Incorporating Probabilistic Parsing into an LR Parser *
- LR Table Engineering (4) -

Virach Sornlertlamvanich†, Kentaro Inui†, Kiyoaki Shirai†, Hozumi Tanaka†
Takenobu Tokunaga† and Toshiyuki Takezawa‡
†Department of Computer Science, Tokyo Institute of Technology
‡ATR Interpreting Telecommunications Research Laboratories
†{virach,inui,kshirai,tanaka,take}@cs.titech.ac.jp
‡takezawa@itl.atr.co.jp

Abstract

We propose a probabilistic model for incorporating probability into a (G)LR parser. The model is formalized based on stack transition during parsing distinguishing it from the existing models proposed by Wright and Wrigley, and Briscoe and Carroll. Our model produces a remarkable improvement in statistical parsing with probability. Associating probabilities directly to actions in an LR parsing table, and theoretically requiring only one probability for each action promise model trainability and potential extension to other related tasks.

1 Introduction

Probabilistic techniques have been introduced to various kinds of natural language processing tasks, due to the increasing availability of text corpora. In syntactical parsing, probabilistic techniques are utilized to rank the potentially high numbers of parses generated for natural language (NL) applications.

Several attempts have been made to prune meaningless parses and aid in the selection of the most likely parse from multiple parse candidates. Fujisaki et al. [3] introduced the notion of a probabilistic context-free grammar (P-CFG), with probabilities trained in the Forward/Backward manner. Wright and Wrigley [6] formalized a method of mapping P-CFGs onto LR parsing tables by way of distributing the probabilities originally associated with a given CFG to each corresponding LR parsing action. As a result, the parser can incrementally compute the probability of each parse. Nevertheless, under Wright and Wrigley's model the resultant probability of a parse is identical to that acquired from the original P-CFG, despite the process of generating the LR parsing table being greatly complicated.

Briscoe and Carroll [2] proposed a simpler way of incorporating trained probabilities into each parsing action of the LR parsing table. Probabilities are computed directly from the frequency of application of each action when parsing the training corpus. Their method seems to be able to exploit the advantages offered by the context-sensitivity of LR parsing. LR parsing is, indeed, context-sensitive in that reduce actions depend on the state and lookahead symbol. But, without the obvious formalization of their model, there is doubt as to the validity of their method of including the left context for reduce actions in an attempt to increase the accuracy of computation of parse probability.

In contrast to the formal models proposed by Wright and Wrigley, and Briscoe and Carroll, we formalize our model based on stack transition during parsing. Parse probability is decomposed into a sequence of actions. Theoretically, our model requires only one probability for each action. Therefore, we can easily train our model by computing the frequency of application of each action when parsing the training corpus (correctly hand-annotated corpus), and directly associate a probability with each action in the LR parsing table. The results of our experiments clearly show that our model outperforms the other two in all cases, and is able to reduce the per-word cross entropy of the task.

In this paper, we propose a probabilistic model for LR parsers, with Section 2 describing the semantics of state transitions. Section 3 briefly reviews Briscoe and Carroll's model, and compares it to our model by means of the spareness in training of free parameters. We then clarify the

*LR パーザへの確立の導入

context-sensitive nature of LR parsing, and point out the dubious nature of Briscoe and Carroll's model in Section 4. Section 5 describes how a probability is allocated to each action of the table, through a simple example. Section 6 shows the results of experiments made on the three models.

2 Language Model for Probabilistic LR Parsing

Suppose we have a context-free grammar (CFG) and its corresponding LR table. Let $\boldsymbol{V}_n$ and $\boldsymbol{V}_t$ be the sets of nonterminal and terminal symbols, respectively, of the CFG. Further, let $\boldsymbol{S}$ and $\boldsymbol{A}$ be respectively, the sets of LR parse states and parsing actions appearing in the LR parsing table (LR table for short). For each state $s \in \boldsymbol{S}$, the LR table specifies a set of possible input symbols: $La(s) \subseteq \boldsymbol{V}_t$. Additionally, 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 sets 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, we can regard each parse derivation as a sequence of transitions between LR parse stacks. Let us consider a stack transition sequence T as:

$$\sigma_0 \overset{l_1,a_1}{\Longrightarrow} \sigma_1 \overset{l_2,a_2}{\Longrightarrow} \ldots \overset{l_{n-1},a_{n-1}}{\Longrightarrow} \sigma_{n-1} \overset{l_n,a_n}{\Longrightarrow} \sigma_n \quad (1)$$

where σ_i is the stack at time t_i, whose stack-top state is denoted by $top(\sigma_i)$, and l_i and a_i are, respectively, an input symbol and a parsing action chosen at time t_i. It can be proven from the LR parsing algorithm that, given a derived 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 σ_{i+1} $(= next(\sigma_i, a_{i+1}))$ can always be uniquely determined. A parse derivation completes successfully if $l_n = \$$ and $a_n = accept$. We say stack transition sequence T is complete if $l_n = \$$, $a_n = accept$, and $st_n = final$, where *final* is a dummy denoting the stack when parsing has completed. Hereafter, we consistently refer to an LR parse state as a *state* and an LR parse stack as a *stack*. Unless defined explicitly, s_i denotes the stack-top state of the stack σ_i at time t_i, i.e. $s_i = top(\sigma_i)$.

Herewith, the stack transition sequence T is decomposed into a sequence of stacks (Σ), input symbols (L) and parsing actions (A).

$$T = \{\Sigma, L, A\} \quad (2)$$

Given an input W, where W is a word sequence, the probability of a parse derivation is defined as follows,

$$\begin{aligned} W &= \{w_1, w_2, \ldots, w_m\} && (3)\\ P(T|W) &= P(\Sigma, L, A|W) && (4)\\ &= \alpha_W \cdot P(\Sigma, L, A) \cdot P(W|\Sigma, L, A) && (5)\\ &\approx \alpha_W \cdot P(\Sigma, L, A) \cdot P(W|L) && (6) \end{aligned}$$

Since our aim is to rank the probability of each stack transition, the scaling factor α_W in equation (6) can be omitted. In addition, the word sequence (W) is assumed to be independent of all other factors except for the sequence of input symbols (L).

In equation (6), $P(\Sigma, L, A)$ is the *LR parsing action probability* and estimates the probability of the parse derivation in terms of the sequence of state transitions, while $P(W|L)$ is the *lexical probability* and estimates the word sequence when the sequence of input symbols is estimated by $P(\Sigma, L, A)$. The probability derived from equation (6) is, in other words, a kind of generation probability. Considering the *LR parsing action probability* term, the probability of a complete stack transition T can be decomposed as follows:

$$\begin{aligned} &P(\Sigma, L, A)\\ &= P(\sigma_0, l_1, a_1, \sigma_1, \ldots, \sigma_{n-1}, l_n, a_n, \sigma_n) && (7)\\ &= P(\sigma_0) \cdot \prod_{i=1}^{n} P(l_i, a_i, \sigma_i| \\ &\quad \sigma_0, l_1, a_1, \sigma_1, \ldots, l_{i-1}, a_{i-1}, \sigma_{i-1}) && (8) \end{aligned}$$

In the LR parsing model, every parse derivation starts from the initial state (s_0), and therefore:

$$P(\sigma_0) = P(s_0) = 1 \quad (9)$$

According to the Markov assumption, it is possible to assume that the stack at time t_i depends only on the state at time t_{i-1}. Therefore, we can simplify equation (8) to:

$$P(T) = \prod_{i=1}^{n} P(\sigma_i, l_i, a_i|\sigma_{i-1}) \quad (10)$$

Furthermore, it is possible decompose the above probability $P(\sigma_i, l_i, a_i|\sigma_{i-1})$ into individual terms which we can estimate as:

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

$$P(l_i|\sigma_{i-1}) \cdot P(a_i|\sigma_{i-1}, l_i) \cdot P(\sigma_i|\sigma_{i-1}, a_i, l_i) \quad (11)$$

Considering the LR table, the state symbol summarizes the information contained in the stack below it, and the combination of the state symbol on top of the stack and the current input symbol is used to index the parsing table and determine the parsing decision. Hence we estimate the state on top of the stack instead of the stack below it.

To begin with, we estimate the first term $P(l_i|\sigma_{i-1})$ of equation (11) according to the type of the state, as follows.

In the initial state $i = 1$, where the stack contains only the initial state, the stack is the initial state itself.

$$P(l_1|\sigma_0) = P(l_1|s_0) \quad (12)$$

For non-initial states, we consider the state as being one of two different classes. The first class consists of states reached after applying a shift action ($\boldsymbol{S}_s$). The next input symbol is newly predicted at parse time by the state on top of the stack. Therefore,

$$P(l_i|\sigma_{i-1}) \approx P(l_i|s_{i-1}) \quad (13)$$

The second class consists of states that are reached after applying a reduce action ($\boldsymbol{S}_r$). Since the input symbol for the current state is already estimated in the state before the reduce action, the input symbol does not change ($l_i = l_{i-1}$). Therefore,

$$P(l_i|\sigma_{i-1}) = 1 \quad (14)$$

For the second term $P(a_i|\sigma_{i-1}, l_i)$ of equation (11), the probability of the current action a_i can be estimated from the state s_{i-1} on top of stack σ_{i-1}, and the input symbol l_i.

$$P(a_i|\sigma_{i-1}, l_i) \approx P(a_i|s_{i-1}, l_i) \quad (15)$$

For the third term $P(\sigma_i|\sigma_{i-1}, a_i, l_i)$ of equation (11), the current stack σ_i is unambiguously determined when the previous stack σ_{i-1} and the current action a_i are fixed, therefore,

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

From equation (11) to (16), we can summarize the *LR parsing action probability* as follows,

$$P(l_i, a_i, \sigma_i|\sigma_{i-1}) \approx \begin{cases} P(l_i, a_i|s_{i-1}) & (s_{i-1} \in \boldsymbol{S}_s) \\ P(a_i|s_{i-1}, l_i) & (s_{i-1} \in \boldsymbol{S}_r) \end{cases} \quad (17)$$

Since $\boldsymbol{S}_s$ and $\boldsymbol{S}_r$ are mutually exclusive, we can assign a probability to each action in the action part of an LR table, according to equation (17). To be more specific, for each state $s \in \boldsymbol{S}_s$, we associate a probability $p(a)$ for each action $a \in Act(s, l)$ ($l \in La(s)$), where $p(a) = P(l, a|s)$, such that:

$$\sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) = 1 \quad (\text{for } s \in \boldsymbol{S}_s) \quad (18)$$

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

$$\sum_{a \in Act(s,l)} p(a) = 1 \quad (\text{for } s \in \boldsymbol{S}_r) \quad (19)$$

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 (1) by computing the product of the probabilities associated with all the actions included in T:

$$P(T) = \prod_{i=1}^{n} p(a_i) \quad (20)$$

We can estimate the *lexical probability* [4], $P(W|L)$ in equation (6), through decomposition and estimation, by assuming that the probability of the current word depends only on its part-of-speech[1]. This estimation is made with the following equation.

$$P(W|L) \approx \prod_{i=1}^{n} P(w_i|l_i) \quad (21)$$

The new input word is taken into account only when the previous action is a shift action. If the previous action is a reduce action, the current input word is always identical to the previous input word. Therefore, in the same manner as for the estimation of the *LR parsing action probability*, the *lexical probability* can be defined as follows:

$$P(w_i|l_i) = \begin{cases} P(w_i|l_i) & (s_{i-1} \in \boldsymbol{S}_s) \\ 1 & (s_{i-1} \in \boldsymbol{S}_r) \end{cases} \quad (22)$$

[1] In our case, the terminal symbols used in the LR table are the parts-of-speech of input words

3 Comparison with Briscoe and Carroll's Model

In this section, we briefly review Briscoe and Carroll's model [2] and make a qualitative comparison with our model.

In out model, the probabilities of transitions between stacks are considered, as given in equation (10), whereas Briscoe and Carroll consider the probabilities of transitions between *LR parse states* as:

$$P(T) \approx \prod_{i=1}^{n} P(l_i, a_i, s_i | s_{i-1}) \quad (23)$$

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

Briscoe and Carroll initially associate a probability $p(a)$ with each action $a \in Act(s, l)$ ($l \in La(s)$) in an LR table, where $p(a)$ corresponds to $P(l_i, a_i | s_{i-1})$, the first term in (24):

$$p(a) = P(l, a | s) \quad (25)$$

such that:

$$\forall s \in \boldsymbol{S}. \sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) = 1 \quad (26)$$

As such, the probability associated with each action is normalized in the same manner for all states. 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 (18) and (19). Without this differentiation, probability $P(l_i | s_{i-1})$ in equation (13) could be duplicated for a single terminal symbol. As a consequence, in Briscoe and Carroll's formulation, the probabilities of all the complete parse derivations may not sum up to one.

Briscoe and Carroll are also forced to include the second term $P(s_i | s_{i-1}, l_i, a_i)$ in (24) since it is not always one. In general, if we only have the information of the current state and apply a reduce action, we cannot always uniquely determine the next state. For this reason, Briscoe and Carroll further subdivide the probabilities assigned to reduce actions according to the state reached after the reduce action is applied. Contrastively, in our model, given the current stack, the stack produced after the application of any action can be uniquely determined as in (16).

4 Context-Sensitivity in GLR Parsing

(Generalized) LR parsers are driven by a precompiled LR table, generated from a context-free grammar. Though the grammar used in generating the table is context-free, the nature of the table and the manner in which the LR parser is driven, make the parser mildly context sensitive. Briscoe and Carroll raised this issue in [2] but misinterpreted some aspects of the context sensitivity, causing the number of free parameters to increase unnecessarily and resulting in on unexpectedly complicated probability model. We will come back to this point after describing the context-sensitivity nature of (G)LR parsing.

In the process of generating an LR table, each state in the table is generated by applying the *goto* function, (as described in [1]), to the previous state ($s_i = goto[s_{i-1}, X_i]$). Each new state (s_i) is generated by consulting the previous state (s_{i-1}) and a grammar symbol (X_i), where X_i is a terminal symbol in the case of the next input symbol being incorporated onto the stack, or a non-terminal symbol when the stack is reduced by way of an appropriate reduction rule. Each state in the LR parsing table thus contains a local *left context* for the parser.

On the other hand, at parse time, the action in the LR table is determined by the pair of the state and input symbols ($a_{i+1} = action[s_i, l_{i+1}]$). This means that at time t_i, when the parser has come to a state (s_i), the parser will consider the next input symbol (l_{i+1}) in determining the next action (a_{i+1}). The next input symbol here provides the parser with a *right context* to aid in the determination process. Basically, the context taken into account during parsing is limited to one viable state and one input symbol.

In parsing natural language, ambiguity inevitably occurs for a fixed state and input symbol. Ambiguity occurs when there is more than one action corresponding to the given state and input symbol. Two cases exist for potential action conflicts: reduce/reduce conflicts (Figure 1) and shift/reduce conflicts (Figures 2 and 3). Due to the properties of LR tables, shift/shift conflicts never occur.

Let us consider the case of parsing with a grammar for constructing a binary tree. Reduce/reduce conflicts represent the dilemma of deciding which non-terminal label should be asso-

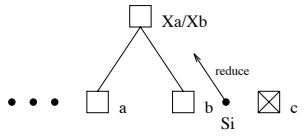

Figure 1: Reduce/reduce conflict

ciated with the structure (for instance, X_a or X_b in Figure 1), and shift/reduce conflicts constitute the problem of deciding whether to incorporate the next input symbol (d) onto the stack and delay construction of hierarchical structure to the next step (see Figure 2) or to create structure based on the previous stack (see Figure 3). In the case of parsing with only P-CFG, the most we can do is to assign a probability to each rule, based on the premise that the probabilities for all the rules that expand a common non-terminal must sum to one. Disambiguation in the case of Figure 1 is then made based on the comparison between the probabilities for $[X_a \rightarrow a\ b]$ and $[X_b \rightarrow a\ b]$. This same methodology is also used to disambiguate between Figures 2 and 3.

For both conflict types, the LR parser at least provides the left/right context to distinguish probabilities for reducing to the same non-terminal. The overall probability for reducing to X_a may be higher than to X_b but in some context, such as Figure 1, the probability for reducing to X_b can be higher than for X_a. The LR parser provides the context of the state number (s_i) and input symbol (c) to determine parse preference. Similarly, the context of state number (s_i) and input symbol (d) in Figures 2 and 3 can be used to give the preference to either shift or reduce.

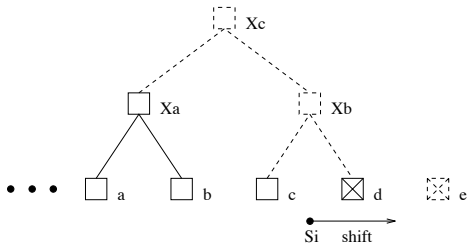

Figure 2: Shift preference for a shift/reduce conflict

Briscoe and Carroll [2] have pointed out some examples of NL phenomena that an LR parser can inherently handle. In the example of **he** *loves* **her**, an LR parser can distinguish between the contexts for reducing the first pronoun and the

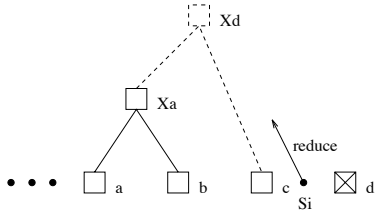

Figure 3: Reduce preference for a shift/reduce conflict

second pronoun to NP, given that the next input symbol after **he** is *loves*, while that for **her** is the sentence end marker (\$). However this does not work if the next input symbols are the same, such as the reduction of pronouns in the examples of **he** *passionately loves her* and *he loves* **her** *passionately*. As previously mentioned, Briscoe and Carroll proposed an approach of subdividing reduce actions according to the state reached after that reduce action is applied. The purpose of this approach is to distinguish between reductions using the left context of the reduction rule.

Subdividing reduce actions according to the state reached after the reduce action is one of the factors that leads to Briscoe and Carroll's model being unexpectedly complicated and including an unnecessarily large number of free parameters.

As we have described above regarding the left context of the LR parsing scheme, every state is generated by consulting the previous state and a grammar symbol. The states contain some local context, with the degree of the context depending directly on the type of the table, namely SLR, LALR or CLR (see [1]). Furthermore, the states reached after reduce actions (for instance, s_i and s_x in Figure 4) are determined deterministically. This is accounted for in our probabilistic modeling in Section 2.

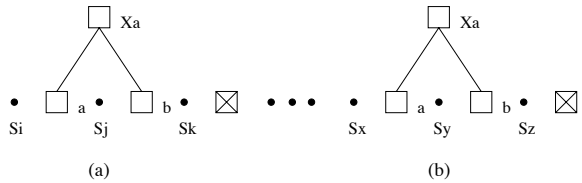

Figure 4: Reduction of $[X_a \rightarrow a\ b]$ in different contexts

The context sensitivity when parsing with either an LALR or CLR table is different, because during the process of generating an LALR table, states are merged together if they fulfill the re-

quirement of have the same *core* in the LR item [1]. As a result, the number of states in an LALR table is drastically decreased when compared a CLR table. Therefore, the left context contained in states in an LALR table is less than that for states in a CLR table (for further discussion see [5]). Despite this, however, the results of an experiment presented in Section 6 confirm that parsing with an LALR table does not significantly decrease accuracy.

5 Incorporating Probability into an LR Parsing Table

Our model described in Section 2 normalizes the probability is differently depending on the class of the state, either $\boldsymbol{S}_s$ or $\boldsymbol{S}_r$. States are distinguishable because of the property that the state classes for an LR table are mutually exclusive. This is because any state can be reached by only one type of grammar symbol. The states in the $\boldsymbol{S}_r$ class are those states referenced by transitions in the *goto* part of the table, and all other states are in the $\boldsymbol{S}_s$ class.

Suppose that we have a simple English context-free grammar as shown in Table 1, with LR table as shown in Table 2. We train our LR parser in the supervised mode, using a correctly hand-annotated corpus to guide the parser in its extraction of the sequence of actions required to produce the correct parse. We sum the frequency of application of each action, and add a part of a count to each action that appears in the table, to smooth the probability for unobserved events. Finally, each action probability is computed according to the state class which the action belongs to.

Table 1: A simple English context-free grammar

(1)	S	→	NP	VP
(2)	NP	→	det	n
(3)	NP	→	n	
(4)	NP	→	NP	PP
(5)	PP	→	p	NP
(6)	VP	→	v	NP
(7)	VP	→	VP	PP

Table 3 shows the artificial probability associated with each action. It is noticeable that the probabilities of actions in states in the $\boldsymbol{S}_s$ class (states 0, 1, 2, 4, 5, and 6) sum to one, but for states in the $\boldsymbol{S}_r$ class (states 3, 7, 8, 9, 10, 11 and 12), the probabilities of the actions in the slot of state and the input symbol sum to one. Table 3 does not show the goto part because actions in the goto part are deterministic and their probabilities are always one.

Table 2: LR table for the English grammar in Table 1

	Action					Goto			
State	det	n	p	v	$	NP	PP	S	VP
0	s2	s1				3		12	
1			r3	r3	r3				
2		s4							
3			s5	s6			7		8
4			r2	r2	r2				
5	s2	s1				9			
6	s2	s1				10			
7			r4	r4	r4				
8			s5		r1		11		
9			r5/s5	r5	r5		7		
10			r6/s5		r6		7		
11			r7		r7				
12					acc				

Table 3: LR table with its associated probabilities

	Action				
State	det	n	p	v	$
0	s2	s1			
	0.6	0.4			
1			r3	r3	r3
			0.4	0.5	0.1
2		s4			
		1			
3			s5	s6	
			1	1	
4			r2	r2	r2
			0.3	0.6	0.1
5	s2	s1			
	0.7	0.3			
6	s2	s1			
	0.7	0.3			
7			r4	r4	r4
			1	1	1
8			s5		r1
			1		1
9			r5/s5	r5	r5
			0.7/0.3	1	1
10			r6/s5		r6
			0.4/0.6		1
11			r7		r7
			1		1
12					acc
					1

6 Experiments

We tested our model (P-LR) on two Japanese corpora. Those two corpora were different in both their sources and their applied context-free grammar, but about the same size. Our experiments are designed to not be biased towards our model in terms of the degree of complexity of the task. For comparison of the performance of the different models, we conducted tests with the P-CFG

model as the baseline, and also with Briscoe and Carroll's model (B/C)

6.1 ATR Corpus and Grammar

The ATR corpus is a tree bank (a collection of trees annotated with a syntactic analysis, or "trees" for short) of Japanese dialogue. We randomly selected about 5% of the corpus to use as the test set and trained each parsing model with the remaining approximately 20,000 trees (see Table 4). The smallest sentence unit is a Japanese morpheme, and the length shown in the Table 4 is the number of morphemes. All trees are licensed by the Japanese phrase structure grammar developed at ATR. We generated an LALR table from the grammar of 360 production rules, comprised of 67 non-terminal symbols and 41 terminal symbols. 376 states were generated in the LALR table.

Table 4: ATR Corpus

		# of Morpheme	
ATR Corpus	*# of Sent.*	*Ave.*	*Range*
Training set	19,586	12.08	2-60
Test set	998	11.68	2-30

6.2 Results of the Experiment on the ATR Corpus

In the test, we used the test set to generate a sequence of parts-of-speech for each sentence, to act as the input for each model. The evaluation data for each model is shown as the percentage of ranked candidate parses containing an exact match to the standard parse. Candidate parses were ranked based on the value of the parse probability. Table 5 shows the percentage of the exact matches in 4 classes. "Exact-1" is the percentage of most probable candidate parses that matched the standard parse. "Exact-5" is the percentage of parse outputs containing an exact matched parse ranked within top 5 parses and so on. To make sure that the models can rank their output parses accurately, we count the rank of the lowest parse for parses of the same probability.

Table 5: Performance on the ATR Corpus

Rank	*P-CFG*	*B/C*	*P-LR*
Exact-1	76.05%	74.45%	82.77%
Exact-5	92.89%	88.78%	94.89%
Exact-10	95.69%	91.98%	96.69%
Exact-20	96.99%	94.79%	97.70%
Cross Entropy	2.72	N/A	2.40

Our model (P-LR) outperformed the other two models in every ranking class, and the per-word cross entropy is also less than that for the P-CFG model. The per-word cross entropy of Briscoe and Carroll's model (B/C) is out of range because of the large number of free parameters used to re-predict the next input symbol after applying a reduce action. Our model returning the highest parsing accuracies and the lowest per-word cross entropy, clearly shows that our model can efficiently use the context information encoded in the LR parsing scheme.

6.3 EDR Corpus and Grammar

In a similar way, we prepared another testing environment using the EDR corpus, an entirely different type of corpus. The EDR corpus is a collection of Japanese documents extracted from various newspaper sources, with the size of the corpus given in Table 6. From the corpus, we selected data which could be constructed by a set of binary production rules, for the purpose of testing the distinguishability of the phrase dependency of each model.

Table 6: EDR Corpus

		# of Bunsetsu	
EDR Corpus	*# of Sent.*	*Ave.*	*Range*
Training set	21,512	8.29	2-26
Test set	1,000	8.16	3-20

The grammar is a set of production rules modeling inter-phrasal structure, consisting of 296 binary rules. The smallest sentence unit is the case-marked phrase (*bunsetsu*). From the grammar, we generated an LALR table consisting of 34 non-terminal symbols, 25 terminal symbols and 478 states.

6.4 Results of the Experiment on the EDR Corpus

Table 7: Performance on the EDR Corpus

Rank	*P-CFG*	*B/C*	*P-LR*
Exact-1	43.00%	46.30%	54.30%
Exact-5	78.10%	76.90%	83.70%
Exact-10	87.20%	84.70%	91.10%
Exact-20	92.10%	91.20%	94.50%
Cross Entropy	2.79	N/A	2.60

Though the complexity of the sentence structure increased for the EDR corpus and most parts of the structure were constructed differently depending on the context, our model still outperformed the other two models, (see Table 7). The

complexity of the task can be ascertained from the per-word cross entropy, which is higher for the EDR test set than for the ATR test set.

6.5 Additional Experiment with CLR Table

We extended our experiment on the ATR corpus to use a CLR table, which is more distinctive in state assignment. The experiment was repeated for both Briscoe and Carroll's model and our P-LR model (note that the result for P-CFG model does not change for different table types). The CLR table occupied 814 states, more than twice the number of states in the corresponding LALR table. The overall performance using the CLR table was slightly higher than that for the LALR table, and the per-word entropy for the CLR table was also slightly less than that for the LALR table, (see Table 8). States in a CLR table are more distinguishable than those in an LALR table, but the drastic increase in states for the CLR table causes data sparseness in training. Thus, whereas we would expect the predictability for a CLR table to be higher than that for an LALR table, LALR tables perform similarly to CLR tables given the same size of training data. Considering the cost of table generation and parsing time, using an LALR table is much more efficient.

Table 8: Performance on the ATR Corpus

	B/C		*P-LR*	
Rank	*LALR*	*CLR*	*LALR*	*CLR*
Exact-1	74.45%	74.75%	82.77%	83.37%
Exact-5	88.78%	86.87%	94.89%	95.19%
Exact-10	91.98%	91.08%	96.69%	96.79%
Exact-20	94.79%	94.39%	97.70%	97.60%
Cross Entropy	N/A	N/A	2.40	2.32

7 Conclusion

We formalized a probabilistic model for GLR parsing and applied the method to the construction of a probabilistic LR table. The results of experiments clearly showed that our model (P-LR) is able to make effective use of both left and right context information within the LR parsing scheme. As a result, our model outperformed both Briscoe and Carroll's model and the P-CFG model in all tests. In addition, our model needs only the probability for each action in the LR table to compute the overall probability of each parse. It is thus tractable to training with the smallest amount of free parameters, and associates a probability directly to each action. Since the parse probability is incrementally calculated from action probabilities, the parser can compute the partial parse probability at any stage of the parse. We plan to extend the P-LR model to the parsing of sentences including unknown words, using the predictability of the model. The P-LR model is also expected to provide a means for recovering from ill-formed input sentences.

8 Acknowledgments

We would like to thank Masahiro Ueki for support in using MSLR and kindly adapting his parser to be able to accumulate action counts during the training phase. Taiichi Hashimoto helped us to improve the probabilistic parser, and Toshiki Ayabe helped with LR table generation, especially in computing the state list for use with the B/C model. In addition, Thanaruk Theeramunkong of JAIST provided helpful discussions on accumulating probabilities into a GLR packed shared parse forest.

References

[1] Aho, A., Sethi, R. and Ullman, J. 1986. *Compilers: Principles, Techniques, and Tools.* Addison-Wesley.

[2] Briscoe, T. and Carroll, J. 1993. Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. *Computational Linguistics*, Vol.19, No.1, pages 25-59.

[3] Fujisaki, T., Jelinek, F., Cocke, J., Black, E. and Nishino, T. 1989. A Probabilistic Parsing Method for Sentence Disambiguation. *Proceedings of 1st International Workshop on Parsing Technologies*, Carnegie-Mellon University, Pittsburgh, PA, pages 85-94.

[4] Inui, K., Shirai, K., Tokunaga, T. and Tanaka, H. 1996. Integration of Statistics-based Techniques for Analysis of Japanese Sentences (in Japanese). *IPSJ-NL*, SIG-NL-116-6.

[5] Inui, K., Sornlertlamvanich, V., Tanaka, H. and Tokunaga, T. 1997. A new probabilistic LR language model for statistical parsing. *Technical Report*, 97TR-0004, Dept. of Computer Science, Tokyo Institute of Technology.

[6] Wright, J. H. and Wrigley, E. N. 1991. GLR Parsing with Probability. *Generalized LR Parsing*, edited by Tomita, M., Kluwer Academic Publishers, pages 113-128.