A New Probabilistic LR Parsing

Virach Sornlertlamvanich, Kentaro Inui, Hozumi Tanaka and Takenobu Tokunaga
Department of Computer Science, Tokyo Institute of Technology
{virach,inui,tanaka,take}@cs.titech.ac.jp

1 Introduction

The probabilistic approach has been introduced to various kinds of natural language processing tasks because the available text corpora are increasing in recent years. In syntactical parsing, the probabilistic approach is introduced for ranking the resultant parses which are generated numerously when handling a natural language.

Several attempts have been made to prune the meaningless parses and to aid in choosing the most likely parse from the huge outputted parses. Fujisaki et al. [3] introduced probabilistic context-free grammar (P-CFG) whose probabilities are trained in the Forward/Backward manner. Wright et al. [4] formalized a way to include the P-CFG into LR parsing table by distributing the probabilities originally associated with the CFG rule to each LR parsing action. As a result, the parser can compute the probability at each step of the parse. The resultant probability of a parse in Wright et al.'s model is identical to the one acquired from the original P-CFG while the process of generating LR parsing table becomes much more complicated.

Briscoe and Carroll [2] proposed a simpler way to incorporate the trained probability into each parsing action of the LR parsing table. The probability is directly computed from the count of action employed during parsing the training text corpora. Their method seems to be able to exploit the advantage in context-sensitivity property of the LR parsing algorithm. The LR parsing algorithm has the context-sensitivity in the way of its grammar reduction depending on the lookahead symbol. But, without the formalization of their method, it is dubious in the way of their including the left context for reduce action aiming to increase the accuracy in computing the probability of the parse.

The LR parsing algorithm has context-sensitivity in the parsing process to some extent. The states in canonical LR (CLR) parsing table are created individually according to the lookahead symbols. The LR parsing table implicitly includes the information of left and right contexts to the reduce action, which will distinguish the context in applying the grammar rule.

In this paper, we formalize probabilistic LR language model for statistical parsing, and review Briscoe and Carroll's model in terms of our formalization. Finally, the experiments comparing the performance of our model with the Briscoe and Carroll's model and P-CFG are conducted.

2 Language Modeling for Probabilistic LR Parsing

Suppose we have a CFG and its corresponding CLR parsing 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 CLR parsing table, respectively. For each state $s \in \boldsymbol{S}$, the CLR parsing table specifies a set of possible input symbols: $La(s) \subseteq \boldsymbol{V}_t$. And, 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, we can regard each parse derivation as a sequence of transitions between LR parse stacks. Let us consider a stack transition sequence T as (1):

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

where σ_i is the i-th stack, whose stack-top state is denoted by $top(\sigma_i)$, and l_i and a_i are, respectively, a input symbol and a parsing action chosen at σ_{i-1}. 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 $next(\sigma_i, a_{i+1})$ $(= \sigma_{i+1})$ can always be uniquely determined. 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 $st_n = final$, where *final* is a dummy denoting the stack when parsing is completed. Hereafter, we consistently refer to an LR parse state as a *state* and an LR parse stack by a *stack*. And, unless defined explicitly, s_i denotes the stack-top state of the i-th stack σ_i, i.e. $s_i = top(\sigma_i)$.

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

$$T = \{\Sigma, L, A\} \tag{2}$$

Given an input of word sequence (W), 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 the stack transition, the scaling factor α_W in equation (5) can be omitted. In addition, the word sequence (W) is assumed to be independent of everything else other than the sequence of input symbol (L).

The first term is called *LR parsing action probability* and the second term is called *lexical probability*. Considering the LR parsing action probability, the probability of a complete stack transition T can be decomposed as follows:

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

Every parse derivation starts from the initial state (s_0), therefore,

$$P(\sigma_0) = P(s_0) = 1 \tag{9}$$

According to Markov assumption, the stack in the parse process at time t_i depends only on the state at time t_{i-1}. Therefore, we here can simplify the equation (8) to,

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

Furthermore, we decompose the above probability $P(\sigma_i, l_i, a_i|\sigma_{i-1})$ into what we can estimate individually.

$$\begin{aligned} &P(\sigma_i, l_i, a_i|\sigma_{i-1}) =\\ &P(l_i|\sigma_{i-1}) \cdot P(a_i|\sigma_{i-1}, l_i) \cdot P(\sigma_i|\sigma_{i-1}, a_i, l_i) \end{aligned} \tag{11}$$

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

The first term $P(l_i|\sigma_{i-1})$ can be estimated as follows.
If $i = 1$, then

$$P(l_1|\sigma_0) = P(l_1|s_0) \tag{12}$$

If the previous action is a shift action then,

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

If the previous action is a reduce action, the input symbol for the current state does not change $(l_i = l_{i-1})$, therefore,

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

For the second term $P(a_i|\sigma_{i-1}, l_i)$, 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.

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

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

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

In concluding, only the first term of equation (11) is differently estimated according to the previous action. On the other hand, the estimation of the probability must be separately considered in two cases depending on the type of

state, namely the state reached immediately after a shift action ($\boldsymbol{S}_s$, including the initial state s_0) and the state reached immediately after a reduce action ($\boldsymbol{S}_r$). 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 parsing table, according to equation (17). To be more specific, for each state $s \in \boldsymbol{S}_s$, we associate a probability $p(a)$ with each action $a \in Act(s,l)$ (for $l \in La(s)$), 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 $p(a)$ with each action $a \in Act(s,l)$ (for $l \in La(s)$), where $p(a) = P(a|s,l)$ such that:

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

Through assigning probabilities to actions in an LR parsing 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:

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

Now, let us consider the second term in equation (6). The *lexical probability* can be decomposed and estimated by assuming that the probability of the current word depends only on its part-of-speech[1] shown in 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 when the previous action is a shift action only. 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 in the estimation of the *LR parsing action probability*, the *lexical probability* is then can be defined as,

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

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 between their model and ours.

We consider the probabilities of transitions between stacks as given in equation (10), whereas Briscoe and Carroll consider the probabilities of transitions between *LR parse states* as below:

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

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

Briscoe and Carroll initially associate a probability $p(a)$ with each action $a \in Act(s,l)$ (for $l \in La(s)$) in an LR parsing 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:

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

In this model, for any state, the probability associated with each action is normalized in the same manner. 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 difference, 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 required 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 have only

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

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 next stack after applying any action can be uniquely determined as in (16).

4 Experiments

We use a grammar for the Japanese inter-phrase construction. Together with this grammar there is also a considerable large enough bracketed corpus (a part of EDR corpus) for statistical training and the evaluation.

4.1 Grammar and Corpus

The grammar consists of 11 terminal symbols and 13 non-terminal symbols. The terminal symbols represent the Japanese phrase types which are subcategorized by the type of phrase dependency. The grammar[2] is the set of production rules for constructing the inter-phrase structure. From this grammar, we generate a CLR parsing table whose number of state is 280.

There are about 20,000 bracketed sentences. We randomly divided the corpus into 2 groups, 95% are used for training and 5% are used for testing. The size of the corpus is shown in the Table 1. The length of sentences are well distributed in both groups.

Corpus	*# of Sents.*	*Ave.*	*Range*
Training	20,470 (95%)	8.28	3-26
Testing	1,077 (5%)	8.27	3-25

Table 1: Corpus

4.2 Results

We trained the amount of 20,000 training sentences according to each model of P-CFG, Briscoe and Carroll (B & C), and our P-LR. Estimation of the free parameter, the Briscoe and Carroll's model has the largest number because a reduce action has to be divided according to all reachable states while for the P-CFG model has the smallest number. We compute the probabilities for the P-CFG model equivalent to the product of all employed grammar rules. Since our bracketed sentences have no ambiguity in the bracketing, it is more meaningful to evaluate the performance by counting the percentage of exact match of the candidate parse to the standard parse rather than the crossing bracket measurement.

Model	*P-CFG*	*B & C's*	*P-LR*
Exact-1	43.6%	33.5%	53.2%
Exact-5	74.3%	70.2%	83.7%
Exact-10	83.8%	80.4%	90.2%
Exact-20	88.6%	87.2%	93.8%
Perplexity	7.15	11.23	2.24

Table 2: Model Performance

The Table 2 shows that our model (P-LR) gives the highest percentage of accuracy in every range of ranking and the smallest perplexity per state transition. Though the Briscoe and Carroll's model requires the largest number of free parameter for training, the performance is apparently lower than the simplest P-CFG model.

5 Conclusion

We formalized the probabilistic model for GLR parsing and exhibited the method in constructing the probabilistic LR parsing table. Our model requires just a sequence of state transition of the parse to compute the probability. Therefore, we need only to associate the probability to each action in the LR parsing table. In the future task, we plan to apply our model to LALR parsing table which is more cost effective parsing table comparing to the CLR parsing table.

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] 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.

[2]The grammar is provided by Shirai, K. at TIT