Representation and Learning of Grammars Beyond the Language Domain

Keweï Tu
ShanghaiTech University
Outline

- Extending grammars beyond the language domain
- Grammar-based probabilistic modeling
- Unsupervised learning of grammars
Outline

- Extending grammars beyond the language domain
  - And-Or Grammars [Zhu & Mumford, 2006], [Tu, IJCAI 2016]
- Grammar-based probabilistic modeling
- Unsupervised learning of grammars
Example

\[ S \rightarrow NP \ VP \] \[ S \rightarrow Aux \ NP \ VP \] \[ S \rightarrow VP \] \[ NP \rightarrow Pronoun \] \[ NP \rightarrow Proper-Noun \] \[ NP \rightarrow Det \ Nominal \] \[ NP \rightarrow Nominal \] \[ Nominal \rightarrow Noun \] \[ Nominal \rightarrow Nominal \ Noun \] \[ Nominal \rightarrow Nominal \ PP \] \[ VP \rightarrow Verb \] \[ VP \rightarrow Verb \ NP \] \[ VP \rightarrow Verb \ NP \ PP \] \[ VP \rightarrow Verb \ PP \] \[ VP \rightarrow Verb \ NP \ NP \] \[ VP \rightarrow VP \ PP \] \[ PP \rightarrow Preposition \ NP \] 

\[ P(T) = 0.05 \times 0.20 \times 0.20 \times 0.20 \times 0.75 \times 0.30 \times 0.60 \] \[ \times 0.10 \times 0.40 = 2.2 \times 10^{-6} \]
And-Or Normal Form of Stochastic CFG

- Two types of non-terminals
  - And-nodes, Or-nodes
- Two types of production rules
  - And-rule (composition): \( A \to N_1 N_2 \cdots \)
  - Or-rule (alternative configurations): \( O \to N_1 | N_2 | t_1 | t_2 | \cdots \)

<table>
<thead>
<tr>
<th>SCFG</th>
<th>The AND-OR Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \to a ) (0.4)</td>
<td>OR(_S) \to a ) (0.4)</td>
</tr>
<tr>
<td>( A \to a ) (1.0)</td>
<td>AND(_{AB}) \to OR(_A) OR(_B)</td>
</tr>
<tr>
<td>( B \to b_1 ) (0.2)</td>
<td>OR(_A) \to a ) (1.0)</td>
</tr>
<tr>
<td>( ) (0.5)</td>
<td>( OR_B \to b_1 ) (0.2)</td>
</tr>
<tr>
<td>( ) (0.3)</td>
<td>( b_2 ) (0.5)</td>
</tr>
<tr>
<td>( ) (0.3)</td>
<td>( b_3 ) (0.3)</td>
</tr>
</tbody>
</table>

[Tu & Honavar, 2008]
And-Or Normal Form of Stochastic CFG

Phrase

NP
Verb
NP

“Concatenating” relation

Word/Letter

And-node
Or-node

Sentiment

0.6
0.4
0.2

0.4
0.2
0.4

start

s
t
a
r

……...
And-Or Normal Form of Stochastic CFG

“They start the book today.”
Stochastic Context-free And-Or Grammar (AOG)

And-node
Or-node

High-level pattern

Relation

Atomic pattern
AOG of Images

Image patch

- Eyes
- Ears
- Nose
- And-node
- Or-node

Images of Animal face

Spatial relation (e.g., relative position)

Visual word (e.g., oriented line)
AOG of Images

Images of Animal face

Dog face

Eyes

Ears

Nose

And-node

Or-node

0.4

0.2

0.4

0.6

0.4

0.4

………

………

………

………

………

………

………

………

………

………
Applications – indoor scene parsing

(a) Grammar

(b) Parse Tree

[Zhao & Zhu, NIPS 2011]
AOG of Events

Event or sub-event

Temporal relation

Atomic action

And-node

Or-node

Office activity

Enter room

Open door

Walk

Close door

Temporal relation

Atomic action

0.4

0.4

0.2

0.4

0.6
Applications – video event parsing

Grammar

Parse Tree

[Pei, Jia & Zhu, ICCV 2011]
With some additional constraints, we get a decomposable sum-product network [Poon & Domingos, 2011]
Formal Definition

- A stochastic context-free And-Or grammar is a 5-tuple
  - A set $\Sigma$ of terminals (atomic patterns)
  - A set $N$ of nonterminals (high-level patterns)
    - Two disjoint subsets: And-nodes, Or-nodes
  - A start symbol $S \in N$ (a complete pattern)
  - A function $\theta$ that maps an instance of a terminal or nonterminal $x$ to a parameter $\theta_x$
  - A set $R$ of production rules
    - Or-rule $\langle r, p \rangle$
      - $r$: $O \rightarrow x$
      - $p$ is the probability of $O$ producing $x$
    - And-rule $\langle r, t, f \rangle$
      - $r$: $A \rightarrow \{x_1, x_2, \ldots, x_n\}$ for some $n \geq 2$
      - $t(\theta_{x_1}, \theta_{x_2}, \ldots, \theta_{x_n})$ is a parameter relation between child nodes
      - $\theta_A = f(\theta_{x_1}, \theta_{x_2}, \ldots, \theta_{x_n})$ is a parameter function
Special cases of AOG

Natural Language Processing
- Stochastic Context-free Grammars
- Hidden Markov Models
- Linear Context-free Rewriting System
- Constraint-based Grammar Formalisms
- etc.

Computer Vision
- Pictorial Structures Models
- Deformable Part Models
- Flexible Mixture-of-Parts Models
- etc.

Machine Learning
- Sum-Product Networks
- Naïve Bayes
- Biclustering
- Thin Junction Trees
- Mixtures of Trees
- Latent Tree Models
- etc.
Logic Interpretation

- Stochastic context-free AOG can be seen as a subset of first-order probabilistic logic
  - grammar rules → material implications
  - terminals, nonterminals → unary relations
  - A possible-world semantics

- Close connection with the field of statistical relational learning
  - Resembles tractable Markov logic [Domingos & Webb, 2012]
  - A special case of stochastic logic programs [Muggleton, 1996]
Outline

- Extending grammars beyond the language domain
- Grammar-based probabilistic modeling
  - Sum-product networks
  - Latent Dependency Forest Models
- Unsupervised learning of grammars
Sum-Product Network (SPN) [Poon & Domingos, UAI 2011]

A new type of probabilistic models
- Linear-time marginal inference
- Capable of modeling context-specific independence

Additional structural restrictions:
- Completeness at sum nodes
- Decomposability at product nodes
Reduction to AOG [Tu, IJCAI 2016]

- **Reduction**
  - Sum $\Leftrightarrow$ Or-node
  - Product $\Leftrightarrow$ And-node
  - Indicator $\Leftrightarrow$ Terminal
  - No relation in And-rules

- Probability computation in SPN $\Leftrightarrow$ Marginal computation in AOG (the inside algorithm)
Benefit of the Reduction

A lot of methods/ideas for inference and learning of grammars in NLP

These methods/ideas may be borrowed for inference and learning of SPNs
Grammars ↔ Probabilistic Models

Context-free Grammar

Dependency Grammar (Non-projective)

Latent Dependency Forest Model
Latent Dependency Forest Model (LDFM)
[Chu, Jiang & Tu, AAAI 2017]

Dependency rules:

- Root $\rightarrow X_1 = T$ [0.2]
- Root $\rightarrow X_1 = F$ [0.1]
- Root $\rightarrow X_2 = T$ [0.4]
- Root $\rightarrow X_2 = F$ [0.3]
- $X_1 = T$ $\rightarrow X_2 = T$ [0.6]
- $X_1 = T$ $\rightarrow X_2 = F$ [0.4]
- $X_1 = F$ $\rightarrow X_2 = T$ [0.1]
- $X_1 = F$ $\rightarrow X_2 = F$ [0.9]

Given $X_1 = T$, $X_2 = F$, we have three parses:

- ROOT $X_1 = T$ $\rightarrow X_2 = F$ $w_1 = 0.2 \times 0.3 = 0.06$
- ROOT $X_1 = T$ $\rightarrow X_2 = F$ $w_1 = 0.2 \times 0.4 = 0.08$
- ROOT $X_1 = T$ $\rightarrow X_2 = F$ $w_1 = 0.3 \times 0.5 = 0.15$

$P(X_1 = T, X_2 = F) \propto w_1 + w_2 + w_3 = 0.29$

- Unnormalized probability: tractable using matrix-tree theorem
- Inference: Gibbs sampling, tree-augmented sampling
- Learning: expectation-maximazation
Outline

- Extending grammars beyond the language domain
- Grammar-based probabilistic modeling
- Unsupervised learning of grammars
Learning a grammar

- Supervised Methods
  - Rely on treebanks (data samples annotated with parse trees)

Training Data

A square is above the triangle.
A triangle rolls.
The square rolls.
A triangle is above the square.
A circle touches a square.

Probabilistic Grammar

\[
S \rightarrow NP \ VP \\
NP \rightarrow \text{Det} \ N \\
VP \rightarrow \text{Vt} \ NP \ (0.3) \\
| \ \text{Vi} \ PP \ (0.2) \\
| \ rolls \ (0.2) \\
| \ bounces \ (0.1) \\
\]

Nonexistent for most types of data!
Learning a grammar

- **Supervised Methods**
  - Rely on treebanks (data samples annotated with parse trees)

- **Unsupervised Methods**
  - Do not require annotated data
Unsupervised grammar learning

- Two tasks
  - Structure search
    - Try to find an optimal set of grammar rules
  - Parameter learning
    - Given a set of grammar rules, try to learn their probabilities

Extremely difficult on real data

Difficult but doable, a lot of work in NLP over the past decade
Unsupervised parameter learning

- Expectation-maximization
  - Given grammar rules, initialize their probabilities
  - E-step
    - Parse the training samples using the current grammar
  - M-step
    - Update the grammar rule probabilities to maximize expected likelihood of the parses
  - Repeat until convergence

- Next
  - Our work in unsupervised parameter learning of natural language grammars
The ambiguity of natural language is remarkably low!

Idea: incorporate unambiguity bias into learning

- Measure ambiguity by entropy of the parse given the sentence and the grammar
- Add it into the objective function using posterior regularization [Ganchev et al. (2010)]

\[
J(\theta) = \log p(\theta|X) - \min_q \left( \text{KL}(q(Z)||p_\theta(Z|X)) + \sigma \sum_i H(z_i) \right)
\]

- Log posterior of the grammar
- Entropy of a proxy distribution \( q \) of the parses given the sentences & grammar
- \( \sigma \) controls regularization strength
Unambiguity regularization [Tu & Honavar, EMNLP 2012]

- Coordinate Ascent
- Algorithm behavior depends on regularization strength $\sigma$
  - $\sigma = 0$: equivalent to standard EM
  - $\sigma \geq 1$: equivalent to hard EM
  - $0 < \sigma < 1$: \textit{softmax-EM} (E-step produces softmax of parse distribution)

Very easy to implement: Just exponentiate all the rule probabilities before the E-step
Experimental results

Dependency Accuracy on WSJ10 Testset (Training with WSJ10, no lexicalization)

Generative Approaches

- DMV (2004)
- LN Families (2009)
- PR-S (2010)
- EVG (2009)
- TSG-DMV (2010)
- UR-A E-DMV (2012)
Unsupervised Neural Dependency Parsing
[Jiang, Han & Tu, EMNLP 2016]

- Represent symbols with vectors
- Learn ANN to predict grammar rule probabilities

\[ P(\text{child} \mid \text{head}, \text{direction}, \text{valency}) \]
Unsupervised Neural Dependency Parsing

[Jiang, Han & Tu, EMNLP 2016]

- Learning by expectation-maximization

![Diagram showing the cycle of P (Forward Evaluating), W (Dynamic Programming), and E (Neural Network Training) with Count Normalizing and Neural Network Training connections.](image)
Experimental results

Dependency Accuracy on WSJ10 Testset
(Training with WSJ10, no lexicalization)

Generative Approaches

- DMV (2004)
- LN Families (2009)
- PR-S (2010)
- EVG (2009)
- TSG-DMV (2010)
- UR-A E-DMV (2012)
- N E-DMV good init (2016)
CRF-autoencoder [Cai, Jiang & Tu, EMNLP 2017]

- Use a discriminative parser in unsupervised grammar learning
- Based on the CRF-autoencoder framework [Ammar, Dyer & Smith, 2014]

Diagram:

- Encoder
  - A discriminative parser (MSTParser)
- Decoder
  - Generating child word from head word

Graph:

- $x$: Input sequence
- $y_1$, $y_2$, $y_3$, $y_4$: Hidden states
- $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$, $\hat{x}_4$: Predicted output

Sequence: These, stocks, eventually, reopened
CRF-autoencoder [Cai, Jiang & Tu, EMNLP 2017]

- Learning objective
  - Reconstruction probability
- Optimization algorithm
  - Coordinate ascent: alternately optimize encoder and decoder
Experimental results

Dependency Accuracy on WSJ10 Testset (Training with WSJ10, no lexicalization)

Generative Approaches
- DMV (2004)
- LN Families (2009)
- PR-S (2010)
- EVG (2009)
- TSG-DMV (2010)

Discriminative Approaches
- UR-A E-DMV (2012)
- NE-DMV good init (2016)
- Convex MST (2015)
- CRF-AE (2017)
Summary
Summary

- Extending grammars beyond the language domain
  - Stochastic And-Or grammars
- Grammar-based probabilistic modeling
  - Sum-product networks
  - Latent Dependency Forest Models
- Unsupervised learning of grammars
  - Unambiguity regularization
  - Unsupervised neural dependency parsing
  - CRF autoencoder
Thank you!

Q&A