Grammars

Machine Learning Reading Group

July 2016

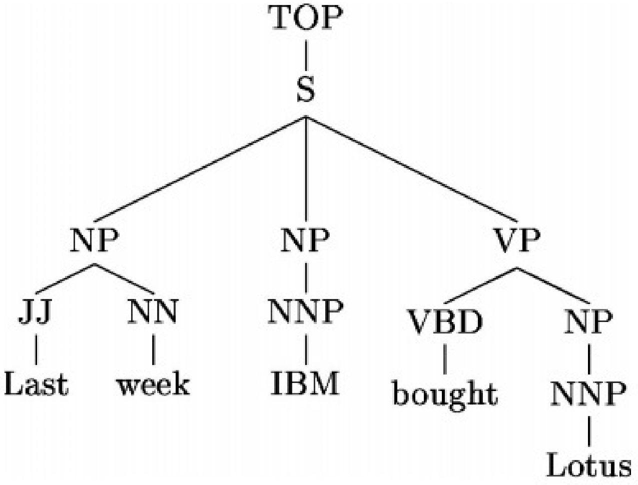


Figure 1: Adapted from [1].

Image Parsing

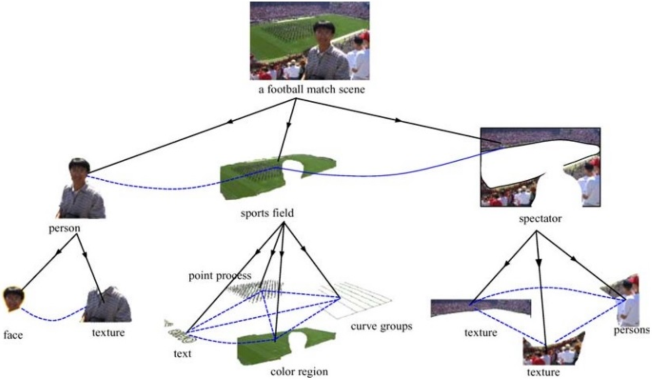


Figure 2: Adapted from [10].

RNA Secondary Structure Prediction

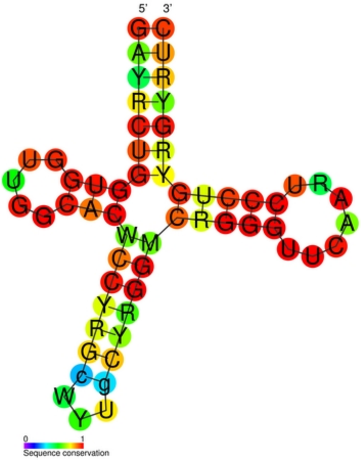


Figure 3: Adapted from Wikimedia Commons.

Outline

- Non-Probabilistic Grammars
- Probabilistic Context Free Grammars (PCFGs)
  - PCFGs in Chomsky Normal Forms (CNFs)
    - Calculating Probability of a Sequence
    - Determining the Most likely Parse for a Sequence
    - Learning Rule Parameters for an Unparameterised PCFG
- Grammar Learning
  - The ADIOS Algorithm

Grammars

- A **grammar** G is a 4-tuple $G = (N, \Sigma, R, S)$, where:
  - N is a finite set of **non-terminal** symbols.
  - Σ is a finite set of **terminal** symbols.
  - R is finite set of **rules** of the form $(\Sigma \cup N)^* N (\Sigma \cup N)^* \to (\Sigma \cup N)^*$
  - $S \in N$ is a distinguished **start** symbol.

Grammar Example

Grammar for $L = \{a_n b_n c_n : n > 0\}$

S → ABCS

S → TC

CA → AC

BA → AB

CB → BC

CTC → TCc

TC → TB

BTB → TBb

TB → TA

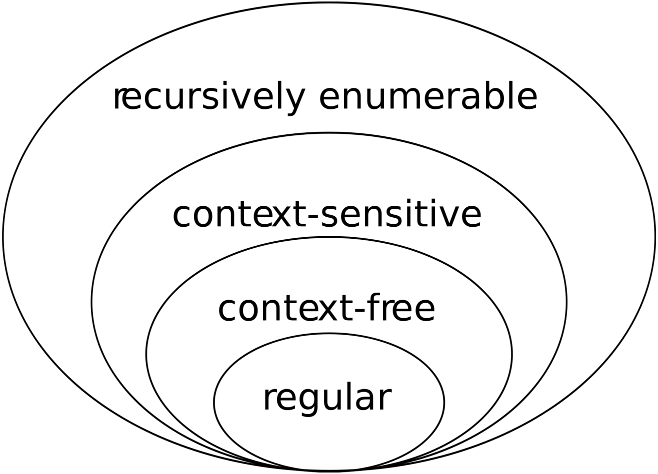
ATA → TAa

TA → ε

$$N = \{A, B, C, S, T\}$$

$$\Sigma = \{c, a, b\}$$

The Chomsky Hierarchy



Context-Free Grammars (CFGs)

- A **CFG** G is a 4-tuple $G = (N, \Sigma, R, S)$, where:
  - N is a finite set of **non-terminal** symbols.
  - Σ is a finite set of **terminal** symbols.
  - R is a finite set of **rules** of the form $X \to Y_1 Y_2 ... Y_n$, where $X \in N, n \geq 0$ and $Y_i \in (N \bigcup \Sigma)^*$ for $i = 1...n$.
  - $S \in N$ is a distinguished **start** symbol.

GFG Example

CFG:

$S \rightarrow ASB$

$A \rightarrow aAS \mid a \mid \varepsilon$

$B \rightarrow SbS \mid A \mid bb$

Chomsky Normal Form (CNF)

- A context-free grammar is in **Chomsky Normal Form** when
  - each rule has the form $X \rightarrow X_1 X_2$, $X \rightarrow a$, or $S \rightarrow \epsilon$ where X, X_1 and X_2 are non-terminals, a is a terminal, and S is the start symbol.
- Any context-free grammar can be converted into an equivalent grammar in Chomsky normal form.
  - **Disadvantages**: (1) more nonterminals & rules (up to quadratic blowup in size), (2) less obvious relation to problem domain.
  - **Advantages**: (1) easy implementation of parsers, (2) CNF conversion is used in some algorithms as a preprocessing step, e.g., CYK.

CNF Example

CFG:

$S \rightarrow ASB$

$A \rightarrow aAS \mid a \mid \varepsilon$

$B \rightarrow SbS \mid A \mid bb$

CNF:

$S_0 \rightarrow AU_1 \mid SB \mid AS$

$S \rightarrow AU_2 \mid SB \mid AS$

$A \rightarrow V_1U_3 \mid a \mid V_1S$

$B \rightarrow SU_4 \mid V_2V_2 \mid V_1U_5 \mid a \mid V_1S$

$U_2 \rightarrow SB$

$U_3 \rightarrow AS$

$U_4 \rightarrow V_2S$

$U_5 \rightarrow AS$

$V_1 \rightarrow a$

$V_2 \rightarrow b$

Probabilistic Grammars

- **Non-probabilistic grammars** either generate a string x or not.
  - Patterns have to be modified to allow the language to grow.
  - It is difficult to create a specific pattern. In some cases (e.g., some protein families) it is impossible to produce a discriminative pattern.
  - As a pattern is loosened, it is possible to match random unrelated sequences.

Probabilistic Grammars

- **Non-probabilistic grammars** either generate a string x or not.
  - Patterns have to be modified to allow the language to grow.
  - It is difficult to create a specific pattern. In some cases (e.g., some protein families) it is impossible to produce a discriminative pattern.
  - As a pattern is loosened, it is possible to match random unrelated sequences.
- **Probabilistic (stochastic) grammars** generate different strings x with probability $P(x|\theta)$.
  - Allows exceptions, but can give exceptions less probability.

Hidden Markov Models (HMMs) are Probabilistic Regular Grammars [3]

- **HMMs** are **probabilistic regular grammars**.
  - **HMMs** are **Moore machines** (emit symbols on states).
  - **Probabilistic regular grammars** are **Mealy machines** (emit a terminal on transition to a non-terminal).
  - Moore and Mealy machines are interchangeable.
- Algorithms for scoring, training, aligning of probabilistic regular grammars are the same used as in HMM.

Probabilistic Context Free Grammars (PCFGs)

- A **PCFG**
  - extends **context-free grammars** by defining a multinomial distribution over the set of derivation rules over each terminal symbol.
  - has a parameter $P(X \rightarrow \gamma)$ for each rule $X \rightarrow \gamma$, where probabilities are normalized at the level of each nonterminal, $\forall X \in N, \sum_{X \rightarrow \gamma} P(X \rightarrow \gamma) = 1$.
  - $\sum_x P(x|\theta) = 1$

PCFG Example

PCFG

S → NP VP 1.0

PP → P NP 1.0

VP → V NP 0.7

VP → VP PP 0.3

P → with 1.0

V → saw 1.0

NP → NP PP 0.4

NP → astronomers 0.1

NP → ears 0.18

NP → saw 0.04

NP → stars 0.18

NP → telescopes 0.1

Basic Problems for PCFGs [3]

- **Scoring** problem: calculate the **probability of a sequence** according to a parametrised PCFG:

$$P(x|\theta) = ? \quad (1)$$

- **Alignment** problem: Determine **most likely parse (alignment) of a sequence** according to a parametrised PCFG:

$$\underset{t}{\text{argmax}}\, P(t|x, \theta) = ? \quad (2)$$

- **Training** problem: Learn **rule probability parameters** for an unparameterised PCFG, given a set of sequences:

$$\underset{\theta}{\text{argmax}}\, P(x_1 ... x_n|\theta) = ? \quad (3)$$

Sequence Probability

- The **probability** of a sequence x according to grammar G with parameters θ:

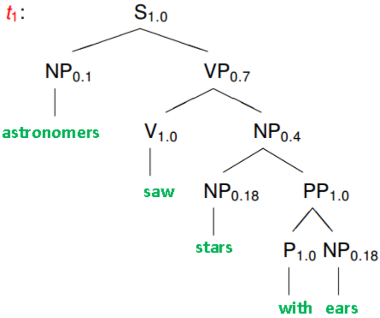
$$P(x|\theta) = \sum_t P(x, t) \tag{4}$$

  where t is a **parse (alignment)** of the sequence.
- **Trivial** solution: find all parse trees, calculate and sum up their probabilities.
  - Problem: exponential time complexity in general
- **Efficient** solution: using inside and outside probabilities.

Probability of Sequence

S → NP VP 1.0
PP → P NP 1.0
VP → V NP 0.7
VP → VP PP 0.3
P → with 1.0
V → saw 1.0
NP → NP PP 0.4
NP → astronomers 0.1
NP → ears 0.18
NP → saw 0.04
NP → stars 0.18
NP → telescopes 0.1

t_1:

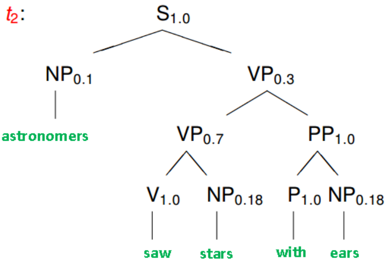


$$
\begin{aligned}
P(x,t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072
\end{aligned}
$$

Probability of Sequence

- S → NP VP 1.0
- PP → P NP 1.0
- VP → V NP 0.7
- VP → VP PP 0.3
- P → with 1.0
- V → saw 1.0
- NP → NP PP 0.4
- NP → astronomers 0.1
- NP → ears 0.18
- NP → saw 0.04
- NP → stars 0.18
- NP → telescopes 0.1

t_2:



$$P(x, t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$
$$= 0.0006804$$

Probability of Sequence

- For the sentence $x =$ astronomers saw stars with ears, we can construct 2 parse trees.
- $P(x|\theta) = P(x, t_1) + P(x, t_2) = 0.0009072 + 0.0006804 = 0.0015876$

The Inside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.

The Inside Probability for CNFs

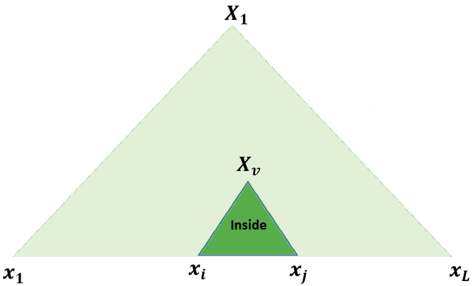
- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\alpha(i, j, v)$ is the joint probability of starting from the non-terminal X_v and generating $x_i, ..., x_j$.

The Inside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\alpha(i, j, v)$ is the joint probability of starting from the non-terminal X_v and generating $x_i, ..., x_j$.
- $P(x|\theta) = \alpha(1, L, 1)$

The Inside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\alpha(i, j, v)$ is the joint probability of starting from the non-terminal X_v and generating $x_i, ..., x_j$.
- $P(x|\theta) = \alpha(1, L, 1)$
- $e_v(x_i)$ is the probability of rule $X_v \rightarrow x_i$.
- $t_v(y, z)$ is the probability of rule $X_v \rightarrow X_y X_z$.
- $M = |X|$ and $L = |x|$.



The Inside Algorithm for CNFs

procedure INSIDE

Initialization:

for $i = 1$ to L, $v = 1$ to M **do**

$\alpha(i, i, v) = e_v(x_i)$

end for

Iteration:

for $i = 1$ to $L - 1, j = i + 1$ to L, $v = 1$ to M **do**

$\alpha(i, j, v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} \alpha(i, k, y)\alpha(k + 1, j, z)t_v(y, z)$

end for

Termination:

$P(x|\theta) = \alpha(1, L, 1)$

end procedure

Illustration of Recursion

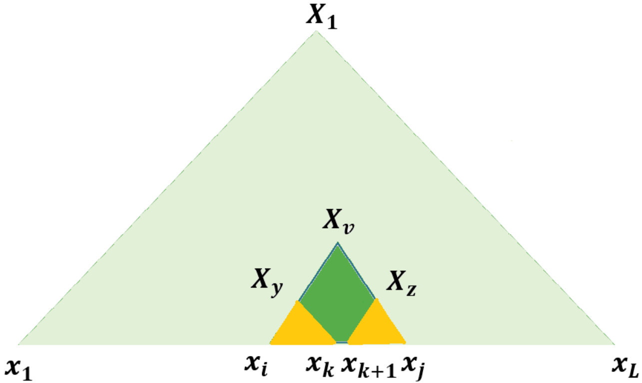


Figure 4: Illustration of the recursion calculation of $\alpha(i, j, v)$.

The Outside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.

The Outside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\beta(i, j, v)$ is the probability of a parse tree rooted at start state X_1 (S) for the complete sequence sequence x, excluding all parse subtrees for the subsequence $x_i, ..., x_j$ rooted at nonterminal X_v.

The Outside Probability for CNFs

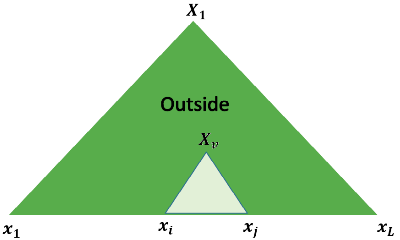
- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\beta(i, j, v)$ is the probability of a parse tree rooted at start state X_1 (S) for the complete sequence sequence x, excluding all parse subtrees for the subsequence $x_i, ..., x_j$ rooted at nonterminal X_v.
- $P(x|\theta) = \sum_{v=1}^{M} \beta(i, i, v) e_v(x_i)$ for any i.

The Outside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\beta(i, j, v)$ is the probability of a parse tree rooted at start state X_1 (S) for the complete sequence sequence x, excluding all parse subtrees for the subsequence $x_i, ..., x_j$ rooted at nonterminal X_v.
- $P(x|\theta) = \sum_{v=1}^{M} \beta(i, i, v) e_v(x_i)$ for any i.
- $e_v(x_i)$ is the probability of rule $X_v \rightarrow x_i$.

The Outside Probability for CNFs

- We want to calculate the probability of a sequence $x = x_1, ...x_L$ according to grammar G with parameters θ, i.e., $P(x|\theta)$.
- $\beta(i, j, v)$ is the probability of a parse tree rooted at start state X_1 (S) for the complete sequence sequence x, excluding all parse subtrees for the subsequence $x_i, ..., x_j$ rooted at nonterminal X_v.
- $P(x|\theta) = \sum_{v=1}^{M} \beta(i, i, v) e_v(x_i)$ for any i.
- $e_v(x_i)$ is the probability of rule $X_v \rightarrow x_i$.
- $t_v(y, z)$ is the probability of rule $X_v \rightarrow X_y X_z$.
- $M = |X|$ and $L = |x|$.



procedure INSIDE ▷

Initialization:

$\beta(1, L, 1) = 1$

for $v = 2$ to M **do**

$\beta(1, L, v) = 0$

end for

Iteration:

for $i = 1$ to $L, j = L$ to i, $v = 1$ to M **do**

$\beta(i, j, v) = \sum_{y,z} \sum_{k=1}^{i-1} \alpha(k, i-1, z)\beta(k, j, y)t_y(y, z)$ + $\sum_{y,z} \sum_{k=j+1}^{L} \alpha(j+1, k, z)\beta(i, k, y)t_y(v, z)$

end for

Termination:

$P(x|\theta) = \sum_{v=1}^{M} \beta(i, i, v)e_v(x_i)$ for any i.

end procedure

Illustration of Recursion

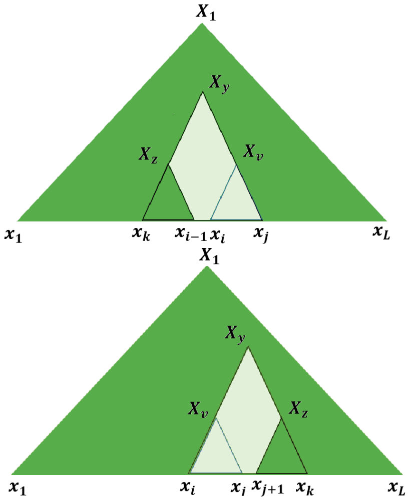


Figure 5: Illustration of the recursion calculation of $\beta(i, j, v)$. The first diagram corresponds to the first part of the recursion and the second diagram corresponds to the second part of the recursion.

The Cocke-Younger-Kasami (CYK) Algorithm for CNFs

procedure CYK ▷ Finds the most likely parse $\hat{t}$ of a sequence x.

Initialization:

for $i = 1$ to L, $v = 1$ to M **do**

$\gamma(i, i, v) = \log e_v(x_i)$

$\tau(i, i, v) = (0, 0, 0)$

end for

Iteration:

for $i = 1$ to $L - 1, j = i + 1$ to L, $v = 1$ to M **do**

$\gamma(i, j, v) = \max_{y,z} \max_{k=i..j-1} \gamma(i, k, y) + \gamma(k + 1, j, z) + \log t_v(y, z)$

$\tau(i, j, v) = \text{argmax}_{(y,z,k),k=i..j-1} \gamma(i, k, y) + \gamma(k + 1, j, z) + \log t_v(y, z)$

end for

Termination:

$\log P(x, \hat{t}|\theta) = \tau(1, L, 1)$.

end procedure

The Cocke-Younger-Kasami (CYK) Traceback

procedure CYK TRACEBACK

Initialization:

Push $(1, L, 1)$ on the stack.

Iteration:

Pop (i, j, v).

$(y, z, k) = \tau(i, j, v)$.

if $\tau(i, j, v) = (0, 0, 0)$ (implying $i == j$) **then**

Attach x_i as the child of v.

else

Attach y, z to parse tree as children of v.

Push $(k + 1, j, z)$.

Push (i, k, y).

end if

end procedure

General Schema for Certain (Multinomial Distributions) EM Algorithms [6] [8]

- Given two events, x and y, the maximum likelihood estimation (MLE) for their conditional probability is:

$$P(x|y) = \frac{count(x,y)}{count(x)} \tag{5}$$

- If they are observable, it is easy to see what to do: just count the events in a representative corpus and use the MLE or a smoothed distribution.

General Schema for Certain (Multinomial Distributions) EM Algorithms [6] [8]

- What if these are hidden variables that cannot be observed directly?
- Use a model θ and iteratively improve the model based on a corpus of observable data (O) generated by the hidden variables:

$$P_{\hat{\theta}}(x|y) = \frac{E_{\mu}[(count(x,y)|O]}{E_{\mu}[count(x)|O]]} \tag{6}$$

- It is worth noting that if you know how to calculate the numerator, the denominator is trivially derivable.
- By updating θ and iterating, the model converges to at least a local maximum.

The Inside-Outside Algorithm for CNFs (Parameter Re-estimation by Expectation Maximization)

1. Begin with a given grammar topology and some initial probability estimates for rules.
   - $\forall X \in N, \sum_{X \to \gamma} P(X \to \gamma) = 1, P(X \to \gamma) \geq 0$
2. The probability of each parse of a training sequence according to G will act as our confidence in it.
3. Sum the probabilities of each rule being used in each place to give an expectation of how often each rule was used.
4. Use the expectations to refine the probability estimates - increase the likelihood of the training corpus according to G.

The Inside-Outside Algorithm for CNFs

- The expected number of times that X_v is used in a derivation:

$$c(v) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \alpha(i,j,v)\beta(i,j,v) \tag{7}$$

- The expected number of times that X_v is occupied and then production rules $X_v \rightarrow X_y X_z$ is used:

$$c(v \rightarrow yz) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} \beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z) \tag{8}$$

The Inside-Outside Algorithm for CNFs

- MLE for $v \to yz$ given v:

$$\hat{t}_v(y,z) = \frac{c(v \to yz)}{c(v)}$$
$$= \frac{\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\sum_{k=i}^{j-1}\beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z)}{\sum_{i=1}^{L}\sum_{j=i}^{L}\alpha(i,j,v)\beta(i,j,v)} \tag{9}$$

The Inside-Outside Algorithm for CNFs

- MLE for $v \to yz$ given v:

$$\hat{t}_v(y,z) = \frac{c(v \to yz)}{c(v)}$$
$$= \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} \beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z)}{\sum_{i=1}^{L} \sum_{j=i}^{L} \alpha(i,j,v)\beta(i,j,v)} \tag{9}$$

- Similarly, MLE for $v \to a$ given v:

$$\hat{e}_v(a) = \frac{c(v \to a)}{c(v)} = \frac{\sum_{i|x_i=a} \beta(i,i,v)e_v(a)}{\sum_{i=1}^{L} \sum_{j=1}^{L} \alpha(i,j,v)\beta(i,j,v)} \tag{10}$$

The Inside-Outside Algorithm for CNFs

- MLE for $v \to yz$ given v:

$$\hat{t}_v(y,z) = \frac{c(v \to yz)}{c(v)}$$
$$= \frac{\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\sum_{k=i}^{j-1}\beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z)}{\sum_{i=1}^{L}\sum_{j=i}^{L}\alpha(i,j,v)\beta(i,j,v)} \tag{9}$$

- Similarly, MLE for $v \to a$ given v:

$$\hat{e}_v(a) = \frac{c(v \to a)}{c(v)} = \frac{\sum_{i|x_i=a}\beta(i,i,v)e_v(a)}{\sum_{i=1}^{L}\sum_{j=1}^{L}\alpha(i,j,v)\beta(i,j,v)} \tag{10}$$

- In the case of multiple independent observed sequences, expected counts are summed over all sequences.

Grammar Learning

- **Grammar learning** refers to the learning of a formal grammar from a set of observations, thus constructing a model which accounts for the characteristics of the observed objects.
- Presentation set:
  - Text (only positive examples).
  - Informant (both positive and negative examples).
- Learning methods:
  - Supervised (use structured sequences).
  - Unsupervised (uses unstructured sequences).
  - Semi-supervised (uses unstructured data to improve supervised learning tasks).

A Survey of Grammatical Inference Methods for Natural Language Learning [2]

	Presentation set		Type of information		
	Text	Informant	Supervised	Unsupervised	Semi-supervised
ADIOS	X			X	
EMILE		X	X		
e-GRIDS	X			X	
CLL	X			X	
CDC	X			X	
INDUCTIVE CYK		X		X	
LAgtS	X			X	
GA-based		X	X		
ALLis	X		X		
ABL	X			X	
UnsuParse	X			X	
Incremental parsing	X			X	
Self-training	X				X
Co-training	X				X

Goal

- Given a corpus of raw text (unstructured positive examples) separated into sequences, we want to derive a specification of the underlying grammar.
- This means we want to
  - Create new unseen grammatically correct sequences.
  - Accept new unseen grammatically correct sequences and reject ungrammatical ones.

Automatic Distillation of Structure (ADIOS) [9] [5] [7]

- **ADIOS** uses statistical information present in raw sequential data to identify significant segments and to distill rule-like regularities that support structured generalization.
- **ADIOS** has three main elements
  - A representational data structure.
  - A segmentation criterion (MEX).
  - A generalization ability.

- **Words** as **vertices** and **sequences** as **paths**.

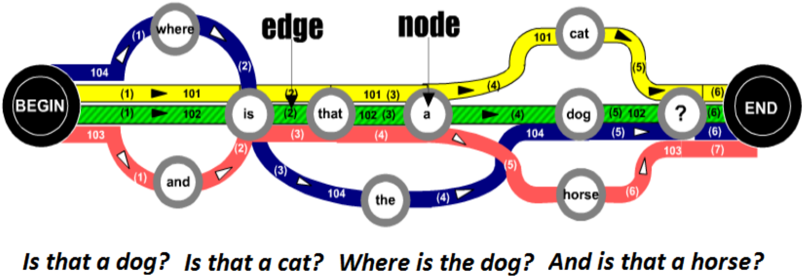


Figure 6: Image adapted from [5]

Automatic Distillation of Structure (ADIOS)

- ADIOS has three main elements
  - A representational data structure.
  - A segmentation criterion (MEX).
  - A generalization ability.

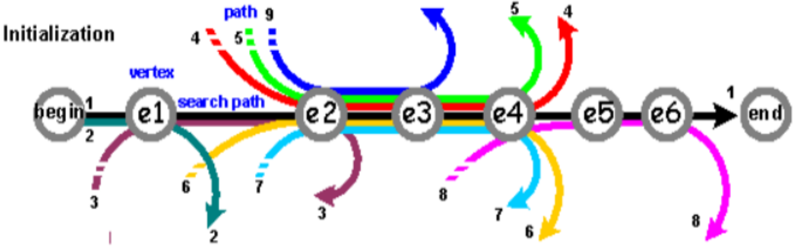


Figure 7: Image adapted from [5]

- Identifying patterns becomes easier on a graph.

Motif EXtraction (MEX)

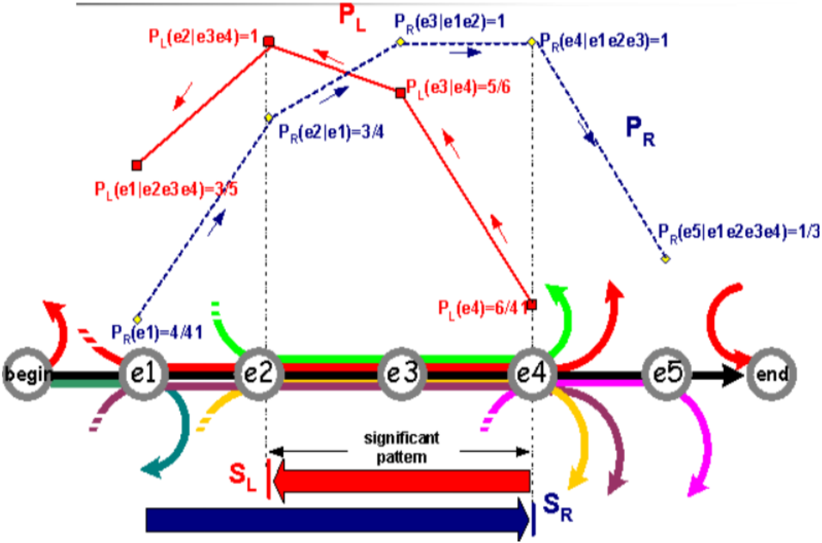
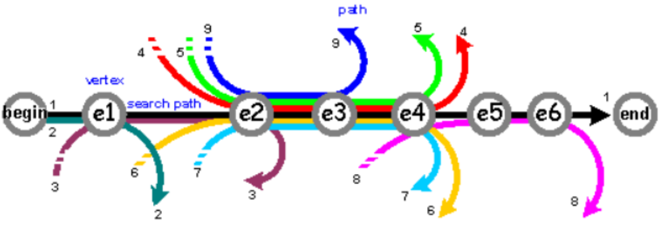


Figure 8: Image adapted from [5]



Once a pattern is identified as significant, the sub-paths it subsumes are merged into a new vertex and the graph is rewired accordingly. Repeating this process, leads to the formation of complex, hierarchically structured patterns.

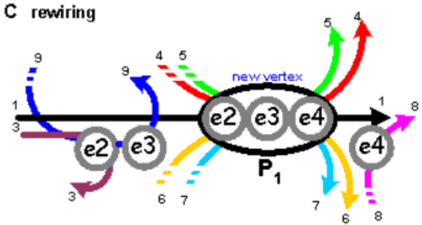


Figure 9: Image adapted from [5]

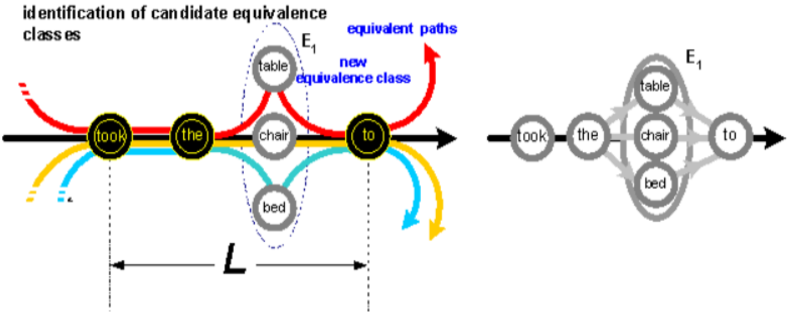
Automatic Distillation of Structure (ADIOS)

- ADIOS has three main elements
  - A representational data structure.
  - A segmentation criterion (MEX).
  - A generalization ability.

Generalization: Defining an Equivalence Class

Figure 10: Image adapted from [7]

Generalization



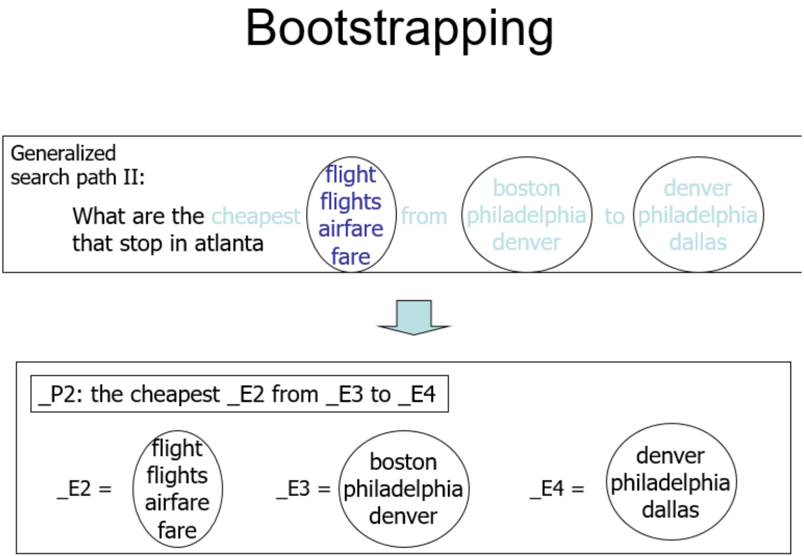


Figure 11: Image adapted from [7]

Bootstrapping

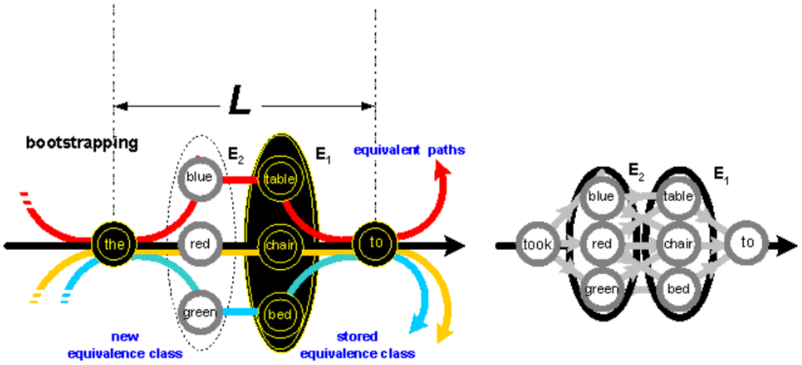


Figure 12: Image adapted from [5]

The ADIOS Algorithm Outline

- Initialization - load all data into a pseudograph
- Until no more patterns are found
  - For each path P
    - Create generalized search paths from P
    - Detect significant patterns using MEX
    - If found, add best new pattern and equivalence classes and rewire the graph

ADIOS:Context-free Mode Example

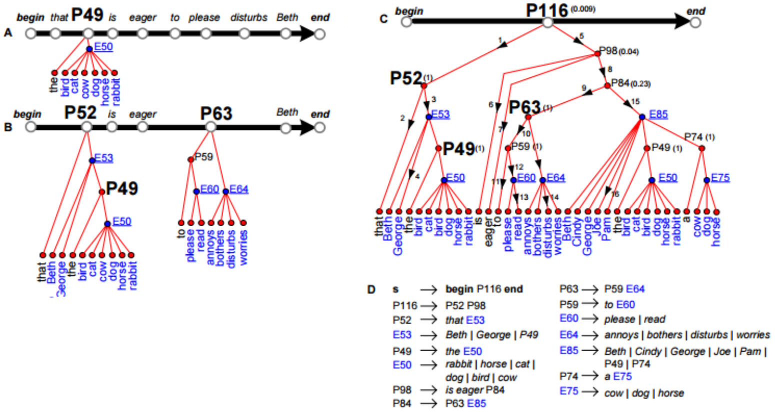


Figure 13: Adapted from [4]

Evaluating Performance

- **Recall**: the probability of ADIOS recognizing an unseen gramatical sequence.
  - Can be assessed by leaving out some of the training corpus.
- **Precision**: the proportion of grammatical ADIOS productions.
  - Can be evaluated by **external referee** (e.g., by a human subject).

ADIOS Drawback

- ADIOS is a heuristic.
  - Once a pattern is created it remains forever - errors conflate
  - Sequence ordering affects outcome.
- Running ADIOS with different orderings gives patterns that *cover* different parts of the grammar.

An Ad-hoc Solution

- Train multiple learners on the corpus.
  - Each on a different sequence ordering.
  - Create a **forest** of learners.
- To create a new sequence
  - Pick one learner at random.
  - Use it to produce sequence.
- To check grammaticality of given sequence
  - If any learner accepts sequence, declare as grammatical.

References I

- These are some of the sources I used to prepare these slides!

[1] Michael Collins. "Head-driven statistical models for natural language parsing". In: Computational linguistics 29.4 (2003), pp. 589–637.

[2] Arianna DUlizia, Fernando Ferri, and Patrizia Grifoni. "A survey of grammatical inference methods for natural language learning". In: Artificial Intelligence Review 36.1 (2011), pp. 1–27.

[3] Richard Durbin et al. Biological sequence analysis. Cambridge university press, 1998.

References II

[4] Shimon Edelman et al. “Learning syntactic constructions from raw corpora”. In: 29th Boston University Conference on Language Development. Citeseer. 2005.

[5] PowerShow.com. Probabilistic Context Free Grammar. http://www.powershow.com/view/1796e4-MjRjY/Probabilistic_Context_Free_Grammar_powerpoint_ppt_presentation. [Online; accessed 5-July-2016].

[6] Philip Resnik. A Simple Recipe for EM Update Equationsr. http://www.umiacs.umd.edu/~resnik/ling773_sp2011/readings/em_recipe.v2.only_hmm.pdf. [Online; accessed 5-July-2016].

[7] Eytan Ruppin.
Boosting Unsupervised Language Acquisition with ADIOS .
www.cri.haifa.ac.il/events/2006/machine_learning/presentations/haifa3.ppt. [Online; accessed 5-July-2016].

[8] Noah A Smith. "Linguistic structure prediction". In: Synthesis lectures on human language technologies 4.2 (2011), pp. 1–274.

[9] Zach Solan et al. "Unsupervised learning of natural languages". In: Proceedings of the National Academy of Sciences of the United States 102.33 (2005), pp. 11629–11634.

[10] Zhuowen Tu et al. “Image parsing: Unifying segmentation, detection, and recognition”. In: International Journal of computer vision 63.2 (2005), pp. 113–140.