A Node-Driven Parse Pruning Technique for Probabilistic GLR Parsing

Virach Sornlertlamvanich
National Electronics and Computer Technology Center
22F Gypsum Metropolitan Tower, 539/2 Sriayudhya Rd.,
Rajthevi, Bangkok 10400, Thailand.
virach@links.nectec.or.th

ABSTRACT — This paper proposes a new technique, *a node-driven parse pruning technique*, in pruning the less probable parses for GLR parsing algorithm. Without decreasing the efficiency of GLR parsing, this technique estimates the number of parses in the GSS (graph-structured stack) based on the number of expanded nodes during the parse process. We show the evaluation results of various beam width settings for pruning, and compare the parse time and space consumption against full parsing results. Our node-driven parse pruning algorithm allows pruning in a left-to-right manner without modifying the GSS.

KEY WORDS — Probabilistic GLR parsing, parse pruning, GSS, beam search.

บทคัดย่อ —บทความนี้เสนอวิธีการในการขจัดผลการแจงส่วนที่มีความเป็นไปได้ต่ำออกไป, สำหรับการใช้งานในการแจงส่วนแบบ GLR. วิธีการนี้ได้อาศัยจำนวนขั้นของการเคลื่อนสถานะมาเป็นตัวประมาณจำนวนของแขนงของการแจงส่วนใน GSS, จึงเรียกวิธีนี้ว่า "การขจัดผลการแจงส่วนโดยอาศัยจำนวนโนด". เราได้ทำการทดลองเพื่อเปรียบเทียบการใช้เวลาและหน่วยความจำสำหรับการขจัดผลการแจงส่วนในเกณฑ์ต่างๆ กับการแจงส่วนปกติ. วิธีการนี้มีข้อดีอีกอย่างหนึ่งคือสามารถคำนวณได้ตามการแจงส่วนจากซ้ายไปขวาโดยไม่ได้เปลี่ยนแปลงโครงสร้างของ GSS ซึ่งมีประสิทธิภาพของการเก็บข้อมูลของการแจงส่วนที่ดีอยู่แล้ว.

คำสำคัญ — การแจกส่วน GLR ด้วยความน่าจะเป็น, การขจัดแขนงของการแจงส่วน, GSS, การสืบค้นแบบบีม.

1 Introduction

Pruning is an essential paradigm to reduce the search space in parsing. The idea of pruning is to exclude hypotheses from further investigation if the parses turn out to be unlikely, based on evaluation of partial data. The efficiency of pruning technique depends mostly on the parsing paradigm which ranks the parse candidates according to their likelihood. The preciseness in distributing the parse probabilities into each partial parse tree is essentially considered for applying the pruning technique. This pruning technique is generally proposed for any probabilistic parsing models applied to the GLR parsing algorithm. However, the probabilistic GLR parsing model [9, 11] is introduced to eveluate the pruning technique because of its efficiency over all other models on GLR parsing as reported in [10]

Extracting all possible parses from a packed parse forest is crucially constrained by both time and memory space concerns. Carroll and Briscoe [2] have proposed a method for extracting the n-best parses from a complete packed parse forest. However, their method still requires the parser to parse to completion, and then identifies the n-best parses from the resultant packed parse forest. Their method can only save time in the actual extraction of the n-best parses, and does not concern itself with time and memory space consumption during the parse process. Sentences, in general, are ambiguous because of the wide-coverage nature of context-free grammars, but most of the possible parses for a sentence are pretty senseless. In practice, it is not reasonable to parse exhaustively to obtain some of the most probable parses. We would obtain the results more quickly if it were possible to prune off the less probable parses as early as possible.

The compaction of the graph-structured stack (GSS) prevents us from applying the Viterbi algorithm [13] directly to GLR parsing. By way of the GSS, the parse stack is dynamically changed and does not keep trace of the various parses. Both of the Viterbi beam-search methods proposed in [14, 15] and [16] need additional storage to keep trace of the parses, which overrides the benefits of using GSS in GLR parsing.

Lavie and Tomita [7] introduced a beam search heuristic for GLR* parsing. GLR* parsing is a noise-skipping parsing algorithm which allows shift operations to be performed from inactive state nodes of the GSS. This amounts to skipping words at any previous state in the GSS. The purpose of introducing the beam search algorithm is to limit the number of inactive state nodes for performing shift operations. The algorithm simply considers performing shift operations from the nearest state nodes until the number of state nodes reaches the limit. In fact, any state nodes may be merged and be common to several different sub-parses. Therefore, any undesirable less-probable parses that end up with the same state nodes are included within the beam, which leads to an inefficient beam search. It is also possible that the most probable parses be overlooked because of inaccuracy in scoring and parse estimation.

Pruning with a beam search technique is widely discussed in the speech processing community [12, 5, 8, 16]. Steinbiss et al. [12] give a good summary of previous research on beam search methods and propose some improvements to beam searching, as *histogram pruning*. This method introduces an additional pre-specified upper limit on the number of active points per frame (or active nodes per time frame) to limit the expansion of hypotheses. The result of their experiments show that the search space is efficiently reduced by observing the distribution of the number of states over the parse.

In this paper, we propose a new method for pruning parses that have a lower probability than parses within a predetermined beam width using a histogram pruning-like algorithm called the *node-driven parse pruning algorithm*. The number of expanded nodes is effectively used to estimate the number of parses in the GSS. We also show the evaluation results of various beam width settings, and compare the parse time and space consumption against full parsing results. Our node-driven parse pruning algorithm allows pruning in a left-to-right manner without modifying the GSS.

2 Probabilistic GLR Parsing

The probabilistic GLR language model (PGLR) has previously been proven to be better than existing models, in particular the model proposed by Briscoe and Carroll [1] and the baseline model using a probabilistic context-free grammar (PCFG), in parsing strings of parts-of-speech (non-word-based parsing) [10]. 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 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 [4] or "pseudo context-sensitive grammar" model (PCSG)—recently presented in [3]—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. [6] 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.

Considering a parse derivation as a sequence of transitions between LR parse stacks (T) and assuming that the current stack σ_i contains all the information of its preceding parse derivation, PGLR defines the probability of a complete stack transition as:

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

where l_i is an input symbol and a_i is an action.

The PGLR model distributes the probability of a complete stack transition into each transition by assuming that the stack-top state (s_i) represents the stack information beneath it, then:

$$P(l_i, a_i, \sigma_i | \sigma_{i-1}) \approx \begin{cases} P(l_i|s_{i-1}) \cdot P(a_i|s_{i-1}, l_i) & \\ \quad = P(l_i, a_i|s_{i-1}) & (s_{i-1} \in S_s) \\ P(a_i|s_{i-1}, l_i) & (s_{i-1} \in S_r) \end{cases}$$

such that:

$$\sum_{l \in La(s)} \sum_{a \in Act(s,l)} p(a) \qquad (\text{for } s \in S_s)$$

$$\sum_{a \in Act(s,l)} p(a) \qquad (\text{for } s \in S_r$$

where $p(a)$ is the probability of an action a, S_s is the class of states reached after applying a shift action, including the initial state, and S_r is the class of states reached after applying a reduce action.

Therefore, the parse probability upto the current state can be calculated and stored in the current stack-top state.

3 The node-driven parse pruning algorithm

It is inefficient to compute parse probabilities for all parses from the initial state successively up to the current state of parsing, because we have to keep trace of all possible parses individually. This also degrades the benefits of local ambiguity packing in the graph-structured stack (GSS). To counter this inefficiency, we observe the number of state nodes at each stage of parsing time and compute all possible parses only when the number is more than a threshold. Since the number of state nodes in a GSS can be viewed as an indicator of the degree of ambiguity, we indirectly estimate the number of parses by observing the number of state nodes in the GSS, and apply this as a threshold for activating the parse pruning process as shown in **Algorithm-1**.

The threshold T_t at time t is computed by:

$$T_t = G_t \cdot n_t \qquad (2)$$

where T_t is the estimated number of parses at time t, G_t is the gain based on the number of state nodes and the length of the input string up to time t, and n_t is the number of state nodes at time t.

The gain G_t can be computed by:

$$G_t = \frac{\sum_{i=1}^{L} n_{t-i} T_{t-i}}{\sum_{i=1}^{L} n_{t-i}^2} \qquad (3)$$

where L is the number of past observations (a good setting for L is 5, as reported in [5]). The gain is used in adaptive pruning [5] by regarding a pruning process as a non-linear time-variant dynamical system. In our implementation, we simply set the gain G_t as a linear time-variant to reduce the computational overhead. Since our beam width is fixed, the estimated number of partial parses at each parsing time is used to activate the parse pruning algorithm.

Algorithm-1. Node-driven parse pruning process.

1. *If the number of parses estimated from the number of state nodes in the GSS is over the threshold T_t, compute the number of partial parses, else return.*
2. *If the number of parses is greater than the predetermined beam width, then compute the probabilities for each partial parse and individually store the sequences of state transitions with corresponding probability at each active state node (top node of each stack), else return.*
3. *Sort the sequences of state transitions according to the probabilities and determine the minimum probability of the parse within the beam width.*
4. *Mark sequences of state transitions that have probabilities less than the minimum as 'pruned'.*
5. *Apply the next actions to the active state nodes only if there is at least one possible parse (unmarked sequence of state transitions). For reduce actions, check reduceable parses with the sequences of state transitions at the active state nodes.*

Figure 1 exemplifies the GSS in the process of parse pruning. Suppose that the beam width is equal to one, and the first parse (state sequence (0,3,6,13,9)) at node 9 is the only parse within the beam width. Here, only the action [rel,14] is executed, with the result as shown in Figure 1 (b). Note that we do not extend the stack at active state node 13 because all parses up to state node 13 are marked to be pruned off. At active state node 17, after trying the action [rel,17], the state sequence (0,4,11,3,9) which is marked to be pruned off, is activated. Therefore, the parse after this action is also disregarded.

As a result, we can parse with smaller memory space and lower computational time overhead than that required in full parsing. However, parsing with this pruning technique gives appropriate results if and only if the exploited probabilistic model provides precise probabilistic estimates for the partial parses.

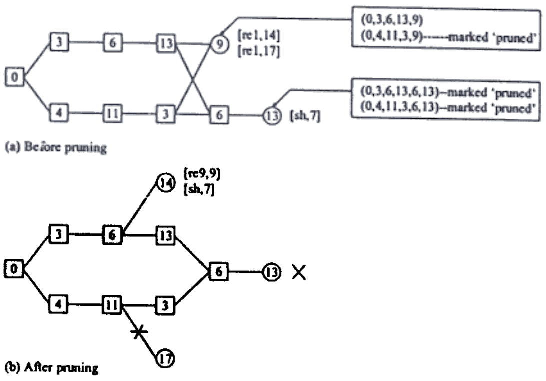
(0,3,6,13,9)
(0,4,11,3,9)——marked 'pruned'
[re1,14]
[re1,17]
(0,3,6,13,6,13)--marked 'pruned'
(0,4,11,3,6,13)--marked 'pruned'
[sh,7]
(a) Before pruning
[re9,9]
[sh,7]
(b) After pruning

Figure 1: Parse pruning within a graph-structured stack. Circled state numbers are active nodes. All possible parses (sequences of state transitions) at each active node are shown in the box pointing to that node. The action/state pairs after applying the actions are shown in the square brackets next to the active nodes.

This is because the beam search is an approximate heuristic method that does not guarantee that the interpretation of a sentence is the best possible interpretation.

4 Efficiency of parse pruning in PGLR

We calculated the efficiency of parse pruning using the PGLR model for a varying beam width. Parsing can be sped up by reducing the beam width, excepting that the correct parses can potentially be pruned off if the beam width is too small. Figure 2 shows that PGLR provides quite a precise probabilistic estimate for partial parses, in that the parsing accuracy increases steeply with a small increase in the beam width. The parser performs equally well with a beam width of around 30 as with full parsing. Time consumption in parsing using our pruning technique is linear in sentence length, while it is exponential for full parsing. For example, our pruning technique requires only $\frac{1}{10000}$ of the parsing time required for full parsing, for 25 character long sentences (there are more than 200,000 parses if completely parsed).

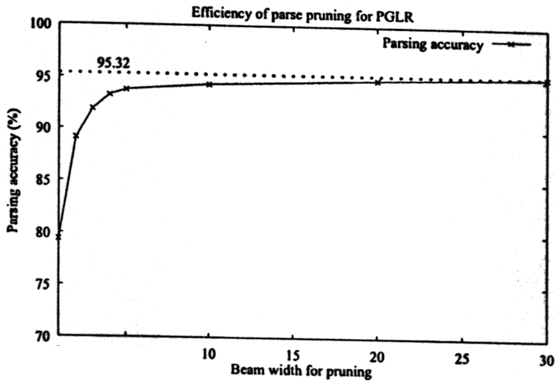

Figure 2: Parsing accuracy under a varying beam width for parse pruning.

We evaluated our pruning algorithm by observing time and space consumption between full parsing and parsing with the pruning algorithm at the beam width of 30. Distributions of state nodes against input symbols in single sentence, are shown in Figures 3 and 4. It is obvious that our parse pruning algorithm can drastically reduce the number of nodes used in parsing both sentences considered. Consequently, the parsing time for both sentences is also visibly reduced because of the reduction in search space. Parsing time is reduced from 1709.62 seconds to 1.0 seconds and 649.49 seconds to 5.52 seconds, in parsing the 33 and 36 character length sentences, respectively.

Table 1 shows the average efficiency of time and space consumption when parsing with the node-driven parse pruning algorithm at a beam width of 30, as compared to full parsing.

The setting for the beam width is a trade-off between parsing accuracy and parsing time. In practice, a beam width of around 20 is likely to be sufficient to produce satisfactory parsing results for the ATR corpus.

Table 1: Average time and space consumption when parsing with the node-driven parse pruning algorithm, as compared to full parsing.

	Average number of state nodes per sentence	Average parse time, seconds per sentence
Full parsing	9,146	163
Beam width = 30	630	0.243

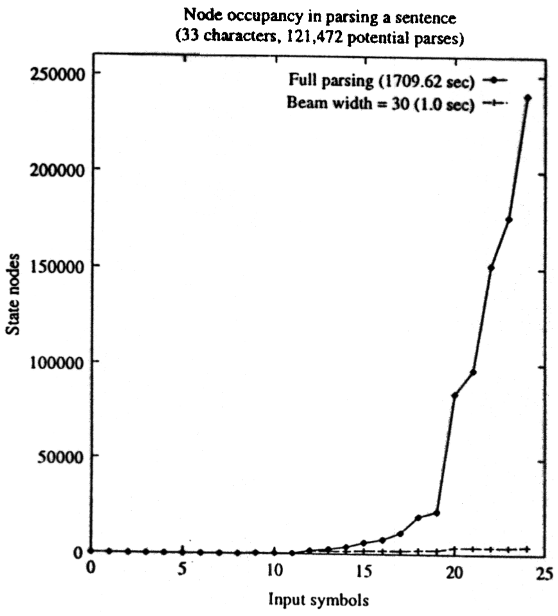

Figure 3: Distribution of state nodes over input symbols in parsing a sentence of 33 characters with 121,472 potential parses.

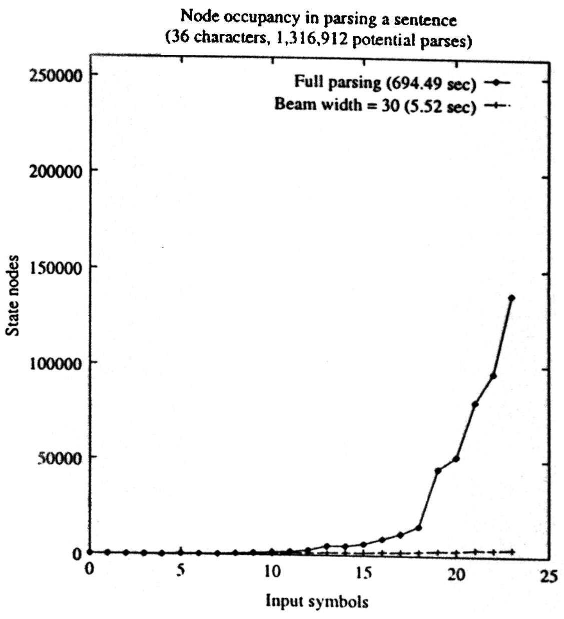

Figure 4: Distribution of state nodes over input symbols in parsing a sentence of 36 characters with 1,316,912 potential parses.

5 Conclusion

Beam searching is an approximate-based heuristic method that does not guarantee that the final interpretation of a sentence is the best possible interpretation. Nevertheless, by carefully managing the number of parses in the GSS using the *node-driven parse pruning technique* we make significant efficiency gains in both time and space consumption. Moreover, when coupled with a precise model for partial parse probability estimation, our pruning technique yields the same results as full parsing at a beam width of about 30, using about 0.07% of the relative parsing space and 0.0015% of relative parsing time over full parsing.

Our *node-driven parse pruning algorithm* allows parse pruning in the GSS in a strict left to right fashion, with the benefit of being able to disregard less-probable parses at as early a stage as possible.

Our pruning algorithm also retains the advantages of the GSS, with neglible cost in estimating the number of parses in the GSS.

References

[1] T. Briscoe and J. Carroll. Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. *Computational Linguistics*, 19(1):25-59, 1993.

[2] J. Carroll and T. Briscoe. Probabilistic Normalisation and Unpacking of Packed Parse Forests for Unification-Based Grammars. In *Proceedings of AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 33-38, 1992.

[3] E. Charniak and G. Carroll. Context-Sensitive Statistics for Improved Grammatical Language Models. In *Proceedings of AAAI-94*, pages 728 733, 1994.

[4] M. Chitrao and R. Grishman. Statistical Parsing of Messages. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 263-266, 1990.

[5] H. V. Hamme and F. V. Aelten. An Adaptive-Beam Pruning Technique for Continuous Speech Recognition. In *Proceedings of International Conference on Spoken Language Processing*, pages 2083-2086, 1996.

[6] K. Inui, V. Sornlertlamvanich, H. Tanaka, and T. Tokunaga. A New Formalization of Probabilistic GLR Parsing. In *Proceedings of the 5th International Workshop on Parsing Technologies*, 1997.

[7] A. Lavie and M. Tomita. GLR* - An Efficient Noise-Skipping Parsing Algorithm for Context-Free Grammars. In *Recent Advances in Parsing Technology*, chapter 10, pages 183-200. Kluwer Academic Publishers, 1996.

[8] S. Ortmanns, H. Ney, and A. Eiden. Language-Model Look-Ahead for Large Vocabulary Speech Recognition. In *Proceedings of International Conference on Spoken Language Processing*, pages 2095-2098, 1996.

[9] V. Sornlertlamvanich. *Probabilistic Language Modeling for Generalized LR Parsing.* Doctoral dissertation, Tokyo Institute of Technology, Tokyo, Japan, March 1998.

[10] V. Sornlertlamvanich, K. Inui, K. Shirai, H. Tanaka, T. Tokunaga, and T. Takezawa. Empirical Evaluation of Probabilistic GLR Parsing. In *Proceedings of the Natural Language Processing Pacific Rim Symposium*, pages 169 174, 1997.

[11] V. Sornlertlamvanich, K. Inui, H. Tanaka, T. Tokunaga, and T. Takezawa. Empirical Support for New Probabilistic Generalized LR Parsing. *Journal of Natural Language Processing*, 6(3):3 22, 1999.

[12] V. Steinbiss, B.-H. Tran, and H. Ney. Improvements in Beam Search. In *Proceedings of International Conference on Spoken Language Processing*, pages 2143–2146, 1994.

[13] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In *Proceedings of IEEE Transactions on Information Theory*, pages 260–269, 1967.

[14] J. H. Wright and E. N. Wrigley. GLR Parsing with Probability. In *Generalized LR Parsing*, pages 113–128. Kluwer Academic Publishers, 1991.

[15] J. H. Wright, E. N. Wrigley, and R. Sharman. Adaptive Probabilistic Generalized LR Parsing. In *Proceedings of the 2nd International Workshop on Parsing Technologies*, pages 100–109, 1991.

[16] T. Yamada and S. Sagayama. LR-Parser-Driven Viterbi Search with Hypotheses Merging Mechanism Context-Dependent Phone Models. In *Proceedings of International Conference on Spoken Language Processing*, pages 2103–2106, 1996.

Virach SORNLERTLAMVANICH:

is a researcher of the National Electronics and Computer Technology (NECTEC) of Thailand since 1992. He received the B.Eng. and M.Eng. degrees from Kyoto University, in 1984 and 1986, respectively. From 1988 to 1992, he joined NEC Corporation and involved in the Multi-lingual Machine Translation Project supported by MITI. He received the D.Eng. degree from Tokyo Institute of Technology in 1998. His research interests are natural language processing, lexical acquisition and information retrieval.