A New Formalization of Probabilistic GLR Parsing

Kentaro Inui, Virach Sornlertlamvanich, Hozumi Tanaka, and Takenobu Tokunaga
Graduate School of Information Science and Engineering
Tokyo Institute of Technology
2-12-1 O-okayama Meguro Tokyo 152, Japan
{inui,virach,take,tanaka}@cs.titech.ac.jp

Abstract

This paper presents a new formalization of probabilistic GLR language modeling for statistical parsing. Our model inherits its essential features from Briscoe and Carroll's generalized probabilistic LR model [3], which obtains context-sensitivity by assigning a probability to each LR parsing action according to its left and right context. Briscoe and Carroll's model, however, has a drawback in that it is not formalized in any probabilistically well-founded way, which may degrade its parsing performance. Our formulation overcomes this drawback with a few significant refinements, while maintaining all the advantages of Briscoe and Carroll's modeling.

1 Introduction

The increasing availability of text corpora has encouraged researchers to explore statistical approaches for various tasks in natural language processing. Statistical parsing is one of these approaches. In statistical parsing, one of the most straightforward methodologies is to generalize context-free grammars by associating a probability with each rule in producing probabilistic context-free grammars (PCFGs). However, as many researchers have already pointed out, PCFGs are not quite adequate for statistical parsing due to the lack of context-sensitivity. Probabilistic GLR parsing is one existing statistical parsing methodology which is more context-sensitive than PCFG-based parsing.

Several attempts have been made to incorporate probability into generalized LR (GLR) parsing [17]. For example, Wright and Wrigley proposed an algorithm to distribute probabilities originally associated with CFG rules to LR parsing actions, in such a way that the resulting model is equivalent to the original PCFG [19]. Perhaps, the most naive way of coupling a PCFG model with the GLR parsing framework would be to assign the probability associated with each CFG rule to the reduce actions for that rule. Wright and Wrigley expanded on this general methodology by distributing probabilities to shift actions as well as reduce actions, so that the parser can prune improbable parse derivations after shift actions as well as reduce actions. This can be advantageous particularly when one considers applying a GLR parser to, for example, continuous speech recognition. However, since their principal concern was in compiling PCFGs into the GLR parsing framework, their language model still failed to capture the context-sensitivity of languages.

Su et al. proposed a way of introducing probabilistic distribution into the shift-reduce parsing framework [16]. Unlike Wright and Wrigley's work, the goal of this research was the construction of a mildly context-sensitive model[1] that is effective for statistical parsing. Their model distributes probabilities to stack transitions between two shift actions, and associates a probability with each parse derivation, given by the product of the probability of each change included in the derivation. Further, they also described an algorithm to handle this model within the GLR parsing framework, gaining parse efficiency. However, since their probabilistic model in itself is not intimately coupled with the GLR parsing algorithm, their model needs an additional complex algorithm for training.

On the other hand, Briscoe and Carroll proposed the distribution of probabilities directly to each action in an LR table to realize mildly context-sensitive parsing [3]. Their model overcomes the drawback of context-insensitivity of PCFGs by estimating the probability of each LR parsing action according to its left (i.e. LR parse state) and right context (i.e. next input symbol). The probability of each parse derivation is computed as the product of the probability assigned to each action included in the derivation. Unlike the approach of

[1] By "a mildly context-sensitive model", we mean a model that is moderately more context-sensitive than context-free models such as PCFGs, but not a *fully* context-sensitive one, which would be intractable in both training and parsing.

Su et al., this makes it easy to implement context-sensitive probabilistic parsing by slightly extending GLR parsers, and the probabilistic parameters can be easily trained simply by counting the frequency of application of each action in parsing the training sentences. Furthermore, their model is expected to be able to allow the parser to prune improbable parse derivations at an equivalently fine-grained level as that of Wright and Wrigley's statistical parser, since it assigns probabilities to both shift and reduce actions. However, in as far as we have tested the performance of Briscoe and Carroll's model (B&C model, hereafter) in our preliminary experiments, it seems that, in many cases, it does not significantly improve on the performance of the PCFG model, and furthermore, in the worst case, it can be even less effective than the PCFG model. According to our analysis, these seem to be result, principally, of the method used for normalizing probabilities in their model, which may not be probabilistically well-founded. In fact, Briscoe and Carroll have not explicitly presented any formalization of their model.

This line of reasoning led us to consider a new formalization of probabilistic GLR (PGLR) parsing. In this paper, we propose a newly formalized PGLR language model for statistical parsing, which has the following advantages:

- It provides probabilistically well-founded distributions.
- It realizes mildly context-sensitive statistical parsing.
- It can be trained simply by counting the frequency of each LR parsing action.
- It allows the parser to prune improbable parse derivations, even after shift actions.

Although the performance of our model should be evaluated through large-scaled experiments, we are achieving promising results in our preliminary experiments. In what follows, we first present our new formalization of PGLR parsing (section 2). We then review B&C model according to our formalization, demonstrating that B&C model may not be probabilistically well-founded through the use of simple examples (section 3). We finally discuss how our refinement is expected to influence parsing performance through a further example (section 4).

2 A PGLR Language Model

Suppose we have a CFG and its corresponding LR table. Let $\boldsymbol{V}_n$ and $\boldsymbol{V}_t$ be the nonterminal and terminal alphabets, respectively, of the CFG. Further, let $\boldsymbol{S}$ and $\boldsymbol{A}$ be the sets of LR parse states and parsing actions appearing in the LR table, respectively. For each state $s \in \boldsymbol{S}$, the LR table specifies a set $La(s) \subseteq \boldsymbol{V}_t$ of possible next input symbols. Further, for each coupling of a state s and input symbol $l \in La(s)$, the table specifies a set of possible parsing actions: $Act(s, l) \subseteq \boldsymbol{A}$. Each action $a \in \boldsymbol{A}$ is either a shift action or reduce action. Let $\boldsymbol{A}_s$ and $\boldsymbol{A}_r$ be the set of shift and reduce actions, respectively, such that $\boldsymbol{A} = \boldsymbol{A}_s \cup \boldsymbol{A}_r \cup \{accept\}$ (*accept* is a special action denoting the completion of parsing).

As with most statistical parsing frameworks, given an input sentence, we rank the parse tree candidates according to the probabilities of the parse derivations that generate those trees. In LR parsing, each parse derivation can be regarded as a sequence of transitions between LR parse stacks, which we describe in detail below. Thus, in the following, we use the terms parse tree, parse derivation, and stack transition sequence interchangeably.

Given an input word sequence $W = \{w_1, \ldots, w_n\}$, we estimate the distribution over the parse tree candidates T as follows:

$$P(T|W) = \alpha \cdot P(T) \cdot P(W|T) \tag{1}$$

The first scaling factor α is a constant that is independent of T, and thus does not need to be considered in ranking parse trees. The second factor $P(T)$ is the distribution over all the possible trees that can be derived from a given grammar, such that, for $\boldsymbol{\mathcal{T}}$ being the infinite set of all possible trees:

$$\sum_{T \in \boldsymbol{\mathcal{T}}} P(T) = 1 \tag{2}$$

We estimate this syntactic distribution $P(T)$ using a PGLR model. The third factor $P(W|T)$ is the distribution of lexical derivations from T, where each terminal symbol of T is assumed to be a part of speech symbol. Most statistical parsing frameworks estimate this distribution by assuming that the probability of the i-th word

$w_i \in W$ depends only on its corresponding terminal symbol (i.e. part of speech) l_i. Since l_i is uniquely specified by T for each i, we obtain equation (3):

$$P(W|T) \approx \prod_{w_i \in W} P(w_i|l_i) \tag{3}$$

One could use a more context-sensitive model to estimate the lexical distribution $P(W|T)$; for example, one could introduce the statistics of word collocations. However, this is beyond the scope of this paper. For further discussion, see [7].

A stack transition sequence T can be described as (4):

$$\sigma_0 \stackrel{l_1,a_1}{\Longrightarrow} \sigma_1 \stackrel{l_2,a_2}{\Longrightarrow} \ldots \stackrel{l_{n-1},a_{n-1}}{\Longrightarrow} \sigma_{n-1} \stackrel{l_n,a_n}{\Longrightarrow} \sigma_n \tag{4}$$

where σ_i is the i-th stack, whose stack-top state is denoted by $top(\sigma_i)$, and $l_i \in La(top(\sigma_{i-1}))$ and $a_i \in Act(top(\sigma_{i-1}), l_i)$ are, respectively, an input symbol and a parsing action chosen at σ_{i-1}. It can be proven from the LR parsing algorithm that, given an input symbol $l_{i+1} \in La(top(\sigma_i))$ and an action $a_{i+1} \in Act(top(\sigma_i), l_{i+1})$, the next (derived) stack $next(\sigma_i, a_{i+1})$ $(= \sigma_{i+1})$ can always be uniquely determined as follows:

- If the current action a_{i+1} is a shift action for an input symbol l_{i+1}, then the parser consumes l_{i+1}, pushing l_{i+1} onto the stack, and then pushes the next state s_{i+1}, which is uniquely specified by the LR table, onto the stack.
- If the current action a_{i+1} is a reduction by a rule $A \rightarrow \beta$, the parser derives the next stack as follows. The parser first pops $|\beta|$ grammatical symbols together with $|\beta|$ state symbols off the stack, where $|\beta|$ is the length of β. In this way, the stack-top state s_j is exposed. The parser then pushes A and s_{i+1} onto the stack, with s_{i+1} being the entry specified in the LR goto table for s_j and A. All these operations are executed deterministically.

A parse derivation completes if $l_n = \$$ and $a_n = accept$. We say stack transition sequence T is complete if $l_n = \$$, $a_n = accept$, and $\sigma_n = final$, where *final* is a dummy symbol denoting the stack when parsing is completed. Hereafter, we consistently refer to an LR parse state as a *state* and an LR parse stack as a *stack*. And, unless defined explicitly, s_i denotes the stack-top state of the i-th stack σ_i, i.e. $s_i = top(\sigma_i)$.

The probability of a complete stack transition sequence T can be decomposed as:

$$\begin{aligned} P(T) &= P(\sigma_0, l_1, a_1, \sigma_1, \ldots, \sigma_{n-1}, l_n, a_n, \sigma_n) & (5) \\ &= P(\sigma_0) \cdot \prod_{i=1}^{n} P(l_i, a_i, \sigma_i | \sigma_0, l_1, a_1, \sigma_1, \ldots, l_{i-1}, a_{i-1}, \sigma_{i-1}) & (6) \end{aligned}$$

Here we assume that σ_i contains all the information of its preceding parse derivation that has any effect on the probability of the next transition, namely:

$$P(l_i, a_i, \sigma_i | \sigma_0, l_1, a_1, \sigma_1, \ldots, l_{i-1}, a_{i-1}, \sigma_{i-1}) \approx P(l_i, a_i, \sigma_i | \sigma_{i-1}) \tag{7}$$

This assumption simplifies equation (6) to:

$$P(T) = \prod_{i=1}^{n} P(l_i, a_i, \sigma_i | \sigma_{i-1}) \tag{8}$$

Now, we show how we estimate each transition probability $P(l_i, a_i, \sigma_i | \sigma_{i-1})$, which can be decomposed as in (9):

$$P(l_i, a_i, \sigma_i | \sigma_{i-1}) = P(l_i | \sigma_{i-1}) \cdot P(a_i | \sigma_{i-1}, l_i) \cdot P(\sigma_i | \sigma_{i-1}, a_i, l_i) \tag{9}$$

To begin with, we estimate the first term $P(l_i|\sigma_{i-1})$ as follows:

Case 1. $i = 1$:

$$P(l_1|\sigma_0) = P(l_1|s_0) \tag{10}$$

Case 2. The previous action a_{i-1} is a shift action, i.e. $a_{i-1} \in \boldsymbol{A}_s$. We assume that only the current stack-top state $s_{i-1} = \mathit{top}(\sigma_{i-1})$ has any effect on the probability of the next input symbol l_i. This means that:

$$P(l_i|\sigma_{i-1}) \approx P(l_i|s_{i-1}) \tag{11}$$

where

$$\sum_{l \in La(s)} P(l|s) = 1 \tag{12}$$

Case 3. The previous action a_{i-1} is a reduce action, i.e. $a_{i-1} \in \boldsymbol{A}_r$. Unlike Case 2, the input symbol does not get consumed for reduce actions, and thus the next input symbol l_i is always identical to l_{i-1}; namely, l_i can be deterministically predicted. Therefore,

$$P(l_i|\sigma_{i-1}) = 1 \tag{13}$$

Next, we estimate the second term $P(a_i|\sigma_{i-1}, l_i)$ relying on the analogous assumption that only the current stack-top state s_{i-1} and input symbol l_i have any effect on the probability of the next action a_i:

$$P(a_i|\sigma_{i-1}, l_i) \approx P(a_i|s_{i-1}, l_i) \tag{14}$$

where

$$\sum_{a \in Act(s,l)} P(a|s,l) = 1 \tag{15}$$

Finally, as mentioned above, given the current stack σ_{i-1} and action a_i, the next stack σ_i can be uniquely determined:

$$P(\sigma_i|\sigma_{i-1}, l_i, a_i) = 1 \tag{16}$$

As shown in equations (11) and (13), the probability $P(l_i|\sigma_{i-1})$ should be estimated differently depending on whether the previous action a_{i-1} is a shift action or a reduce action. Let $\boldsymbol{S}_s$ be the set containing s_0 and all the states reached immediately after applying a shift action, and $\boldsymbol{S}_r$ be the set of states reached immediately after applying a reduce action:

$$\boldsymbol{S}_s \stackrel{\text{def}}{=} \{s_0\} \cup \{s | \exists a \in \boldsymbol{A}_s, \sigma : \ s = \mathit{top}(\mathit{next}(\sigma, a))\} \tag{17}$$

$$\boldsymbol{S}_r \stackrel{\text{def}}{=} \{s | \exists a \in \boldsymbol{A}_r, \sigma : \ s = \mathit{top}(\mathit{next}(\sigma, a))\} \tag{18}$$

where s_0 is the initial state. Note that these two sets are mutually exclusive[2]:

$$\boldsymbol{S} = \boldsymbol{S}_s \cup \boldsymbol{S}_r \quad \text{and} \quad \boldsymbol{S}_s \cap \boldsymbol{S}_r = \emptyset \tag{19}$$

Equations (9) through (18) can be summarized as:

$$P(l_i, a_i, \sigma_i|\sigma_{i-1}) \approx \begin{cases} P(l_i, a_i|s_{i-1}) & (\text{for } s_{i-1} \in \boldsymbol{S}_s) \\ P(a_i|s_{i-1}, l_i) & (\text{for } s_{i-1} \in \boldsymbol{S}_r) \end{cases} \tag{20}$$

Since $\boldsymbol{S}_s$ and $\boldsymbol{S}_r$ are mutually exclusive, we can assign a single probabilistic parameter to each action in an LR table, according to equation (20). To be more specific, for each state $s \in \boldsymbol{S}_s$, we associate a probability $p(a)$ with each action $a \in \mathit{Act}(s, l)$ (for $l \in \mathit{La}(s)$), where $p(a) = P(l, a|s)$ such that:

$$\sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) = 1 \quad (\text{for } s \in \boldsymbol{S}_s) \tag{21}$$

[2]It is obvious from the algorithm for generating an LR(1) goto graph[1] that, for each state s ($\neq s_0$), if there exist states s_i and s_j whose goto transitions on symbol X_i and X_j, respectively, both lead to s, then $X_i = X_j$. Namely, for any given state s, the symbol X required to reach s by way of a goto transition is always uniquely specified. On the other hand, if the current state is in S_s, then it should have been reached through a goto transition on a certain terminal symbol $X \in V_t$, whereas, if the current state is in S_r, then it should have been reached through a goto transition on a certain nonterminal symbol $X \in V_n$. Given these facts, it is obvious that S_s and S_r are mutually exclusive.

On the other hand, for each state $s \in \boldsymbol{S}_r$, we associate a probability $p(a)$ with each action $a \in Act(s,l)$ (for $l \in La(s)$), where $p(a) = P(a|s,l)$ such that:

$$\sum_{a \in Act(s,l)} p(a) = 1 \quad (\text{for } s \in \boldsymbol{S}_r) \tag{22}$$

Through assigning probabilities to actions in an LR table in this way, we can estimate the probability of a stack transition sequence T as given in (4) by computing the product of the probabilities associated with all the actions included in T:

$$P(T) = \prod_{i=1}^{n} p(a_i) \tag{23}$$

Before closing this section, we describe the advantages of our PGLR model. Our model inherits some of its advantages from B&C model. First, as in equation (14), the probabilistic distribution of each parsing action depends on both its left context (i.e. LR parse state) and right context (i.e. input symbol). This incorporates context-sensitivity into the model. We elaborate this through an example in section 4. Second, since the probability of each parse derivation can be estimated simply as the product of the probabilities associated with all the actions in that derivation, we can easily implement a probabilistic LR parser through a simple extension to the original LR parser. We can also easily train the model, as we need only count the frequency of application of each action in generating correct parse derivations for each entry in the training corpus. Third, both B&C model and our model are expected to be able to allow the parser to prune improbable parse derivations at an equivalently fine-grained level as that for Wright and Wrigley's statistical parser, since these two models assign probabilities to both shift and reduce actions. Furthermore, since our model assigns a single probabilistic parameter to each action in an LR table similarly to B&C model, the algorithm proposed by Carroll and Briscoe [4] for efficient unpacking of packed parse forests with probability annotations can be equally applicable to our model. Finally, although not explicitly pointed out by Briscoe and Carroll, it should also be noted that PCFGs give long-term preference over structures but do not sufficiently reflect short-term bigram statistics of terminal symbols, whereas both B&C model and our PGLR model reflect these types of preference simultaneously. $P(l_i|s_{i-1})$ in equation (11) is a model that predicts the next terminal symbol l_i for the current left context $s_{i-1} \in \boldsymbol{S}_s$. In this case of $s_{i-1} \in \boldsymbol{S}_s$, since s_{i-1} uniquely specifies the previous terminal symbol l_{i-1}, $P(l_i|s_{i-1}) = P(l_i|s_{i-1}, l_{i-1})$, which is a slightly more context-sensitive version of the bigram model of terminal symbols $P(l_i|l_{i-1})$. This feature is expected to be significant particularly when one attempts to integrate syntactic parsing with morphological analysis in the GLR parsing framework (e.g. [11]), since the bigram model of terminal symbols has been empirically proven to be effective in morphological analysis.

Besides these advantages, which are all shared with B&C model, our model overcomes the drawback of B&C model; namely, our model is based on a probabilistically well-founded formalization, which is expected to improve the parsing performance. We discuss this issue in the remaining sections.

3 Comparison with Briscoe and Carroll's Model

In this section, we briefly review B&C model, and make a qualitative comparison between their model and ours.

In our model, we consider the probabilities of transitions between stacks as given in equation (8), whereas Briscoe and Carroll consider the probabilities of transitions between *LR parse states* as below:

$$P(T) \approx \prod_{i=1}^{n} P(l_i, a_i, s_i|s_{i-1}) \tag{24}$$

$$= \prod_{i=1}^{n} P(l_i, a_i|s_{i-1}) \cdot P(s_i|s_{i-1}, l_i, a_i) \tag{25}$$

Briscoe and Carroll initially associate a probability $p(a)$ with each action $a \in Act(s,l)$ (for $s \in \boldsymbol{S}_s$, $l \in La(s)$) in an LR table, where $p(a)$ corresponds to $P(l_i, a_i|s_{i-1})$, the first term in (25):

$$p(a) = P(l, a|s) \tag{26}$$

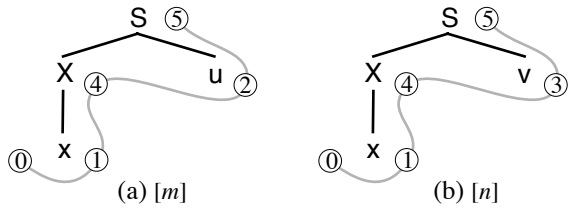


Figure 1. Parse trees derived from grammar $G1$ (The square-bracketed value below each tree denotes the number of occurrences of that tree.)

Table 1. LR table for grammar $G1$, with trained parameters (The numbers given in the middle and bottom of each row the parameters for B&C model and our model, respectively.)

state	**action**				**goto**	
	u	v	x	$	X	S
0			**sh1** $(m+n)$		4	5
(S_s)			1			
			1			
1	**re3** (m)	**re3** (n)				
(S_s)	$m/(m+n)$	$n/(m+n)$				
	$m/(m+n)$	$n/(m+n)$				
2				**re1** (m)		
(S_s)				1		
				1		
3				**re2** (n)		
(S_s)				1		
				1		
4	**sh2** (m)	**sh3** (n)				
(S_r)	$m/(m+n)$	$n/(m+n)$				
	1	1				
5				**acc** $(m+n)$		
(S_r)				1		
				1		

such that:

$$\forall s \in \boldsymbol{S}. \sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) = 1 \tag{27}$$

In this model, the probability associated with each action is normalized in the same manner for any state. However, as discussed in the previous section, the probability assigned to an action should be normalized differently depending on whether the state associated with the action is of class $\boldsymbol{S}_s$ or $\boldsymbol{S}_r$ as in equations (21) and (22). Without this treatment, probability $P(l_i|s_{i-1})$ in equation (11) could be incorrectly duplicated for a single terminal symbol, which would make it difficult to give probabilistically well-founded semantics to the overall score. As a consequence, in B&C formulation, the probabilities of all the complete parse derivations may not sum up to one, which would be inconsistent with the definition of $P(T)$ (see equation (2)).

To illustrate this, let us consider grammar $G1$ as follows.

Grammar G1:

(1) S → X u
(2) S → X v
(3) X → x

This grammar allows only two derivations as shown in Figure 1. Suppose that we have tree (a) with frequency m, and (b) with frequency n in the training set. Training B&C model and our model, respectively, with these trees, we obtain the models as shown in Table 1, where, for each LR parse state, each bracketed value in the top row denotes the number of occurrences of the action associated with it, and the numbers in the middle and bottom of each row denote the probabilistic parameters of B&C model and our model, respectively.

Given this setting, the probability of each tree in Figure 1 is computed as follows (see Figure 1, where each circled number denotes the LR parse state reached after parsing has proceeded from the left-most corner to the

location of that number):

$$P_{\mathrm{B\&C}}(\mathrm{tree(a)}) = 1 \cdot \frac{m}{m+n} \cdot \frac{m}{m+n} \cdot 1 = \left(\frac{m}{m+n}\right)^2 \tag{28}$$

$$P_{\mathrm{B\&C}}(\mathrm{tree(b)}) = 1 \cdot \frac{n}{m+n} \cdot \frac{n}{m+n} \cdot 1 = \left(\frac{n}{m+n}\right)^2 \tag{29}$$

$$P_{\mathrm{PGLR}}(\mathrm{tree(a)}) = 1 \cdot \frac{m}{m+n} \cdot 1 \cdot 1 = \frac{m}{m+n} \tag{30}$$

$$P_{\mathrm{PGLR}}(\mathrm{tree(b)}) = 1 \cdot \frac{n}{m+n} \cdot 1 \cdot 1 = \frac{n}{m+n} \tag{31}$$

where B&C denotes B&C model and PGLR denotes our model. This computation shows that our model correctly fits the distribution of the training set, with the sum of the probabilities being one. In the case of B&C model, on the other hand, the sum of these two probabilities is smaller than one. The reason can be described as follows. After shifting the left-most x, which leads the process to state 1, the model predicts the next input symbol as either u or v, and chooses the reduce action in each case, reaching state 4. So far, both B&C model and our model behave in the same manner. In state 4, however, B&C model repredicts the next input symbol, despite it already having been determined in state 1. This duplication makes the probability of each tree smaller than what it should be. In our model, on the other hand, the probabilities in state 4, which is of class $\boldsymbol{S}_r$, are normalized for each input symbol, and thus the prediction of the input symbol u (or v) is not duplicated.

Briscoe and Carroll are also required to include the second factor $P(s_i|s_{i-1}, l_i, a_i)$ in (25) since this factor does not always compute to one. In fact, if we have only the information of the current state and apply a reduce action in that state, the next state is not always uniquely determined. For this reason, Briscoe and Carroll further subdivide probabilities assigned to reduce actions according to the stack-top states exposed immediately after the pop operations associated with those reduce action. Contrastively, in our model, given the current stack, the next stack after applying any action can be uniquely determined as in (16), and thus we do not need to subdivide the probability for any reduce action.

To illustrate this, let us take another simple example in grammar $G2$ as given below, with all the possible derivations shown in Figure 2. Further, the LR table is shown in Table 2.

Grammar G2:

(1) S → u X
(2) S → v X
(3) X → x

Let us compute again the probability of each tree for the two models:

$$P_{\mathrm{B\&C}}(\mathrm{tree(a)}) = \frac{m}{m+n} \cdot 1 \cdot \frac{m}{m+n} \cdot 1 = \left(\frac{m}{m+n}\right)^2 \tag{32}$$

$$P_{\mathrm{B\&C}}(\mathrm{tree(b)}) = \frac{n}{m+n} \cdot 1 \cdot \frac{n}{m+n} \cdot 1 = \left(\frac{n}{m+n}\right)^2 \tag{33}$$

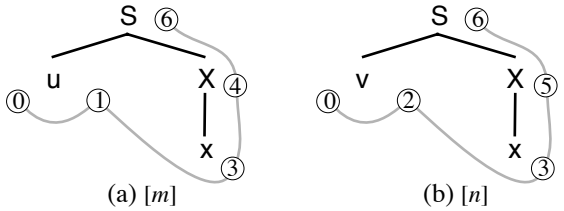
$$P_{\mathrm{PGLR}}(\mathrm{tree(a)}) = \frac{m}{m+n} \cdot 1 \cdot 1 \cdot 1 = \frac{m}{m+n} \tag{34}$$

$$P_{\mathrm{PGLR}}(\mathrm{tree(b)}) = \frac{n}{m+n} \cdot 1 \cdot 1 \cdot 1 = \frac{n}{m+n} \tag{35}$$

In B&C model, the probability assigned to the reduce action in state 3 with the next input symbol being \$ is subdivided according to whether the state reached after the stack-pop operation is state 1 or 2 (see Table 2). This makes the probability of each tree smaller than what it should be.

The above examples illustrate that, in B&C model, the probabilities of all the possible parse derivations may not necessarily sum up to one, due to the lack of probabilistically well-founded normalization. In our model, on the other hand, the probabilities of all the derivations are guaranteed to always sum to one[3]. This flaw in B&C model can be considered to be related to Briscoe and Carroll's claim that their model tends to

[3]Precisely speaking, this is the case if the model is based on a canonical LR (CLR) table. In the case of lookahead LR (LALR) tables, the probabilities of all the complete derivations may not sum up to one even for the case of our model, since some stack



(a) [m] (b) [n]

Figure 2. Parse trees derived from grammar $G2$ (The square-bracketed value below each tree denotes the number of occurrences of that tree.)

Table 2. LR table for grammar $G2$, with trained parameters (For each state, the numbers given in the middle and bottom of each row denote the parameters for B&C model and our model, respectively. Each middle bracketed number for state 3 denotes the state exposed by the stack-pop operation associated with the corresponding reduce action.)

state	**action**				**goto**	
	u	v	x	$	X	S
0	**sh1** (m)	**sh2** (n)				6
(S_s)	$m/(m+n)$	$n/(m+n)$				
	$m/(m+n)$	$n/(m+n)$				
1			**sh3** (m)		4	
(S_s)			1			
			1			
2			**sh3** (n)		5	
(S_s)			1			
			1			
3				**re3** ($m+n$)		
(S_s)				$^{(1)}m/(m+n)$; $^{(2)}n/(m+n)$		
				1		
4				**re1** (m)		
(S_r)				1		
				1		
5				**re2** (n)		
(S_r)				1		
				1		
6				**acc** ($m+n$)		
(S_r)				1		
				1		

favor parse trees involving fewer grammar rules, almost regardless of the training data. In B&C model, stack transition sequences involving more reduce actions tend to be assigned much lower probabilities for the two reasons mentioned above: (a) the probabilities assigned to actions following reduce actions tend to be lower than what they should be, since B&C model repredicts the next input symbols immediately after reduce actions, (b) the probabilities assigned to reduce actions tend to be lower than what they should be, since they are further subdivided according to the stack-top states exposed by the stack-pop operations. Therefore, given the fact that stack transition sequences involving fewer reduce actions correspond to parse trees involving fewer grammar rules, it is to be expected that B&C model tends to strongly prefer parse trees involving fewer grammar rules.

transitions may not be accepted (for details of CLR and LALR, see, for example, [1, 5]). However, this fact will never prevent our model from being applicable to LALR. Let $\mathcal{T}_{acc} \subset \mathcal{T}$ be the infinite set of all possible complete and *acceptable* stack transition sequences. The second factor $P(T)$ in equation (1) is a distribution over complete and acceptable stack transition sequences such that the probabilities of all the transition sequences in $\mathcal{T}_{acc}$ sum up to one. However, if one considers an LALR-based model, since there may be rejected transition sequences in $\mathcal{T}$, equation (1) should be replaced as follows:

$$P(T|W) = \alpha \cdot P(T|T \in \mathcal{T}_{acc}) \cdot P(W|T) = \alpha' \cdot P(T) \cdot P(W|T)$$

where α' is a constant that is independent of T, and $P(T)$ is a distribution over all the possible complete transition sequences, whether acceptable or not, such that $\sum_{T \in \mathcal{T}} P(T) = 1$. Thus, one can rank the parse tree candidates for any given input sentence according to $P(T)$ and $P(W|T)$, whether one bases the model on CLR, LALR, or even LR(0) (i.e. SLR). For further discussion, see [8].

To solve this problem, Briscoe and Carroll proposed calculating the geometric mean of the probabilities of the actions involved in each stack transition sequence. However, this solution makes their model even further removed from a probabilistically well-founded model. In our model, on the other hand, any bias toward shorter derivations is expected to be much weaker, and thus we do not require the calculation of the geometric mean.

One may wonder to what extent these differences matter for practical statistical parsing. Although this issue needs to be explored through large-scaled empirical evaluation, it must be still worthwhile to consider some likely cases where the difference discussed here will influence parsing performance. We discuss such a case through a further example in the next section.

4 Expected Impact on Parsing Performance

In this section, we first demonstrate through an example how B&C model and our model, which we class as GLR-based models here, realize mildly context-sensitive statistical parsing, as compared to the PCFG model. We then return to the issue raised at the end of the previous section.

Suppose we have grammar $G3$ as follows:

Grammar G3:

(1) S → u S
(2) S → v S
(3) S → x
(4) S → S S

Further, let us assume that we train the PCFG model, B&C model, and our PGLR model, respectively, using a training set as shown in Figure 3, where trees (a) and (b) are the parse trees for input sentence $W_1 = \{\mathsf{u}, \mathsf{x}, \mathsf{x}\}$, and (c) and (d) are those for $W_2 = \{\mathsf{v}, \mathsf{x}, \mathsf{x}\}$. Table 3 shows the LR table for grammar $G3$, with the trained

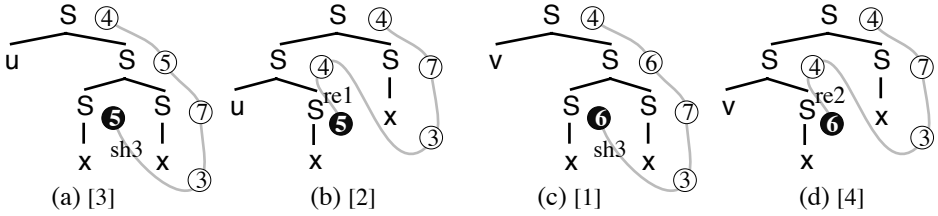


Figure 3. Training set (The square-bracketed number below each tree denotes the number of occurrences of that tree.)

parameters[4].

According to the training data in Figure 3, right branching (i.e. tree (a)) is preferred for input sentence W_1, whereas left branching (i.e. tree (d)) is preferred for input sentence W_2[5]. It is easy to see that the PCFG model does not successfully learn these preferences for either of the sentences, since all the parse trees produced for each sentence with involve the same set of grammar rules.

Unlike the PCFG model, both the GLR-based models can learn these preferences in the following way. In the LR parsing process for sentence W_1, the point where the parser must choose between parse trees (a) and (b) is in state 5, which is reached after the reduction of the left-most x into S (see Figure 3). In state 5, if the shift action is chosen, parse tree (a) is derived, while, if the reduce action is chosen, (b) is derived. Thus, the preference for (a) to (b) is reflected in the distribution over the shift-reduce conflict in this state. Table 3 shows that both B&C model and our model correctly prefer the shift action in state 5 with the next input symbol being x. For input sentence W_2, on the other hand, the left branching tree (d) is preferred. This preference is also reflected in the distribution over the shift-reduce conflict in the state reached after the reduction of the left-most x into S, but, this time, the state is state 6 instead of state 5. According to Table 3, state 6 with the next input symbol being x correctly prefers the reduce action, which derives the left-branching tree (d). In sum,

[4]In practical applications, when computing parameters, one would need to use some smoothing technique in order to avoid assigning zero to any parameter associated with an action that had never occurred in training.

[5]Such preferences are likely to be observed in real corpora. For example, in Japanese sentences, a conjunction at the beginning of a sentence tends to modify the rest of the sentence at the top-most level of the sentence. This corresponds to the preference for sentence W_1 with u equating to a conjunction. On the other hand, a postpositional phrase tends to be subordinated by its left-most verb, which corresponds to the preference for sentence W_2 with v and x being a postpositional phrase and a verb, respectively.

Table 3. LR table for grammar $G3$, with trained parameters (For each state, the numbers given in the middle and bottom of each row denote the parameters for B&C model and our model, respectively. Each middle bracketed number denotes the state exposed by the stack-pop operation associated with the reduce action corresponding to that number.)

state	**action**				**goto**
	x	y	z	$	S
0	**sh1**	**sh2**	**sh3**		4
(S_s)	.5	.5	0		
	.5	.5	0		
1	**sh1**	**sh2**	**sh3**		5
(S_s)	0	1	0		
	0	1	0		
2	**sh1**	**sh2**	**sh3**		6
(S_s)	0	0	1		
	0	0	1		
3	**re3**	**re3**	**re3**	**re3**	
(S_s)	0	0	$^{(1)}$.25 ; $^{(2)}$.25	$^{(4)}$.3 ; $^{(5)}$.15 ; $^{(6)}$.05	
	0	0	.5	.5	
4	**sh1**	**sh2**	**sh3**	**acc**	7
(S_r)	0	0	.23	.77	
	1	1	1	1	
5	**sh1/re1**	**sh2/re1**	**sh3/re1**	**re1**	7
(S_r)	0/0	0/0	.38/$^{(0)}$.25	.38	
	.5/.5	.5/.5	.6/.4	1	
6	**sh1/re2**	**sh2/re2**	**sh3/re2**	**re2**	7
(S_r)	0/0	0/0	.17/$^{(0)}$.67	.17	
	.5/.5	.5/.5	.2/.8	1	
7	**sh1/re4**	**sh2/re4**	**sh3/re4**	**re4**	7
(S_r)	0/0	0/0	0/0	$^{(0)}$.6 ; $^{(1)}$.3 ; $^{(2)}$.1	
	.5/.5	.5/.5	.5/.5	1	

the different preferences for W_1 and W_2 are reflected separately in the distributions assigned to the different states (i.e. states 5 and 6).

As illustrated in this example, for each parsing choice point, the LR parse state associated with it can provide a context for specifying the preference for that parse choice. This feature of the GLR-based models enables us to realize mildly context-sensitive parsing. Furthermore, although not explicitly demonstrated in the above example, it should also be noted that the GLR-based models are sensitive to the next input symbol as shown in (14) in section 2.

Now, let us see how the probabilities assigned to LR parsing actions are reflected in the probability of each parse tree. Table 4 shows the overall distributions provided by the PCFG model, B&C model, and our model, respectively, to the trees in Figure 3[6]. According to the table, our model accurately learns the distribution of the training data, whereas B&C model does not fit the training data very well. In particular, for sentence W_1, it goes as far as incorrectly preferring parse tree (b). This occurs due to the lack of well-founded normalization of probabilities as discussed in section 3. As mentioned above, B&C model correctly prefers the shift action in state 5, as does our model. However, for the rest of the parsing process, B&C model associates a considerably

[6]Although Briscoe and Carroll proposed to take the geometric mean of peripheral distributions as mentioned in section 3, we did not apply this operation when computing the probabilities in Table 4, to give the reader a sense of the difference between the probabilities given by B&C model and our model. Note that, in our example, since the number of state transitions involved in each parse tree is always the same for any given sentence, and given that the grammar generates only binary trees, taking the geometric mean would not change the preference order.

Table 4. Distributions over the parse trees from Figure 3 (trees (a) and (b) are the parse trees for input sentence $W_1 = \{\mathsf{u}, \mathsf{x}, \mathsf{x}\}$, and (c) and (d) are those for $W_2 = \{\mathsf{v}, \mathsf{x}, \mathsf{x}\}$)

	$P(\text{tree(a)}\|W_1)$	$P(\text{tree(b)}\|W_1)$	$P(\text{tree(c)}\|W_2)$	$P(\text{tree(d)}\|W_2)$
PCFG	.50	.50	.50	.50
B&C	.37	.63	.005	.995
PLR	.60	.40	.20	.80
training data	.60	.40	.20	.80

higher probability to the process from state 4 through 3 and 7 to 4, which derives tree (b), than the process from 3 through 7 and 5 to 4, which derives tree (a), since, in their model, the former process is inappropriately supported by the occurrence of tree (d). For example, in both parsing processes for (b) and (d), the pop operation associated with the reduction in state 3 exposes state 4, and B&C model thus assigns an inappropriately high probability to this reduction, compared to the reduction in state 3 for tree (a).

Of course, as far as various approximations are made in constructing a probabilistic model similar to both B&C model and our model, it is always the case that the model may not fit the training data precisely due to the insufficiency of the model's complexity. Analogous to B&C model, our model does not always fit the training data precisely due to the independence assumptions such as equations (7), (11), etc. However, it should be noted that, as illustrated by the above example, there is a likelihood that B&C model not fitting the training data is due not only to the insufficiency of complexity, but also the lack of well-founded normalization.

5 Conclusion

In this paper, we newly presented a formalization of probabilistic LR parsing. Our modeling inherits some of its features from B&C model. Namely, it is mildly context-sensitive, and naturally integrates short-term bigram statistics of terminal symbols and long-term preference over structures of parse trees. Furthermore, since the model is tightly coupled with GLR parsing, it can be easily implemented and trained. Inheriting these advantages, our formalization additionally overcomes an important drawback of B&C model: the lack of well-founded normalization of probabilities. We demonstrated through examples that this refinement is expected to improve parsing performance. Those examples may seem to be relatively artificial and forced. However, in our preliminary experiments, we are achieving some promising results, which support our claim (see [15] for a summary of preliminary results). We are now planning to conduct further large-scaled experiments.

It should also be noted that our modeling is equally applicable to both CLR tables and LALR tables. Since it is a highly empirical issue whether it is better to use CLR-based models or LALR-based models, it may be interesting to make experimental comparisons between these two types (for a qualitative comparison, see [8]).

Other approaches to statistical parsing using context-sensitive language models have also been proposed, such as [2, 9, 12, 14]. We need to make theoretical and empirical comparisons between these models and ours. The significance of introducing lexical sensitivity into language models should also not be underestimated. In fact, several attempts to use lexically sensitive models already exist: e.g. [6, 10, 13]. Our future research will also be directed towards this area [7].

Acknowledgements

The authors would like to thank the reviewers for their suggestive comments. They would also like to thank Mr. UEKI Masahiro and Mr. SHIRAI Kiyoaki (Tokyo Institute of Technology) for their fruitful discussion on the formalization of the proposed model. Finally, they would like to thank Mr. Timothy Baldwin (Tokyo Institute of Technology) for his help in writing this paper.

References

[1] A.V. Aho, S. Ravi, and J.D. Ullman. *Compilers, Principle, Techniques, and Tools.* Addision Wesely, 1986.

[2] E. Black, F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 31–37, 1993.

[3] T. Briscoe and J. Carroll. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, Vol. 19, No. 1, 1993.

[4] J. Carroll and E. Briscoe. Probabilistic normalization and unpacking of packed parse forests for unification-based grammars. In *Proceedings, AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pp. 33–38., 1992.

[5] N. P. Chapman. *LR Parsing — Theory and Practice.* Cambridge University Press, 1987.

[6] M. J. Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 1996.

[7] K. Inui, K. Shirai, H. Tanaka, and T. Tokunaga. Integrated probabilistic language modeling for statistical parsing. Technical Report TR97-0005, Dept. of Computer Science, Tokyo Institute of Technology, 1997. Available from `http://www.cs.titech.ac.jp/tr.html`.

[8] K. Inui, V. Sornlartlamvanich, H. Tanaka, and T. Tokunaga. A new probabilistic LR language model for statistical parsing. Technical Report TR97-0004, Dept. of Computer Science, Tokyo Institute of Technology, 1997. Available from `http://www.cs.titech.ac.jp/tr.html`.

[9] K. Kita. Spoken sentence recognition based on HMM-LR with hybrid language modeling. *IEICE Trans. Inf. & Syst.*, Vol. E77-D, No. 2, 1994.

[10] H. Li. A probabilistic disambiguation method based on psycholinguistic principles. In *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4)*, 1996. cmp-lg/9606016.

[11] H. Li and H. Tanaka. A method for integrating the connection constraints into an LR table. In *Proceedings of Natural Language Processing Pacific Rim Symposium '95*, pp. 703–708, 1995.

[12] Magerman. D. M. and Marcur. M. Pearl: A probabilistic chart parser. In *Proceedings of the 5th Conference of European Chapter of the Association for Computational Linguistics*, pp. 15–20, 1991.

[13] Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistics*, Vol. 2, pp. 425–432, 1992.

[14] S. Sekine and R. Grishman. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings of the International Workshop on Parsing Technologies '95*, 1995.

[15] V. Sornlertlamvanich, K. Inui, K. Shirai, H. Tanaka, T. Tokunaga, and T. Takezawa. Incorporating probabilistic parsing into an LR parser – LR table engineering (4) –. *Information Processing Sciety of Japan, SIG-NL-119*, 1997. Available from `http://tanaka-www.cs.titech.ac.jp/~inui/`.

[16] K.-Y. Su, J.-N. Wang, M.-H. Su, and J.-S. Chang. GLR parsing with scoring. In Tomita [18], chapter 7.

[17] M. Tomita. *An Efficient Parsing for Natural Languages.* Kluwer, Boston, Mass, 1986.

[18] M. Tomita, editor. *Generalised LR Parsing.* Kluwer Academic Publishers, 1991.

[19] J. H. Wright and E. N. Wrigley. GLR parsing with probability. In Tomita [18], chapter 8.