Probabilistic Language Modeling for GLR Parsing

(GLR %Q!<%6$r%Y!<%9$K7?3NN(E*8@8l%b%G%k!K

Virach Sornlertlamvanich
Department of Computer Science
Tokyo Institute of Technology
June 1998

Table of contents

Background

- Probabilistic parsing:
  - to aid in choosing/ranking for the most likely interpretation
  - to filter out meaningless parses

Background

- Various approaches in probabilistic parsing:
  Original PCFG—insufficient context

  - Chitrao and Grishman (90): Two-level PCFG—non-terminal's parent node.
  - Su et al. (91): Shift-reduce parsing framework—normalized at shift action.
  - Black et al. (92): History-Based Grammar (HBG)—{*Lex, Syn, Sem, Str*} information of parent node.
  - Magerman et al. (95): Statistical decision-tree on Chart, CKY.
  - Charniak (97): Head-word context, lexicalization.
  - etc.

⇒ Originated from PCFG.

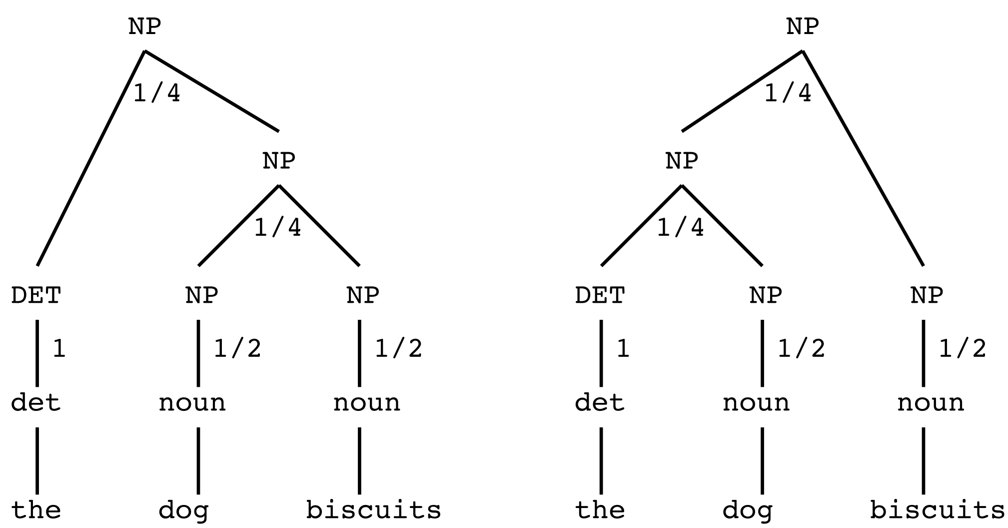
⇒ Extended to include more contextual information.

⇒ Modeled independently from the parsing algorithms.

Probabilistic CFG

- Context-Free Grammar with the probabilities:

(1)	NP	$\rightarrow$	NP	NP	(1/4)
(2)	NP	$\rightarrow$	DET	NP	(1/4)
(3)	NP	$\rightarrow$	noun		(1/2)
(4)	DET	$\rightarrow$	det		(1)



(a) The dog biscuits...   (b) The dog named "Biscuits"...

Background

- Probabilistic models in the GLR parsing framework:
  - Wright and Wrigley (91): identical to PCFG
  - Goddeau and Zue (92): input symbol prediction at each state
  - Briscoe and Carroll (93): action probabilities
  - Li et al. (96): pre-terminal bi-gram constraints
  - etc.

⇒ Inherit the efficiency of GLR parsing.

⇒ Use the provided contextual information within the GLR parsing framework.

Aims of this research

- Construct and verify probabilistic language models for the GLR parsing framework.
  - Evaluate the PGLR model against the existing Briscoe & Carroll (B&C) and Two-level PCFG models.
  - Analytical discussion on the experimental results. How the models reflect language phenomena.
  - Model trainability and tractability, PGLR(LALR) vs PGLR(CLR).
- Parse pruning technique.

GLR parsing

- A table-driven shift-reduce left-to-right parser for context-free grammars, constructing a rightmost derivation in reverse.

$$action_{i+1} = [state_i, symbol_{i+1}]$$

- Configurations:

	stack	input
current configuration:		
	$(s_0 X_1 s_1 X_2 s_2 \cdots X_m \underline{s_m},$	$\underline{a_i} a_{i+1} \cdots a_n \$)$
shift action:		
	$(s_0 X_1 s_1 X_2 s_2 \cdots X_m s_m a_i s,$	$a_{i+1} \cdots a_n \$)$
reduce action:		
	$(s_0 X_1 s_1 X_2 s_2 \cdots X_{m-r} s_{m-r} A s,$	$a_i a_{i+1} \cdots a_n \$)$

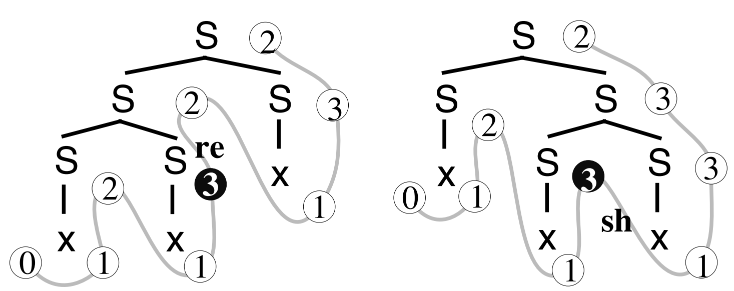
$\Rightarrow$ GLR parsing is a kind of stack transitions, or state transitions

GLR parsing

- Grammar:
  (1) S → S S
  (2) S → x

- LR table:

state	action		goto
	x	$	S
0	sh1		2
1	re2	re2	
2	sh1	acc	3
3	**re1 / sh1**	re1	3



⇒ LR table generated from a CFG.
→ **Global context**.

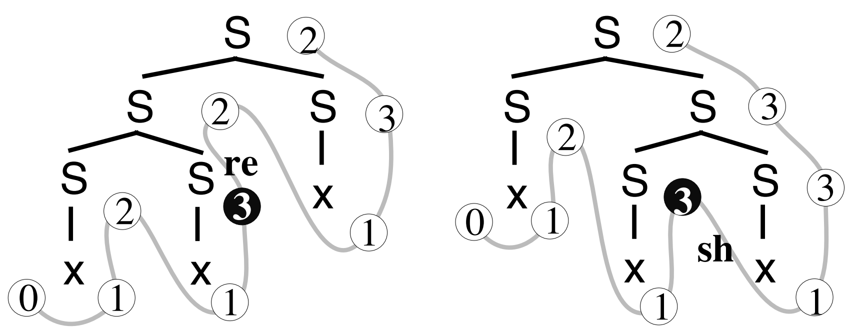
⇒ A pair of state and input symbol is the constraint for selecting the parsing action. → **Local n-gram context**.

Briscoe and Carroll's model

- A parse tree is regarded as a sequence of state transitions.
- Action probability is the probability of a transition out of a state. Therefore, action probabilities are normalized within each state.
- Probability for a reduce action is subdivided according to the state reached after applying the action, aiming at capturing the left context during the parse.
- Parse probability is the geometric mean of the applied action probabilities, to avoid the bias in favor of parsing involving fewer rules.

Briscoe and Carroll's model

state	action		goto
	x	$	S
0	**sh1** (5) 1.0		2
1	**re2** (10) $^{(0)}.33;^{(2)}.33$	**re2** (5) $^{(2)}.26;^{(3)}.08$	
2	**sh1** (9) .64	**acc** (5) .36	3
3	**re1** (4) / **sh1** (1) $^{(0)}.36$ / .09	**re1** (6) $^{(0)}.45;^{(2)}.09$	3



(a) [4] (b) [1]

⇒ Trained by counting the number of parsing actions, guided by the bracketed sentences.

Briscoe and Carroll's model

- Advantages:
  - Inherit the efficiency of GLR parsing.
  - Use the provided context by the nature of the GLR parsing.
    Left context: parsing state
    Right context: input symbol
- Problematic issues:
  - No probabilistic formalization.
  - Re-prediction of the input symbol after applying a reduce action.
  - Stack-top state after stack-pop operation is not deterministic.

PGLR model

- A parse derivation is a sequence of stack transitions:

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

- Probability of a complete stack transition sequence T:

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

PGLR model: estimation of transition probabilities

Stack-top state represents the information contained in the stack below it.

- Estimate for next input symbol:

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

- Estimate for next action:

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

- Estimate for next stack:

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

  The next stack after applying an action is deterministic.

Summary: B&C vs PGLR

- <u>Normalization</u>

  B&C : within each state.

  PGLR : according to state membership, i.e. in $\boldsymbol{S_s}$ or $\boldsymbol{S_r}$, because the input symbol after applying a reduce action is not changed.

<u>Transition probability:</u>

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

$\boldsymbol{S_s}$: s_0 and all the states reached after a shift action

$\boldsymbol{S_r}$: all the states reached after a reduce action

$\boldsymbol{S_s} \cap \boldsymbol{S_r} = \emptyset$: deterministic finite automaton (DFA) of GLR parsing

Summary: B&C vs PGLR

- Action probabilities

  B&C : reduce actions are subdivided according to the state reached after applying the action.

  PGLR : one action one probability.

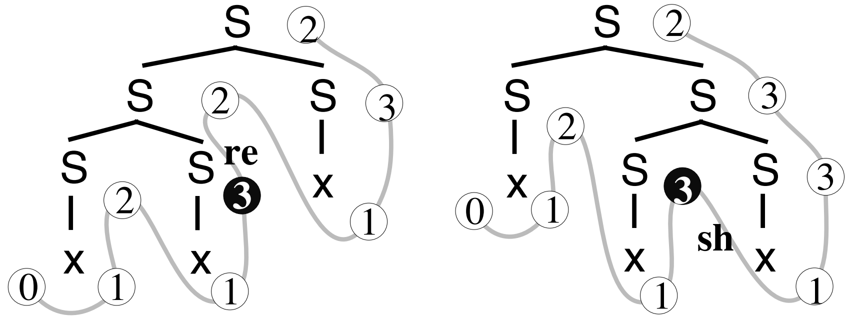
- Parse probabilities

  B&C : geometric mean of action probabilities applied in the parse.

  PGLR : product of action probabilities applied in the parse.

Summary: B&C vs PGLR

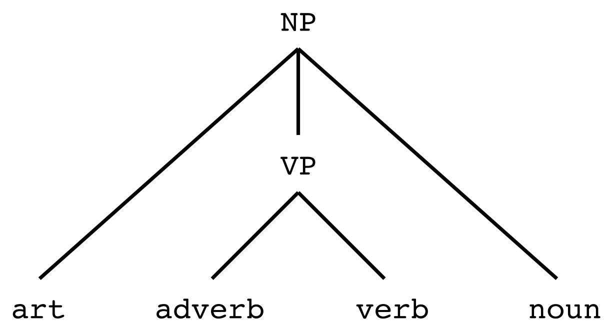
state	action		goto
	x	$	S
0	**sh1** (5)		2
S_s	1.0		
	1.0		
1	**re2** (10)	**re2** (5)	
S_s	$^{(0)}.33;^{(2)}.33$	$^{(2)}.26;^{(3)}.08$	
	.67	.33	
(2)	**sh1** (9)	**acc** (5)	3
S_r	.64	.36	
	1.0	1.0	
(3)	**re1** (4) / **sh1** (1)	**re1** (6)	3
S_r	$^{(0)}.36$ / .09	$^{(0)}.45;^{(2)}.09$	
	.80 / .20	1.0	



(a) [4] (b) [1]

Two-level PCFG

- Two-level PCFG (Chitrao and Grishman, 1990)
- Pseudo Context-sensitive Grammar (Charniak and Carroll, 1994)



$$P(VP \rightarrow adverb,\ verb \mid \rho(VP) = NP)$$

⇒ Incorporate context for PCFG.

⇒ Accurately reflect the true distribution of English (word based) language string.

⇒ Minimize the model's per-word (per-tag) cross entropy.

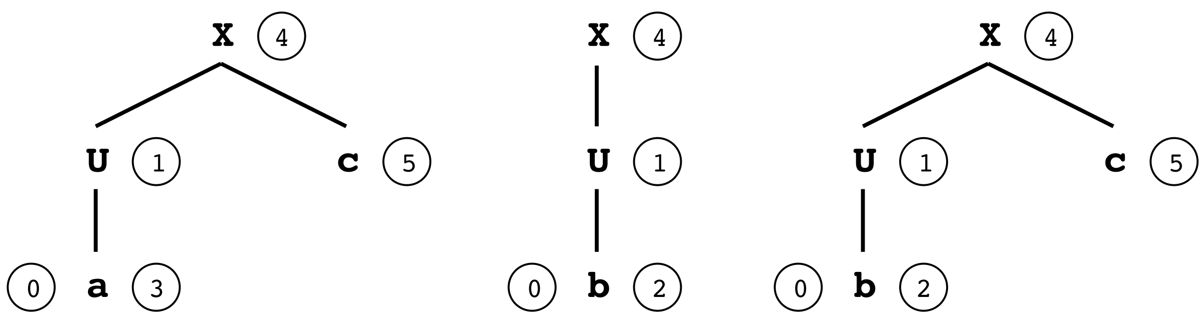
Evaluation

- Morphological and syntactic analysis:
  - Given a string of characters as the input
  - The task includes:
    word segmentation, POS tagging and parse tree construction
- ATR Japanese corpus
- Grammar:
  - 762 rules of the Japanese phrase structure grammar
  - 137 non-terminal symbols
  - 407 terminal symbols

Model analyses

- Grammar:

(1)	X	$\rightarrow$	U	c
(2)	X	$\rightarrow$	U	
(3)	U	$\rightarrow$	a	
(4)	U	$\rightarrow$	b	

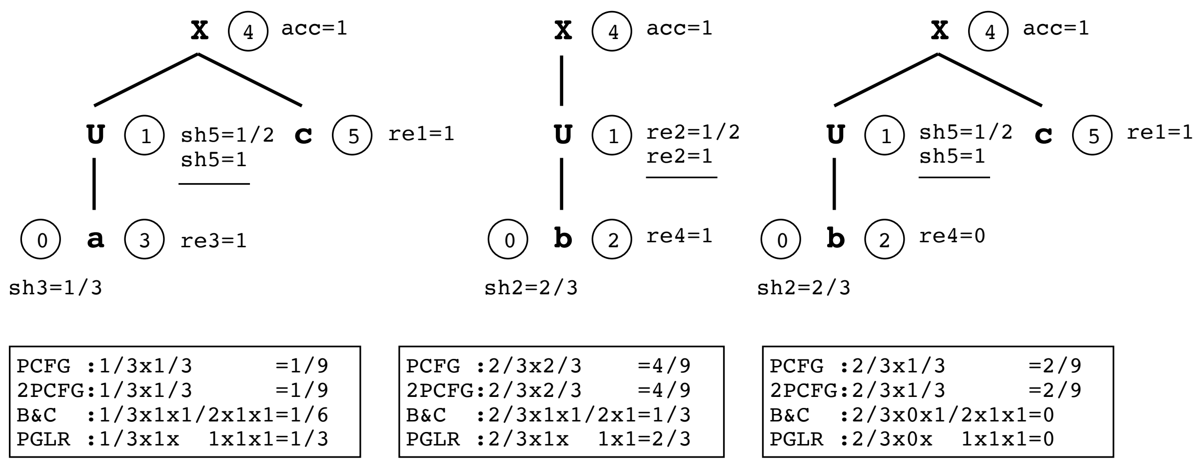


(S1)[1] **(S2)[2]** **(S3)[0]**

- Rule probabilities for Two-level PCFG:

(1)	S	; X	$\rightarrow$	U	c	(1/3)
(2)	S	; X	$\rightarrow$	U		(2/3)
(3)	X	; U	$\rightarrow$	a		(1/3)
(4)	X	; U	$\rightarrow$	b		(2/3)

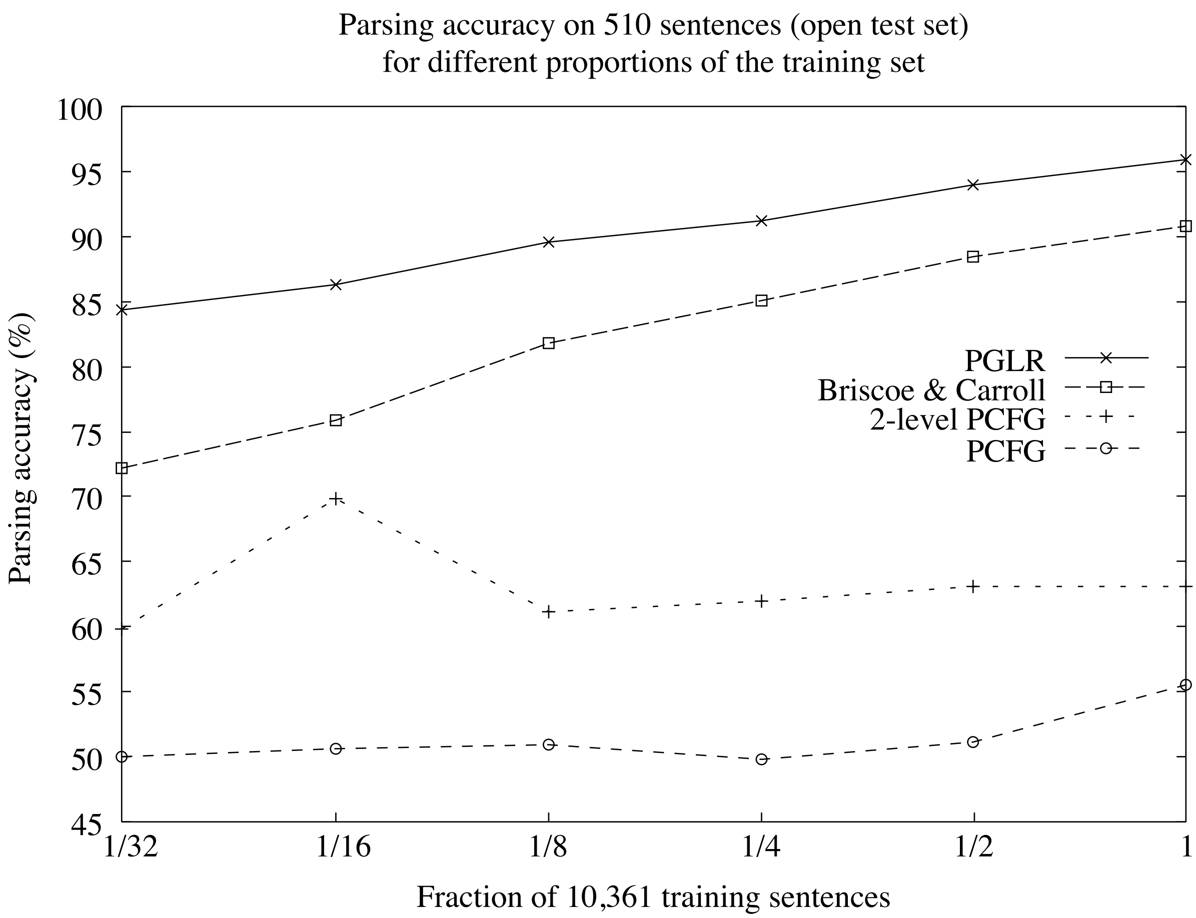
Comparative results for Two-level PCFG, B&C and PGLR



(S1)[1] **(S2)[2]** **(S3)[0]**

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

Model trainability



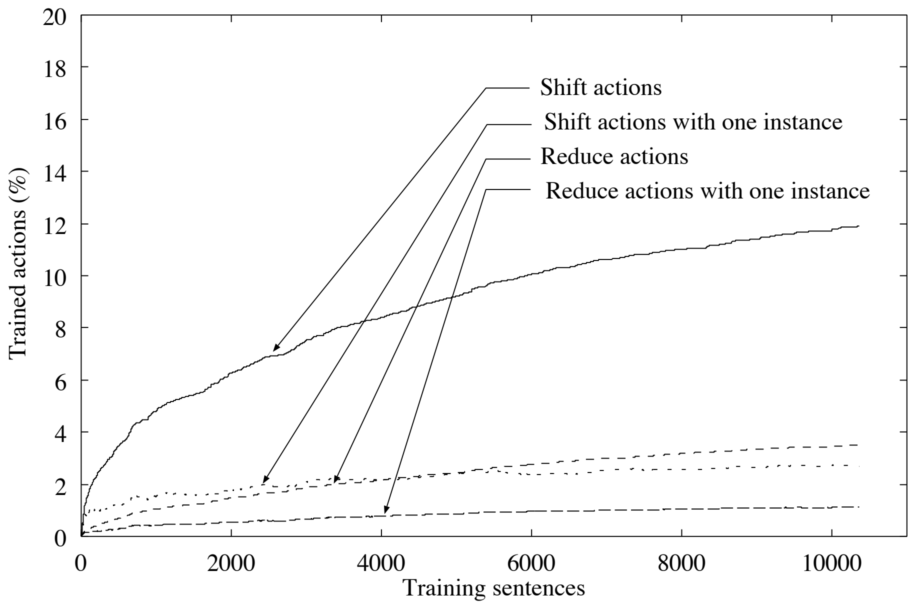
LALR and CLR table-based PGLR

- The degree of context-sensitivity of the states in a CLR table is higher than those in an LALR table.
- Data sparseness problems in using a CLR table.

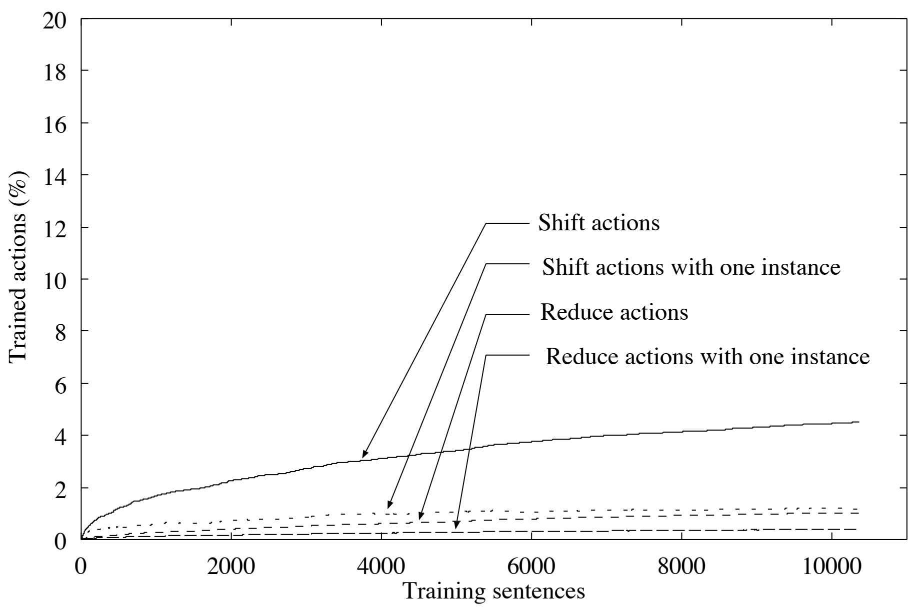
	LALR table	**CLR table**
States	856	3,715
Shift	11,445	43,833
Reduce	164,058	756,715
Goto	4,682	19,733
States in S_s	488	2,539
States in S_r	368	1,176

LALR and CLR table-based PGLR

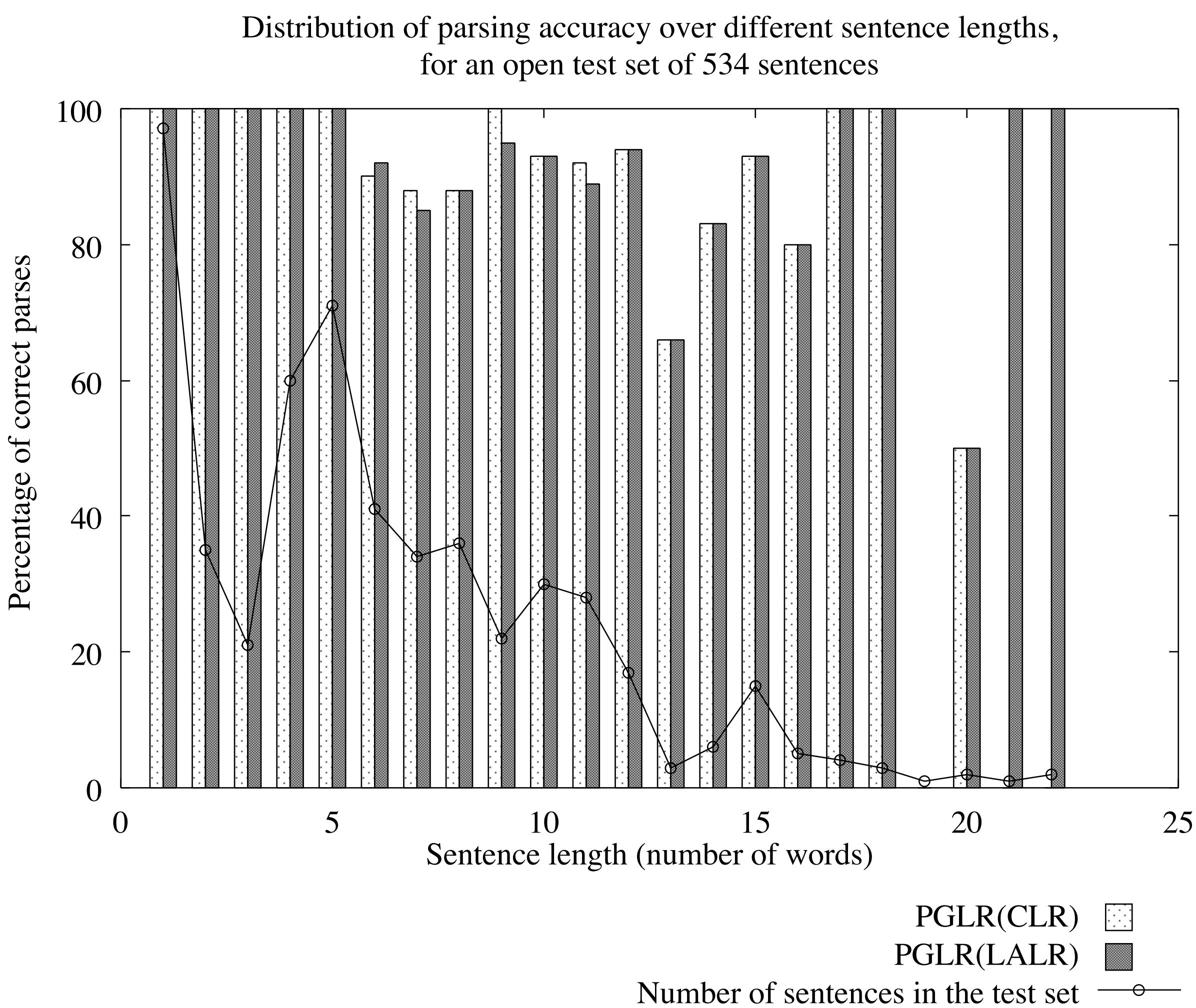
Learning curve of actions in PGLR using an LALR table
(total of 11,445 shift and 164,058 reduce actions)



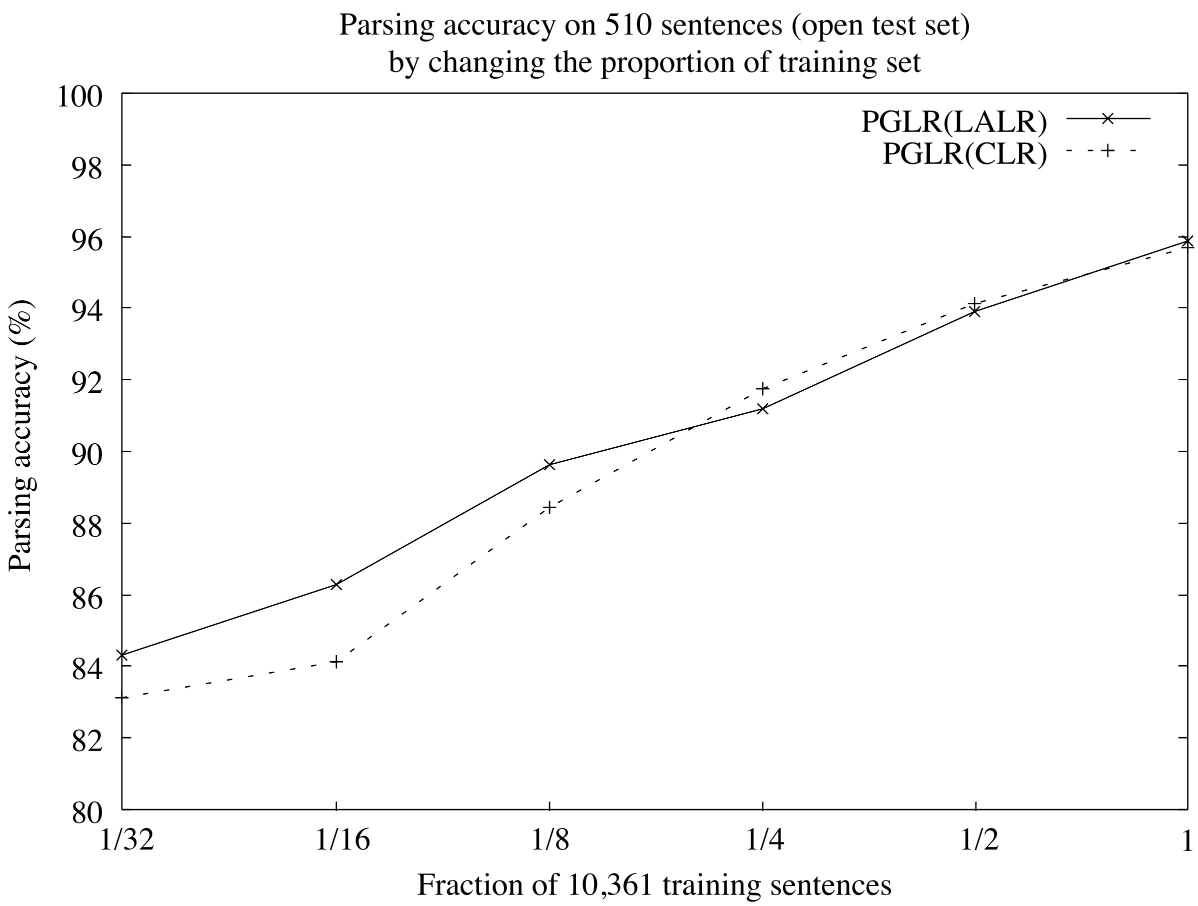
Learning curve of actions in PGLR using a CLR table
(total of 43,833 shift and 756,715 reduce actions)



LALR and CLR table-based PGLR



LALR and CLR table-based PGLR



Identifying n-best parses

- Previous research:
  - Pull out n-best parses from the full parsed packed parse forest without exhaustive search.
    - Extend the n-best parses from left-to-right using two heuristic methods (Carroll and Briscoe, 92).
    - Store the sorted node probabilities at each node in the packed parse forest. Then, pull out parses according to the product of node probabilities and truncate away the less probable parses (Wright et al., 91).
  - Beam-search in graph-structured stack (GSS)
    - Beam-search in GLR* limits the number of inactive state nodes to be extended (Lavie and Tomita, 96).

Node-driven parse pruning technique

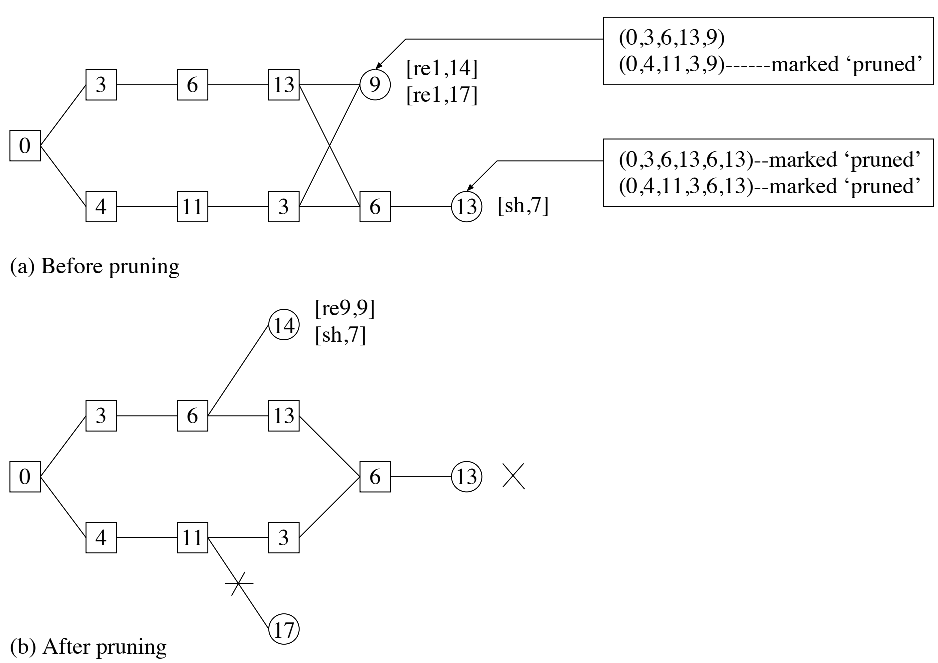
- Prune off less probable parses from the GSS during parsing:
  - Node-driven parse pruning technique.

$$T_t = G_t \cdot n_t$$

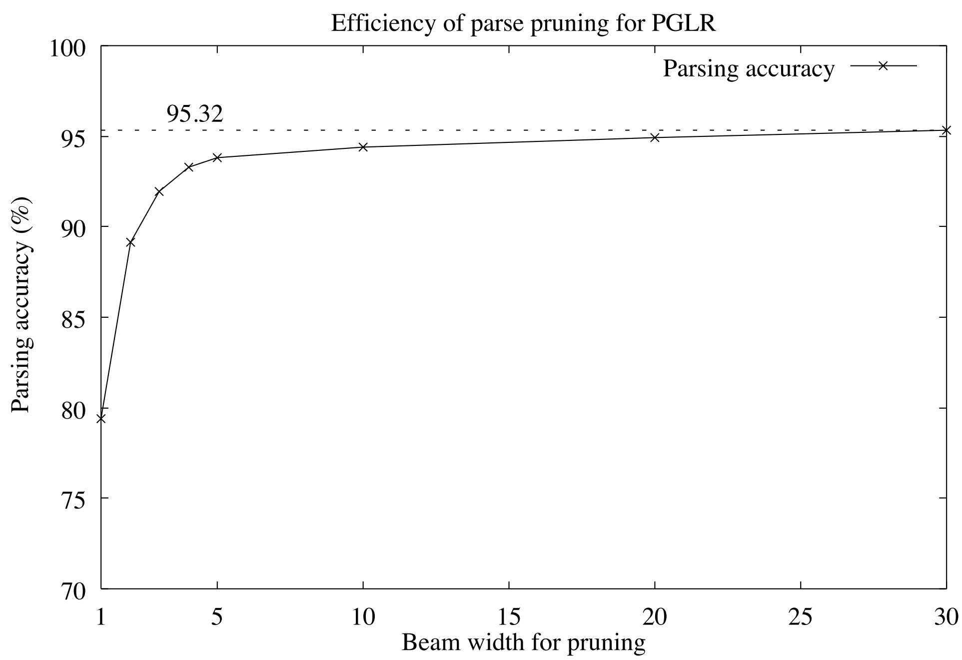
T_t: beam width

n_t: number of state nodes at time t

G_t: gain based on the number of state nodes at time t, and the beam width

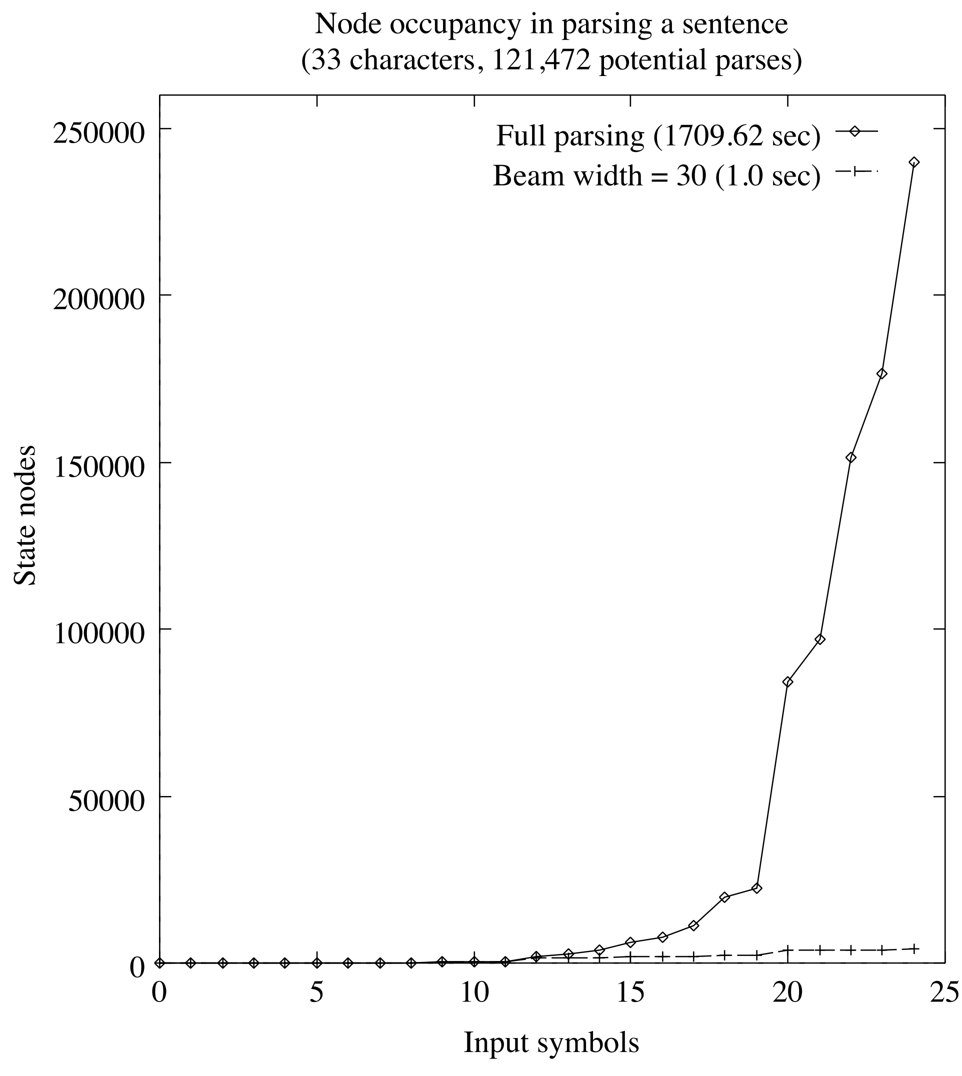


Evaluation of the node-driven parse pruning technique

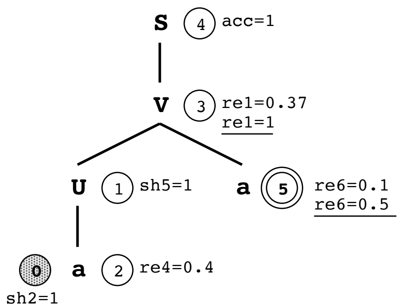


	Ave mem node/sent	Ave time sec/sent
Full parsing	9146	163
Beam width = 30	630	0.243

Evaluation of the node-driven parse pruning technique

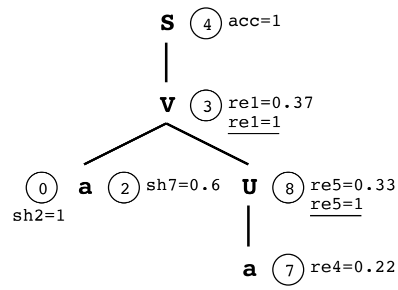


Problematic issues



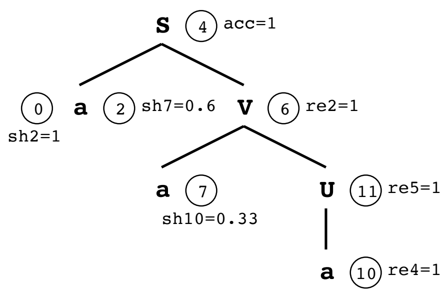
B&C :1x0.4x1x0.1x0.37x1 = 0.015
PGLR:1x0.4x1x0.5x 1x1 = 0.2

(S1-a)[1]



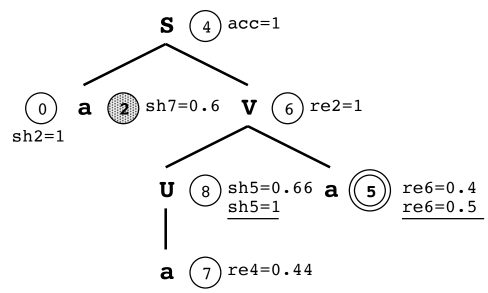
B&C :1x0.6x0.22x0.33x0.37x1 = 0.016
PGLR:1x0.6x0.22x 1x 1x1 = 0.132

(S1-b)[2]



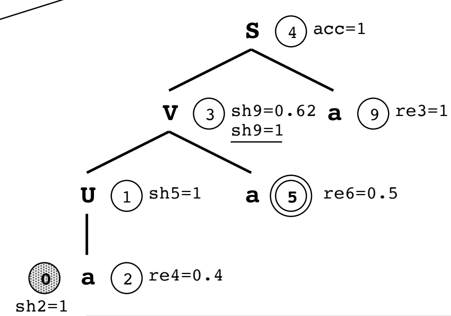
B&C :1x0.6x0.33x1x1x1x1 = 0.198
PGLR:1x0.6x0.33x1x1x1x1 = 0.198

(S2-a)[3]



B&C :1x0.6x0.44x0.66x0.4x1x1 = 0.069
PGLR:1x0.6x0.44x 1x0.5x1x1 = 0.132

(S2-b)[4]



B&C :1x0.4x1x0.5x0.62x1x1 = 0.124
PGLR:1x0.4x1x0.5x 1x1x1 = 0.2

(S2-c)[5]

PGLR model-2

- A parse derivation is a sequence of state transitions:

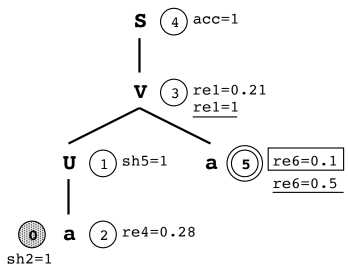
$$s_0 \overset{l_1,a_1}{\Longrightarrow} s_1 \overset{l_2,a_2}{\Longrightarrow} \ldots \overset{l_{n-1},a_{n-1}}{\Longrightarrow} s_{n-1} \overset{l_n,a_n}{\Longrightarrow} s_n$$

- Probability of a complete stack transition sequence T:

$$\begin{aligned}
&P(T)\\
&\quad= P(s_0, l_1, a_1, s_1, \ldots, s_{n-1}, l_n, a_n, s_n)\\
&\quad= P(s_0) \cdot \prod_{i=1}^{n} P(l_i, a_i, s_i | s_0, l_1, a_1, s_1,\\
&\qquad \ldots, l_{i-1}, a_{i-1}, s_{i-1})\\
&\quad= \prod_{i=1}^{n} P(l_i, a_i, s_i | s_{i-1})\\
&\quad= \prod_{i=1}^{n} P(l_i | s_{i-1}) \cdot P(a_i | s_{i-1}, l_i)\\
&\qquad \cdot P(s_i | s_{i-1}, l_i, a_i)
\end{aligned}$$

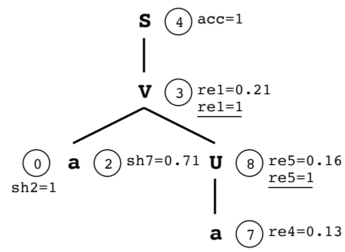
  - Estimate for next input symbol: $P(l_i|s_{i-1})$
  - Estimate for next action: $P(a_i|s_{i-1}, l_i)$
  - Estimate for next stack: $P(s_i|s_{i-1}, l_i, a_i)$

PGLR model-2 vs PGLR and B&C



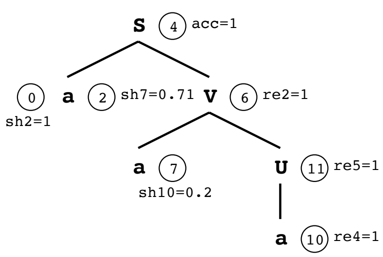

```
B&C   :1x0.28x1x0.1x0.21x1 = 0.006
PGLR  :1x0.28x1x0.5x   1x1 = 0.14
PGLR-2:1x0.28x1x0.1x   1x1 = 0.028
```

(S1-a)[1]



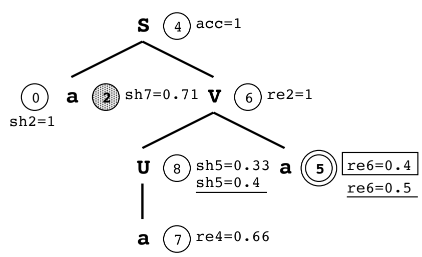

```
B&C   :1x0.71x0.13x0.16x0.21x1 = 0.003
PGLR  :1x0.71x0.13x   1x   1x1 = 0.092
PGLR-2:1x0.71x0.13x   1x   1x1 = 0.092
```

(S1-b)[2]



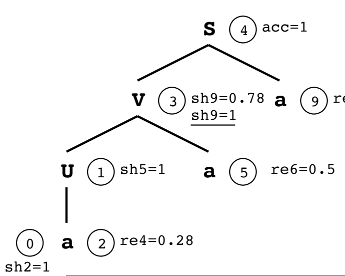

```
B&C   :1x0.71x0.2x1x1x1x1 = 0.142
PGLR  :1x0.71x0.2x1x1x1x1 = 0.142
PGRL-2:1x0.71x0.2x1x1x1x1 = 0.142
```

(S2-a)[3]



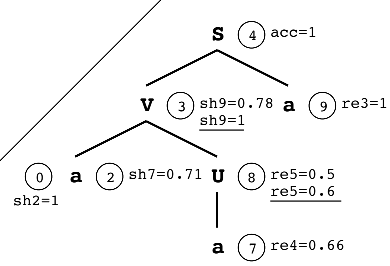

```
B&C   :1x0.71x0.66x0.33x0.4x1x1 = 0.062
PGLR  :1x0.71x0.66x 0.4x0.5x1x1 = 0.094
PGLR-2:1x0.71x0.66x 0.4x0.4x1x1 = 0.075
```

(S2-b)[4]




```
B&C   :1x0.28x1x0.5x0.78x1x1 = 0.109
PGLR  :1x0.28x1x0.5x   1x1x1 = 0.14
PGLR-2:1x0.28x1x0.5x   1x1x1 = 0.14
```

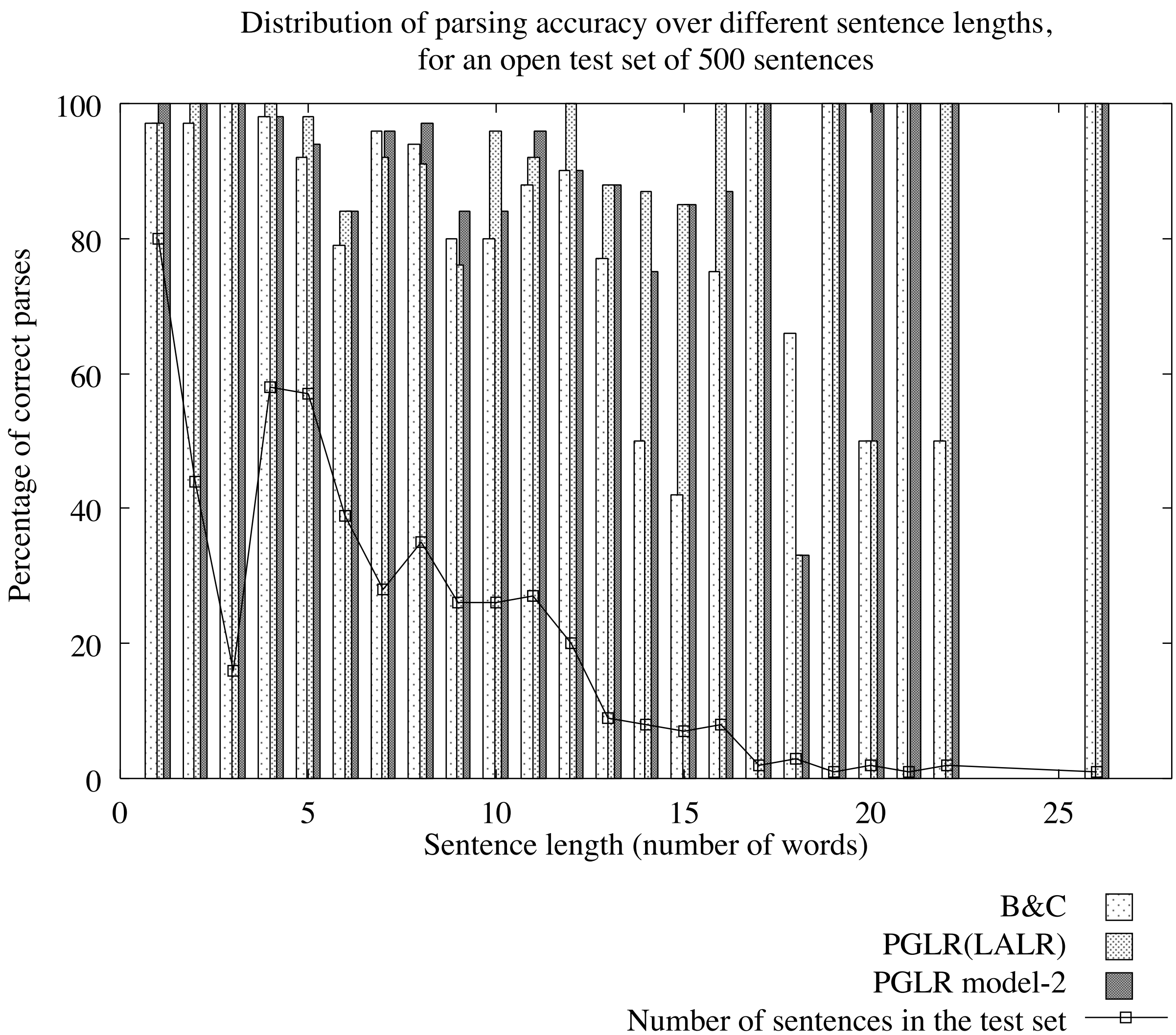
(S2-c)[5]



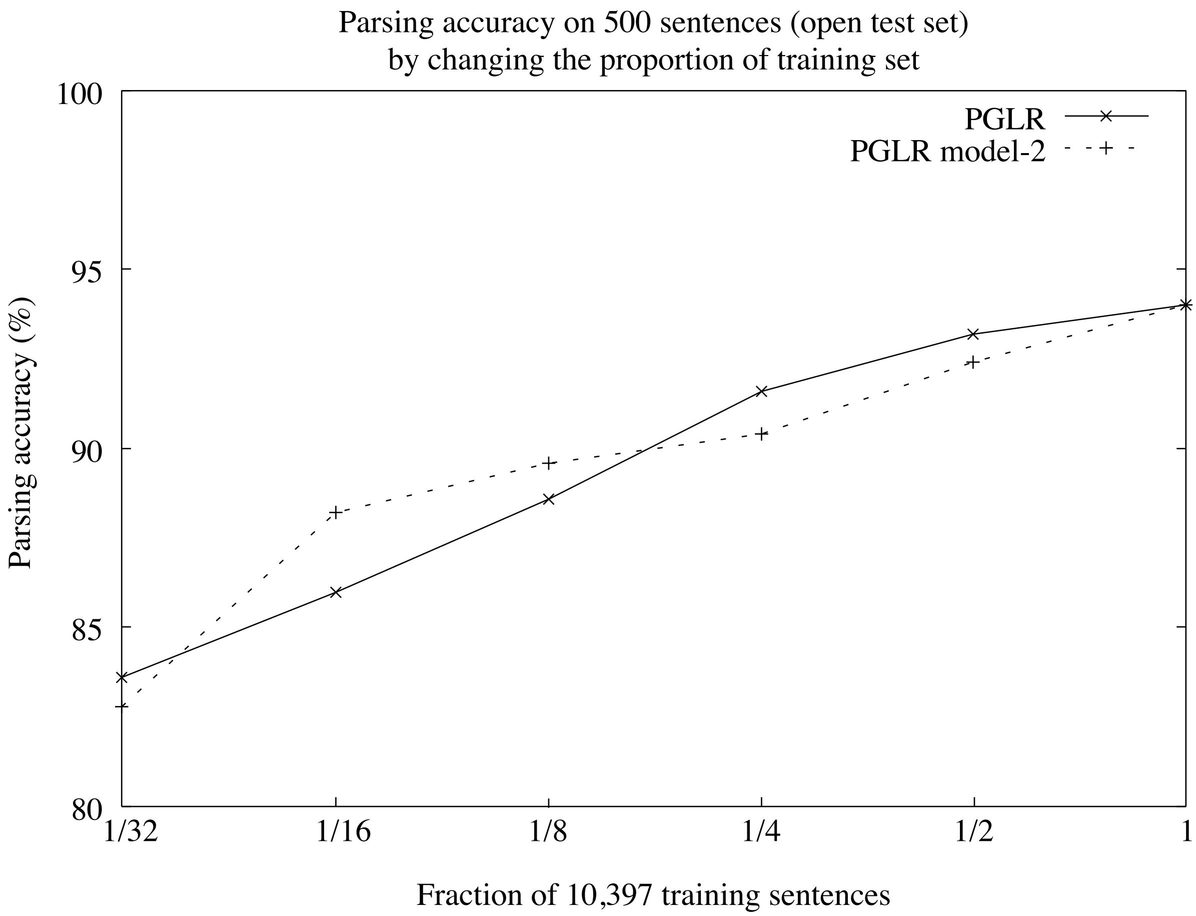

```
B&C   :1x0.71x0.66x0.5x0.78x1x1 = 0.183
PGLR  :1x0.71x0.66x0.6x   1x1x1 = 0.281
PGLR-2:1x0.71x0.66x0.6x   1x1x1 = 0.281
```

(S2-d)[6]

PGLR model-2 vs PGLR and B&C



Model trainability of PGLR model-2 vs PGLR



Conclusion

- Two new PGLR models are proposed:
  PGLR: Precise probabilistic model for GLR parsing.
  PGLR model-2: More precise PGLR.
- Parse performance:
  PGLR model-2 $>$ PGLR $>$ B&C $>$ Two-level PCFG $>$ PCFG
- The PGLR models effectively make use of both global CFG and local n-gram context in the GLR parsing framework.
- No statistically significant distinction between the results of PGLR(LALR) and PGLR(CLR).
- The node-driven parse pruning technique:
  (i) a space and time efficient left-to-right parse pruning technique.
  (ii) facilitates parsing highly ambiguous sentences, maintaining the use of GSS.
  (iii) applicable for speech recognition.

Future work

- N-best parses extraction from the packed parse forest.
- Lexicalize the probabilistic models.
- Include long distance constraints.
- Verify the PGLR models with a larger corpus.
- Smoothing methods for the PGLR models.