 1998). We binarized by inserting $n-2$ additional nodes into each local tree with $n > 2$ children. We binarized by first joining the head to all of the constituents to its right, and then joining the resulting structure with constituents to the left. The label of a new node is the label of the head followed by the suffix "-1" if the head is (contained in) the right child or "-2" if the head is (contained in) the left child. Figure 3 depicts an example of this transformation.

The Structured Language Model is described in detail in Jelinek (2000), so it is only reviewed here. Each parser's stack is a sequence of node

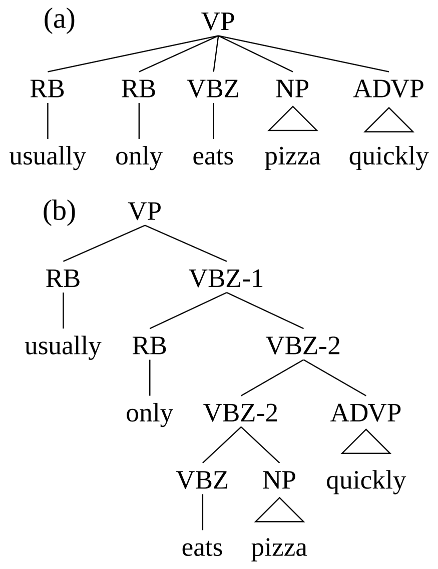

Figure 3: The binarization transformation used in the shift-reduce parser experiments transforms tree (a) into tree (b).

labels (possibly including labels introduced by binarization). In what follows, s_1 refers to the top element of the stack, or '$\star$' if the stack is empty; similarly s_2 refers to the next-to-top element of the stack or '$\star$' if the stack contains less than two elements. We also append a '$\star$' to end of the actual terminal string being parsed (just as with the HMMs above), as this simplifies the formulation of the parsers, i.e., if the string to be parsed is $w_1 \ldots w_m$, then we take $w_{m+1} = \star$.

A shift-reduce parse is defined in terms of moves. A move is either $\mathrm{shift}(w)$, $\mathrm{reduce}_1(c)$ or $\mathrm{reduce}_2(c)$, where c is a nonterminal label and w is either a terminal label or '$\star$'. Moves are partial functions from stacks to stacks: a $\mathrm{shift}(w)$ move pushes a w onto the top of stack, while a $\mathrm{reduce}_i(c)$ move pops the top i terminal or nonterminal labels off the stack and pushes a c onto the stack. A shift-reduce parse is a sequence of moves which (when composed) map the empty stack to the two-element stack whose top element is '$\star$' and whose next-to-top element is the start symbol. (Note that the last move in a shift-reduce parse must always be a $\mathrm{shift}(\star)$ move; this corresponds to the final "accept" move in an LR parser). The isomorphism between shift-reduce parses and standard parse trees is well-known (Hopcroft and Ullman, 1979), and so is not described here.

A (joint) shift-reduce parser is defined by a distribution $\mathrm{P}(m|s_1, s_2)$ over next moves m given the top and next-to-top stack labels s_1 and s_2. To ensure that the next move is in fact a possible move given the current stack, we require that $\mathrm{P}(\mathrm{reduce}_1(c)|\star,\star) = 0$ and $\mathrm{P}(\mathrm{reduce}_2(c)|c',\star) = 0$ for all c, c', and that $\mathrm{P}(shift(\star)|s_1, s_2) = 0$ unless s_1 is the start symbol and $s_2 = \star$. Note that this extends to a probability distribution over shift-reduce parses (and hence parse trees) in a particularly simple way: the probability of a parse is the product of the probabilities of the moves it consists of. Assuming that P meets certain tightness conditions, this distribution over parses is properly normalized because there are no "dead" stack configurations: we require that the distribution over moves be defined for all possible stacks.

A conditional shift-reduce parser differs only minimally from the shift-reduce parser just described: it is defined by a distribution $\mathrm{P}(m|s_1, s_2, t)$ over next moves m given the top and next-to-top stack labels s_1, s_2 and the next input symbol w (w is called the *look-ahead symbol*). In addition to the requirements on P above, we also require that if $w' \neq w$ then $\mathrm{P}(\mathrm{shift}(w')|s_1, s_2, w) = 0$ for all s_1, s_2; i.e., shift moves can only shift the current look-ahead symbol. This restriction implies that all non-zero probability derivations are derivations of the parse string, since the parse string forces a single sequence of symbols to be shifted in all derivations. As before, since there are no "dead" stack configurations, so long as P obeys certain tightness conditions, this defines a properly normalized distribution over parses. Since all the parses are required to be parses of of the input string, this defines a conditional distribution over parses given the input string.

It is easy to show that the MLE for the joint model, and the MCLE for the conditional model, are just the empirical distributions from the training data. We ran into sparse data problems using the empirical training distribution as an estimate for $\mathrm{P}(m|s_1, s_2, w)$ in the conditional model, so in fact we used deleted interpolation to interpolate $\hat{\mathrm{P}}(m|s_1, s_2, w)$, and $\hat{\mathrm{P}}(m|s_1, s_2)$ to estimate $\mathrm{P}(m|s_1, s_2, w)$. The models were estimated from sections 2–21 of the Penn treebank, and tested on the 2245 sentences of length 40 or less in section 23. The deleted interpolation parameters were estimated using heldout training data from section

	Joint SR	Conditional SR	PCFG
Precision	0.666	0.633	0.700
Recall	0.650	0.639	0.657

Table 2: Labelled precision and recall results for joint and conditional shift-reduce parsers, and for a PCFG.

22.

We calculated the most probable parses using a dynamic programming algorithm based on the one described in Jelinek (2000). Jelinek notes that this algorithm's running time is n^6 (where n is the length of sentence being parsed), and we found exhaustive parsing to be computationally impractical. We used a beam search procedure which thresholded the best analyses of each prefix of the string being parsed, and only considered analyses whose top two stack symbols had been observed in the training data. In order to help guard against the possibility that this stochastic pruning influenced the results, we ran the parsers twice, once with a beam threshold of 10^{-6} (i.e., edges whose probability was less than 10^{-6} of the best edge spanning the same prefix were pruned) and again with a beam threshold of 10^{-9}. The results of the latter runs are reported in table 2; the labelled precision and recall results from the run with the more restrictive beam threshold differ by less than 0.001, i.e., at the level of precision reported here, are identical with the results presented in table 2 except for the Precision of the Joint SR parser, which was 0.665. For comparision, table 2 also reports results from the non-lexicalized treebank PCFG estimated from the transformed trees in sections 2-21 of the treebank; here exhaustive CKY parsing was used to find the most probable parses.

All of the precision and recall results, including those for the PCFG, presented in table 2 are much lower than those from a standard treebank PCFG; presumably this is because the binarization transformation depicted in Figure 3 loses information about pairs of non-head constituents in the same local tree (Johnson (1998) reports similiar performance degradation for other binarization transformations). Both the joint and the conditional shift-reduce parsers performed much worse than the PCFG. This may be due to the pruning effect of the beam search, although this seems unlikely given that varying the beam threshold did not affect the results. The performance difference between the joint and conditional shift-reduce parsers bears directly on the issue addressed by this paper: the joint shift-reduce parser performed much better than the conditional shift-reduce parser. The differences are around a percentage point, which is quite large in parsing research (and certainly highly significant).

The fact that the joint shift-reduce parser outperforms the conditional shift-reduce parser is somewhat surprising. Because the conditional parser predicts its next move on the basis of the lookahead symbol as well as the two top stack categories, one might expect it to predict this next move more accurately than the joint shift-reduce parser. The results presented here show that this is not the case, at least for non-lexicalized parsing. The *label bias* of conditional models may be responsible for this (Bottou, 1991; Lafferty et al., 2001).

5 Conclusion

This paper has investigated the difference between maximum likelihood estimation and maximum conditional likelihood estimation for three different kinds of models: PCFG parsers, HMM taggers and shift-reduce parsers. The results for the PCFG parsers suggested that conditional estimation might provide a slight performance improvement, although the results were not statistically significant since computational difficulty of conditional estimation of a PCFG made it necessary to perform the experiment on a tiny training and test corpus. In order to avoid the computational difficulty of conditional estimation, we compared closely related (but not identical) HMM tagging and shift-reduce parsing models, for some of which the maximum likelihood estimates were easy to compute and for others of which the maximum conditional likelihood estimates could be easily computed. In both cases, the joint models outperformed the conditional models by quite large amounts. This suggests that it may be worthwhile investigating methods for maximum (joint) likelihood estimation for model classes for which only maximum conditional likelihood estimators are currently used, such as Maximum Entropy models and MEMMs, since if the results of the experiments presented in this paper extend to these models, one might expect a modest performance improvement.

As explained in the introduction, because maximum likelihood estimation exploits not just the conditional distribution of hidden variable (e.g., the tags or the parse) conditioned on the visible variable (the terminal string) but also the marginal distribution of the visible variable, it is reasonable to expect that it should outperform maximum conditional likelihood estimation. Yet it is counter-intuitive that joint tagging and shift-reduce parsing models, which predict the next tag or parsing move on the basis of what seems to be less information than the corresponding conditional model, should nevertheless outperform that conditional model, as the experimental results presented here show. The recent theoretical and simulation results of Lafferty et al. (2001) suggest that conditional models may suffer from *label bias* (the discovery of which Lafferty et. al. attribute to Bottou (1991)), which may provide an insightful explanation of these results.

None of the models investigated here are state-of-the-art; the goal here is to compare two different estimation procedures, and for that reason this paper concentrated on simple, easily implemented models. However, it would also be interesting to compare the performance of joint and conditional estimators on more sophisticated models.

References

Steven Abney, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 542–549, San Francisco. Morgan Kaufmann.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

J. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.

Léon Bottou. 1991. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. Ph.D. thesis, Université de Paris XI.

Ted Briscoe and John Carroll. 1993. Generalized probabilistic LR parsing of natural language (corpora) with unification-based methods. *Computational Linguistics*, 19:25–59.

Paul R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

K. Inui, V. Sornlertlamvanich, H. Tanaka, and T. Tokunaga. 1997. A new formalization of probabilistic GLR parsing. In *Proceedings of the Fifth International Workshop on Parsing Technologies (IWPT-97)*, pages 123–134, MIT.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Frederick Jelinek. 2000. Stochastic analysis of structured language modeling. Technical report, Center for Language and Speech Modeling, Johns Hopkins University.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *The Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, pages 535–541, San Francisco. Morgan Kaufmann.

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Machine Learning: Proceedings of the Eighteenth International Conference (ICML 2001)*.

K. Lari and S.J. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum Entropy Markov Models for information extraction and segmentation. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, pages 591–598, Stanford, California.

Judea Pearl. 1988. *Probabalistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.