Empirical Support for New Probabilistic Generalized LR Parsing

Virach Sornlertlamvanich,† Kentaro Inui,† Hozumi Tanaka,†
Takenobu Tokunaga† and Toshiyuki Takezawa††

This paper shows the empirical results of our probabilistic GLR parser based on a new probabilistic GLR language model (PGLR) against existing models based on the same GLR parsing framework, namely the model proposed by Briscoe and Carroll (B&C), and two-level PCFG or pseudo context-sensitive grammar (PCSG) which is claimed to be a context-sensitive version of PCFG. We evaluate each model in character-based parsing (morphological and syntactic analysis) tasks, in which we have to consider the word segmentation and multiple part-of-speech problems. Parsing a sentence from the morphological level makes the task much more complex because of the increase of parse ambiguity stemming from word segmentation ambiguities and multiple corresponding sequences of parts-of-speech. As a result of the well-founded probabilistic nature of PGLR, the model accurately incorporates probabilities for word prediction, by way of encoding pre-terminal n-gram constraints into LR parsing tables. The PGLR model empirically outperforms the other two models in all measures, on experimentation with the ATR Japanese corpus. To examine the appropriateness of PGLR using an LALR table, we test the PGLR model using both an LALR and CLR table. The results show that parsing with the PGLR model using LALR table returns the best performance in parse accuracy, parsing time and memory space consumption.

KeyWords: *Probabilistic parsing, GLR parser, Corpus, LALR table, CLR table*

1 Introduction

The probabilistic GLR language model (PGLR) (Inui, Sornlertlamvanich, Tanaka, and Tokunaga 1997) has previously been proven to be better than existing models, in particular the model proposed by Briscoe and Carroll (Briscoe and Carroll 1993) and the baseline model using a probabilistic context-free grammar (PCFG), in parsing strings of parts-of-speech (non-word-based parsing) (Sornlertlamvanich, Inui, Shirai, Tanaka, Tokunaga, and Takezawa 1997). Parsing a sentence from the morphological level makes the task much more complex because of the increase of parse ambiguity stemming from word segmentation ambiguities and multiple corresponding sequences of parts-of-speech. In this paper, we empirically evaluate

† Tokyo Institute of Technology, Department of Computer Science
†† ATR Interpreting Telecommunications Research Laboratories

the preciseness of a probabilistic model for PGLR against that for Briscoe and Carroll's model (B&C), which is based on the same GLR parsing framework. We also examine the benefits of context-sensitivity in GLR parsing, of the PGLR model against the "two-level PCFG" model (Chitrao and Grishman 1990) or "pseudo context-sensitive grammar" model (PCSG)—recently presented in (Charniak and Carroll 1994)—which has been shown to capture greater context-sensitivity than the original PCFG model, by empirical results and qualitative analysis.

Like the B&C model, PGLR inherits the benefits of context-sensitivity in generalized LR parsing (GLR). Its LR parsing table ("LR table" for short) is generated from a context-free grammar (CFG) by decomposing a parse into a sequence of actions. Every action in the LR table is determined by the pairing of a state and input symbol, so that it is valid to regard the state/input symbol pair as the context for determining an action. As a result, PGLR inherently captures two levels of context, i.e. global context over structures from the source CFG, and local n-gram context from adjoining pre-terminal constraints. Inui et al. (Inui et al. 1997) showed that B&C has some defects in distributing parse probabilities over the actions of an LR table. One is that, in B&C, no distinction is made between actions when normalizing action probabilities over the states in an LR table, while PGLR distinguishes the action probability normalization of states reached immediately after applying a shift action, from states reached immediately after applying a reduce action. B&C repeatedly counts the next input symbol when computing the probabilities (though the next input symbol is deterministic), if parsing is at the state reached immediately after applying a reduce action. Redundantly including the probabilities of the preceding input symbols in this case significantly distorts the overall parse probabilities. Moreover, subdividing reduce action probabilities according to the states reached after applying reduce actions is also redundant because resultant stack-top states after popping for reduce actions are always deterministic. B&C thus estimates parse probabilities lower than they should be.

Sornlertlamvanich et al. (Sornlertlamvanich et al. 1997) demonstrated the superior performance of PGLR over B&C and PCFG in a syntactic analysis task involving a determined sequence of parts-of-speech as input—non-word-based parsing. Most syntactic parsing model evaluation takes a string of parts-of-speech as input and leaves the problems of word segmentation and part-of-speech determination to other morphological analysis modules, such as part-of-speech taggers. Since GLR parsing has the ability to integrate morphological and syntactic analysis (Tanaka, Tokunaga, and Izawa 1996), we can easily realize PGLR parsing for morphosyntactic analysis, by adding lexical probabilities. To prove that local n-gram

constraints in PGLR are effective in morphological parsing, we conducted a character-based experiment on the ATR Japanese corpus, and compared the resulting performance with the existing B&C model and two-level PCFG, an extension of PCFG which is claimed to yield a significant performance advantage over the original PCFG framework. Since no spaces are placed between words in Japanese sentences, the models can in this way be evaluated in terms of both morphological and syntactic analysis.

Parsing highly ambiguous sentences or long sentences can cause extended parse time and exhaustive use of computing resources. We propose a new technique for pruning parses that have a lower probability than parses within a predetermined beam width, called the *node-driven parse pruning algorithm*. Unlike the method proposed by Carroll and Briscoe (Carroll and Briscoe 1992), which has to parse a sentence completely before extracting n-best parses, our technique allows pruning the less probable parses at any stage of parsing. By using our parse pruning technique, the parser needs to extend only the probable parses within a beam width, resulting in reduction of both parsing time and memory space.

Section 2 briefly reviews the various probabilistic models, namely B&C, two-level PCFG and PGLR, which are evaluated through character-based parsing on the ATR Japanese corpus. Section 3 shows the results of experiments carried out on the three models and the baseline model of PCFG. We discuss the empirical results and give case analyses in Section 4. Section 5 shows the relative results for LALR and CLR table-based PGLR and their parameter trainability.

2 Probabilistic Models

In this section, we briefly describe the existing probabilistic models, namely B&C and two-level PCFG, which are to be evaluated against the PGLR model. B&C is a high-performance probabilistic model proposed for the GLR parsing framework, as presented in (Briscoe and Carroll 1993). Two-level PCFG is an extended PCFG model for yielding greater context-sensitivity than the original paradigm. It was recently explored more thoroughly by Charniak and Carroll (Charniak and Carroll 1994), using the terminology "pseudo context-sensitive grammar" (PCSG), showing the improvement in per-word cross-entropy over the original PCFG model. Our motivation in selecting B&C and two-level PCFG for the comparative evaluation of PGLR is to examine the effectiveness of LR table context-sensitivity (global context over CFG-derived structures and local n-gram context from adjoining pre-terminal constraints), and the appropriateness of PGLR for GLR parsing.

In character-based parsing, given a string of characters $C = c_1, \ldots, c_n$ as an input, the

joint probability of a parse tree (T) and word sequence (W) is:

$$P(T,W|C) = \frac{P(T)\cdot P(W|T)\cdot P(C|W,T)}{P(C)} \quad (1)$$

$$\propto P(T)\cdot P(W|T) \quad (2)$$

The term $P(C|W,T)$ becomes one when word sequence W is determined, and $P(C)$ is a constant scaling factor, independent of T and W, which is not worthy of consideration in ranking parse trees and word sequences.

Probabilistic models allow us to estimate parse tree probabilities ($P(T)$). For the lexical probability $P(W|T)$, in our evaluation, we naively assume that word w_i in word sequence $W = w_1, \ldots, w_m$ depends only on its part-of-speech (l_i). Therefore,

$$P(W|T) \approx \prod_{i=1}^{m} P(w_i|l_i) \quad (3)$$

The estimation of lexical probability is applied identically in all models.

2.1 Briscoe and Carroll's Model (B&C)

Briscoe and Carroll (Briscoe and Carroll 1993) introduced probability to the GLR parsing algorithm in the light of the fact that LR tables do provide appropriate contextual information for solving the context-sensitivity problems observable in real world NL applications. They pointed out that an LR parse state encodes information about the left and right context of the parse. This results in distinguishability of context for an identical rule reapplied in different ways across different derivations. Briscoe and Carroll's method allows us to associate probabilities with an LR table directly, rather than simply with the rules of the grammar.

They consider the LR table as a nondeterministic finite-state automaton. Each row of the LR table corresponds to the possible transitions out of the state represented by that row, and each transition is associated with a particular lookahead item[1] and a parsing action. Nondeterminism arises when more than one action is possible given a particular input symbol. The following is a review of B&C in terms of our formalization.

Briscoe and Carroll regard a parse derivation as a sequence of state transitions (T):-

$$s_0 \overset{l_1,a_1}{\Longrightarrow} s_1 \overset{l_2,a_2}{\Longrightarrow} \cdots \overset{l_{n-1},a_{n-1}}{\Longrightarrow} s_{n-1} \overset{l_n,a_n}{\Longrightarrow} s_n \quad (4)$$

where a_i is an action, l_i is an input symbol and s_i is the state at time t_i. The probability of

1 The term "lookahead", originally used in (Aho, Sethi, and Ullman 1986), refers to the extra information that is incorporated into a state by redefining *items* to include a terminal symbol as a second component. An LR *item* of a grammar G is a production of G with a dot at some position of the right side. In this case, however, Briscoe and Carroll refer to lookahead as an "input symbol".

the parse derivation T is estimated by equation (5):-

$$\begin{aligned} P(T) &\approx \prod_{i=1}^{n} P(l_i, a_i, s_i | s_{i-1}) \\ &= \prod_{i=1}^{n} P(l_i, a_i | s_{i-1}) \cdot P(s_i | s_{i-1}, l_i, a_i) \end{aligned} \tag{5}$$

Based on B&C, the following is a summary of the scheme for deriving the action probabilities ($p(a)$) from the count of state transitions resulting from parsing a training set.

(1) The probability of an action given an input symbol is conditioned by the state it originated from. The probabilities assigned to each action at a given state must sum to one. Therefore,

$$\sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) = 1 \quad (\text{for } \forall s \in \boldsymbol{S}) \tag{6}$$

where $La(s)$ is the set of input symbols at state s, $Act(s, l)$ is the set of actions given a pair of state s and input symbol l, and $\boldsymbol{S}$ is the set of all states of the LR table. This means that the actions in the LR table are normalized within each state.

(2) In the case of a shift action ($\boldsymbol{A}_s$), $P(s_i | s_{i-1}, l_i, a_i)$ in equation (5) is equal to one because shift conflict never occurs in an LR table. Therefore,

$$p(a_i) = P(l_i, a_i | s_{i-1}) \quad (\text{for } a_i \in \boldsymbol{A}_s) \tag{7}$$

(3) In the case of a reduce action ($\boldsymbol{A}_r$), the probability is subdivided according to the state reached after applying the reduce action. The reason for this is that Briscoe and Carroll associate probabilities with transitions in the automaton rather than with actions in the action part of the LR table. In this case $P(s_i | s_{i-1}, l_i, a_i)$ in equation (5) is not one. Therefore,

$$\begin{aligned} p(a_i) = P(l_i, a_i | s_{i-1}) \cdot P(s_i | s_{i-1}, l_i, a_i) \\ (\text{for } a_i \in \boldsymbol{A}_r) \end{aligned} \tag{8}$$

B&C employs the geometric mean of the probabilities of the actions ($p(a_i)$) for state transitions across the whole parse derivation as the probability of a parse derivation, to avoid bias in favor of parses involving fewer rules or equivalently smaller trees. Therefore,

$$P(T) = (\prod_{i=1}^{n} p(a_i))^{1/n} \tag{9}$$

2.2 Two-level Probabilistic Context-Free Grammar (two-level PCFG)

Two-level PCFG is an extended version of PCFG, deriving from the idea of providing context-sensitivity for a context-free grammar.

In the original PCFG model, the probability of a parse derivation (T) is regarded as the product of probabilities of the rules which are employed for deriving that parse derivation. Each production rule of the grammar (r_i) is of the form $< A \rightarrow \alpha, P(r_i) >$ where $P(r_i)$ is the associated probability, and the probabilities associated with all rules with a given nonterminal A on the left-hand side must sum to one. $P(\alpha|A)$ is the probability of a rule whose left-hand side nonterminal symbol is "A", and the right-hand side "α" is a sequence of terminal and/or non-terminal symbols. Therefore,

$$\sum_{\alpha} P(\alpha|A) = 1 \tag{10}$$

$$P(T) = \prod_{i} P(r_i) \tag{11}$$

where $P(r)$ corresponds to $P(\alpha|A)$.

Two-level PCFG utilizes extra information provided by the parent of nonterminals in expanding rules (r_i) through assignment of rule probabilities. Thus, the rule probability in equation (10) can be rewritten as:

$$\sum_{\alpha} P(\alpha|\rho(A)) = 1 \tag{12}$$

where $\rho(A)$ is the nonterminal that immediately dominates A (i.e. its parent).

2.3 Probabilistic Generalized LR (PGLR)

Inui et al. (Inui et al. 1997) recently proposed a new formalization of a probabilistic model for GLR parsing. Unlike B&C, a parse derivation is regarded as a sequence of transitions between LR parse stacks (T) as shown in (13), where σ_i is the stack at time t_i, a_i is an action, and l_i is an input symbol. Schema (13) shows the inherent diversion from B&C in the definition of parse derivation.

$$\sigma_0 \stackrel{l_1,a_1}{\Longrightarrow} \sigma_1 \stackrel{l_2,a_2}{\Longrightarrow} \ldots \stackrel{l_{n-1},a_{n-1}}{\Longrightarrow} \sigma_{n-1} \stackrel{l_n,a_n}{\Longrightarrow} \sigma_n \tag{13}$$

Based on the above definition, the probability of a complete stack transition sequence T can be represented with equation (14), by assuming that σ_i contains all the information of its

preceding parse derivation:-

$$P(T) = \prod_{i=1}^{n} P(l_i, a_i, \sigma_i | \sigma_{i-1}) \tag{14}$$

To estimate each transition probability $P(l_i, a_i, \sigma_i | \sigma_{i-1})$, we decompose it to:

$$P(l_i, a_i, \sigma_i | \sigma_{i-1}) = P(l_i | \sigma_{i-1}) \cdot P(a_i | \sigma_{i-1}, l_i) \cdot P(\sigma_i | \sigma_{i-1}, l_i, a_i) \tag{15}$$

By the nature of GLR parsing, the stack σ_i is determined if the previous stack σ_{i-1}, the next input symbol l_i and action a_i are given. Therefore, the term $P(\sigma_i | \sigma_{i-1}, l_i, a_i)$ is equal to one. We hence have to estimate only the terms $P(l_i | \sigma_{i-1})$ and $P(a_i | \sigma_{i-1}, l_i)$.

The first term $P(l_i | \sigma_{i-1})$ is a conditional probability for estimating an input symbol l_i, given the current stack σ_{i-1}. According to the LR parsing algorithm, the input symbol after applying a reduce action remains unchanged. Therefore, we have to consider $P(l_i | \sigma_{i-1})$ separately in terms of the actions that are applied to reach the stack σ_{i-1}:

(1) If the previous action a_{i-1} is a shift action, and we assume that the stack-top state represents the stack information beneath it, then

$$P(l_i | \sigma_{i-1}) \approx P(l_i | s_{i-1}) \tag{16}$$

(2) If the previous action a_{i-1} is a reduce action, the input symbol is not changed. Therefore,

$$P(l_i | \sigma_{i-1}) = 1 \tag{17}$$

The second term $P(a_i | \sigma_{i-1}, l_i)$ is a conditional probability for estimating an action a_i, given the current stack σ_{i-1} and input symbol l_i. By the same assumption as applied above, that the stack-top state represents the stack information beneath it, this term can be estimated by:

$$P(a_i | \sigma_{i-1}, l_i) \approx P(a_i | s_{i-1}, l_i) \tag{18}$$

The first term in transition probability (15) is estimated differently depending on the type of action applied to reach the current stack. Therefore, we divide states into two classes, namely the class of states reached after applying a shift action ($\boldsymbol{S}_s$), and the class of states reached after applying a reduce action ($\boldsymbol{S}_r$). Fortunately, these two classes are mutually exclusive because state transition in LR parsing is representable by a deterministic finite automaton[2] (DFA). A state can be reached by a unique grammar symbol, which distinguishes states that are reached by a terminal symbol from states that are reached by a non-terminal symbol.

2 A deterministic finite automaton has at most one transition from each state on any input.

Therefore,

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

To summarize, the transition probability can be rewritten as:

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

such that:

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

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

where $p(a)$ is the probability of an action a, $\boldsymbol{S}_s$ is the class of states reached after applying a shift action, including the initial state, and $\boldsymbol{S}_r$ is the class of states reached after applying a reduce action.

The probability of a parse derivation is the product of the probabilities of the actions ($p(a_i)$) for stack transitions across the whole parse derivation:-

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

3 Experimental Results

We evaluated the various probabilistic models (i.e. B&C, two-level PCFG and PGLR) on a portion of the ATR Japanese corpus, called the Spoken Language Database (SLDB) (Takezawa 1997). Given an input string of Japanese characters, each model produces probabilistically ranked parses together with the associated parse probabilities, computed as described in Section 2.

As the dictionary to generate word candidates and their corresponding parts-of-speech, we collected all the words used in the corpus. Each word in the dictionary retains lexical probability $P(w|l)$, which is the probability of generating word 'w' from an arbitrary part-of-speech 'l'.

Each model was trained equally with the same hand-annotated training set. For unseen events, we simply added part of a count to smooth the model probabilities. Evaluation of the smoothing method is beyond the scope of this paper. We additionally present the results of the original PCFG framework as the baseline for the evaluation.

3.1 ATR Corpus and Grammar

The "Spoken Language Database" (SLDB) is a treebank (a collection of trees annotated with a syntactic analysis, or "trees" for short) developed by ATR based on Japanese dialogue. A portion of the corpus has been revised through application of a more detailed phrasal grammar developed by Tanaka et al. (Tanaka, Takezawa, and Etoh 1997). We randomly selected about 5% of the revised corpus to use as a test set and trained each parsing model with the remaining approximately 10,000 trees. Table 1 describes a breakdown of the corpus. The range and average of sentence length in both the training and test sets are very close, from which it is plain that the test set was appropriately selected from the corpus.

Table 1 The ATR Corpus.

		# of Morphemes		# of Characters	
ATR Corpus	**# of Sent.**	**Range**	**Average**	**Range**	**Average**
Training set	10,361	1-34	6.69	2-58	12.57
Test set	545	1-22	6.36	2-42	12.03

We implemented all the models using a GLR parser. We generated an LALR table from the treebank governing context-free grammar of 762 production rules, comprised of 137 non-terminal symbols and 407 terminal symbols. The generated LALR table contained 856 states.

3.2 Parsing the ATR Corpus

We used PARSEVAL measures (Black et al. 1992) to compare the performance of the top-ranked parses for each model.

Table 2 shows that the PGLR model outperformed the other models in every metric. Looking at the metrics of BP, BR, 0-CB and m-CB, where all structure labels are ignored, every model returned very good results ($> 93\%$). The disparity between models becomes significant when bracket labels are taken into consideration, such as in LP/LR and PA. Therefore, the difficulty lies not in bracketing but in labeling, including determining pre-terminals which is an essential task in character-based parsing.

One reason for this is that, the context-free grammar used for this corpus is relatively restricted in terms of terminal assignment. Information about word form (e.g. *sahen-meisi*—Sino-Japanese verbal noun), post position (e.g. *-/ga* and *-/ni*), for example, is explicitly included in non-terminal symbol labels. Therefore, word connection constraints within the rules can somehow exclude the invalid word combinations. Results from our preliminary test

Table 2 Performance on the ATR Corpus. **PA** is the parse accuracy and indicates the percentage of top-ranked parses that match standard parses. **LP/LR** are label precision/recall. **BP/BR** are bracket precision/recall. **0-CB** and **m-CB** are zero crossing brackets and mean crossing brackets per sentence, respectively.

	2-42 Characters (545 sentences)						
Models	**PA**	**LP**	**LR**	**BP**	**BR**	**0-CB**	**m-CB**
B&C	88.62	97.72	97.50	98.48	98.05	93.94	0.15
Two-leval PCFG	62.39	96.28	95.32	98.61	97.38	95.23	0.10
PCFG	53.03	95.67	94.54	98.77	97.35	94.86	0.08
PGLR	95.23	99.08	98.50	99.54	98.76	98.53	0.03

	15-42 Characters (160 sentences)						
Models	**PA**	**LP**	**LR**	**BP**	**BR**	**0-CB**	**m-CB**
B&C	73.75	96.00	97.26	96.84	98.14	83.75	0.44
Two-level PCFG	56.25	97.44	97.31	98.90	98.76	93.13	0.18
PCFG	35.62	95.86	95.64	98.60	98.39	90.63	0.17
PGLR	90.00	98.98	98.99	99.49	99.50	96.25	0.08

on part-of-speech input sequences showed that structural ambiguity hardly occurred. Most of the sentences had no ambiguity. From this study, it was observed that most of the sentences had only one parse if the parts-of-speech of the words in that sentence were defined. The ambiguity increases, however, when we consider parsing input strings of characters.

The average parse base[3] (APB) of the test set is as high as 1.348 in the character-based measure. This is comparable with the SUSANNE corpus (1.256) and SEC corpus (1.239), as reported in (Briscoe and Carroll 1995). Therefore, the performance of character-based parsing in this test mainly depends on the accuracy of selecting words and their corresponding parts-of-speech. This means that a model that can provide local context in addition to the global context would result in higher performance.

As expected, the models which make effective use of the local context modeling nature of GLR parsing, namely B&C and PGLR, returned significantly better results than PCFG-based parsing. Although the PCFG rule context in two-level PCFG extends a step higher (i.e. to the parent of the reduced rule), the model still failed to include appropriate context in some

3 Briscoe and Carroll (Briscoe and Carroll 1995) defined APB as the measure of ambiguity for a given corpus. It is the geometric mean over all sentences in the corpus of $\sqrt[n]{p}$, where n is the number of words in a sentence, and p is the number of parses for that sentence.

cases. One such case is shown in Section 4.

Parse accuracy (PA) shows the percentage of correct parses that are ranked topmost according to the model probability. By this measure, PGLR maintained the highest accuracy in ranking parses, while the PCFG-based models dropped down to slightly higher than 50%. Since the corpus is a kind of spoken language database, there are a lot of short response utterances i.e. "yes", "no" and "take care". The lower table in Table 2 is added to show the performance on sentences ranging from 15 to 42 characters. The difference in performance becomes obvious in parsing longer sentences.

Parse performance partly depends on the grammar and the corpus. According to the report of an experiment on the SUSANNE English corpus by Carroll (Carroll 1997), the difference between the performance of PGLR and B&C was not significantly observed. However, the input test set was a set of part-of-speech sequences, excluding ambiguity in word and part-of-speech selection. Even here, though, PGLR returned the best result in terms of the m-CB metric.

4 Discussion

It is obvious that two-level PCFG shows the benefits of context-sensitivity and yields significant gains over the original PCFG model. However, the results are still far below those for the probabilistic GLR-based parsing models. One reason would be the advantages of local context, i.e. pre-terminal n-gram constraints encoded in the LR table. The n-gram constraints are distributed over the actions of the table. Therefore, the parse trees generated by probabilistic GLR-based parsers include pre-terminal n-gram constraints in the parse probabilities.

The example below shows that probabilistic GLR-based parsing can successfully exploit the advantages of pre-terminal n-gram constraints, and assign parse probabilities in a more accurate manner. Based on Grammar-1, the three parse tree types in Figure 1 can be generated. Supposing that (S1) and (S2) are found one and two times respectively in our training set, but (S3) does not occur. (S3) can be found very rarely, or alternatively never occur because it may have no obvious meaning. This actually happens for most wide-coverage grammars.

The case shown in Figure 1 is simplified from one of the cases we have found in our test set. It is the case of selecting the appropriate part-of-speech for a sentence-ending word "su", which can be "infl-masu-su" or "infl-desu-su". It must be assigned "infl-masu-su", if it follows a word "ma" having the part-of-speech of "auxstem-masu", and assigned "infl-desu-su" if it follows a word "de" having the part-of-speech of "auxstem-desu". In this case, only the pre-terminal n-gram constraints are effective, rather than the constraints from the parent node

which are the same in both cases. In Figure 1, 'a', 'b' and 'c' correspond to "auxstem-masu", "auxstem-desu" and "infl-masu-su", respectively.

Grammar-1. A context-free grammar.

(1) X → U c
(2) X → U
(3) U → a
(4) U → b

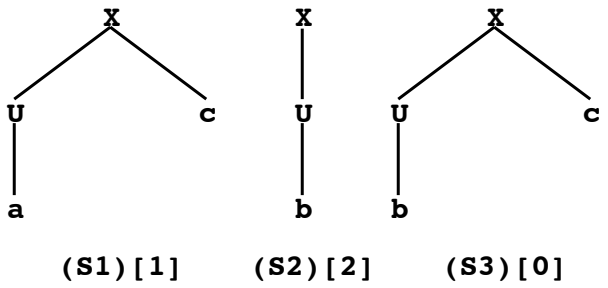


Fig. 1 Parse trees, with the frequency in the training set shown in brackets.

Probability-1. Rule probabilities for two-level PCFG.

(1) S ; X → U c (1/3)
(2) S ; X → U (2/3)
(3) X ; U → a (1/3)
(4) X ; U → b (2/3)

The bracketed values given for Probability-1 are the rule probabilities estimated according to the two-level PCFG model from the training set in Figure 1. In fact, they are the same as for PCFG because the parents of rules (1) and (2) are not different, and neither are the parents of rules (3) and (4). This means that the extended context in two-level PCFG does not have any effect if direct parents are the same. We need more information to distinguish the cases. Unfortunately, however, there are no other parent nodes in this case.

Table 3 is an LALR table generated from Grammar-1. The associated probabilities below each action are estimated according to B&C and PGLR, indicated in the first and second lines of each state row, respectively. For the sake of brevity, we do not consider any smoothing technique in this table, although smoothing was performed in the experiments described in Section 3.

Table 3 The LALR table with its associated probabilities. Probabilities in the first line of each state row are those estimated by B&C and the bracketed values in the second line are those estimated by PGLR.

State	Action				Goto	
	a	b	c	$	U	X
0	sh3	sh2			1	4
	1/3	2/3				
	(1/3)	(2/3)				
1			sh5	re2		
			1/2	1/2		
			(1)	(1)		
2			re4	re4		
			0	1		
			(0)	(1)		
3			re3	re3		
			1	0		
			(1)	(0)		
4				acc		
				1		
				(1)		
5				re1		
				1		
				(1)		

Applying the probabilities prepared in Probability-1 for two-level PCFG (as well as PCFG), and Table 3 for B&C and PGLR, to estimate the parse probabilities of (S1), (S2) and (S3) in Figure 1, we obtain the results shown in Table 4. Two-level PCFG (and PCFG) wrongly assigned preference to (S3) over (S1), whereas (S3) never occurs in the training set. Although B&C yields correct preference, the probabilities are smaller than what they should be. In this case, there is no difference between B&C and PGLR in ranking the parses. The side-effects of inappropriate normalization of probabilities in B&C has already been explored in (Inui et al. 1997) and empirically confirmed in the evaluation in Section 3.

Table 4 Probabilities of parse trees, (S1), (S2) and (S3), estimated according to each model.

Models	(S1)	(S2)	(S3)
PCFG	1/9	4/9	2/9
Two-level PCFG	1/9	4/9	2/9
B&C	1/6	1/3	0
PGLR	1/3	2/3	0

5 Comparative Results for LALR and CLR Table-based PGLR

Canonical LR (CLR), or LR with one lookahead, is the most powerful and the most expensive. In the LR table generating process, one lookahead symbol is taken into account by adding as the second component in constructing an *item*, where the first component in the *item* is a dot notation, called the *core* of the *item*. An LR *item* of a grammar G is a production of G with a dot at some position of the right side.

Table 5 (Aho et al. 1986) shows the CLR and LALR tables generated from the same grammar. It is obvious that the LALR table requires fewer states than the CLR table does. According to the process of generating the LALR table, state 1 and 6, state 2 and 7, and state 5 and 9 in the CLR table are merged to be state 1, 2 and 5, respectively, because they have the same *core* in their LR *item*.

Table 5 The CLR table and the corresponding LALR table, generated from the same grammar.

[CLR table]

	Action			Goto	
State	c	d	$	S	C
0	sh1	sh2		3	4
1	sh1	sh2			5
2	re3	re3			
3			acc		
4	sh6	sh7			8
5	re2	re2			
6	sh6	sh7			9
7			re3		
8			re1		
9			re2		

[LALR table]

	Action			Goto	
State	c	d	$	S	C
0	sh1	sh2		3	4
1	sh1	sh2			5
2	re3	re3	re3		
3			acc		
4	sh1	sh2			6
5	re2	re2	re2		
6			re1		

Consequently, the degree of context sensitivity when parsing with an LALR and CLR table can be different, and the left context encoded in states in an LALR table can be decreased compared with that for states in a CLR table. However, the tremedous number of states in the CLR table can cause the data sparseness problem. To verify our PGLR model on both tables, we thus extended our experiment to examine the performance of PGLR using an LALR table (PGLR(LALR)) against using a CLR table (PGLR(CLR)).

The CLR table contained 3,715 states, more than fourfold the number of states in the corresponding LALR table (856 states) for the same ATR Japanese grammar. Table 6 shows that PGLR(LALR) returned slightly better results than PGLR(CLR). However, the difference

is not statistically significant.

Table 6 Performance on the ATR Corpus. Comparative results for PGLR using an LALR and CLR table.

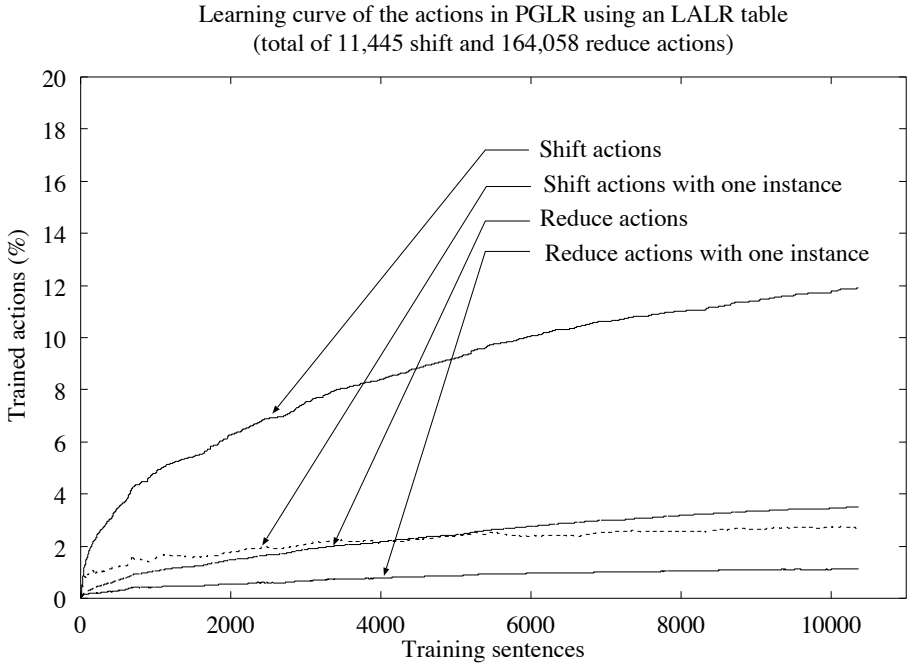
	2-42 Characters (534 sentences)						
Models	**PA**	**LP**	**LR**	**BP**	**BR**	**0-CB**	**m-CB**
PGLR(CLR)	95.13	99.04	98.40	99.46	98.61	97.57	0.04
PGLR(LALR)	95.32	99.06	98.47	99.53	98.73	98.50	0.03

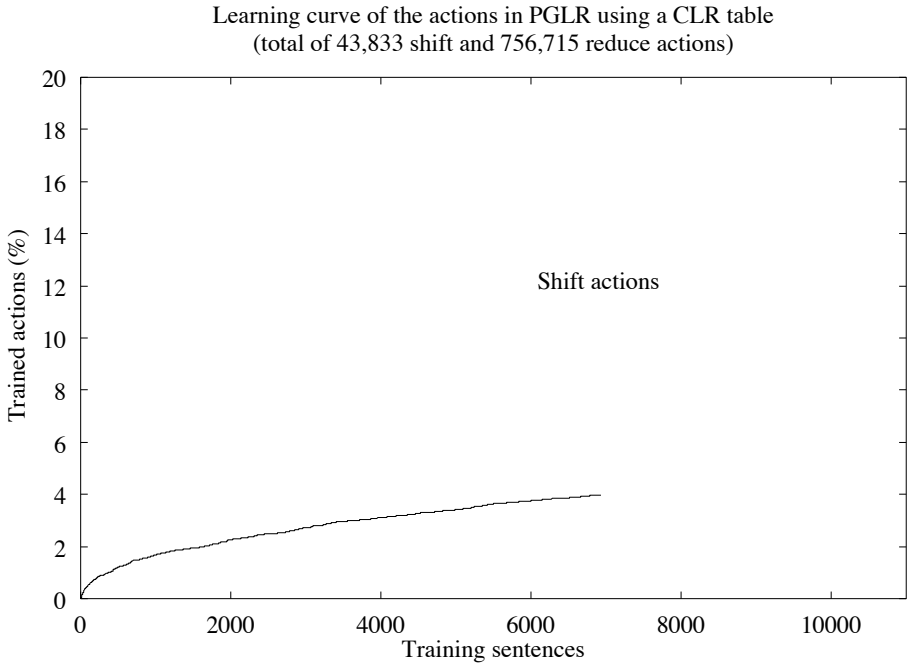
Based on the above, we may conclude that, empirically, PGLR using an LALR table can perform as well as one using a CLR table. State merging in generating an LALR table does not affect the parsing context a great deal. On the contrary, the drastic increase in states for the CLR table causes parameter sparseness. In PGLR, we distribute parse probabilities to each action appearing in the table, so that the number of training parameters is identical to the number of distinct actions in the LR table.

Compared with PGLR using a CLR table, the percentage of trained actions for that using an LALR table increases more significantly, as shown in Figure 2. The percentage of actions (both shift and reduce) in the LALR table observed in parsing the training set is around three times higher than in the CLR table. This means that actions in the LALR table are trained more effectively than those in the CLR table. Although the number of trained actions tends to increase as the number of training sentences is increased, the number of actions observed only once in training becomes saturated at an early stage of training, with most of the observed actions repeated in the expansion of the training set. Perhaps most of the actions in the LR tables are not really used, though they are allowed under the original context-free grammar; this is especially the case for reduce actions with a very low learning curve. It may be possible to generate a more efficient LR table if we can drop the actions that cause the parser to generate practically unacceptable parses. However, it is beyond the scope of this paper to discuss this possibility.

Figure 3 explicitly shows that with a small training set, the results when using an LALR table are better than when using a CLR table. Training the model with only 1/8 of the training set, or about 1,250 sentences, PGLR(LALR) returned a parse accuracy as high as 90%, while PGLR(CLR) needed more training sentences to catch up with the performance of PGLR(LALR).

In the sense of context-sensitivity, states in a CLR table are more distinguishable than


Learning curve of the actions in PGLR using an LALR table
(total of 11,445 shift and 164,058 reduce actions)
Shift actions
Shift actions with one instance
Reduce actions
Reduce actions with one instance
Trained actions (%)
Training sentences
0
2
4
6
8
10
12
14
16
18
20
0
2000
4000
6000
8000
10000


Learning curve of the actions in PGLR using a CLR table
(total of 43,833 shift and 756,715 reduce actions)
Shift actions
Trained actions (%)
Training sentences
0
2
4
6
8
10
12
14
16
18
20
0
2000
4000
6000
8000
10000