Journal of Computer Science 3 (1): 9-13, 2007
ISSN 1549-3636

Packed Forest Chart Parser

Qi Haoliang, Li Sheng, Yang Muyun, Zhao Tiejun
Ministry of Education, Microsoft Key Laboratory of Natural Language Processing and Speech
Harbin Institute of Technology, China

Abstract: Packed forest chart parser, like the traditional chart parsing algorithm, uses charts to store all the information produced while parsing to avoid redundant works. Its advantage over the traditional chart parser was the packed forest representation. The algorithm not only shares the non-terminal categories as what was done in the shared parse forest, but also shares the leftmost common elements. The number of active edges in packed forest chart parser was decreased, so memory requirement was reduced and parsing was speeded up. The effectiveness of our approach has been evaluated on Chinese parsing. Results show that packed forest chart parser significantly outperforms packed chart parser, with the former 10 times faster than the latter.

Key words: Parsing algorithm, packed forest chart parser, active edge

INTRODUCTION

Context Free Grammar (CFG) parsing algorithms might produce many parsing results, out of which one wants to extract the most plausible parsing results for the semantic processing. For many years probabilistic CFG (PCFG), which is a CFG with probabilities added to the rules, has been used to ranking parsing results using scores based on statistical information. PCFG is also called stochastic context free grammar (SCFG). The parsing algorithms for PCFG stem from the parsing algorithm for CFG. CYK algorithm[1], Earley algorithm[2] and Generalized LR (GLR) algorithm (Tomita algorithm)[3,4] have been adapted to PCFG. Chart parser, a classical parsing algorithm, is evolved into the packed chart parser[5].

The number of possible parse trees may become very large when the size of sentences increases: it may grow exponentially with the size[6]. Since it is often desirable to consider all possible parse trees (e.g. for semantic processing), it is convenient to merge as much as possible these parse trees into a single structure that allows them to share common elements. This sharing saves the needed space to represent the trees and also on the later processing cost of these trees since it may allow to share between two trees the processing of some common elements. The shared representation of parse trees is called the shared parse forest, or just the parse forest[7]. The drawback of the shared parse forest is that only nonterminal category is shared. Chart parser is a type of shared parse forest algorithm. Shann also points out that the deficiency of chart parser is its parsing tree representation[8].

Packed Forest Chart Parsing (PFCP) algorithm is proposed for natural language parsing in this paper. The algorithm not only shares the nonterminal categories as what is done in the shared parse forest, but also shares the leftmost common elements. The presentation of an active edge is improved, which can present lots of active edges in traditional definition. Because active edges are the vast majority[9], our algorithm heavily reduces the memory requirement and speeds up parsing.

We take the running time as the metric to evaluate the algorithm because it can provide the real performance of the algorithm. Our approach is evaluated on Chinese parsing. The results show that the proposed algorithm is about 10 times faster than the packed chart parsing algorithm. The proposed algorithm also decreases the memory requirement because of its packed forest representation.

PACKED FOREST CHART PARSING ALGORITHM

Chart parsing algorithm uses a data structure called chart as a book-keeping storage for all the information produced while parsing. The information in the chart is used to avoid redundant works, which comes the parsing efficiency.

Allen put forward the packed chart parser for PCFG[5]. A packed chart representation stores only one constituent of any type for the same input. Packing provides quite efficient representations of chart presentations without information loss in PCFG.

The key point here is to improve the parsing efficiency without pruning, which will cause some information loss. The speeds of chart parser and the packed chart parser are very slow when processing natural language without pruning. A great number of parsing trees should be created due to the ambiguity of natural language. Creating trees and computing their probabilities lead to huge computation and memory burden. PFCP presented in this paper packs the parsing tree and speeds up parsing.

Corresponding Author: Haoliang Qi, Box 321, Harbin Institute of Technology, Harbin, P.R. China, 150001, Tel: +86045186402448, Fax: +86045186402448

The following of this section describes PFCP algorithm. Firstly, we introduce the basic concepts, terms used in packed chart parser; secondly, PFCP algorithm is described in details; finally, a parsing example is illustrated to make the algorithm more clearly.

Concepts and terms: We often use a parse tree to describe the structure when parsing a sentence. Figure 1 shows the parse tree of the sentence "the old man the boat". Figure 2 shows the chart corresponding to this sentence.

The symbols used in the following are described in this paragraph. Greek symbols α, β and γ are a sequence of terminal and/or nonterminal symbols and capital letters A, B, C are nonterminal symbols. S is a special symbol representing the parsing goal symbol, i.e. the start symbol(root symbol). X is a nonterminal variable. Lowercase w_i is the *ith* terminal symbol. For a parsing rule (production rule) $P \rightarrow Ch_1\ Ch_2 \ldots Ch_n$, P is called the parent of the elements in the right part (i.e. $Ch_1\ Ch_2 \ldots Ch_n$); and $Ch_1\ Ch_2 \ldots Ch_n$ are the children of P.

The information kept in the chart is divided into a set of active edges and a set of inactive (complete) edges. An inactive edge represents a constituent that has been completed. An active edge represents a constituent with some elements called remainder children left to be satisfied. To make elements in charts clear, we label the terminal symbols with position numbers. Table 1 shows the example of the sentence with position numbers. The word "old" starts at position 1 and ends at position 2 in the example sentence.

An active edge has the format [X→α,i,j,(applied parsing rules),the number of first remainder child], where

* "X" represents the parents which can be not assigned at current. The parents can be assigned only after all children are satisfied according to the parsing rules. Note that, there are lots of rules whose children or parts of children are the same;
* "α" is a sequence of terminal and/or nonterminal symbols which are satisfied;
* "i" and "j" refer to position i and position j. The words spanning position i and position j are satisfied;
* "applied parsing rules" indicates the parsing rules applied in the active edges;
* "the number of the first remainder child" is the number of the first child to be stratified.

It is easy to get the first remainder child by using "applied parsing rules" and "the number of the first remainder child". For example, for an active edge[X→NP,0,2,(VP→NP ADJP VP, S→NP VP, S→NP VP PP,),2], NP is satisfied, which spans position 0 and position 2. For the sentence in the Fig. 1, NP contains "the" and "old"; "applied parsing rules" includes 3 parsing rules VP→NP ADJP VP, S→NP VP and S→NP VP PP. The remainder children can be gotten by applied parsing rules and the number of first

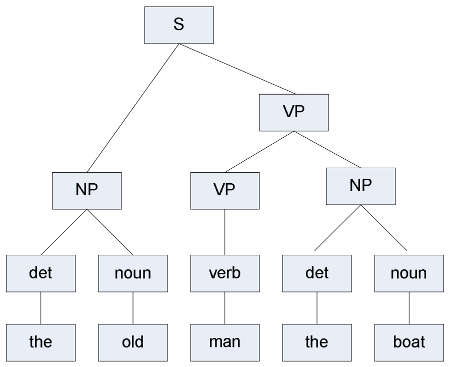
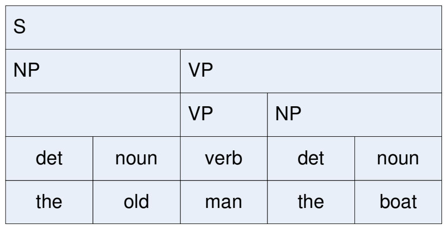


Fig. 1: Example of parsing tree



S				
NP		VP		
		VP	NP	
det	noun	verb	det	noun
the	old	man	the	boat

Fig. 2: The example of the chart in the parsing tree

Table 1: The example of the sentence with position numbers

	the		old		man		the		boat	
0		1		2		3		4		5

remainder child. For the rule VP→NP ADJP VP, the remainder children are ADJP VP, the first remainder child is ADJP; For the rule S→NP VP, the first remainder child is VP (the only one remainder child is VP); For the rule S→NP VP PP, the remainder children are VP PP, the first remainder child is VP. Note that the definition of an active edge in this paper is different to traditional one, which should identify all remainder children.

The rules, whose first child is B, can be represented by X→B β. The rules with the same left most common elements α can be represented by X→α β.

The traditional active edge form is [A→α·β,i,j] where the dot "·" is used to separate the satisfied children α and the remainder children β; the left to the dot "·" are satisfied and the right are not satisfied. There are three active edges if using traditional definition for the active edge in the above example, which are [VP→NP·ADJP VP,0,2], [S→NP·VP,0,2] and [S→NP·VP PP,0,2]. Comparing our definition and traditional one, we can see that our definition is more flexible. In our definition, an active edge is a set, which can include lots of edges of traditional definition.

By the new definition of an active edge, we can share the leftmost common elements. So the number of active edges those should be created in parsing drops down. The efficiency of parsing is improved.

All the children in the inactive edge are satisfied. An inactive edge has the format [A→α·,i,j]. The dot "·" means that the left elements to the dot "·" are stratified.

The other symbols applied in the definition are same to the definition of the active edge. For example, [NP→det noun·,0,2] is an inactive edge, which uses the parsing rule NP→det noun and spans position 0 and position 2. For the sentence in Fig. 1, NP contains "the" and "old". All the charts in Fig. 2 are inactive edges.

Packed chart parsing algorithm: Assuming that there are the parsing rules $A_1 \rightarrow B\ \beta_1$, $A_2 \rightarrow B\ \beta_2$,... , $A_n \rightarrow B\ \beta_n$, if phrase B is generated, then the traditional chart parsing algorithm would create active edges [$A_1 \rightarrow B \cdot\beta_1$], [$A_2 \rightarrow B \cdot\beta_2$], ... , [$A_n \rightarrow B \cdot\beta_n$]. We should create one active edge for each parsing rule whose leftmost child is B. There are lots of such rules in natural language parsing, so lots of active edges are created. The similar phenomena happen in others steps when creating active edges. Until now, for these active edges, the satisfied children are the same. If we only record the satisfied children in the chart, muliti-edge can be presented by a single active edge. The recording method of an active edge has been illustrated in previous texts. By this way, the number of active edges is decreased and the parsing trees are packed. Packed forest chart paring algorithm is named according to this feature. The number of active edges in PFCP is decreased, so memory requirement is reduced and parsing is speeded up. The parsing efficiency is improved. At the same time, the memory that the algorithm used drops.

Now we will give a left to right bottom-up PFCP algorithm.

Input: the parsing rules which have been sorted by their children

a sentence (w_1,w_2,...,w_n)

output: the parsing tree

(a)For each entry C → w_{k+1}, span an inactive edge [C→w_{k+1} ·,k,k+1] between positions k and k+1.

Then for each inactive edge [B→ γ ·,j;k+1] between positions j and k+1(j<k+1), do the following until no new item can be created:

{

(b) For each rule A→B, span an inactive edge [A→B·, j,k+1].

(c) For all rules X→B β , i.e. the first child is B, span an active edge[X→B,j,k+1, (applied parsing rules), 2]. "applied parsing rules" includes all rules X→B β whose first child is B.

For each active edge starting at position i, ending at position j and having the first remainder child B, which can be gotten by applied parsing rules and the number of first remainder child, with the form [X→α,i,j,(applied parsing rules),m], do step d and e.

(d) If the active edge [X→α,i,j,(applied parsing rules),m] has only one remainder child B, create inactive edges [A→αB·,i,k+1] between positions i and k+1, the parent A is assigned according to the rule's children. Note that more than one inactive edge might be created in this step.

(e) If the active edge [X→α,i,j,(applied parsing rules),m] has more than one remainder children having the first remainder child B, create an active edge [X→α B, i, k+1, (new applied parsing rule set),m+1] between positions i and k+1. "new applied parsing rules" is all rules of "applied parsing rules" whose the first remainder child is B.

}

(f)If we find an edge of the form [S→α·,0,n], then accept, else reject.

In step d and e, the active edge [X→α,i,j,(applied parsing rules),m] having first remainder child B and ending at position j is processed. The new charts are created by combining the active edge [X→α,i,j,(applied parsing rules),m] and the inactive edge [B→ γ ·,j;k+1]. If there is no remainder children, inactive edges are created; otherwise, an active edge is created. For example, for an active edge [X→NP,i,j,(S→NP VP, S→NP VP PP, NP→NP VP),2] and the inactive edge [VP→ γ ,j,k+1], the inactive edges [S→NP VP, i, k+1] and [NP→NP VP, i, k+1] are created by applying step d and an active edge [X→NP VP,i,j,(S→NP VP PP),3] is created by applying step e.

Step c, d and e are different to the chart parsing algorithm. For step c, if having the rules $A_1 \rightarrow B\ \beta_1$, $A_2 \rightarrow B\ \beta_2$,..., $A_n \rightarrow B\ \beta_n$ and the phrase B with matching position, only one edges should be created in our approach; The chart parsing algorithm should create n edges. For step d, our algorithm may create more than one inactive edge from single active edge because one active edge in our algorithm presents multi-edges in traditional one and the chart parsing algorithm only creates one inactive edge for a single input inactive edge. For step e, for an active edge [X→α,i,j,(applied parsing rules),m] and an inactive edge [B→ γ ·,j;k+1], only one edge should be created in our approach but more edges should be created in traditional algorithm, which is similar to step c.

Parsing example: To understand the algorithm more clearly, let us consider parsing the input sentence "John likes Mary" using the following grammar. The input sentence is translated into the terminal sequence "n v n". The probability of the rules is omitted here.

[1] S → NP VP
[2] NP → n
[3] NP → v n
[4] VP → v n
[5] VP → v n PP

PFCP will proceed as Fig. 3. Contents in column "Charts (Chart ID)" are generated charts (i.e. active edges or inactive edges) during the parsing and their IDs. "Current input" is the input of this step. They are identified by w_i=POS and/or Chart ID. For example, for the first step the input is w_1=n and the output chart is [NP → n·, 0,1], whose ID is "1". The input ID of the second step is "1" (i.e. inactive edge [NP → n·, 0,1]) and the output is the active edge[X→ NP, 0,1,(Rule 1),2], whose ID is "2".

Step	Charts (Chart ID)	Current input	Applied parsing rule	Steps in algorithm
k=0				
	[NP → n·, 0,1] (1)	w_1=n	[2]	(b)
	[X→ NP, 0,1,(Rule 1),2] (2)	(1)	[1]	(c)
k=1				
	[X → v,1,2,(Rule 3,4,5),2] (3)	w_2=v	[3][4][5]	(c)
k=2				
	[NP → n , 2,3] (4-1) [NP → v n·,1,3] (4-2) [VP → v n·,1,3] (4-3) [X→ v n,1,3,(Rule 5),3] (4-4)	w_3= n w_3= n, (3) w_3= n, (3) w_3= n, (3)	[2] [3] [4] [5]	(b) (d) (d) (e)
	[X → NP, 2,3,(Rule1),2] (5)	(4-1)	[1]	(c)
	[S → NP VP·, 0,3] (6)	(1),(4-3)	[4]	(d)(f)
	[X → NP, 1,3,(Rule1),2] (7)	(4-2)	[1]	(c)

Fig. 3: The parsing example

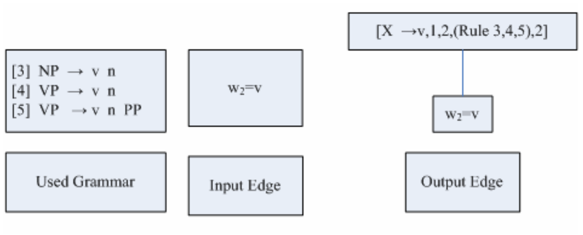


Fig. 4: The third step in the example

In the third step of this example, the algorithm only creates one active edge [X → v,1,2,(Rule 3,4,5),2] (Fig. 4). Three active edges should be created in the traditional chart paring algorithm, which are [NP → v· n, 1,2] [VP → v· n, 1,2] and [VP → v ·n PP, 1,2]. These 3 active edges have the same leftmost child w_2=v.

Our approach has advantage when processing the parsing rules, whose leftmost elements or the whole of their children are the same. PFCP shares the same leftmost partial text; it can decrease the number of inactive edges, so it is very efficient.

In the 4th step, two inactive edges [NP → v n·,1,3] and [VP → v n·,1,3] are created for the input edge w_3= n and the edge [X → v,1,2,(Rule 3,4,5),2] (chart ID is "3"). When creating the edge [NP → v n·,1,3], we apply Rule 3; and when creating the edge [VP → v n·,1,3], we apply Rule 4.

EXPERIMENTS AND RESULTS

We take the running time as the metric to evaluate the algorithm because it can provide the real performance of the algorithm. Our approach is evaluated on Chinese parsing. The corpus used in experiments is HIT (Harbin Institute of Technology) Chinese Treebank version 1.0, which include 8661 sentence. We randomly select 500 as test data and other sentences are used as training data. 4083 PCFG rules are acquired from training data. The test corpus includes 15568 bytes, about 7784 Chinese characters.

Table 2: Comparison two parsing algorithms

	# of Created Active Edges	Running Time (seconds)	Processing Speed (bytes/second)
Packed chart Parser	12,835,628	938	16.60
PFCP	1,401,536	82	189.85

We do word segmentation and POS tagging before parsing just as usual. Because there is no space to separate words, word segmentation is often the first step in Chinese text processing.

We compare PFCP to packed chart parser[5]. In packed chart parser, a packed chart representation stores only one constituent of any type over the same input and any others found are collapsed into the existing one. No pruning is done for two algorithms. The computer used in the experiments is HP Proliant server（Xeon 3G CPU, 1G MM, SCSI 134G）. The experimental results are shown in Table 2.

The efficiency of the PFCP algorithm comes from its packed presentation. Because active edges are the vast majority of edges and the number of active edges in packed forest chart parser is decreased, memory requirement is decreased and parsing is speeded up. The efficiency of the new algorithm is very high. The experimental results show that our approach is very efficient. The new algorithm is 10 times faster than packed chart parser.

We also test PCFG performance. The corpus used for open test includes 1000 sentence in HIT Chinese Treebank. PFCF algorithm and the packed chart parsing algorithm achieve the same effectiveness. The precision is 73.3%, recall is 72.3% and $F_{\beta=1}$=(2*precision*recall)/(precision + recall)=72.8%.

CONCLUSIONS AND THE FUTURE WORK

The packed forest chart parser is proposed in this paper. Its advantage over traditional chart parser is the packed forest representation. The algorithm not only shares the non-terminal categories as what is done in the shared parse forest, but also shares the leftmost common elements. The number of active edges in PFCP is decreased, so memory requirement is reduced and parsing is speeded up. The effectiveness of our approach has been evaluated on Chinese parsing.

Results show that the packed forest chart parser significantly outperforms the packed chart parser, with the former 10 times faster than the latter.

The representation of an active edge can be improved in the future. If an active edge shares any common elements, the parsing efficiency can be improved.

ACKNOWLEDGEMENT

This study was supported by the High Technology Research and Development Program of China (Grant No. 2004AA117010-08), National Natural Science Foundation of China (Grant No. 60373101), Pecan Information System Inc (now it is Cannon Research China) and Heilongjiang Outstanding Yong University Teacher(Grant No. 1151G037). Our thank also goes to Dr. Meng Yao in Fujitsu Research Center China and Dr. Lv Yajuan in Microsoft Research Asia.

REFERENCES

1. Fujisaki, T., F. Jelinek and J. Cocke et al., 1991. A probabilistic parsing method for sentence disambiguation. Tomita Med. Current Issues in Parsing Technology, International Work shop on Parsing Technologies. Pittsburgh, Boston, Kluwer Academic Publishers, pp:139-152.
2. Stolcke, A., 1995. An efficient probabilistic context free parsing algorithm that computes prefix probabilities. Computational Linguistics, 21: 165-201.
3. Wright, J.H., 1990. LR parsing of probabilistic grammars with input uncertainty for speech recognition. Computer Speech and Language, 4: 297-23.
4. Zhu S., M. Zhou, X. Liu and C. Huang, 1998. An efficient stochastic context-free parsing algorithm. J. Software, 9: 59-87 (in Chinese).
5. Allen, J., 1995. Natural Language Understanding. The Benjamin/Cummings Publishing Company, Inc. Sec. Edn.
6. Pereira, F.C.N. and D.H.D., 1983. Warren. parsing as deduction. Proc. 21st Ann. Meeting of the Association for Computational Linguistics. Cambridge (Massachusetts), pp: 137-144.
7. Bernard, L., 1991. Towards a uniform formal framework for parsing. Current Issues in Parsing Technology. Ed. Tomita M., Kluwer Academic Publisher, pp: 153-171.
8. Shann, P., 1991. Experiments with GLR and chart parsing. Intl. Work shop on Parsing Technologies (Tomita M. Ed.). Pittsburgh, Boston, Kluwer Academic Publishers, pp: 17-34.
9. Klein, D. and C. Manning, 2001. Parsing with treebank grammars: Empirical bounds, theoretical models and the structure of the Penn treebank. Proc. 39th Ann. Meeting of the ACL, Toulouse, France, pp: 330-337.