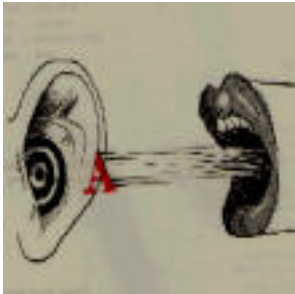


Lecture 19: Language Acquisition II



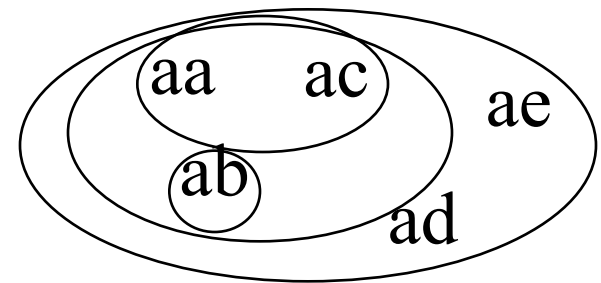
Professor Robert C. Berwick
berwick@csail.mit.edu

The Menu Bar

- Administrivia: lab 5-6 due this Weds!
- Language acquisition – the “Gold standard” & basic results *or* “the (Evil) Babysitter is Here” (apologies to Dar Williams)
 - Informal version
 - Formal version
- Can we meet the Gold standard? What about probabilistic accounts?
- Stochastic CFGs & Bayesian learning

Conservative Strategy

- Baby's hypothesis should always be smallest language consistent with the data
- Works for finite languages? Let's try it ...
 - **Language 1:** {aa,ab,ac}
 - **Language 2:** {aa,ab,ac,ad,ae}
 - **Language 3:** {aa,ac}
 - **Language 4:** {ab}



Babysitter	aa	ab	ac	ab	aa	...
Baby	L3	L1	L1	L1	L1	

Evil Babysitter

- To find out whether Baby is perfect, we have to see whether it gets 100% correct even in the most adversarial conditions
- Assume Babysitter is trying to *fool* Baby
 - although she must speak only sentences from L_T
 - and she must eventually speak each such sentence
- Does C-Baby's strategy work on *every* possible 'fair sequence' for *every* possible language?

In finite # of languages case, Yes – why?

A learnable (“identifiable”) family of Languages

- Family of languages:
 - Let $L_n =$ set of all strings of length $< n$, over some fixed alphabet $= \{a, b\}$
 - What is L_0 ?
 - What is L_1 ?
 - What is L_n ?
 - Let the family $\mathcal{L} = \{L_0, L_1, \dots, L_n\}$
 - No matter what the L_i can Babysitter really follow rules?
 - Must eventually speak every sentence of L . Is this possible?
 - Yes: \emptyset ; **a, b**; **aa, ab, ba, bb**; **aaa, aab, aba, abb, baa,**

An Unlearnable Family of Languages: so-called “Superfinite” family

- Let $L_n =$ set of all strings of length $< n$
 - What is L_0 ?
 - What is L_1 ?
 - What is L_∞ ?
- Our (infinite) family is $\mathcal{L} = \{L_0, L_1, \dots, L_n, \dots, L_\infty\}$
- A perfect C-baby must be able to distinguish among all of these depending on a finite amount of input
- But there is no perfect C-baby ...

An Unlearnable Family

- Our class is $\mathcal{L} = \{L_0, L_1, \dots, L_\infty\}$
- C-Baby adopts conservative strategy, always picking smallest possible language in \mathcal{L}
- So if Babysitter's longest sentence so far has 75 words, baby's hypothesis is L_{75} .
- This won't always work for all languages in \mathcal{L}

What language can't a conservative Baby learn?

So, C-baby cannot always pick *smallest* possible language and win...

An Unlearnable Family

- Could a non-conservative baby be ‘almost’ a perfect C-Baby, and eventually converge to any of the languages in the family of languages?
- **Claim:** *Any* perfect C-Baby must be “quasi-conservative”:
 - If the true language is L_{75} , and baby posits something else, baby must still eventually come back and guess L_{75} (since it’s perfect).
 - So if longest sentence so far is 75 words, and Babysitter keeps talking from L_{75} , then eventually baby *must* actually return to the conservative guess L_{75} .
 - Agreed?

The “Evil Babysitter”

If longest sentence so far is 75 words, and Babysitter keeps talking from L_{76} , then eventually a perfect C-baby must actually return to the conservative guess L_{75} .

- But suppose the true target language is L_{∞} .
- Evil Babysitter can then prevent our supposedly perfect C-Baby from converging to it
- If Baby ever guesses L_{∞} , say when the longest sentence is 75 words:
 - Then Evil Babysitter keeps talking from L_{75} until Baby capitulates and revises her guess to L_{75} – as any perfect C-Baby must.
 - So Baby has *not* stayed at L_{∞} as required.
- Then Babysitter can go ahead with longer sentences. If Baby ever guesses L_{∞} again, she plays the same trick again (and again)

The “Evil Babysitter”

If longest sentence so far is 75 words, and Babysitter keeps talking from L_{76} , then eventually a perfect C-baby must actually return to the conservative guess L_{76} .

- Suppose true language is L_∞ .
- Evil Babysitter can prevent our supposedly perfect C-Baby from converging to it in the limit
- If Baby ever guesses L_∞ , say when the longest sentence is 75 words:
 - Then Evil Babysitter keeps talking from L_{76} until Baby capitulates and revises her guess to L_{76} – as any perfect C-Baby must.
 - So Baby has *not* stayed at L_∞ as required.
- **Conclusion:** There’s no perfect Baby that is guaranteed to converge to L_0, L_1, \dots or L_∞ as appropriate. If C-Baby always succeeds on finite languages, Evil Babysitter can trick it on infinite language; if C-Baby succeeds on the infinite L_∞ then Evil Babysitter can force it to never learn finite L ’s

What does this result imply?

- *Any* family of languages that includes all the finite languages and at least this one super-finite language *is not* identifiable *in the limit* from positive-only evidence
- This includes the family of all finite-state languages; the family of all context-free languages; etc. etc.

Is this too ‘adversarial’?

- Should we assume Babysitter is *evil*?
- Maybe more like Google....
- Perhaps Babysitter isn’t trying to *fool* the baby - not an adversarial situation

Formally: Notation & definitions

1. The *target language* L_T is some language drawn from the class of possible languages \mathcal{L} (equivalently, grammars \mathcal{G}). This is what the learner must identify on the basis of examples.
2. The *example sentences* $s_i \in L_T$ are drawn from L_T and presented to the learner. Here, s_i is the i^{th} such example sentence
3. The *hypothesis languages* $h \in \mathcal{H}$ are the (indexes of languages) drawn from the space of possible languages the learner constructs on the basis of the example sequences
4. The *learning algorithm* \mathcal{A} is any effective (computable) procedure by which languages from \mathcal{H} are chosen on the basis of example sentences.

Notation & definitions

The family of grammars: \mathcal{G}

The family of languages: $\mathcal{L} = \{L_g | g \in \mathcal{G}\}$

The enumerable set of hypothesis grammars: \mathcal{H}

sequence of k example sentences: (s_1, s_2, \dots, s_k)

The set of all possible sequences of k example sentences:

$$D^k = \{(s_1, \dots, s_k) | s_i \in \Sigma^*\} = (\Sigma^*)^k$$

The set of all (enumerable) finite data sequences:

$$\mathcal{D} = \cup_{k>0} D^k$$

The learning algorithm $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{H}$

Notation and definitions

Definition 1. A text t for a language L is an infinite sequence of examples $s_1, s_2, \dots, s_n, \dots$ such that:

- (i) each $s_i \in L$
- (ii) every element of L appears at least once in t (fair presentation)

Definition 2. Fix a distance metric d , a target grammar $g_T \in \mathcal{G}$, and text t for the target language L_{g_T} .

The learning algorithm \mathcal{A} identifies (learns) the target $g_T(L_{g_T})$ in the limit on the text t if:

$$\lim_{k \rightarrow \infty} d(\mathcal{A}(t_k), g_T) = 0$$

Definition 3. Given a finite sequence $x = s_1, s_2, \dots, s_n$ (of length n), the *length* of the sequence is $lh(x) = n$.

range(x) is the set of all unique elements of x

The *concatenation* of two sequences $x = x_1, \dots, x_n$ and $y = y_1, \dots, y_m$

is $x \circ y = x_1, \dots, x_n, y_1, \dots, y_m$.

Notation and definitions

If the target language or grammar g_T is identified (learned) by \mathcal{A} for *all* texts of L_{g_T} :

The learning algorithm is said to *identify (learn) g_T in the limit*.

If *all* grammars (correspondingly languages) in \mathcal{G} can be learned, then the class of grammars \mathcal{G} is *learnable*.

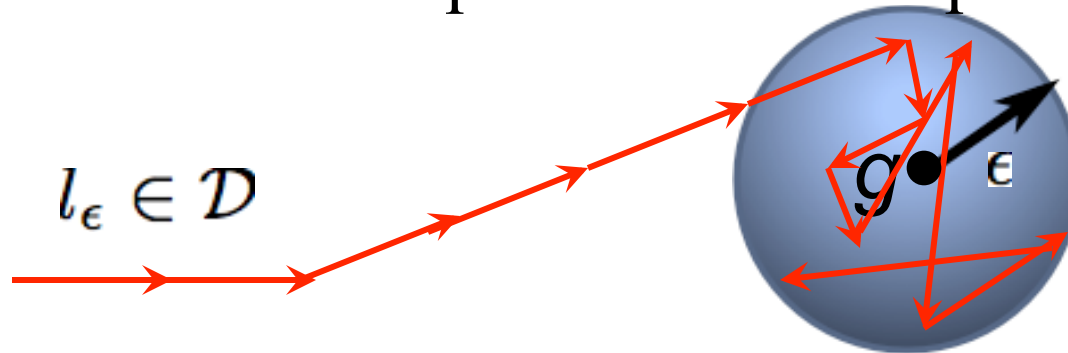
Thus learnability is equivalent to identifiability in the limit.

The “locking sequence” (evil babysitter) theorem

Theorem 1. (after Blum and Blum; ϵ -version) If \mathcal{A} identifies a target grammar g (generating language L_g) in the limit, then, for every $\epsilon > 0$, there exists a (finite) *locking data sequence* $l_\epsilon \in \mathcal{D}$ such that:

- (i) $l_\epsilon \subseteq L_g$;
- (ii) $d(\mathcal{A}(l_\epsilon), g) < \epsilon$; and
- (iii) $d(\mathcal{A}(l_\epsilon \circ \sigma), g) < \epsilon$ for all $\sigma \in \mathcal{D}$ where $\sigma \subseteq L_g$.

After ‘lock sequence’ seen, then “happily ever after”
inside the sphere of radius epsilon...



Proof

Proof: By contradiction. Assume no such locking sequence exists. Then for every $l \in \mathcal{D}$ s.t. $l \subseteq L_g$ and $d(\mathcal{A}(l), g) < \epsilon$,

$$\exists \sigma_l \in \mathcal{D} \text{ s.t. } d(\mathcal{A}(l \circ \sigma_l), g) \geq \epsilon$$

Use this fact to construct the **Evil Babysitter**

Construct “Evil babysitter” text

Construct a text for L_g on which \mathcal{A} will *not* identify the target grammar, a contradiction. Text $r = s_1, s_2, \dots, s_n, \dots$ for L_g

Construct new text q :

1. Let $q^{(1)} = s_1$.
2. If $d(\mathcal{A}(q^{(i)}), L_g) < \epsilon$, then pick $\sigma_{q^{(i)}}$ violating locking property and update text: $q^{(i+1)} = q^{(i)} \circ \sigma_{q^{(i)}} \circ s_{i+1}$.
3. Otherwise, leave text alone and just let $q^{(i+1)} = q^{(i)} \circ s_{i+1}$

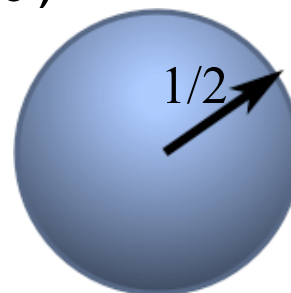
q obviously a valid text. But \mathcal{A} can never converge to g on q

Because: every time the learner conjectures a grammar h such that $d(h, g) < \epsilon$ (say at $q^{(j)}$)

The sequence $q^{(j)} \circ \sigma_{q^{(j)}}$ forces learner away from target;
s.t. $d(\mathcal{A}(q_i), L_g) > \epsilon$ infinitely many times

To get classical result for exact identification
(0–1 metric)

Set $\epsilon = \frac{1}{2}$



Theorem 2 (Blum and Blum 1975) If \mathcal{A} identifies a target grammar g in the limit, then there exists a finite locking data sequence $l \in \mathcal{D}$ such that

- (i) $l \subseteq L_g$;
- (ii) $d(\mathcal{A}(l), g) = 0$; and
- (iii) $d(\mathcal{A}(l \circ \sigma), g) = 0$ for all $\sigma \in \mathcal{D}$ where $\sigma \subseteq L_g$

Classic Gold Result (“Superfinite theorem”)

Theorem 3 (Gold 1967) If the family of languages \mathcal{L} consists of all the finite languages and at least one infinite language, then it is not learnable (identifiable) in the limit.

Proof: By contradiction.

Suppose \mathcal{A} can identify the family \mathcal{L} .

Therefore, \mathcal{A} can identify the infinite language L_∞ .

Therefore, \exists **finite** locking sequence for L_∞ , call it σ_{inf} .

But $L = \text{range}(\sigma_{\text{inf}})$ is a **finite** language, and so $L \in \mathcal{L}$

Then $t_k = \sigma_{\text{inf}}$, $k = \text{length}(\sigma_{\text{inf}})$ is a text for L .

Since \mathcal{A} learns L on all fair texts for L , it must converge to L on t_k .

Therefore, \mathcal{A} does not identify L_∞ , a contradiction.

Extensions reveal the breadth of Gold's result

1. $L \subseteq \Sigma^*$ – this is the central and traditional notion of a language both in computer science and linguistics.
2. $L \subseteq \Sigma_1^* \times \Sigma_2^*$ – a subset of form-meaning pairs and in a formal sense no different from notion 1.
3. $L : \Sigma^* \rightarrow [0, 1]$ – a language maps every expression to a real number between 0 and 1.
4. L is a probability distribution μ on Σ^* – this is the usual notion of a language in statistical language modeling.
5. L is a probability distribution μ on $\Sigma_1^* \times \Sigma_2^*$.

What happens if we ‘go probabilistic’?

- Everyone always *complains* about the Gold results...
- Gold is “too stringent” about way texts are used... identification on *all* texts.
- Suppose we relax this... get *measure-1 learnability*
- Upshot is: this does not enlarge the class of learnable languages *unless*...
- Two senses...
 - (1) Distribution-free (modern sense) - pay attention to complexity
 - (2) Some assumed distribution (eg, exponentially declining, as for CFGs)

What is different?

For (2), not much:

What if we make the grammars probabilistic?

- Horning, 1969: Class of unambiguous probabilistic CFGs *is* learnable in the limit [why unambiguous?]
 - Intuition: since the probability of long sentences becomes vanishingly small, in effect, the language is finite
 - If Baby hasn't heard a sentence beyond a sentence length/complexity, they never will
- (This idea can be pursued in other ways.)

Punchline

- What about the class of probabilistic CFGs?
- Suppose Babysitter has to output sentences randomly **with the appropriate probabilities**, (what does that mean?)
 - Is s/he unable to be too evil?
 - Are there then perfect Babies that are guaranteed to converge to an appropriate probabilistic CFG?
- I.e., from hearing a finite number of sentences, Baby can correctly converge on a grammar that predicts an infinite number of sentences...
- But only if Baby knows the probability distribution function of the sentences *a priori* (Angluin)
- Even then, what is the *complexity* (# examples, time)?

Learning probabilistically

Definition 4 Let g be a target grammar and texts be presented to the learner in i.i.d. fashion according to a probability measure μ on L_g .

If there exists a learning algorithm \mathcal{A} such that

$$\mu_\infty(\{t \mid \lim_{k \rightarrow \infty} d(A(t_k), g) = 0\}) = 1$$

then the target is said to be *learnable with measure 1*.

The family \mathcal{G} is said to be *learnable with measure 1* if all grammars in \mathcal{G} are learnable with measure 1 by some algorithm \mathcal{A} .

And the envelope please...

Theorem 6 With strong prior knowledge about the nature of the μ_i 's, the family \mathcal{L} of r.e. languages is measure 1 learnable.

If the measure μ is *unknown*, the Superfinite languages (those having all the finite languages and at least one infinite language) are *not learnable*.

Beyond this...

Theorem 7 (Angluin 1988) If a family of grammars \mathcal{G} is learnable with measure 1 (on almost all texts) in a distribution-free sense, then it is learnable in the limit in the Gold sense (on all texts). As a matter of fact, one can prove the even stronger theorem:

Theorem 8 (Pitt 1989) If a family of grammars \mathcal{G} is learnable with measure $p > \frac{1}{2}$ then it is learnable in the limit in the Gold sense.

Pac-learning: ‘probably approximately correct’

$$\lim_{k \rightarrow \infty} \mathbb{P}[d(A(t_k), g) > \epsilon] = 0$$

Learning Probabilistic Grammars: Horning

- Need criterion to select among grammars
- Horning uses a Bayesian approach
- To develop this idea, we need the idea of a ‘grammar-grammar’, that is, a grammar that itself generates the family of possible grammars
- If the grammar-grammar is probabilistic, it defines a probability distribution over the the class of grammars it generates
- The *complexity* of a grammar G is then defined as,
$$-\log_2 p(G)$$

Horning's approach II

- In this metric, the more variables (nonterminals) in the grammar-grammar, the more alternatives for each, or the longer the alternatives in a generated grammar, the smaller its probability & the greater its complexity
- This provides a metric for selecting the 'simplest' grammar compatible with data seen so far
- Example

Example grammar-grammar

- Let G be the probabilistic grammar-grammar with the following productions, which generates regular grammars with 1 or 2 variables (S, A) and 1 or 2 terminal symbols

1. $S \rightarrow R$ [0.5]

2. $S \rightarrow RR$ [0.5]

3. $R \rightarrow N \rightarrow P$ [0.5]

4. $P \rightarrow A$ [0.5]

5. $P \rightarrow P, A$ [0.5]

6. $A \rightarrow T$ [0.5]

7. $A \rightarrow TN$ [0.5]

8. $T \rightarrow a$ [0.5]

9. $T \rightarrow b$ [0.5]

10. $N \rightarrow S$ [0.5]

11. $N \rightarrow A$ [0.5]

Example left-most derivation of a ‘sentence’ = a grammar

- Grammar (sentence) is:

$$S \rightarrow b, bS, aA$$

$$A \rightarrow a, bA, aS \quad \text{or as one ‘sentence’:}$$

$$S \rightarrow b, bS, aA \quad A \rightarrow a, bA, aS$$

This takes 27 (left-most) steps in the grammar-
grammar!

Derivation of the grammar from the grammar-grammar

$S \Rightarrow RR$ [0.5]	$\Rightarrow S \rightarrow A, A, AR$
$\Rightarrow N \rightarrow PR$ [1.0]	$\Rightarrow S \rightarrow T, A, AR$
$\Rightarrow S \rightarrow PR$	$\Rightarrow S \rightarrow b, A, AR$
$\Rightarrow S \rightarrow P, AR$	$\Rightarrow S \rightarrow b, TN, AR$
$\Rightarrow S \rightarrow P, A, AR$	$\Rightarrow S \rightarrow b, bN, AR$

Derivation of grammar

$\Rightarrow S \rightarrow b, bN, AA$
 $\Rightarrow S \rightarrow b, bS, AR$
 $\Rightarrow S \rightarrow b, bS, TNR$
 $\Rightarrow S \rightarrow b, bS, aNR$
 $\Rightarrow S \rightarrow b, bS, aAR$
 $\Rightarrow S \rightarrow b, bS, aAN \rightarrow P$ [1.0]
 $\Rightarrow S \rightarrow b, bS, aAN \rightarrow P$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow P$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow P, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow P, A, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow A, A, A$

$\Rightarrow S \rightarrow b, bS, aAA \rightarrow T, A, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, A, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, TN, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, bN, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, bA, A$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, bA, TN$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, bA, aN$
 $\Rightarrow S \rightarrow b, bS, aAA \rightarrow a, bA, aS$

Whew!! 27 steps. Done.

$$p(G) = -\log_2(0.5)^{25} = -25 \log_2(1/2) = 25$$

Note that if change the productions of the grammar-grammar we can ‘change’ what the output grammars look like

- For example, if we change Rule 7, $A \rightarrow TN$ [0.5] so that the pr is less, then we penalize the ‘length’ of a right-hand side
- Now we can play the Bayesian game, since we can compute the *prior* probability of each grammar, as above, by its generation probability
- We can also compute the probability of a sentence, if we assume probabilities to each production in the generated grammar, in the usual way (viz, CGW or lab 5/6); assume these to be uniform at first

Horning's Bayesian game

- *Prior* probability of a grammar G_i in the hypothesis space is denoted $p(G_i)$
- Probability of an input sequence of sentences S_j given a grammar G_i is denoted $p(S_j | G_i)$ and is just the product of the probabilities that G_i generated each sentence s_1, s_2, \dots, s_k , i.e., $p(s_1 | G_i) \times \dots \times p(s_k | G_i)$
- But we want to find the probability that G_i is really the correct grammar, given data sequence S_j i.e., we want to find: $p(G_i | S_j)$ [the *posterior* probability]
- Now, we can use Bayes' rule that determines this as:

$$p(G_i | S_j) = \frac{p(G_i) \times p(S_j | G_i)}{p(S_j)}$$

We want the best (highest probability) G_i
given the data sequence S_j

$$\begin{aligned} p(G_i | S_j) &= \frac{p(G_i) \times p(S_j | G_i)}{p(S_j)} \\ &= \frac{p(G_i) \times p(S_j | G_i)}{p(S_j)} \\ \arg \max & \quad p(S_j) \\ &= \\ \arg \max & \quad p(G_i) \times p(S_j | G_i) \quad (\text{since } S_j \text{ constant}) \end{aligned}$$

And we can compute this! We just need to ‘search’ through all the grammars and find the one that maximizes this... can this be done? Horning has a general result for unambiguous CFGs; for a more recent (2011) approach that works with 2 simple grammar types & child language – see Perfors *et al.* Note: again, the G ’s only ‘approach’ the best G with increasing likelihood 6.863J SP11

Another view of this ‘maximize posterior probability’
view

$$\underset{\text{arg max}}{=} p(G_i) \times p(S_j | G_i)$$

Now let’s assume:

- (1) that $p(G_i) \propto 2^{-|G_i|}$ so that smaller grammars are more probable;
- (2) by Shannon’s source coding theorem, optimal encodings of the data S_j wrt grammar G_i approaches $-\log_2 p(S_j | G_i)$

Then maximizing this ‘posterior probability’ becomes, after taking \log_2 , is equivalent to finding the minimum of:

$$|G_i| - \log_2 p(S_j | G_i)$$

This is usually called minimum description (MDL)

We want to find the shortest (smallest) grammar *plus* the encoding of the data using that grammar

- Most restrictive grammar just lists all possible utterances
 - Only the observed data is grammatical, so it has a high probability
- A simple grammar could be made that allowed any sentences
 - Grammar would have a high probability
 - But data a *very* low one

MDL finds a middle ground between always generalizing and never generalizing

Complexity and Probability

- More complex grammar
 - Longer coding length, so lower probability
- More restrictive grammar
 - Fewer choices for data, so each possibility has a higher probability

Minimum description length as a criterion has a long pedigree...

Given the fixed notation, the criteria of simplicity governing the ordering of statements are as follows: that the shorter grammar is the simpler, and that among equally short grammars, the simplest is that in which the average length of derivation of sentences is least.

Chomsky, 1949, *Morphophonemics of Modern Hebrew*

So, this MDL criterion was there from the start:
Minimize the grammar size and
Minimize the length of the exceptions that can't be compressed by the grammar + data that can be...

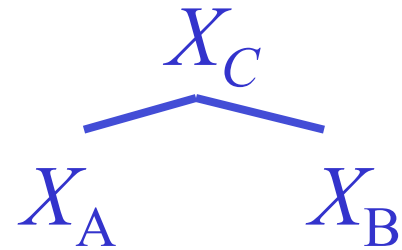
What about actually learning stochastic CFGs?

- Basic idea (from Suppes, 1970 onwards to Perfors *et al.*)
- Start with uniform probabilities on the rules
- Then adjust according to maximum likelihood counts to find best $p(G | D)$
- Use a search method, because exhaustive search using Horning's idea has too many possibilities
- Standard search method to find maximum likelihood is 'expectation maximization' (EM)
- Measure of merit is how well grammar *predicts* the sentences

Idea: Learn PCFGs with EM

- Classic experiments on learning PCFGs with Expectation-Maximization [Lari and Young, 1990]

$\{ X_1 , X_2 \dots X_n \}$



- Full binary grammar over n symbols
- Parse uniformly/randomly at first
- Re-estimate rule expectations off of parses
- Repeat

Re-estimation of PCFGs

- Basic quantity needed for re-estimation with EM:

$$P(X_c | i, j, S) = \frac{\sum_{T:(X_k, i, j) \wedge \text{yield}(T)=S} P(T)}{\sum_{T:\text{yield}(T)=S} P(T)}$$

- Can calculate in cubic time with the Inside-Outside algorithm.
- Consider an initial grammar where all productions have equal weight:

$$P(X_a X_b | X_c) = 1/n^2$$

- Then all trees have equal probability initially.
- Therefore, after one round of EM, the posterior over trees will (in the absence of random perturbation) be approximately uniform over all trees, and symmetric over symbols.

An Example of a run: learning “English” vs. “German”

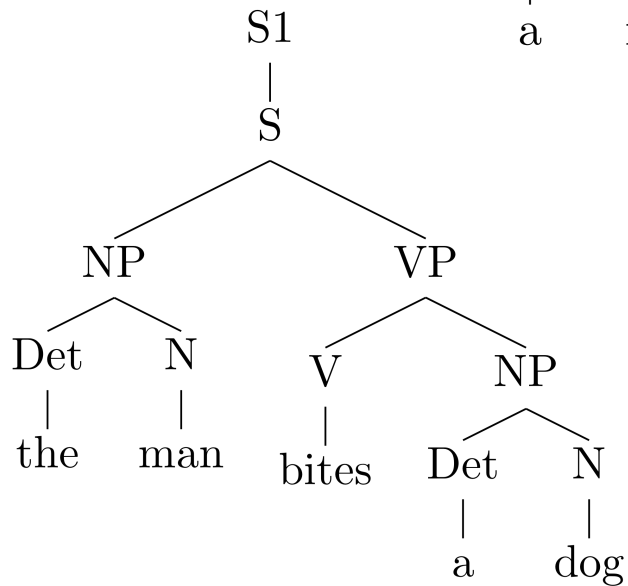
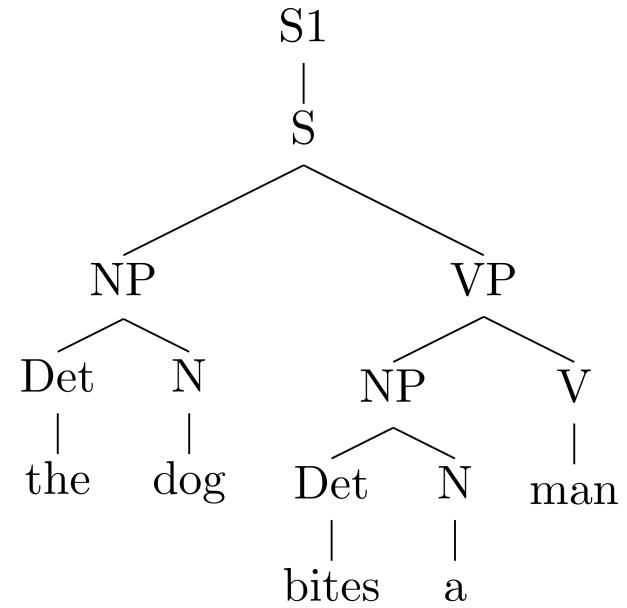
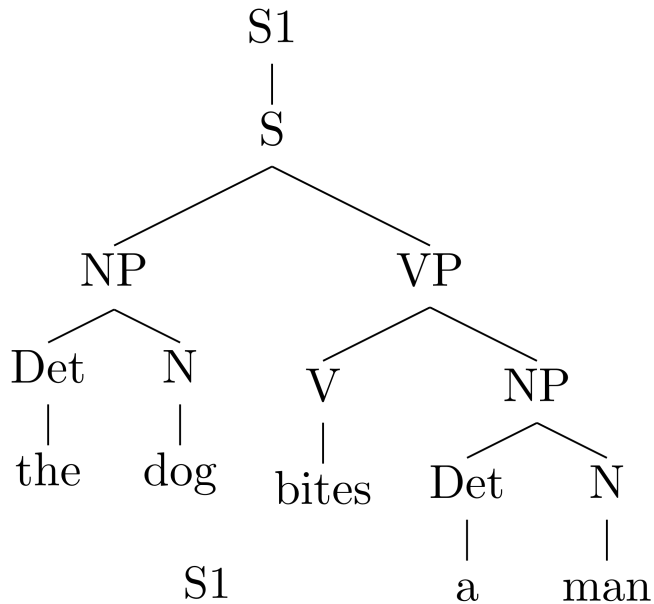
1.0	$S_1 \rightarrow S$	1.0	$V \rightarrow a$
1.0	$S \rightarrow NP VP$	1.0	$Det \rightarrow dog$
1.0	$NP \rightarrow Det N$	1.0	$N \rightarrow dog$
1.0	$VP \rightarrow V$	1.0	$V \rightarrow dog$
1.0	$VP \rightarrow V NP$	1.0	$Det \rightarrow man$
1.0	$VP \rightarrow NP V$	1.0	$N \rightarrow man$
1.0	$VP \rightarrow V NP NP$	1.0	$V \rightarrow man$
1.0	$VP \rightarrow NP NP V$	1.0	$Det \rightarrow bone$
1.0	$Det \rightarrow the$	1.0	$N \rightarrow bone$
1.0	$N \rightarrow the$	1.0	$V \rightarrow bone$
1.0	$V \rightarrow the$	1.0	$Det \rightarrow bites \mid gives$
1.0	$Det \rightarrow a$	1.0	$N \rightarrow bites \mid gives$
1.0	$N \rightarrow a$	1.0	$V \rightarrow bites \mid gives$

Example sentences fed in

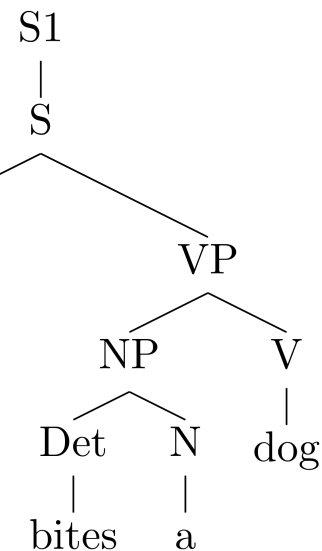
- the dog bites a man
- the man bites a dog
- a man gives the dog a bone
- the dog gives a man the bone
- a dog bites a bone

What is this doing?

Does it always work so well?



6.863J SP11

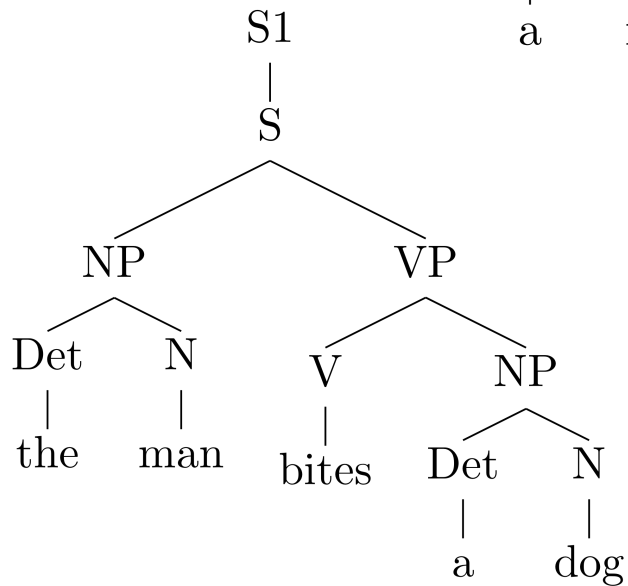
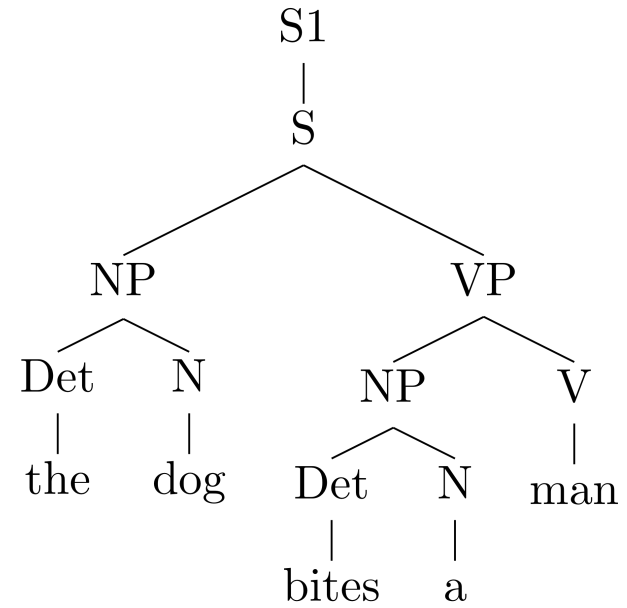
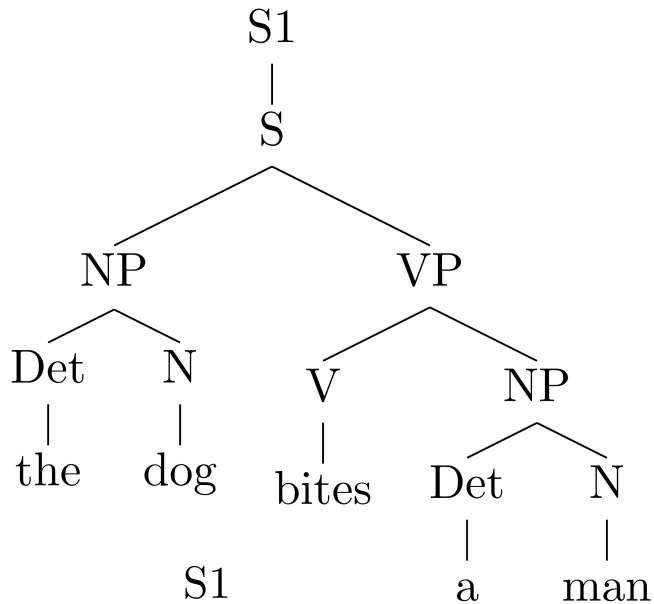


Resulting grammar

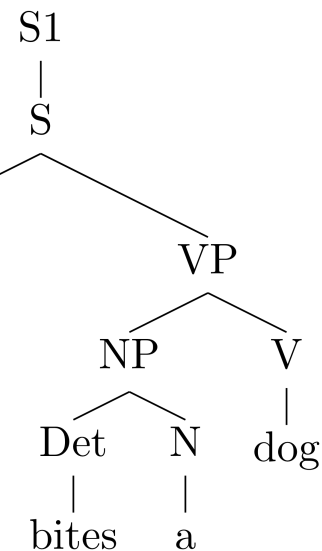
1	$S_1 \rightarrow S$	
1	$S \rightarrow NP VP$	
1	$NP \rightarrow Det N$	
0.6	$VP \rightarrow V NP$	
0.4	$VP \rightarrow V NP NP$	
0.416667	$Det \rightarrow the$	
0.583333	$Det \rightarrow a$	But this is not surprising!
0.416667	$N \rightarrow dog$	
0.333333	$N \rightarrow man$	
0.25	$N \rightarrow bone$	
0.6	$V \rightarrow bites$	
0.4	$V \rightarrow gives$	

What is this doing?

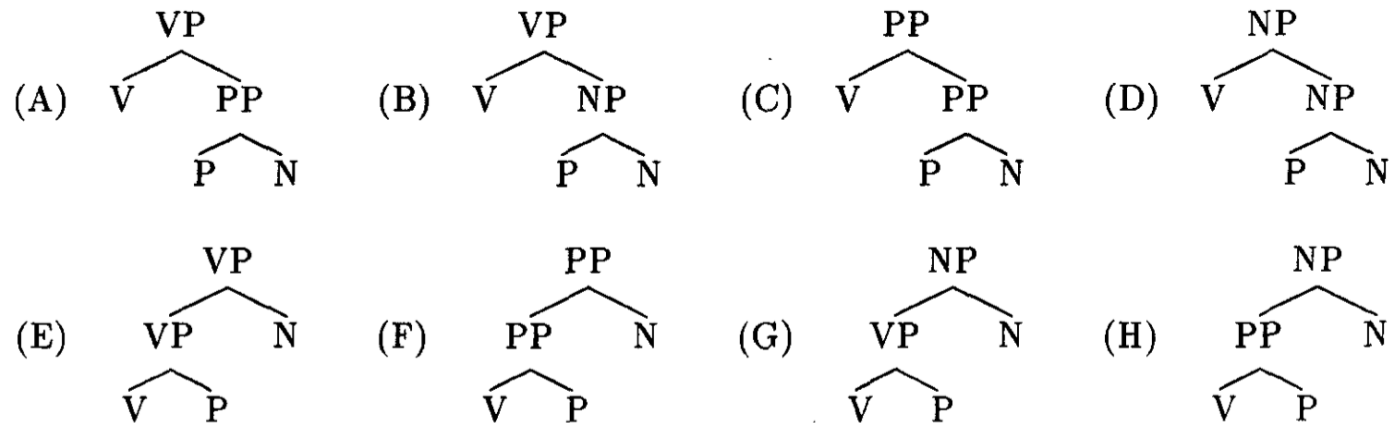
Does it always work so well?



6.863J SP11



“walking on ice”



(A) Is the right structure. Why? Can a stochastic CFG learning algorithm find (A), rather than the other structures?

In fact, this turns out to be hard. The SCFG picks (E)! Why? Entropy of (A) turns out to be higher (worse) than (E)-(H). Learner that uses this will go wrong.