Grammar Induction

Regina Barzilay

MIT

October, 2005

Three non-NLP questions

1. Which is the odd number out?

   625,361,256,197,144

2. Insert the missing letter:

   B,E,?,Q,Z

3. Complete the following number sequence:

   4, 6, 9, 13

   7, 10, 15, ?

How do you solve these questions?

- Guess a pattern that generates the sequence

Insert the missing letter:
B,E,?,Q,Z
2,5,?,17, 26
$k^2 + 1$

- Select a solution based on the detected pattern
  $k = 3 \rightarrow$ 10th letter of the alphabet $\rightarrow J$

More Patterns to Decipher: Byblos Script

Image removed for copyright reasons.

More Patterns to Decipher: Lexicon Learning

Ourenemiesareinnovativeandresourceful,andsoarewe. Theyneverstopthinkingaboutnewwaystoharmourcountry andourpeople,andneitherdowe.

Which is the odd word out?

Ourenemies . . .
Enemies . . .
We . . .

More Patterns to Decipher: Natural Language Syntax

Which is the odd sentence out?

The cat eats tuna.
The cat and the dog eats tuna.

Today

- Vocabulary Induction
  - Word Boundary Detection
- Grammar Induction
  - Feasibility of language acquisition
  - Algorithms for grammar induction

Vocabulary Induction

Task: Unsupervised learning of word boundary segmentation

- Simple:

  Ourenemiesareinnovativeandresourceful,andsoarewe.

  Theyneverstopthinkingaboutnewwaystoharmourcountry

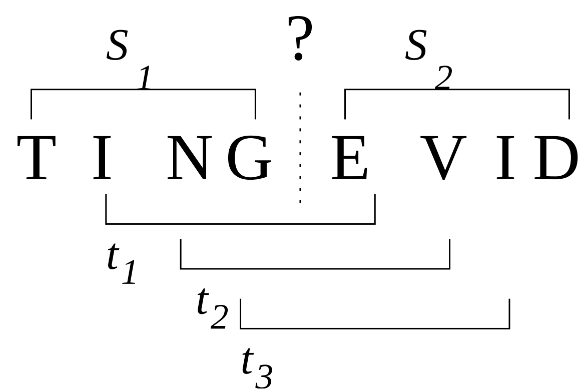
  andourpeople,andneitherdowe.

- More ambitious:

Image of Byblos script removed for copyright reasons.

Word Segmentation (Ando&Lee, 2000)

Key idea: for each candidate boundary, compare the frequency of the n-grams adjacent to the proposed boundary with the frequency of the n-grams that straddle it.



For $N = 4$, consider the 6 questions of the form:
"Is $\#(s_i) \geq \#(t_j)$?", where #(x) is the number of occurrences of x

Example: Is "TING" more frequent in the corpus than "INGE"?

Algorithm for Word Segmentation

s_1^n	non-straddling n-grams to the left of location k
s_2^n	non-straddling n-grams to the right of location k
t_j^n	straddling n-gram with j characters to the right of location k
$I_{\geq}(y, z)$	indicator function that is 1 when $y \geq z$, and 0 otherwise.

1. Calculate the fraction of affirmative answers for each n in N:

$$v_n(k) = \frac{1}{2*(n-1)} \sum_{i=1}^{2} \sum_{j=1}^{n-1} I_{\geq}(\#(s_i^n), \#(t_j^n))$$

2. Average the contributions of each $n-gram$ order

$$v_N(k) = \frac{1}{N} \sum_{n \in N} v_n(k)$$

Algorithm for Word Segmentation (Cont.)

Place boundary at all locations l such that either:

- l is a local maximum: $v_N(l) > v_N(l-1)$ and $v_N(l) > v_N(l+1)$
- $v_N(l) \geq t$, a threshold parameter

Experimental Framework

- Corpus: 150 megabytes of 1993 Nikkei newswire
- Manual annotations: 50 sequences for development set (parameter tuning) and 50 sequences for test set
- Baseline algorithms: Chasen and Juman morphological analyzers (115,000 and 231,000 words)

Evaluation

- Precision (P): the percentage of proposed brackets that exactly match word-level brackets in the annotation
- Recall (R): the percentage of word-level annotation brackets that are proposed by the algorithm
- $F = 2\frac{PR}{(P+R)}$
- $F = 82\%$ (improvement of 1.38% over Jumann and of 5.39% over Chasen)

Grammar Induction

- Task: Unsupervised learning of a language's syntax from a corpus of observed sentences
  - Ability to uncover an underlying grammar
  - Ability to parse
  - Ability to judge grammaticality

Plato's Problem

Logical problem of language acquisition:
(Chomsky 1965, Pinker 1994, Pullum 1996)

- A child hears a finite number of utterances from a target language
- This finite experience is consistent with infinitely many targets
- The child manages to select the correct target language

Gold's Formalization(1967)

- Given: A target language L from a set $\mathcal{L}$ of possible languages
- A learner C is shown a set of positive examples $[s_i], s_i \in L$
- C is never given negative examples
- Each $s \in L$ will be presented at some point i (no guarantees on the order or frequency of examples)
- C maintains a hypothesis $L(C, [s_0, \ldots, s_n]) \in \mathcal{L}$

Identifiability in the Limit

- A language family $\mathcal{L}$ is identifiable in the limit if for any target language and example sequence, the learner's hypothesis is eventually correct

- A language family $\mathcal{L}$ is identifiable in the limit if there is some learner C such that, for any $L \in \mathcal{L}$ and any legal presentation of examples $[s_i]$, there is some point k such that for all $j > k$, $L(C, [s_0, \ldots, s_k]) = L$

Example: $\mathcal{L} = \{\{a\}, \{a, b\}\}$

Gold's Results

A wide variety of language families are not learnable (proof based on recursive function theory)

- Superfinite family (all the finite languages and at least one infinite language)
- Family of regular languages
- Family of context-free languages

Issues to Consider (Pullman 2003)

- Learners may receive considerable information about which strings are not grammatical (perhaps indirectly)
- It is not clear that real language learners ever settle on a grammar at all
- Learners could *approximate* rather than exactly identify grammars
- The learner may operate over strings paired with meaning
- Learning can be viewed as partial characterization of linguistic structure (rather than defining a unique set of grammatical strings)

Horning(1969): probabilistic context free grammars are learnable if some Gold’s constraints are relaxed

Nativism

- *Poverty of stimulus* (Chomsky, 1965): the lack of crucial relevant data in the learner's experience
- Richness of constraint: human languages are highly constrained, since the actual family of human languages is relatively small

Grammar Induction: Evaluation

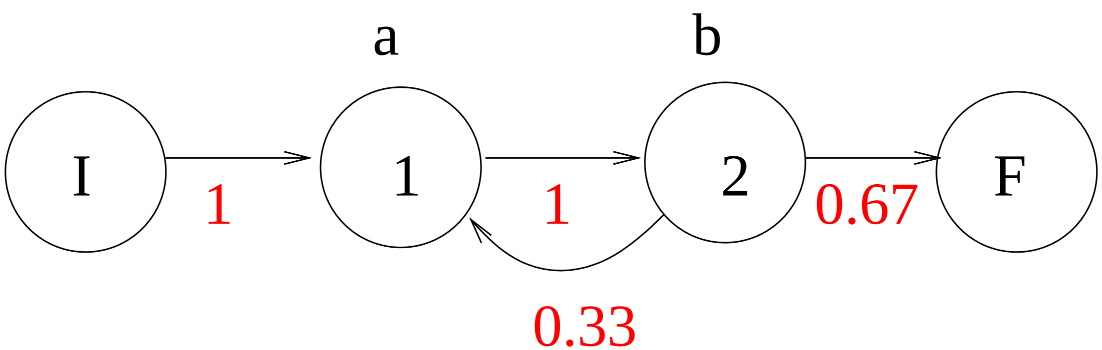
- Evaluation
  - Compare grammars
  - Compare trees
- Baselines
  - Random trees
  - Left- and Right-Branching Trees

Grammar Induction: Approaches

- Structure search
  - Add productions to a context-free grammar
  - Select HMM topology
- Parameter search
  - Determine parameters for a fixed PCFG

Structure search: Example

- Input: $\{ab, abab\}$
- Possible output: $L = (ab)^n$



Model Merging

- A method to construct an initial model from data
- A way to merge submodels
- An error measure to compare the goodness of various candidates for merging and to limit generalization
- A strategy to pick merging operators, search the model space

Model Merging (Stolcke&Omohundro, 1994)

- **Data Incorporation:** Given a body of data X, build an initial model M_0 by explicitly accommodating each data point individually

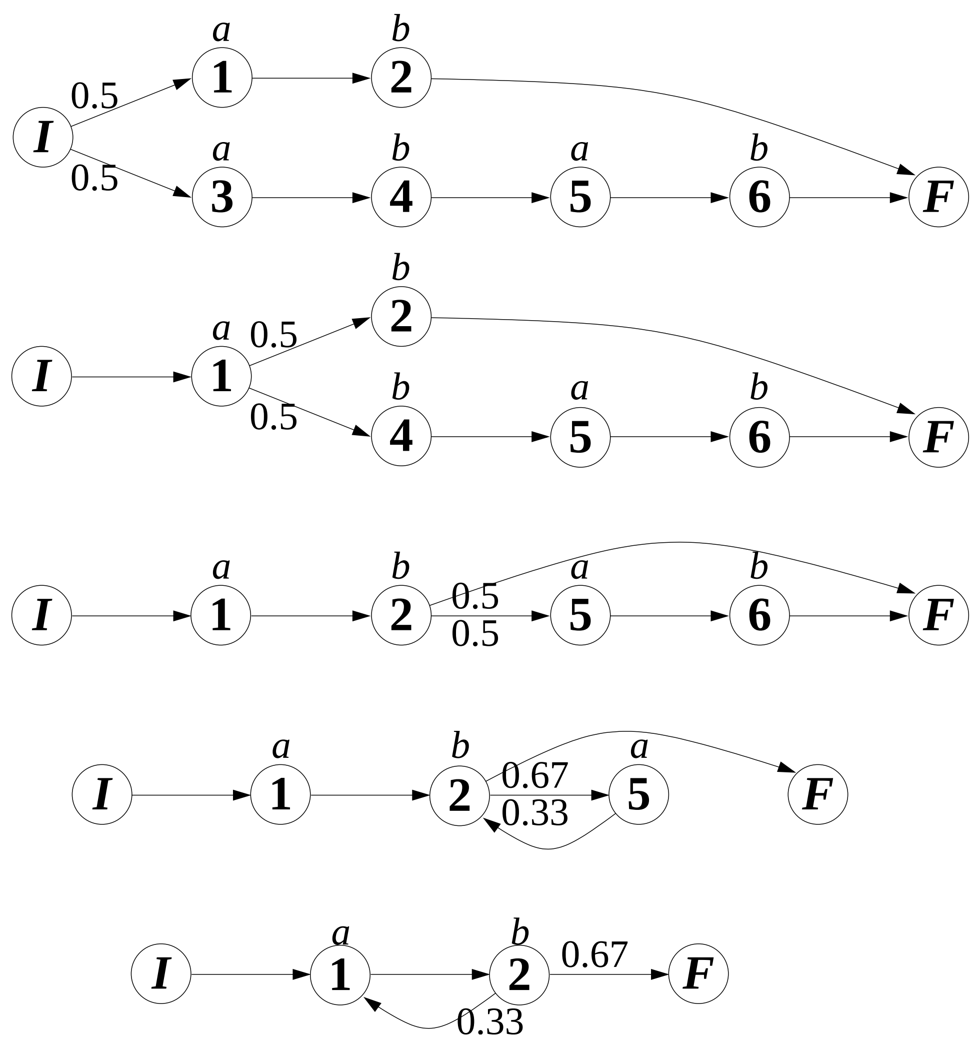
- **Generalization:** Build a sequence of new models, obtaining M_{i+1} from M_i by applying a merging operator m that coalesces substructures in M_i, $M_{i+1} = m(M_i)$

- **Utility function:** Maximize posterior probability $P(M|X)$

- **Search:** Greedy or beam search through the space of possible merges

HMM Topology Induction

- **Data Incorporation:** For each observed sample, create a unique path between the initial and final states by assigning a new state to each symbol token in the sample
- **Generalization:** Two HMM states are replaced by a single new state, which inherits the union of the transitions and emissions from the old states



a
b
0.5
0.5
a
b
a
b
I
1
2
3
4
5
6
F
b
a
0.5
0.5
b
a
b
I
1
2
4
5
6
F
a
b
a
b
0.5
0.5
I
1
2
5
6
F
a
b
a
0.67
0.33
I
1
2
5
F
a
b
0.67
0.33
I
1
2
F

Posterior Computation

Goal: maximize posterior $P(M|X) = \frac{P(M)P(X|M)}{P(X)}$

- We will maximize $P(M|X) \propto P(M)P(X|M)$
- We know how to compute $P(X|M)$
- We need to compute prior $P(M)$

Prior Distribution

Model M is defined by topology M_s and θ_M

$$P(M) = P(M_s)P(\theta_M|M_s)$$

- $P(M_s) \propto \exp(-l(M_s))$, where $l(M_s)$ is the number of bits required to encode M_s
  - Each transition is encoded using $\log(|Q|+1)$ bits, where $|Q|$ is the number of states
  - The total description length for all transitions from state q is $n_t^{(q)} \log(|Q|+1)$ bits, where $n_t^{(q)}$ – the number of transitions from state q

- The total emission length for state q is $n_e^{(q)} \log(|\Sigma| + 1)$ bits, where $n_e^{(q)}$ – the number of state q emissions, and $|\Sigma|$ is the size of the alphabet
- The resulting prior

$$P(M_s^{(q)}) \propto (|Q| + 1)^{-n_t^{(q)}} (|\Sigma| + 1)^{-n_e^{(q)}}$$

- $P(\theta_M | M_s)$ are defined as Dirichlet priors

Algorithm

1. Build the initial, maximum-likelihood model M_0 from the dataset X
2. Let $i := 0$. Loop:
   (a) Compute a set of candidate merges K among the states of model M_i
   (b) For each candidate $k \in K$ compute the merged model $k(M_i)$, and its posterior probability $P(k(M_i)|X)$
   (c) Let k^* be the merge that mazimizes $P(k(M_i)|X)$. Then let $M_{i+1} := k^*(M_i)$
   (d) If $P(M_{i+1}|X) > P(M_i|X)$, return M_i as the induced model.
   (e) Let $i := i + 1$

Evaluation

Method	Cross-Entropy	Language
Merging	2.158	$ac^*a \cup bc^*b$
Baum-Welch+	2.105	
Baum-Welch-	2.825	
Merging	5.623	$a^+b^+a^+b^+$
Baum-Welch+	5.688	
Baum-Welch-	8.395	

Learning PCFGs

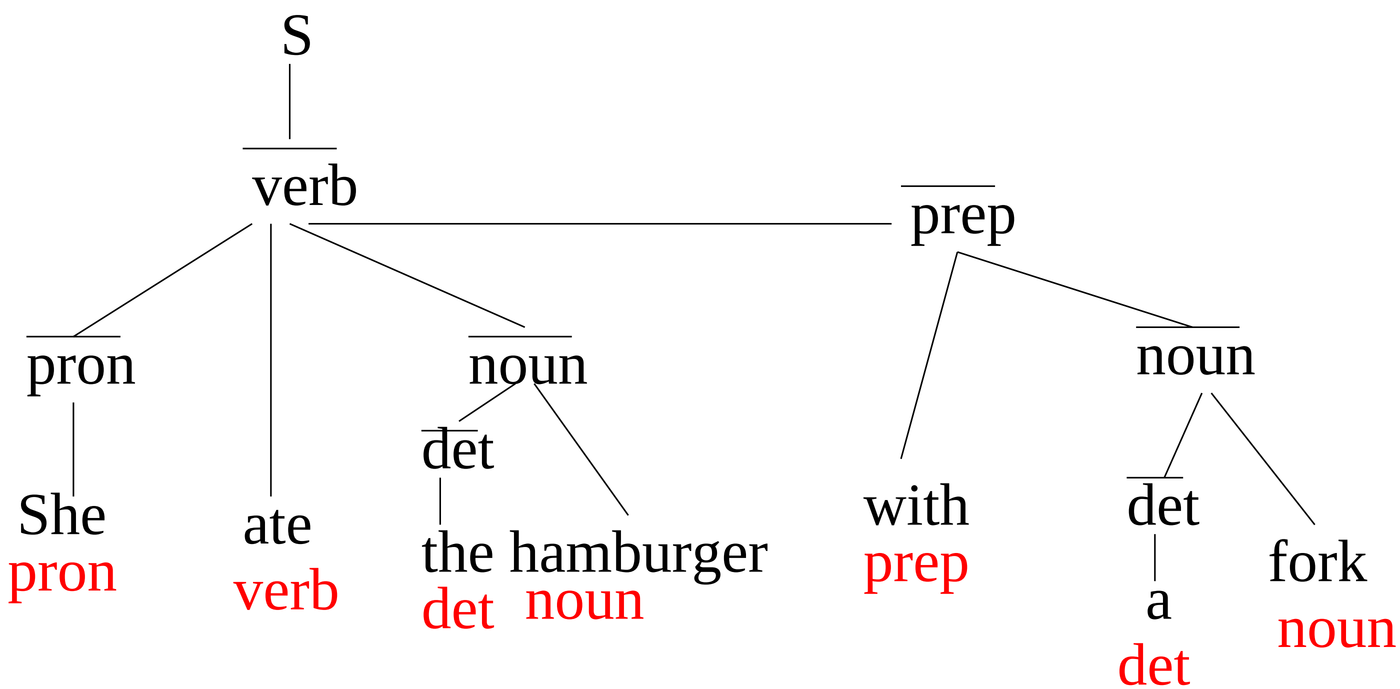
(Carroll&Charniak, 1992)

Goal: Learning grammars for natural language

- Divide the corpus into two parts: the rule corpus and the training corpus.
- For all the sentences in the rule corpus, generate all rules which might be used to parse the sentence, subject to constraints which we will specify later.
- Estimate the probabilities for the rules.
- Using the training corpus, improve our estimate of probabilities.
- Delete all rules with probability $\leq \delta$ for some small δ.

Rule Generation: Dependency Format

Informally, a dependency grammar produces a set of terminals connected by a set of directed arcs — one arc for every terminal except the root terminal



Dependency Grammar

- Target: a dependency grammar $< S, N, R >$
  S is the start symbol
  N is a set of terminals
  R is a set of rewrite rules, where
  $R \subseteq \{S \rightarrow \bar{n} | n \in N\} \cup \{\bar{n} \rightarrow \alpha n \beta | n \in N, \alpha, \beta \in \Gamma\}$,
  Γ is a set of strings of zero or more $\bar{a}$, for $a \in N$

- Assumption: POS tags are provided

- Theorem: A sentence of length n, consisting of all distinct terminals will have $n(2^{n-1} + 1)$ dependency grammar rules to confirm to it

Rule Generation

We have to prune rule space!

- Order sentences by length and generate rules incrementally
- Do not consider rules that were discarded on previous stages
- Limit the number of symbols on the right-hand side of the rule

Algorithm

Loop for i from 2 until $i >$ sentence-length-stopping point

> Add rules required for the sentences with length i from the rule creation subset
>
> Estimate the probabilities for all rules, based upon all sentences of length $\leq i$ from the rule training subset
>
> Remove any rules with probability $\leq \delta$ if its probability doesn't increase

Reestimation

- We have sentences $S_1, \ldots, S_n$. Trees are hidden variables.

$$L(\theta) = \sum_i \log \sum_T P(S_i, T|\theta)$$

- Basic quantity needed for re-estimating with EM:

$$\theta_{\alpha \to \beta} = \frac{\sum_i Count(S_i, \alpha \to \beta)}{\sum_i \sum_{s \in R(\alpha)} Count(S_i, s)}$$

- There are efficient algorithms for calculating

$$Count(S_i, r) = \sum_T P(T|S_i, \theta^{t-1}) Count(S_i, T, r)$$

for a PCFG. See Inside-Outside algorithm (Baker, 1979)

Example

Induce PCFG, given the following corpus:

“noun verb”
“verb noun”
“verb”
“det noun verb”
“verb det noun”

Rule			1 ITER	6 ITER	20 ITER
S	$\longrightarrow$	$\bar{det}$	0.181818	0.0	0.0
S	$\longrightarrow$	$\bar{noun}$	0.363636	0.0	0.0
S	$\longrightarrow$	$\bar{verb}$	0.454545	1.0	1.0
$\bar{det}$	$\longrightarrow$	det	0.250000	1.0	1.0
$\bar{det}$	$\longrightarrow$	$det\ \bar{noun}$	0.250000	0.0	0.0
$\bar{det}$	$\longrightarrow$	$det\ \bar{verb}$	0.125	0.0	0.0
$\bar{det}$	$\longrightarrow$	$verb\ \bar{det}$	0.125	0.0	0.0
$\bar{det}$	$\longrightarrow$	$verb\ \bar{det}\ \bar{noun}$	0.125	0.0	0.0
$\bar{noun}$	$\longrightarrow$	$noun$	0.333333	0.781317	0.998847
$\bar{noun}$	$\longrightarrow$	$\bar{det}\ noun$	0.166667	0.218683	0.01153
$\bar{verb}$	$\longrightarrow$	$\bar{noun}\ verb$	0.153846	0.286749	0.200461
$\bar{verb}$	$\longrightarrow$	$verb\ \bar{noun}$	0.153846	0.288197	0.200461

Experiment 1

- Use grammar from the handout
- Randomly generate 1000 words for the rule corpus, and 9000 for the training corpus
- Evaluation: compare the output with the generated grammar
- Constraint: rules were required to have fewer than five symbols on their right-hand side

Results

- Successfully minimizes a cross entropy (1.245 bits/word on the training of the learned grammar vs. 1.220 bits/word of the correct grammar)
- Miserably fails to recover the correct grammar
  - 300 unsuccessful attempts

$$
\begin{array}{llll}
.220 & \bar{pron} & \rightarrow & pron\ \bar{verb} \\
.214 & \bar{pron} & \rightarrow & \bar{prep}\ pron \\
.139 & \bar{pron} & \rightarrow & pron\ \bar{verb}\ \bar{det} \\
.118 & \bar{pron} & \rightarrow & \bar{verb}\ pron
\end{array}
$$

Experiment 2

Place more restrictions on the grammar

Specify what non-terminals may appear on the right-hand side of a rule with a particular non-terminal on the left

- The algorithm converges to the correct grammar

	noun	verb	pron	det	prep	adj	wh	.
noun				+	+	+	+	
verb	+		+		+			
pron		–						
det						–		

Adding Knowledge to Grammar Induction Algorithms

- Carrol&Charniak (1992): restrictions on the rule format
- Magerman&Marcus (1990): use a di-stituent grammar to eliminate undesirable rules
- Pereira&Schabes (1992): use partially bracketed corpora

Learning Constituents

Are syntactic patterns evident in a corpus? (Klein, 2005)

- Compute context for each POS

Tag	Top Context by Frequency
DT	(IN-NN), (IN-JJ), (IN-NNP), (VB-NN)
JJ	(DT-NN), (IN-NNS), (IN-NN), (JJ-NN)

- Cluster POS based on their context

Learning Constituents

The most similar POS pairs based on their context

Rank	Tag Pairs
1	(VBZ, VBD)
2	(DT, PRP$)
3	(NN, NNS)
4	(WDT, WP)
5	(VBG, VBN)

Learning Constituents

The most similar POS sequence pairs based on their context

Rank	Tag Pairs
1	(NNP NNP, NNP NNP NNP)
2	(DT JJ NN IN, DT NN IN)
3	(NNP NNP NNP NNP, NNP NNP NNP)
4	(DT NNP NNP, DT NNP)
5	(IN DT JJ NN, IN DT NN)

Learning Constituents (Clark, 2001)

- Identify frequent POS sequences in a corpus
- Cluster them based on their context
- Filter out spurious candidates
  - Based on mutual information before the candidate constituent and the symbol after — they are not independent

Summary

- Language acquisition problem
- Three unsupervised induction algorithms:
  - Vocabulary Induction
  - HMM-topology induction
  - PCFG induction