Probabilistic Language Modeling for GLR Parsing

Virach Sornlertlamvanich
Department of Computer Science
Tokyo Institute of Technology
June 1998
# Table of contents

1. Introduction
2. Language Modeling
3. Context-Sensitivity in GLR Parsing
4. Incorporating Probability into an LR Table
5. Experimental Results
6. Model Analyses
7. A Node-Driven GLR Parse Pruning Technique
8. Probabilistic GLR Model-2
9. Conclusions
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) \( NP \rightarrow \text{noun} \ (1/2) \)
  (4) \( DET \rightarrow \text{det} \ (1) \)

(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(CLAR).

- Parse pruning technique.
GLR parsing

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

\[ \text{action}_{i+1} = [\text{state}_i, \text{symbol}_{i+1}] \]

- Configurations:

  stack \hspace{1cm} \text{input}

  current configuration:

  \((s_0X_1s_1X_2s_2\cdots X_ms_m, \quad a_ia_{i+1}\cdots a_n\$)\)

  shift action:

  \((s_0X_1s_1X_2s_2\cdots X_ms_ma_is, \quad a_{i+1}\cdots a_n\$)\)

  reduce action:

  \((s_0X_1s_1X_2s_2\cdots X_{m-r}s_mA_{s}, \quad a_ia_{i+1}\cdots a_n\$)\)

⇒ GLR parsing is a kind of stack transitions, or state transitions
GLR parsing

• Grammar:
  
  (1)  \( S \rightarrow S \ S \)
  
  (2)  \( S \rightarrow x \)

• LR table:

<table>
<thead>
<tr>
<th>state</th>
<th>action</th>
<th>goto</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>$</td>
</tr>
<tr>
<td>1</td>
<td>re2</td>
<td>re2</td>
</tr>
<tr>
<td>2</td>
<td>sh1</td>
<td>acc</td>
</tr>
<tr>
<td>3</td>
<td>rel1 / sh1</td>
<td>rel1</td>
</tr>
</tbody>
</table>

⇒ 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

<table>
<thead>
<tr>
<th>state</th>
<th>action</th>
<th>goto</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>sh1</strong> (5)</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>re2</strong> (10)</td>
<td><strong>re2</strong> (5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>sh1</strong> (9)</td>
<td><strong>acc</strong> (5)</td>
</tr>
<tr>
<td>3</td>
<td><strong>rel</strong> (4) / <strong>sh1</strong> (1)</td>
<td><strong>rel</strong> (6)</td>
</tr>
</tbody>
</table>

![Diagram](image)

(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 \( T \):

\[
P(T) \\
= P(\sigma_0, l_1, a_1, \sigma_1, \ldots, \sigma_{n-1}, l_n, a_n, \sigma_n) \\
= P(\sigma_0) \cdot \prod_{i=1}^{n} P(l_i, a_i, \sigma_i | \sigma_0, l_1, a_1, \sigma_1, \ldots, l_{i-1}, a_{i-1}, \sigma_{i-1}) \\
= \prod_{i=1}^{n} P(l_i, a_i, \sigma_i | \sigma_{i-1}) \\
= \prod_{i=1}^{n} P(l_i | \sigma_{i-1}) \cdot P(a_i | \sigma_{i-1}, l_i) \cdot P(\sigma_i | \sigma_{i-1}, l_i, a_i) \]
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

- **Normalization**

  **B&C**: within each state.
  
  **PGLR**: according to state membership, i.e. in $S_s$ or $S_r$, because the input symbol after applying a reduce action is not changed.

  **Transition probability:**

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

  $S_s$: $s_0$ and all the states reached after a shift action
  
  $S_r$: all the states reached after a reduce action
  
  $S_s \cap S_r = \emptyset$: 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

<table>
<thead>
<tr>
<th>state</th>
<th>action</th>
<th>goto</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>action</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>$</td>
<td>S</td>
</tr>
<tr>
<td>0</td>
<td>sh1 (5)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>re2 (10)</td>
<td>re2 (5)</td>
</tr>
<tr>
<td></td>
<td>(0).33; (2).33</td>
<td>(2).26; (3).08</td>
</tr>
<tr>
<td></td>
<td>.67</td>
<td>.33</td>
</tr>
<tr>
<td>(2)</td>
<td>sh1 (9)</td>
<td>acc (5)</td>
</tr>
<tr>
<td></td>
<td>.64</td>
<td>.36</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(3)</td>
<td>rel1 (4) / sh1 (1)</td>
<td>rel1 (6)</td>
</tr>
<tr>
<td></td>
<td>(0).36 / .09</td>
<td>(0).45; (2).09</td>
</tr>
<tr>
<td></td>
<td>.80 / .20</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Diagrams

1. [Diagram a] [4]
2. [Diagram b] [1]
Two-level PCFG

- Two-level PCFG (Chitrawo and Grishman, 1990)
- Pseudo Context-sensitive Grammar (Char-niak and Carroll, 1994)

\[ P(VP \to \text{adverb, verb} \mid \rho(VP) = NP) \]

⇒ 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) \( U \rightarrow b \)

- 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

\[
\begin{array}{c}
\text{PCFG : } \frac{1}{3} \times \frac{1}{3} = \frac{1}{9} \\
\text{2PCFG: } \frac{2}{3} \times \frac{2}{3} = \frac{4}{9} \\
\text{B&C : } \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{3} \\
\text{PGLR : } \frac{1}{3} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{6}
\end{array}
\]

\[
\begin{array}{c}
\text{PCFG : } \frac{2}{3} \times \frac{2}{3} = \frac{4}{9} \\
\text{2PCFG: } \frac{2}{3} \times \frac{2}{3} = \frac{4}{9} \\
\text{B&C : } \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{3} \\
\text{PGLR : } \frac{2}{3} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{6}
\end{array}
\]

\[
\begin{array}{c}
\text{PCFG : } \frac{2}{3} \times \frac{2}{3} = \frac{4}{9} \\
\text{2PCFG: } \frac{2}{3} \times \frac{2}{3} = \frac{4}{9} \\
\text{B&C : } \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{3} \\
\text{PGLR : } \frac{2}{3} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{6}
\end{array}
\]

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

<table>
<thead>
<tr>
<th>Models</th>
<th>(S1)</th>
<th>(S2)</th>
<th>(S3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFG</td>
<td>1/9</td>
<td>4/9</td>
<td>2/9</td>
</tr>
<tr>
<td>Two-level PCFG</td>
<td>1/9</td>
<td>4/9</td>
<td>2/9</td>
</tr>
<tr>
<td>B&amp;C</td>
<td>1/6</td>
<td>1/3</td>
<td>0</td>
</tr>
<tr>
<td>PGLR</td>
<td>1/3</td>
<td>2/3</td>
<td>0</td>
</tr>
</tbody>
</table>
Model trainability

Parsing accuracy on 510 sentences (open test set) for different proportions of the training set

![Graph showing parsing accuracy for different proportions of the training set](image)
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.

<table>
<thead>
<tr>
<th></th>
<th>LALR table</th>
<th>CLR table</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>856</td>
<td>3,715</td>
</tr>
<tr>
<td>Shift</td>
<td>11,445</td>
<td>43,833</td>
</tr>
<tr>
<td>Reduce</td>
<td>164,058</td>
<td>756,715</td>
</tr>
<tr>
<td>Goto</td>
<td>4,682</td>
<td>19,733</td>
</tr>
<tr>
<td>States in $S_S$</td>
<td>488</td>
<td>2,539</td>
</tr>
<tr>
<td>States in $S_T$</td>
<td>368</td>
<td>1,176</td>
</tr>
</tbody>
</table>
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)
Distribution of parsing accuracy over different sentence lengths, for an open test set of 534 sentences

- PGLR(CLR)
- PGLR(LALR)

Number of sentences in the test set
LALR and CLR

Table-based PGLR

Parsing accuracy on 510 sentences (open test set)
by changing the proportion of training set

Fraction of 10,361 training sentences

PGLR(LALR)  
PGLR(CLR)  

Parsing accuracy (%)
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:

\[ T_t = G_t \cdot n_t \]

*\( T_t \): 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

![Graph showing efficiency of parse pruning for PGLR]

<table>
<thead>
<tr>
<th>Method</th>
<th>Ave mem node/sent</th>
<th>Ave time sec/sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full parsing Beam width = 30</td>
<td>9146</td>
<td>163</td>
</tr>
<tr>
<td>Beam width = 30</td>
<td>630</td>
<td>0.243</td>
</tr>
</tbody>
</table>
Evaluation of the node-driven parse pruning technique

Node occupancy in parsing a sentence
(33 characters, 121,472 potential parses)

- Full parsing (1709.62 sec)
- Beam width = 30 (1.0 sec)
(S1-a)[1]

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

(S1-b)[2]

B&C : 1x0.6x0.22x0.33x0.37x1 = 0.016
PGLR: 1x0.6x0.22x 1x1x1 = 0.132

(S2-a)[3]

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

(S2-b)[4]

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

(S2-c)[5]

B&C : 1x0.4x1x0.5x0.62x1x1 = 0.124
PGLR: 1x0.4x1x0.5x 1x1x1 = 0.2
PGLR model-2

- A parse derivation is a sequence of state transitions:

\[ s_0 \xrightarrow{l_1,a_1} s_1 \xrightarrow{l_2,a_2} \ldots \xrightarrow{l_{n-1},a_{n-1}} s_{n-1} \xrightarrow{l_n,a_n} s_n \]

- Probability of a complete stack transition sequence \( T \):

\[
P(T) = P(s_0, l_1, a_1, s_1, \ldots, s_{n-1}, l_n, a_n, s_n)
\]

\[
= P(s_0) \cdot \prod_{i=1}^{n} P(l_i, a_i, s_i|s_0, l_1, a_1, s_1, \ldots, l_{i-1}, a_{i-1}, s_{i-1})
\]

\[
= \prod_{i=1}^{n} P(l_i, a_i, s_i|s_{i-1})
\]

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

- 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

(S1-a) [1]

(S1-b) [2]

(S2-a) [3]

(S2-b) [4]

(S2-c) [5]

(S2-d) [6]
PGLR model-2 vs PGLR and B&C

Distribution of parsing accuracy over different sentence lengths, for an open test set of 500 sentences

Percentage of correct parses

Sentence length (number of words)

Number of sentences in the test set
Model trainability of PGLR model-2 vs PGLR

Parsing accuracy on 500 sentences (open test set) by changing the proportion of training set

Parsing accuracy (%)

Fraction of 10,397 training sentences
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.