Introduction

FSNLP, chapter 1

Christopher Manning and
Hinrich Schütze

NLP: An Age of Engineering

- 70s/80s: Science of the mind
  - Big questions of cognition
  - Small simulations (SHRDLU, LUNAR, . . .)
- 1990s: Real problems; rigorous evaluation
  - Big corpora on big hard disks
  - Applications: web, speech, (vertical)
  - Greatly favors statistical techniques
- 2000s: The future is meaning?

What is statistical NLP?

- $P(\textit{to}|\textit{Sarah drove})$
- $P(\textit{time}\text{ is a verb}|S = \textit{Time flies like an arrow})$
- $P(\text{NP} \rightarrow \text{Det Adj N}|\textit{Mother} = \text{VP}[\textit{drive}])$
- Statistical NLP methods:
  - Involve deriving numerical data from text
  - Are usually but not always probabilistic (broad church – we include e.g., vector spaces)

StatNLP: Relation to wider context

- Matches move from logic-based AI to probabilistic AI
  - Knowledge → probability distributions
  - Inference → conditional distributions
- Probabilities give opportunity to unify reasoning, planning, and learning, with communication
- There is now widespread use of machine learning (ML) methods in NLP (perhaps even overuse?)
- Use of approximation for hard problems

Questions that linguistics should answer

- What kinds of things do people say?
- What do these things say/ask/request about the world?

Example: *In addition to this, she insisted that women were regarded as a different existence from men unfairly.*

- Text corpora give us *data* with which to answer these questions
- They are an externalization of linguistic knowledge
- What words, rules, statistical facts do we find?
- Can we build programs that learn effectively from this data, and can then do NLP tasks?

The big questions for linguistics/NLP

- What kinds of things do people say?
- What do these things say/ask/request about the world?

These involve questions of frequency, probability, and likelihood

"Statistical considerations are essential to an understanding of the operation and development of languages" – Lyons (1968: 98)

Probabilistic grammars in linguistics

- The predictions about grammaticality and ambiguity of categorical grammars do not accord with human perceptions or engineering needs
- Categorical grammars aren’t predictive
  - They don’t tell us what “sounds natural”
  - Grammatical but unnatural e.g.: *In addition to this, she insisted that women were regarded as a different existence from men unfairly.*

Big picture claims

- Human cognition has a probabilistic nature
- We are continually faced with uncertain and incomplete information, and have to reason and interpret as best we can with the information available
- Language understanding is a case of this
- Language understanding involves mapping from ideas expressed in a symbol system to an uncertain and incomplete understanding

Symbol system $\leftrightarrow$ Probabilistic cognition

All parts of natural language text are ambiguous

- Real language is highly ambiguous at all levels
- It is thus hard to process
- Humans mostly do not notice the high level of ambiguity because they resolve ambiguities in real time, by incorporating diverse sources of evidence, including frequency information (cf. recent psycholinguistic literature)
- Goal of computational linguistics is to do as well
- Use of probabilities allows effective evidence combination within NLP models

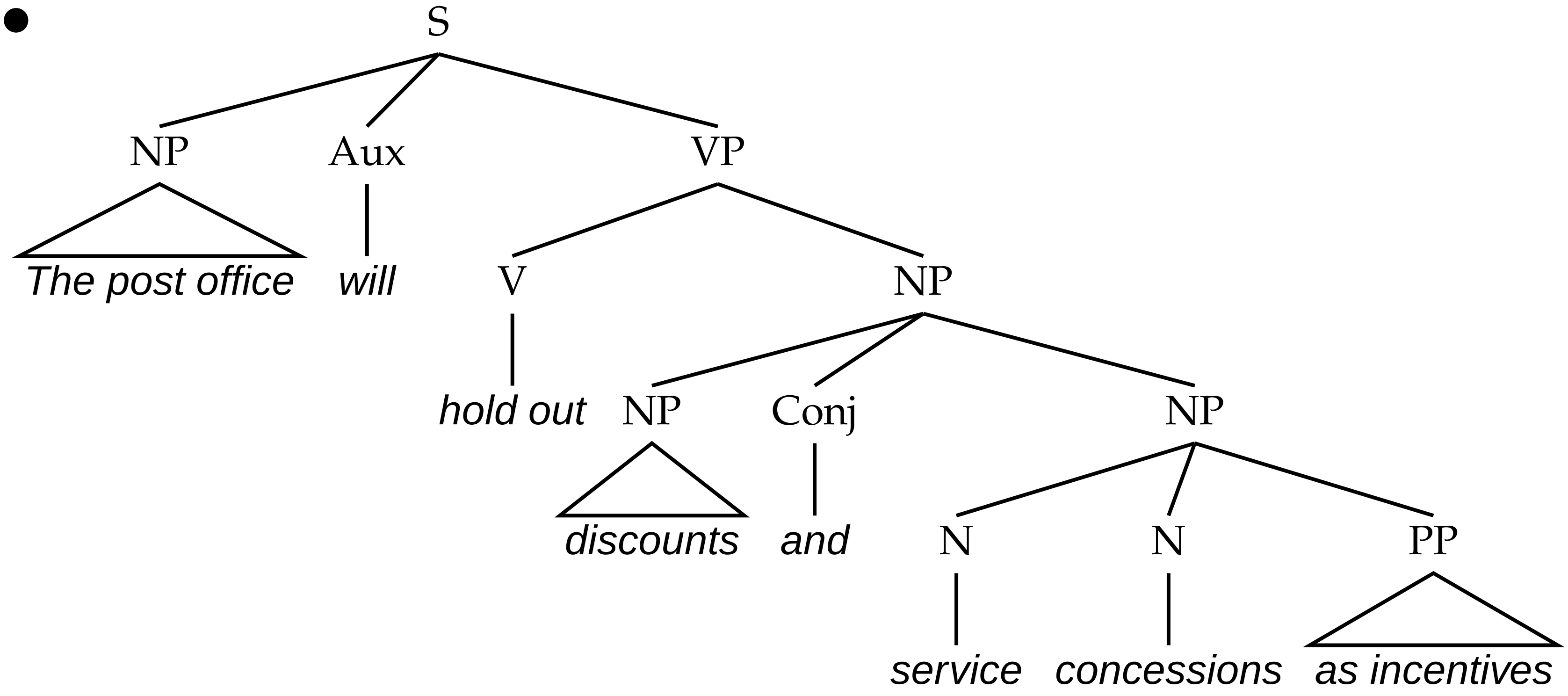
Contextuality of language

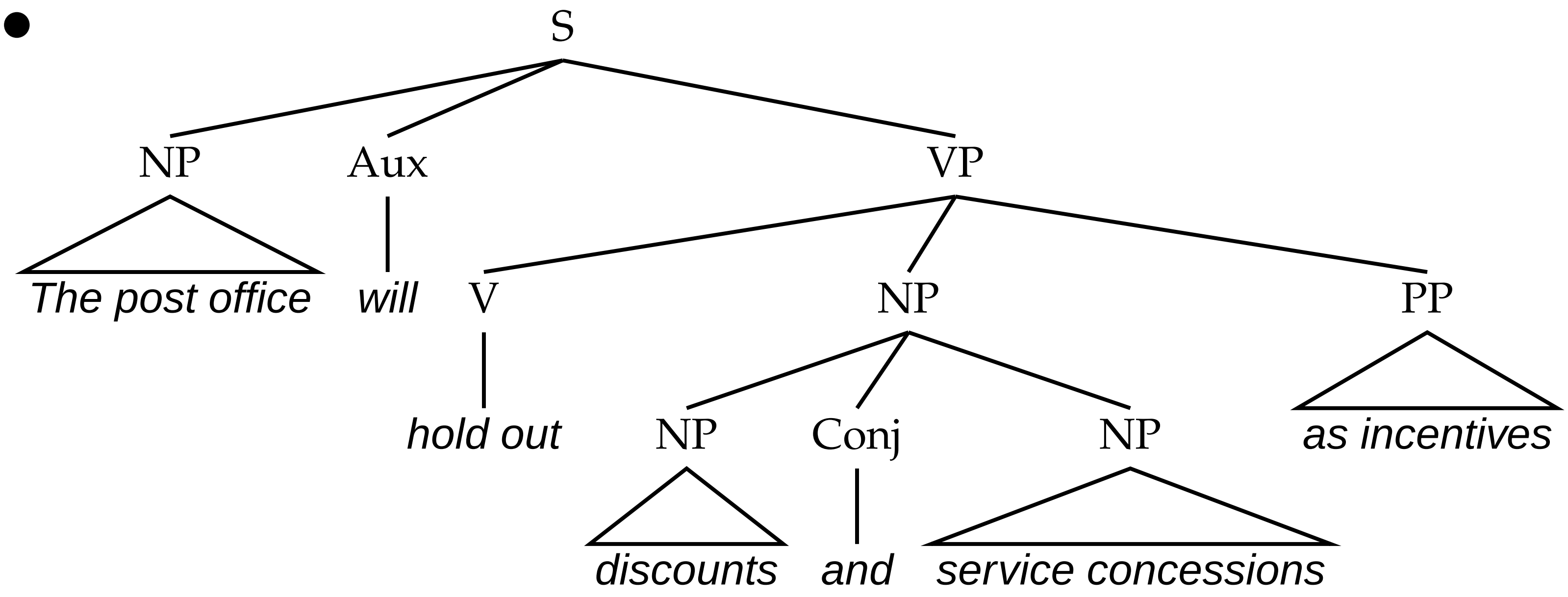
- Language use is situated
- People say the little that is needed to be understood in a certain situation
- Consequently
  - language is highly ambiguous
  - tasks like translation involve (probabilistically) reconstructing world knowledge not in the source text
- We also need to explore quantitative techniques to move away from the unrealistic categorical assumptions of much of formal linguistics

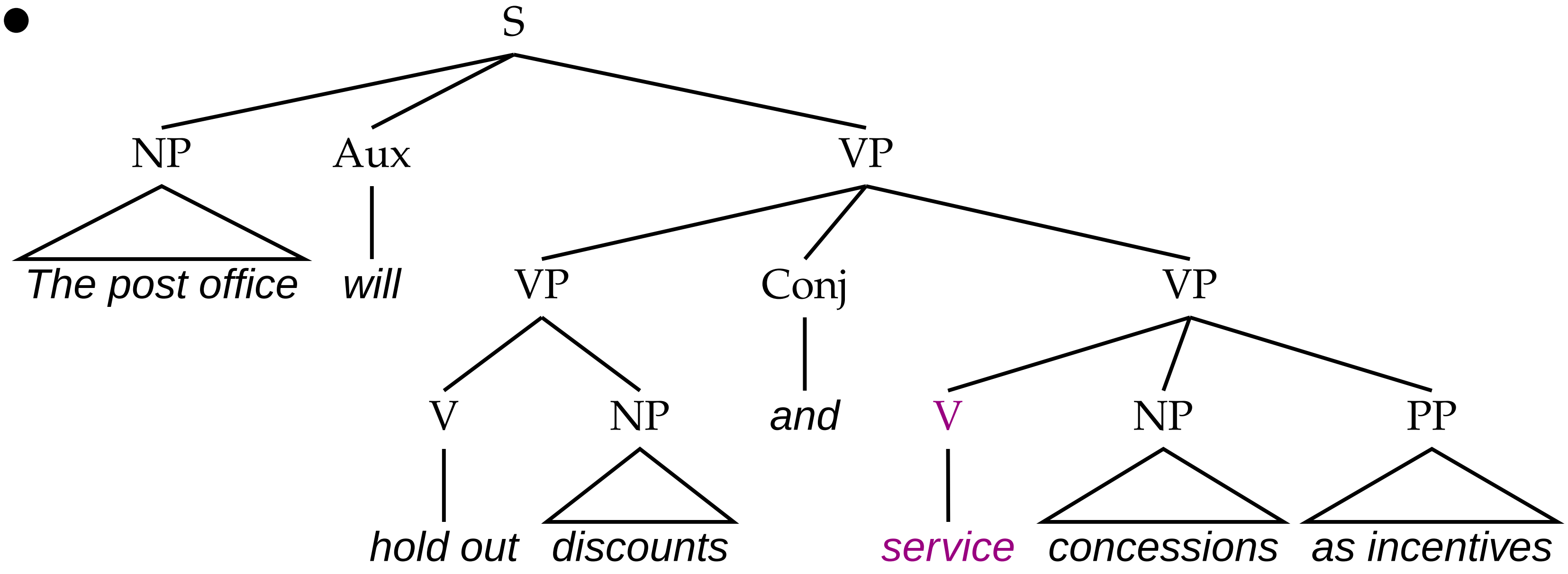
Computer NLP

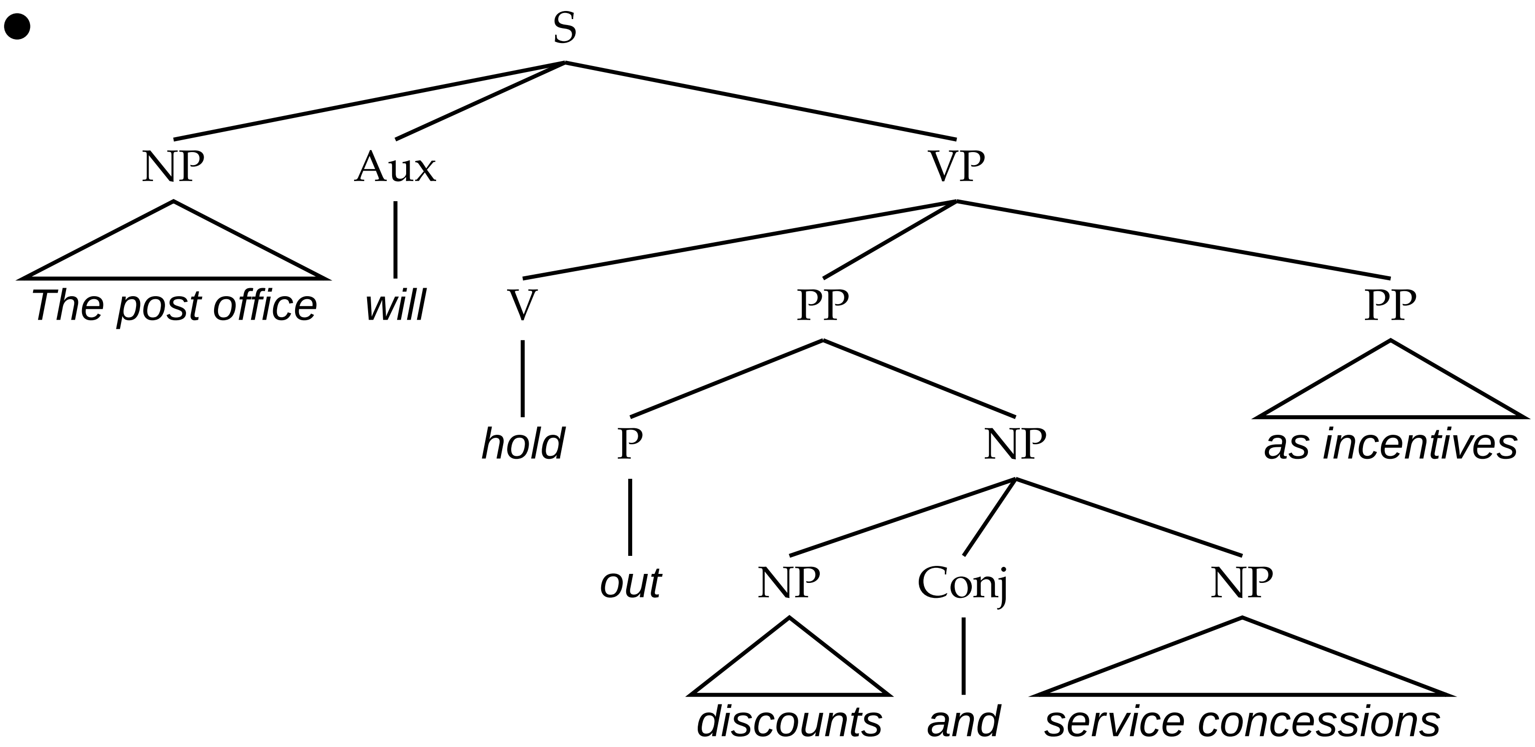
- Is often serial through a pipeline (not parallel)
- All components resolve ambiguities
- Something like an n-best list or word lattice is used to allow some decisions to be deferred until later
- Progressively richer probabilistic models can filter less likely word sequences, syntactic structures, meanings, etc.

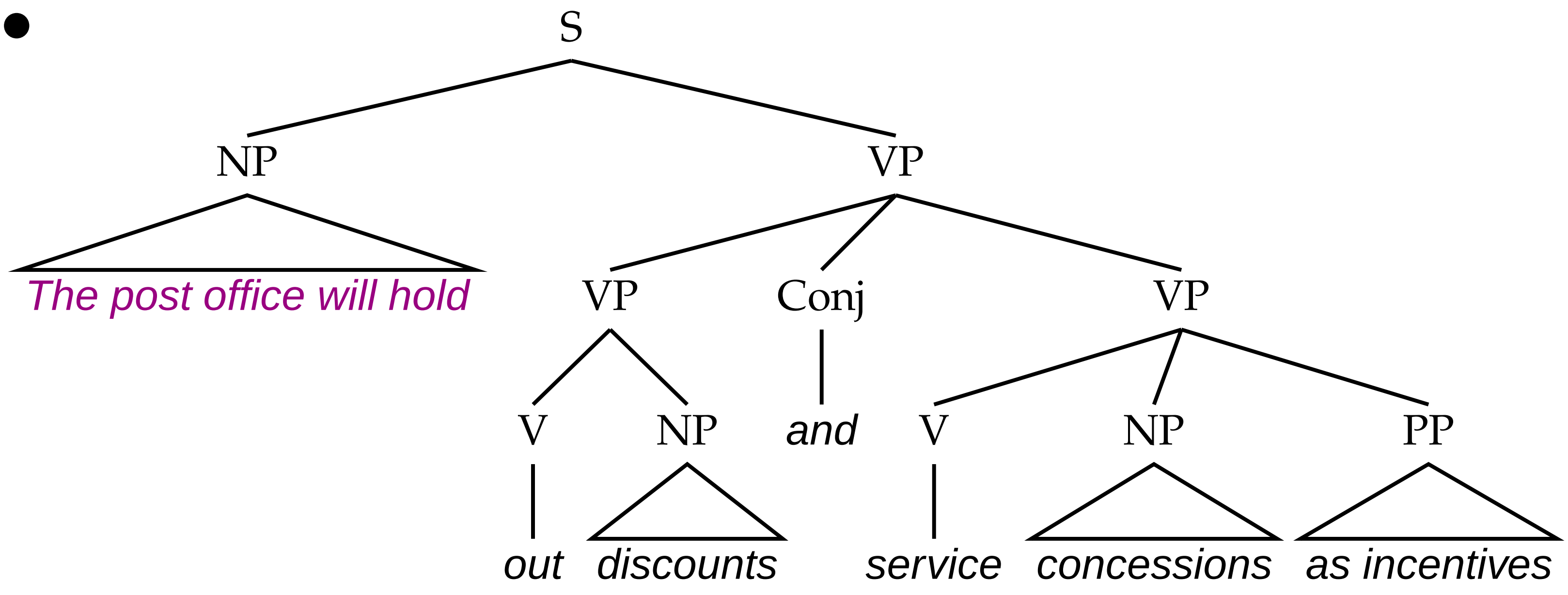
The huge ambiguity of language

- A few of 83+ syntactic parses for: *The post office will hold out discounts and service concessions as incentives.* [Shortened WSJ sentence.]
- 

- 
S
NP
Aux
VP
The post office
will
V
NP
PP
hold out
NP
Conj
NP
as incentives
discounts
and
service concessions


S
NP
Aux
VP
The post office
will
VP
Conj
VP
V
NP
and
V
NP
PP
hold out
discounts
service
concessions
as incentives

- 
S
NP
Aux
VP
The post office
will
V
PP
PP
hold
P
NP
as incentives
out
NP
Conj
NP
discounts
and
service concessions

- 
S
NP
VP
The post office will hold
VP
Conj
VP
V
NP
and
V
NP
PP
out
discounts
service
concessions
as incentives

Where do problems come in?

Syntax

- Part of speech ambiguities
- Attachment ambiguities

Semantics

- Word sense ambiguities ⟶ *we'll start here*
- (Semantic interpretation and scope ambiguities)

How do we solve them?

Hand-crafted NLP systems

- Easy to encode linguistic knowledge precisely
- Readily comprehensible rules
- Construction is costly
- Feature interactions are hard to manage
- Systems are usually nonprobabilistic

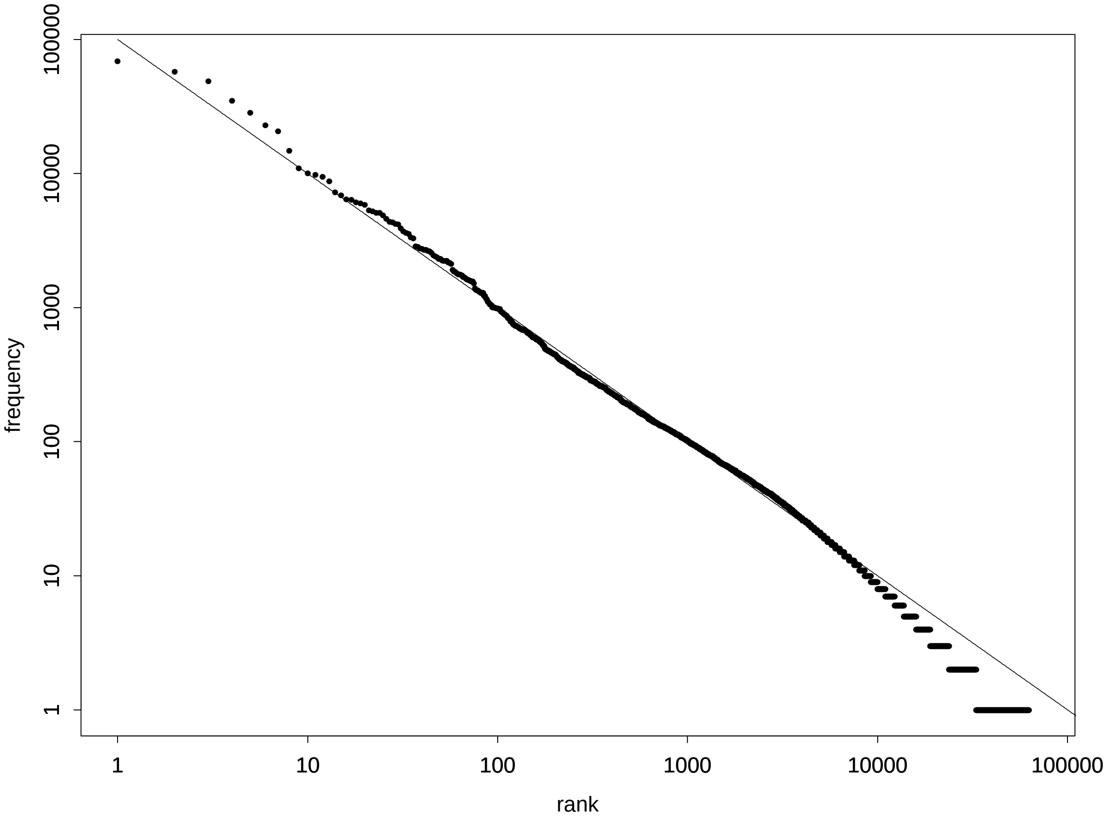
Statistical Computational Methods

- Many techniques are used:
  - n-grams, history-based models, decision trees / decision lists, memory-based learning, loglinear models, HMMs, neural networks, vector spaces, graphical models, PCFGs, . . .
- Robust
- Good for learning (well, supervised learning works well; unsupervised learning is still hard)
- More work needed on encoding subtle linguistic phenomena

Distinctiveness of NLP as an ML problem

- Language allows the complex compositional encoding of thoughts, ideas, feelings, . . . , intelligence.
- We are minimally dealing with hierarchical structures (branching processes), and often want to allow more complex forms of information sharing (dependencies).
- Enormous problems with data sparseness
- Both features and assigned classes regularly involve multinomial distributions over huge numbers of values (often in the tens of thousands)
- Generally dealing with discrete distributions though!
- The distributions are very uneven, and have fat tails

The obligatory Zipf’s law slide:
Zipf’s law for the Brown corpus



Zipf’s law

$f \propto \frac{1}{r}$ or, there is a k such that $f \cdot r = k$

(Now frequently invoked for the web too!

See http://linkage.rockefeller.edu/wli/zipf/)

Mandelbrot’s law

$$f = P(r + \rho)^{-B}$$

$$\log f = \log P - B \log(r + \rho)$$

Corpora

- A *corpus* is a *body* of naturally occurring text, normally one organized or selected in some way
  - Latin: one corpus, two corpora
- A *balanced corpus* tries to be representative across a language or other domain
- Balance is something of a chimaera: What is balanced? Who spends what percent of their time reading the sports pages?

The Brown corpus

- Famous early corpus. Made by W. Nelson Francis and Henry Kučera at Brown University in the 1960s. A balanced corpus of written American English in 1960 (except poetry!).
- 1 million words, which seemed huge at the time.

  Sorting the words to produce a word list took 17 hours of (dedicated) processing time, because the computer (an IBM 7070) had the equivalent of only about 40 kilobytes of memory, and so the sort algorithm had to store the data being sorted on tape drives.
- Its significance has increased over time, but also awareness of its limitations.
- Tagged for part of speech in the 1970s
  - The/AT General/JJ-TL Assembly/NN-TL ,/, which/WDT adjourns/VBZ today/NR ,/, has/HVZ performed/VBN

Recent corpora

- British National Corpus. 100 million words, tagged for part of speech. Balanced.
- Newswire (*NYT* or *WSJ* are most commonly used): Something like 600 million words is fairly easily available.
- Legal reports; UN or EU proceedings (parallel multilingual corpora – same text in multiple languages)
- The Web (in the billions of words, but need to filter for distinctness).
- Penn Treebank: 2 million words (1 million *WSJ*, 1 million speech) of parsed sentences (as phrase structure trees).

Common words in *Tom Sawyer* (71,370 words)

Word	Freq.	Use
the	3332	determiner (article)
and	2972	conjunction
a	1775	determiner
to	1725	preposition, verbal infinitive marker
of	1440	preposition
was	1161	auxiliary verb
it	1027	(personal/expletive) pronoun
in	906	preposition
that	877	complementizer, demonstrative
he	877	(personal) pronoun
I	783	(personal) pronoun
his	772	(possessive) pronoun
you	686	(personal) pronoun
Tom	679	proper noun
with	642	preposition

Frequencies of frequencies in *Tom Sawyer*

Word Frequency	Frequency of Frequency		
1	3993	71,730	word tokens
2	1292	8,018	word types
3	664		
4	410		
5	243		
6	199		
7	172		
8	131		
9	82		
10	91		
11–50	540		
51–100	99		
> 100	102		

Zipf’s law in *Tom Sawyer*

Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
Oh	116	90	10440
two	104	100	10400

Zipf’s law in *Tom Sawyer*

Word	Freq. (f)	Rank (r)	$f \cdot r$
turned	51	200	10200
you’ll	30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

Zipf's law

$$f \propto \frac{1}{r} \tag{1}$$

There is a constant k such that

$$f \cdot r = k \tag{2}$$

(Now frequently invoked for the web too!

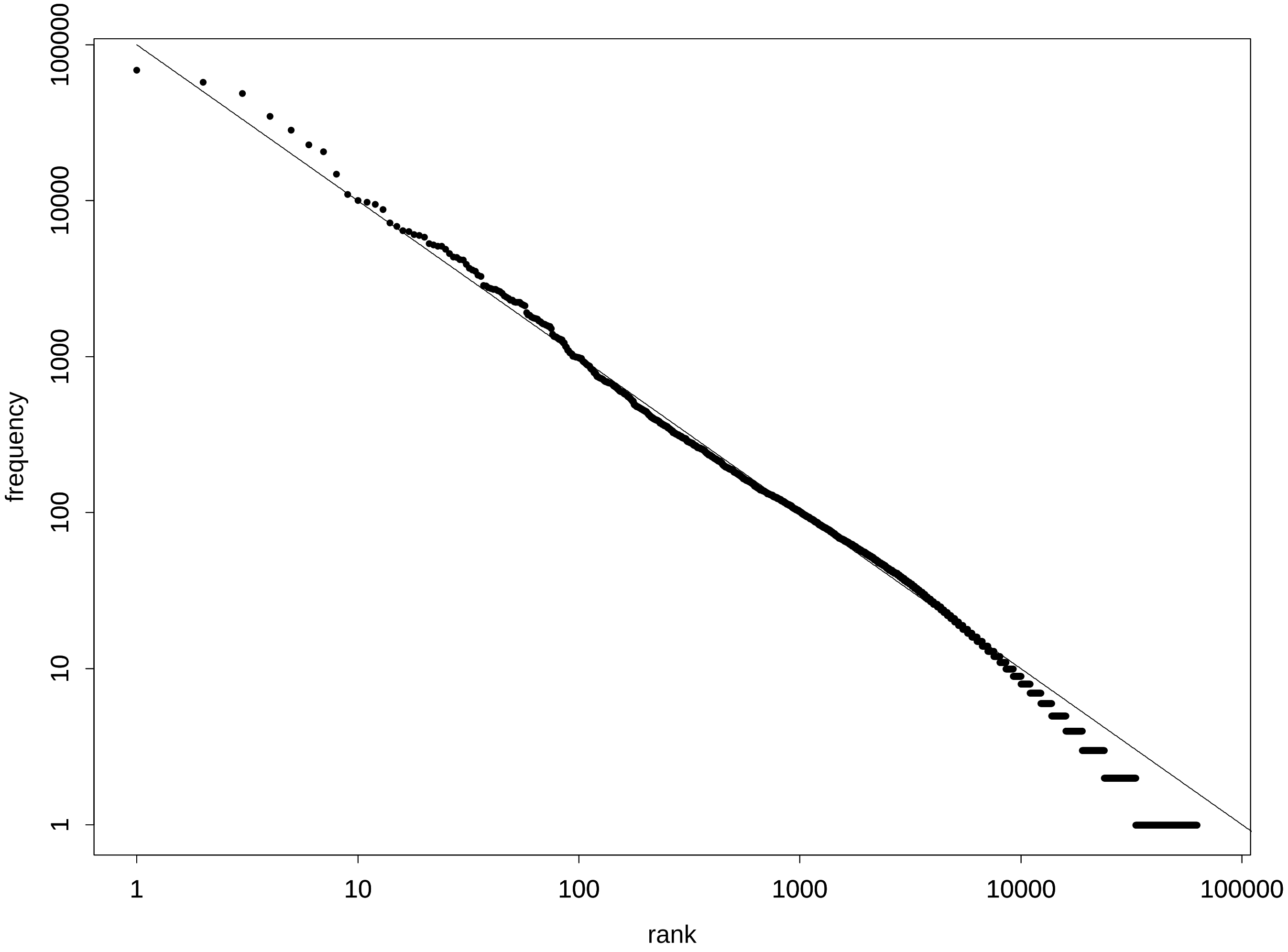
See http://linkage.rockefeller.edu/wli/zipf/)

Mandelbrot's law

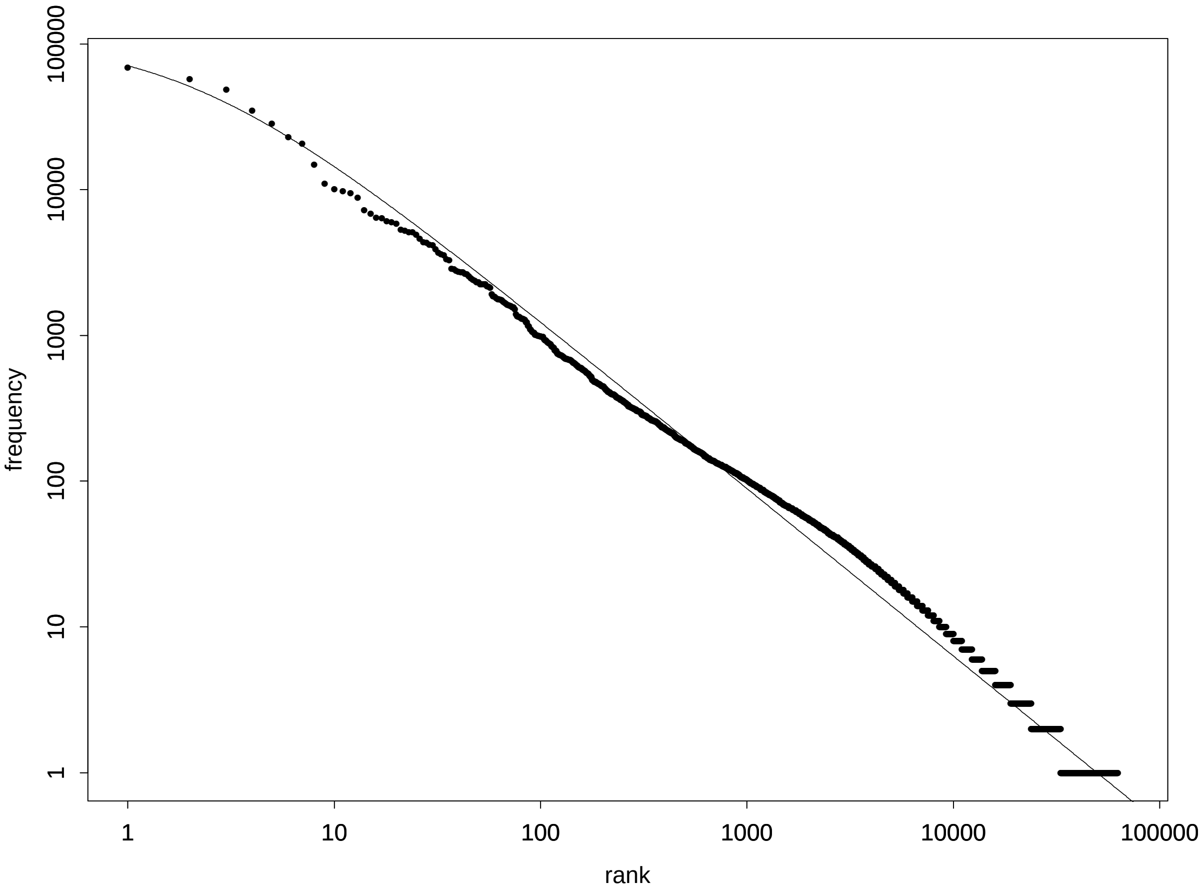
$$f = P(r + \rho)^{-B} \tag{3}$$

$$\log f = \log P - B \log(r + \rho) \tag{4}$$

Zipf’s law for the Brown corpus



Mandelbrot’s formula for the Brown corpus



$P = 10^{5.4}$, $B = 1.15$, $\rho = 100$

Commonest bigrams in the *NYT*

Frequency	Word 1	Word 2
80871	of	the
58841	in	the
26430	to	the
21842	on	the
21839	for	the
18568	and	the
16121	that	the
15630	at	the
15494	to	be
13899	in	a
13689	of	a
13361	by	the
13183	with	the
12622	from	the
11428	New	York
10007	he	said
9775	as	a
9231	is	a
8753	has	been
8573	for	a

Filtered common bigrams in the *NYT*

Frequency	Word 1	Word 2	POS pattern
11487	New	York	A N
7261	United	States	A N
5412	Los	Angeles	N N
3301	last	year	A N
3191	Saudi	Arabia	N N
2699	last	week	A N
2514	vice	president	A N
2378	Persian	Gulf	A N
2161	San	Francisco	N N
2106	President	Bush	N N
2001	Middle	East	A N
1942	Saddam	Hussein	N N
1867	Soviet	Union	A N
1850	White	House	A N
1633	United	Nations	A N
1337	York	City	N N
1328	oil	prices	N N
1210	next	year	A N
1074	chief	executive	A N
1073	real	estate	A N

KWIC display

```
 1   could find a target. The librarian  “showed  off” - running hither and thither w
 2  elights in. The young lady teachers  “showed  off” - bending sweetly over pupils
 3  ingly. The young gentlemen teachers  “showed  off” with small scoldings and other
 4  seeming vexation). The little girls  “showed  off” in various ways, and the littl
 5  n various ways, and the little boys  “showed  off” with such diligence that the a
 6  t genuwyne?” Tom lifted his lip and   showed  the vacancy. “Well, all right,” sai
 7  is little finger for a pen. Then he   showed  Huckleberry how to make an H and an
 8  ow’s face was haggard, and his eyes   showed  the fear that was upon him. When he
 9  not overlook the fact that Tom even   showed  a marked aversion to these inquests
10  own. Two or three glimmering lights   showed  where it lay, peacefully sleeping,
11  ird flash turned night into day and   showed  every little grass-blade, separate
12   that grew about their feet. And it   showed  three white, startled faces, too. A
13  he first thing his aunt said to him   showed  him that he had brought his sorrows
14  p from her lethargy of distress and   showed  good interest in the proceedings. S
15  ent a new burst of grief from Becky   showed  Tom that the thing in his mind had
16   shudder quiver all through him. He   showed  Huck the fragment of candle-wick pe
```

Syntactic frames for *showed* in *Tom Sawyer*

$\text{NP}_{\textit{agent}}$ showed off $(\text{PP}[\textit{with/in}]_{\textit{manner}})$

$$\text{NP}_{\textit{agent}} \text{ showed } (\text{NP}_{\textit{recipient}}) \left(\left\{\begin{array}{l} \text{NP}_{\textit{content}} \\ \text{CP}[\textit{that}]_{\textit{content}} \\ \text{VP[inf]}_{\textit{content}} \\ \textit{how}\ \text{VP[inf]}_{\textit{content}} \\ \text{CP}[\textit{where}]_{\textit{content}} \end{array}\right\}\right)$$

$\text{NP}_{\textit{agent}}$ showed $\text{NP}[\textit{interest}]$ $\text{PP}[\textit{in}]_{\textit{content}}$

$\text{NP}_{\textit{agent}}$ showed $\text{NP}[\textit{aversion}]$ $\text{PP}[\textit{to}]_{\textit{content}}$

Basic issues in text

- Upper and lower case
  - When is it useful to treat *black*, *Black* and *BLACK* the same or differently?
- Tokenization: what is a word?
  - Whitespace separated? *“I’m bored!”*
  - Oddities like *Micro$oft* or *C|net*, or even : -)
  - May want to have whitespace-separated single words: *New York* or *make up*. Or *by and large*?
  - Word segmentation is much harder in other languages (Chinese, Thai, etc. – no word breaks marked)

Basic issues in text

- Periods/Sentence boundaries
  - Normally ends of sentences. But also used for abbreviations: *Calif.*
  - Can have both functions at once (haplology): *I like you, etc.*
  - People have researched sentence segmentation.
- Single apostrophes
  - How does one treat *I'll* or *isn't* and/or *dog's*
  - With final quotes like *the kids' toys* you need longer distance context to tokenize well

Basic issues in text

- Hyphenation:
  - Traditionally, for text line breaks
  - *e-mail* or *co-operate* or *A-1-plus* [vs. *cooperate*]
  - *the aluminum-export ban* or *a text-based medium*
  - *the idea of a child-as-required-yuppie-possession must be motivating them*
  - In Dow Jones newswire get all of *data base*, *data-base* and *database* (with choice partly depending on section of paper).
- Homographs: *saw*

Much of the structure is implicit, traditionally

- Two carriage returns indicate a paragrah break
- Now, often SGML or XML gives at least some of the macro structure (sentences, paragraphs). Commonly not micro-structure
- `<p><s>And then he left.</s>`
  `<s>He did not say another word.</s></p>`
- `<utt speak="Fred" date="10-Feb-1998">That is an ugly couch.</utt>`
- May not be semantic markup:
  - `<B><font="+3">Relevant prior approaches</font></`

Distinctiveness of NLP as an ML problem

- Language allows the complex, compositional encoding of thoughts, ideas, feelings, . . . , intelligence.
- Most structure is hidden
- Relational, constraint satisfaction nature
- Long pipelines
- Large and strange, sparse, discrete distributions
- Large scale
- Feature-driven; performance-driven

Distinctiveness of NLP as an ML problem

- Much hidden structure; long processing pipelines
  - Long pipelines of probabilistic decompositions, through which errors can – and do – propagate
  - The problem has a relational/CSP nature. It's not just doing a series of (assumed iid) simple classification tasks. There are a lot of decisions to coordinate.
  - We are often dealing with hierarchical structures (branching processes), and often want to allow more complex forms of information sharing (dependencies/relational structure).

NLP: Large, sparse, discrete distributions

- Both features and assigned classes regularly involve multinomial distributions over huge numbers of values (often in the tens of thousands).
- The distributions are very uneven, and have fat tails
- Enormous problems with data sparseness: much work on smoothing distributions/backoff (shrinkage), etc.
- We normally have inadequate (labeled) data to estimate probabilities
- Unknown/unseen things are usually a central problem
- Generally dealing with discrete distributions though

Sparsity

- How often does an every day word like *kick* occur in a million words of text?
  - *kick*: about 10 [depends vastly on genre, of course]
  - *wrist*: about 5
- Normally we want to know about something bigger than a single word, like how often you *kick a ball*, or how often the conative alternation *he kicked* ***at*** *the balloon* occurs.
- How often can we expect that to occur in 1 million words?
- Almost never.
- “There’s no data like more data” [if of the right domain]

Distinctiveness of NLP as an ML problem

- Large scale
  - Scale is extremely large: tens or hundreds of thousands of features/parameters.
  - Need to balance beauty and practicality. Dynamic programming techniques and speed issues are key. Need to exploit sparseness.
- Feature-driven; performance-driven
  - The emphasis is on developing and finding ways to effectively exploit useful features from a rich but sparse representation, rather than on trade-offs between machine learning methods (there's still linguistics there!)

The Approach of Probabilistic Natural Language Understanding

- Language use is situated in a world context
- People say the little that is needed to be understood by their interlocutors in a certain situation
- Consequently
  - Language is highly ambiguous
  - Tasks like interpretation and translation involve (probabilistically) reasoning about meaning, using world knowledge not in the source text
- Suggests exploring probabilistic and quantitative techniques, and moving away from the unrealistic categorical assumptions of much of formal linguistics

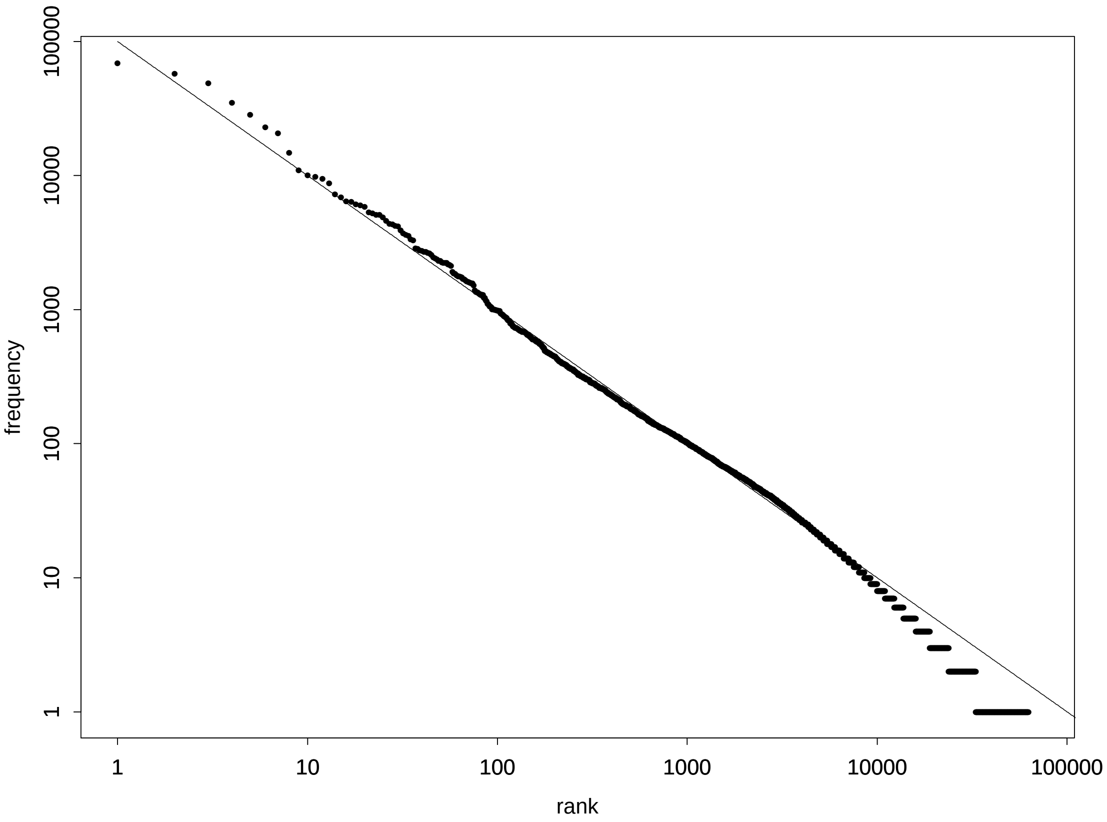
Distinctiveness of NLP as an ML problem

- Language allows the complex compositional encoding of thoughts, ideas, feelings, . . . , intelligence.
- We are minimally dealing with hierarchical structures (branching processes), and often want to allow more complex forms of information sharing (dependencies).
- Enormous problems with data sparseness
- Both features and assigned classes regularly involve multinomial distributions over huge numbers of values (often in the tens of thousands)
- Generally dealing with discrete distributions though!
- The distributions are very uneven, and have fat tails

Frequencies of frequencies in *Tom Sawyer*

Word Frequency	Frequency of Frequency		
1	3993	71,730	word tokens
2	1292	8,018	word types
3	664		
4	410		
5	243		
6	199		
7	172		
8	131		
9	82		
10	91		
11–50	540		
51–100	99		
> 100	102		

The obligatory Zipf’s law slide: Zipf’s law for the Brown corpus



Zipf's law

$f \propto \frac{1}{r}$ or, there is a k such that $f \cdot r = k$

(Now frequently invoked for the web too!

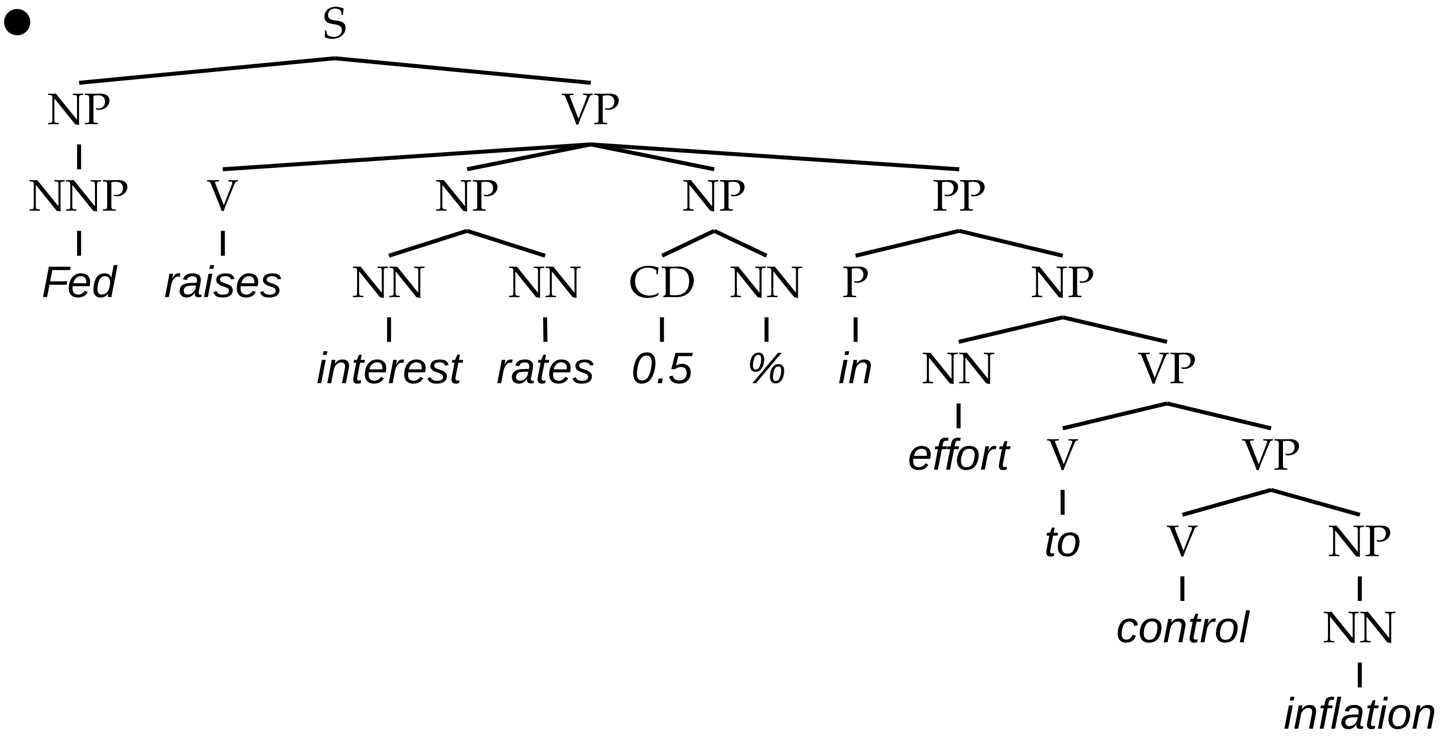
See http://linkage.rockefeller.edu/wli/zipf/)

Mandelbrot's law

$$f = P(r + \rho)^{-B}$$

$$\log f = \log P - B \log(r + \rho)$$

Why is NLU difficult? The hidden structure of language is hugely ambiguous

- Structures for: *Fed raises interest rates 0.5% in effort to control inflation* (*NYT* headline 17 May 2000)
- 

Where are the ambiguities?

Part of speech ambiguities

						Syntactic	
		VB					*attachment*
	VBZ	VBP	VBZ				*ambiguities*
NNP	NNS	NN	NNS	CD	NN		
Fed	*raises*	*interest*	*rates*	*0.5*	*%*	*in*	*effort*
						to	*control*
							inflation

Word sense ambiguities: Fed → *"federal agent"*
interest → *a feeling of wanting to know or learn more*

Semantic interpretation ambiguities above the word level

Mathematical Foundations

FSNLP, chapter 2

Christopher Manning and
Hinrich Schütze

Entropy

- Entropy is the average uncertainty of a random variable:

  Entropy $H(\mathrm{p}) = H(X) = -\sum_{x \in \mathcal{X}} \mathrm{p}(x) \log_2 \mathrm{p}(x)$

- Suppose you are reporting the result of rolling an 8-sided die. Then the entropy is:

$$H(X) = -\sum_{i=1}^{8} \mathrm{p}(i) \log \mathrm{p}(i) = -\sum_{i=1}^{8} \frac{1}{8} \log \frac{1}{8} = -\log \frac{1}{8} = 3 \text{ bits}$$

- Entropy measures are expectations:

$$H(X) = E\left(\log \frac{1}{\mathrm{p}(X)}\right)$$

Simplified Polynesian

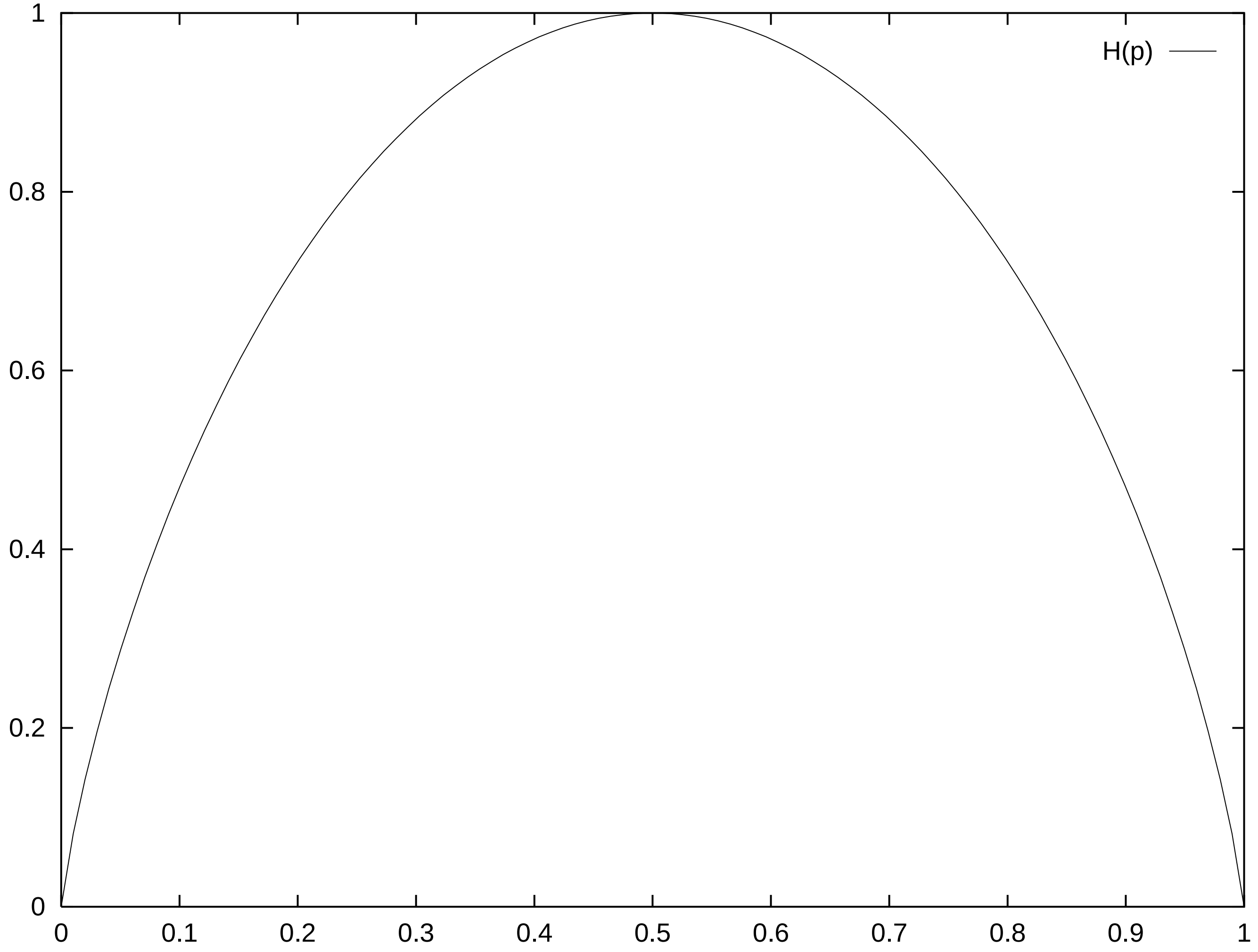
- Simplified Polynesian appears to be just a random sequence of letters, with the letter frequencies as shown:
p	t	k	a	i	u
1/8	1/4	1/8	1/4	1/8	1/8
- Then the per-letter entropy is:

$$\begin{aligned} H(P) &= -\sum_{i \in \{p,t,k,a,i,u\}} P(i) \log P(i) \\ &= -[4 \times \frac{1}{8} \log \frac{1}{8} + 2 \times \frac{1}{4} \log \frac{1}{4}] = 2\frac{1}{2} \text{ bits} \end{aligned}$$

We can design a code that on average takes $2\frac{1}{2}$ bits a letter:

p	t	k	a	i	u
100	00	101	01	110	111

The entropy of a weighted coin



Joint/conditional entropy

- Joint entropy:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \mathsf{p}(x,y)\log \mathsf{p}(x,y)$$

- Conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} \mathsf{p}(x) H(Y|X=x) \\ &= \sum_{x\in\mathcal{X}} \mathsf{p}(x)\left[-\sum_{y\in\mathcal{Y}} \mathsf{p}(y|x)\log \mathsf{p}(y|x)\right] \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \mathsf{p}(x,y)\log \mathsf{p}(y|x) \end{aligned}$$

The relation of mutual information I to entropy H

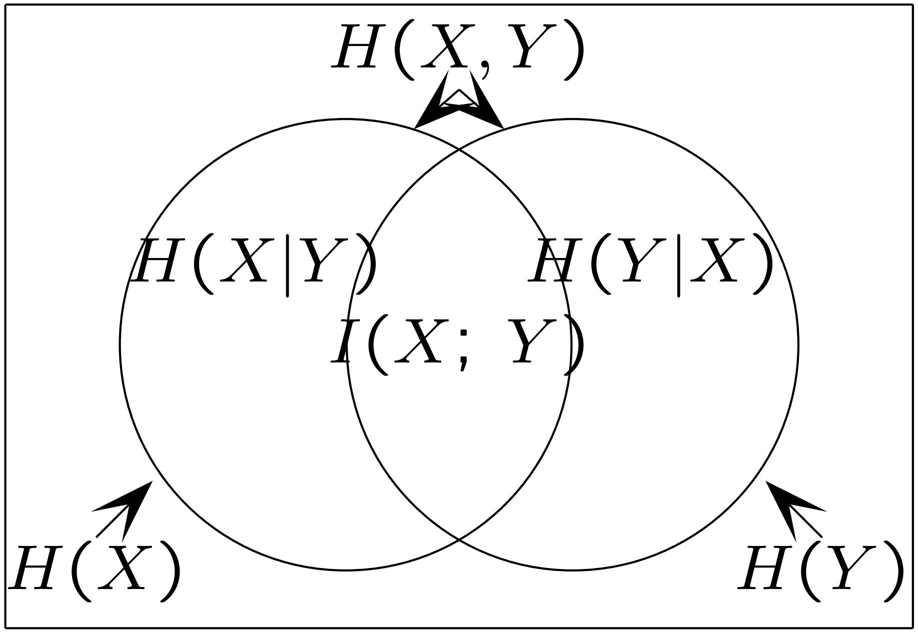
- By the chain rule for entropy,

  $$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

  Therefore,

  $$H(X) - H(X|Y) = H(Y) - H(Y|X) = I(X;\ Y)$$

- This is called the *mutual information* between X and Y


$H(X,Y)$
$H(X|Y)$
$H(Y|X)$
$I(X;Y)$
$H(X)$
$H(Y)$

Mutual information

$$\begin{aligned} I(X;\ Y) &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X,Y) \\ &= \sum_x \mathrm{p}(x) \log \frac{1}{\mathrm{p}(x)} + \sum_y \mathrm{p}(y) \log \frac{1}{\mathrm{p}(y)} + \sum_{x,y} \mathrm{p}(x,y) \log \mathrm{p} \\ &= \sum_{x,y} \mathrm{p}(x,y) \log \frac{\mathrm{p}(x,y)}{\mathrm{p}(x)\,\mathrm{p}(y)} \end{aligned}$$

- Since $H(X|X) = 0$, note that:

$$H(X) = H(X) - H(X|X) = I(X;\ X)$$

- Hence, entropy is also called self-information
- MI of dependent variables depends on their entropy

Mutual information in classification problems

- A large mutual information means that knowing a lot about one variable tells you a lot about the other
- It is thus a useful metric for various problems in building supervised classification systems:
  - The standard decision tree induction algorithm uses MI to decide the attribute on which to split a node:
    - information gain$= I(C;\ A)$
  - Brown et al. (1991a) used MI to find the most informative single feature for WSD, and to partition its values to correspond to senses
  - MI is often used for feature selection in text categorization (e.g., with Bernoulli Naive Bayes models)

KL divergence or relative entropy

Two pmfs $\mathsf{p}(x)$ and $\mathsf{q}(x)$:

$$D(\mathsf{p} \,\|\, \mathsf{q}) = \sum_{x \in \mathcal{X}} \mathsf{p}(x) \log \frac{\mathsf{p}(x)}{\mathsf{q}(x)}$$

Say $0 \log \frac{0}{\mathsf{q}} = 0$, otherwise $\mathsf{p} \log \frac{\mathsf{p}}{0} = \infty$.

$$D(\mathsf{p} \,\|\, \mathsf{q}) = E_{\mathsf{p}} \left(\log \frac{\mathsf{p}(X)}{\mathsf{q}(X)} \right)$$

$$I(X;\ Y) = D(\mathsf{p}(x,y) \| \, \mathsf{p}(x)\, \mathsf{p}(y))$$

KL divergence

$$D(\mathsf{p} \parallel \mathsf{q}) = \sum_{x \in \mathcal{X}} \mathsf{p}(x) \log \frac{\mathsf{p}(x)}{\mathsf{q}(x)}$$

- Measure of how different two probability distributions are
- The average number of bits that are wasted by encoding events from a distribution p with a code based on a not-quite-right distribution q.
- $D(\mathsf{p} \parallel \mathsf{q}) \geq 0$; $D(\mathsf{p} \parallel \mathsf{q}) = 0$ iff $\mathsf{p} = \mathsf{q}$
- Not a metric: not commutative, doesn't satisfy triangle equality

[Slide on $D(p\|q)$ vs. $D(q\|p)$]

Cross entropy

- Entropy = uncertainty
- Lower entropy = efficient codes = knowing the structure of the language = measure of model quality
- Entropy = measure of surprise
- How surprised we are when w follows h is pointwise entropy:

$$H(w|h) = -\log_2 \mathrm{p}(w|h)$$

$\mathrm{p}(w|h) = 1$? $\mathrm{p}(w|h) = 0$?

- Total surprise:

$$\begin{aligned} H_{\text{total}} &= -\sum_{j=1}^{n} \log_2 \mathrm{m}(w_j|w_1, w_2, \ldots, w_{j-1}) \\ &= -\log_2 \mathrm{m}(w_1, w_2, \ldots, w_n) \end{aligned}$$

Formalizing through cross-entropy

- Our model of language is $\mathsf{q}(x)$. How good a model is it?
- Idea: use $D(\mathsf{p} \,\|\, \mathsf{q})$, where p is the correct model.
- Problem: we don’t know p.
- But we know roughly what it is like from a corpus
- Cross entropy:

$$\begin{aligned}
H(X, \mathsf{q}) &= H(X) + D(\mathsf{p} \,\|\, \mathsf{q}) \\
&= -\sum_x \mathsf{p}(x) \log \mathsf{q}(x) \\
&= E_{\mathsf{p}}(\log \frac{1}{\mathsf{q}(x)})
\end{aligned}$$

Cross entropy

- Cross entropy of a language $L = (X_i) \sim \mathsf{p}(x)$ according to a model m:

$$H(L, \mathsf{m}) = -\lim_{n\to\infty} \frac{1}{n} \sum_{x_{1n}} \mathsf{p}(x_{1n}) \log \mathsf{m}(x_{1n})$$

- If the language is 'nice':

$$H(L, \mathsf{m}) = -\lim_{n\to\infty} \frac{1}{n} \log \mathsf{m}(x_{1n})$$

  I.e., it's just our average surprise for large n:

$$H(L, \mathsf{m}) \approx -\frac{1}{n} \log \mathsf{m}(x_{1n})$$

- Since $H(L)$ is fixed if unknown, minimizing cross-entropy is equivalent to minimizing $D(\mathsf{p} \,\|\, \mathsf{m})$

- Assuming: independent test data, $L = (X_i)$ is stationary [does’t change over time], ergodic [doesn’t get stuck]

Entropy of English text

Character n-grams over a 27 letter alphabet

Model		Cross entropy (bits)	
Uniform	uniform	4.75	$(\log 27)$
Zeroth Order	unigram	4.03	
First Order	bigram	3.32	
Second Order	trigram	3.1	
Third Order	fourgram	2.8	
Shannon's experiment		1.3 (1.34)	

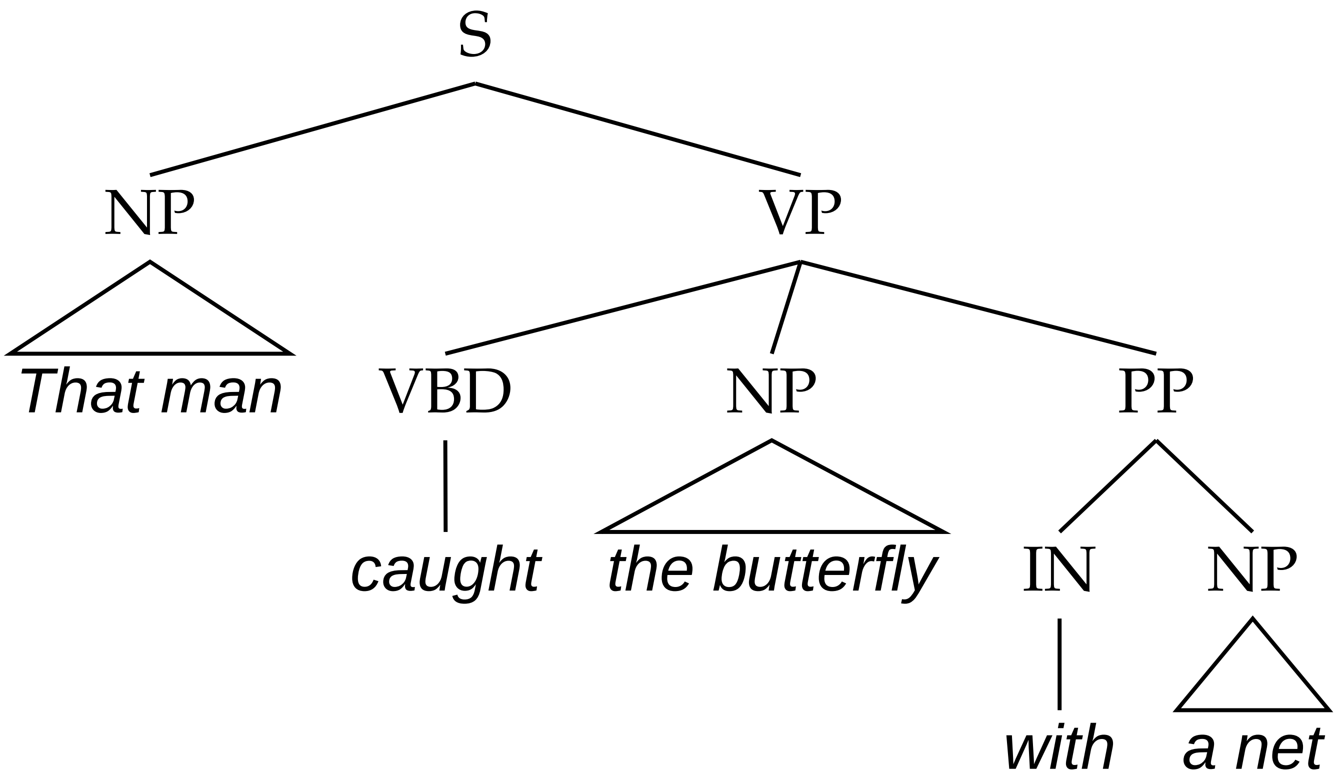
(Cover and Thomas 1991: 140) (though order is named nonstandardly there)

Perplexity

$$\begin{aligned} \mathsf{perplexity}(x_{1n}, \mathsf{m}) &= 2^{H(x_{1n}, \mathsf{m})} \\ &= \mathsf{m}(x_{1n})^{-\frac{1}{n}} \end{aligned}$$

(Note that $H(\cdot)$ has been overloaded between total entropy and entropy rate; perplexity is always per decision.)

Constituent phrases in language are represented by phrase structure trees



General Context-Free Grammar Parsing: A phrase structure grammar

- Also known as a context-free grammar (CFG)
- $$\begin{array}{lll@{\qquad}lll}
\text{S} & \longrightarrow & \text{NP VP} & \text{DT} & \longrightarrow & \textit{the} \\
\text{NP} & \longrightarrow & \left\{\begin{array}{l}\text{DT NNS}\\ \text{DT NN}\\ \text{NP PP}\end{array}\right\} & \text{NNS} & \longrightarrow & \left\{\begin{array}{l}\textit{children}\\ \textit{students}\\ \textit{mountains}\end{array}\right\} \\
\text{VP} & \longrightarrow & \left\{\begin{array}{l}\text{VP PP}\\ \text{VBD}\\ \text{VBD NP}\end{array}\right\} & \text{VBD} & \longrightarrow & \left\{\begin{array}{l}\textit{slept}\\ \textit{ate}\\ \textit{saw}\end{array}\right\} \\
\text{PP} & \longrightarrow & \text{IN NP} & \text{IN} & \longrightarrow & \left\{\begin{array}{l}\textit{in}\\ \textit{of}\end{array}\right\} \\
 & & & \text{NN} & \longrightarrow & \textit{cake}
\end{array}$$

Application of grammar rewrite rules

- S

  → NP VP

  → DT NNS VBD

  → *The children slept*

- S

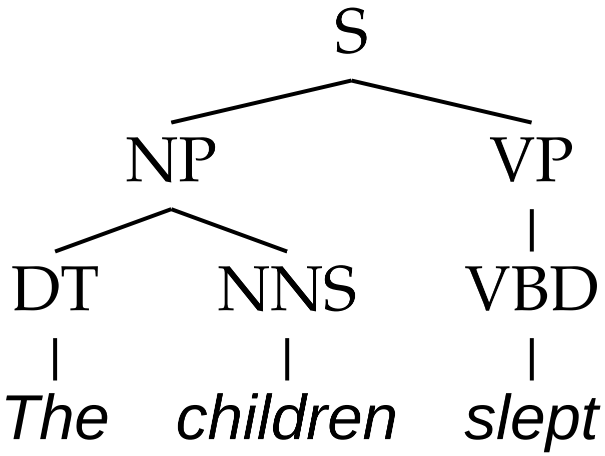
  → NP VP

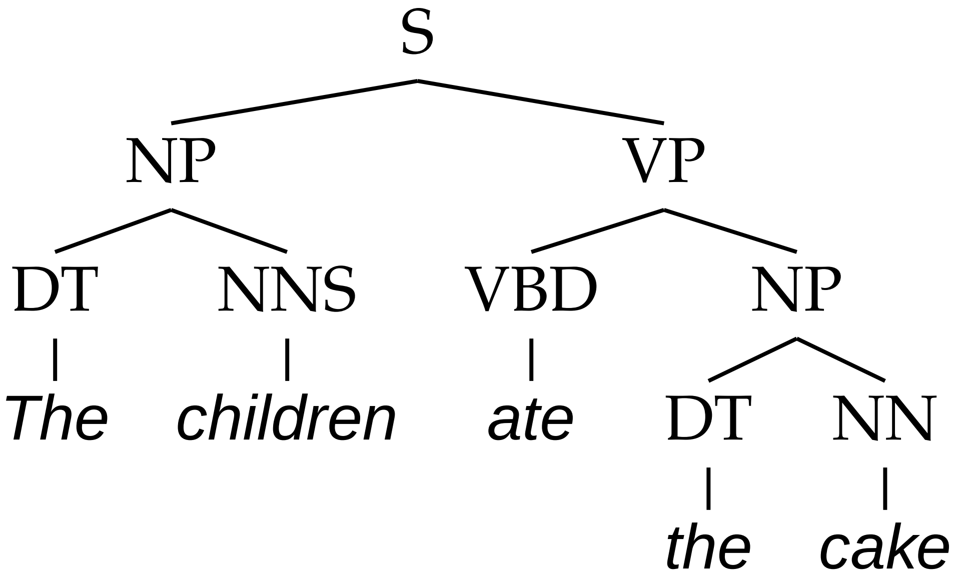
  → DT NNS VBD NP

  → DT NNS VBD DT NN

  → *The children ate the cake*

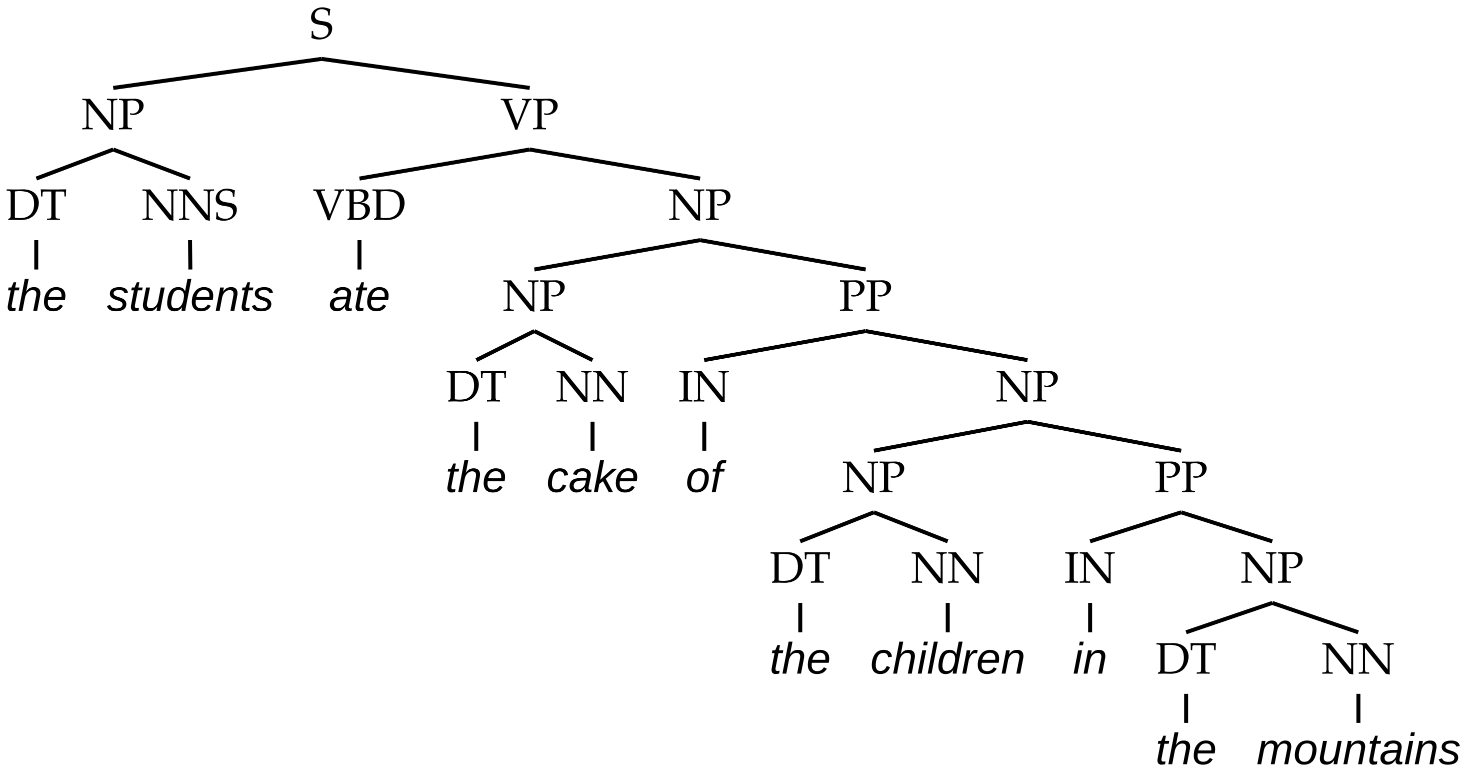
Phrase structure trees

- 

- 

Phrase structure is recursive

So we use at least context-free grammars, in general

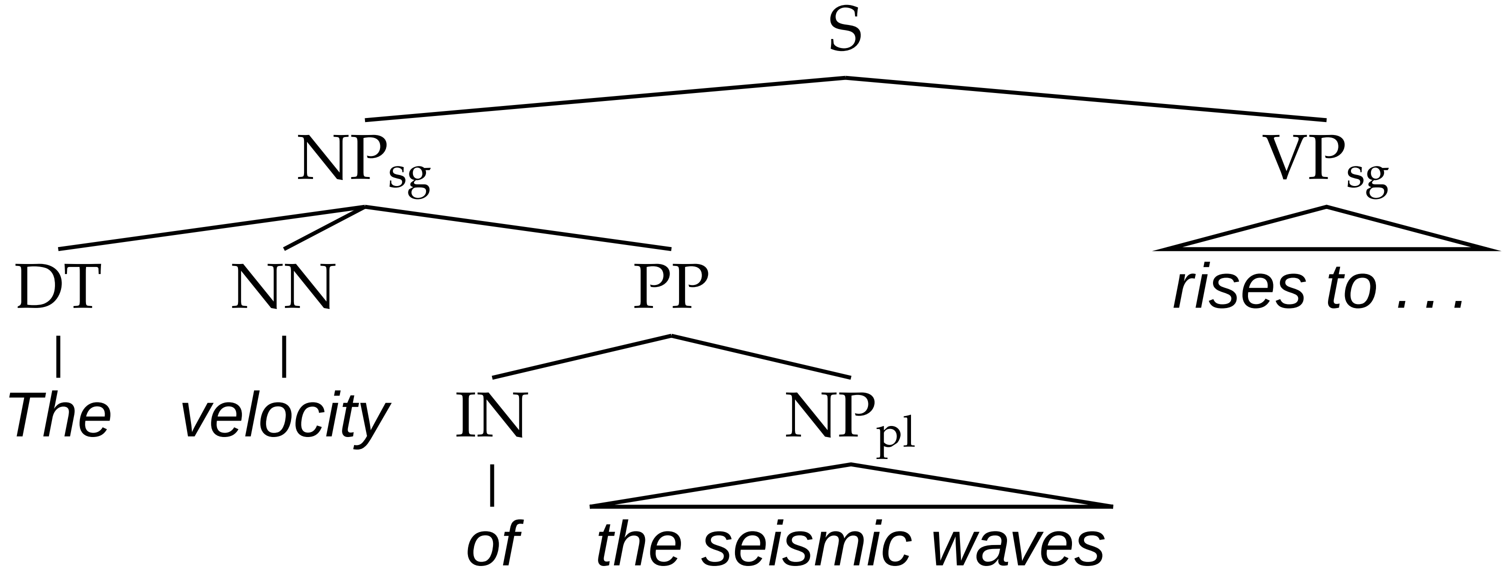


Why we need recursive phrase structure

- Kupiec (1992): Sometimes HMM tagger goes awry:

  waves → verb

- *The velocity of the seismic waves rises to . . .*



- Language model: There are similar problems.

The captain of the ship yelled out.

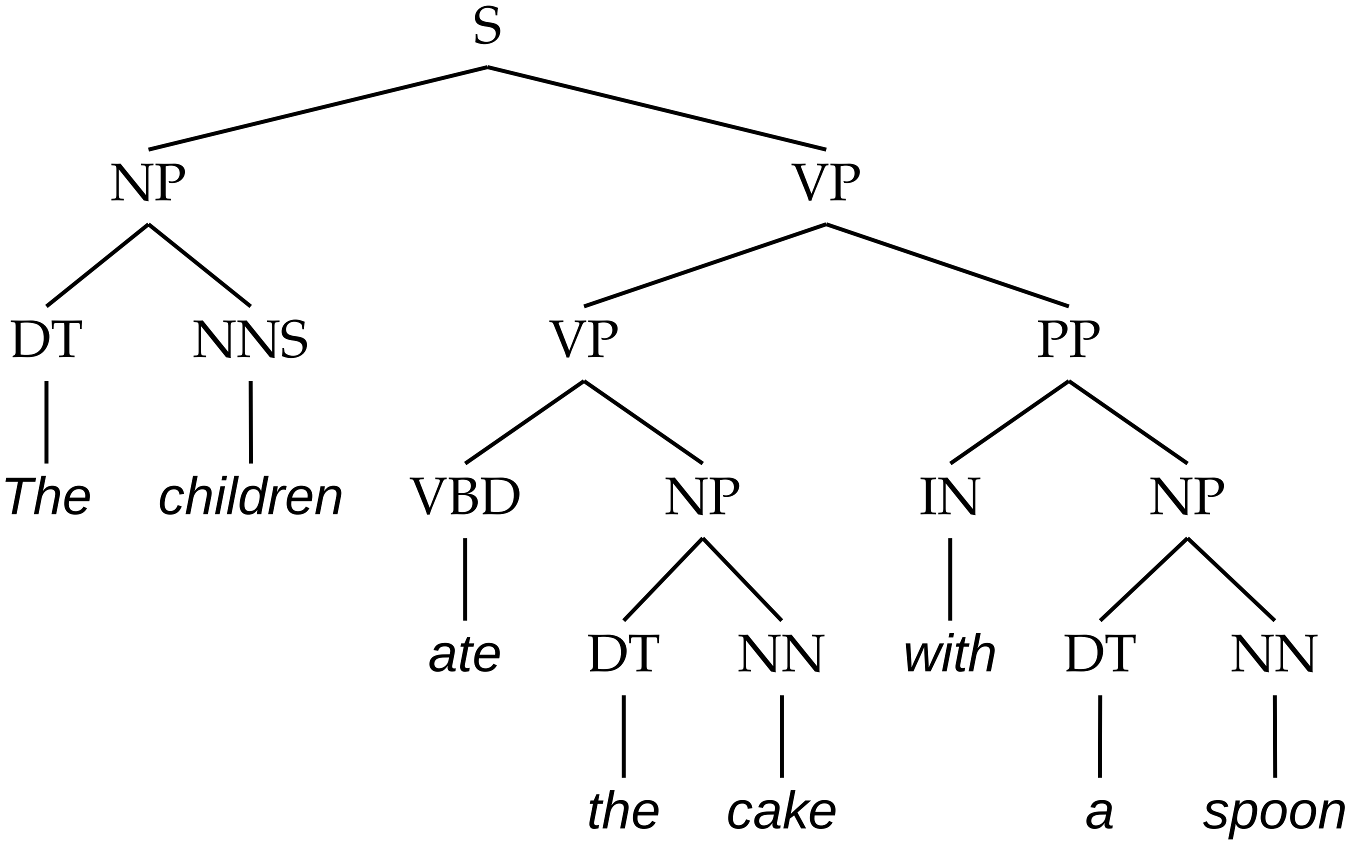
Why we need phrase structure (2)

- Syntax gives important clues in information extraction tasks and some cases of named entity recognition
- *We have recently demonstrated that stimulation of [*$_{\text{CELLTYPE}}$*human T and natural killer cells] with [*$_{\text{PROTEIN}}$*IL-12] induces tyrosine* ***phosphorylation*** *of the [*$_{\text{PROTEIN}}$*Janus family tyrosine kinase] [*$_{\text{PROTEIN}}$*JAK2] and [*$_{\text{PROTEIN}}$*Tyk2].*
- Things that are the object of *phosphorylate* are likely proteins.

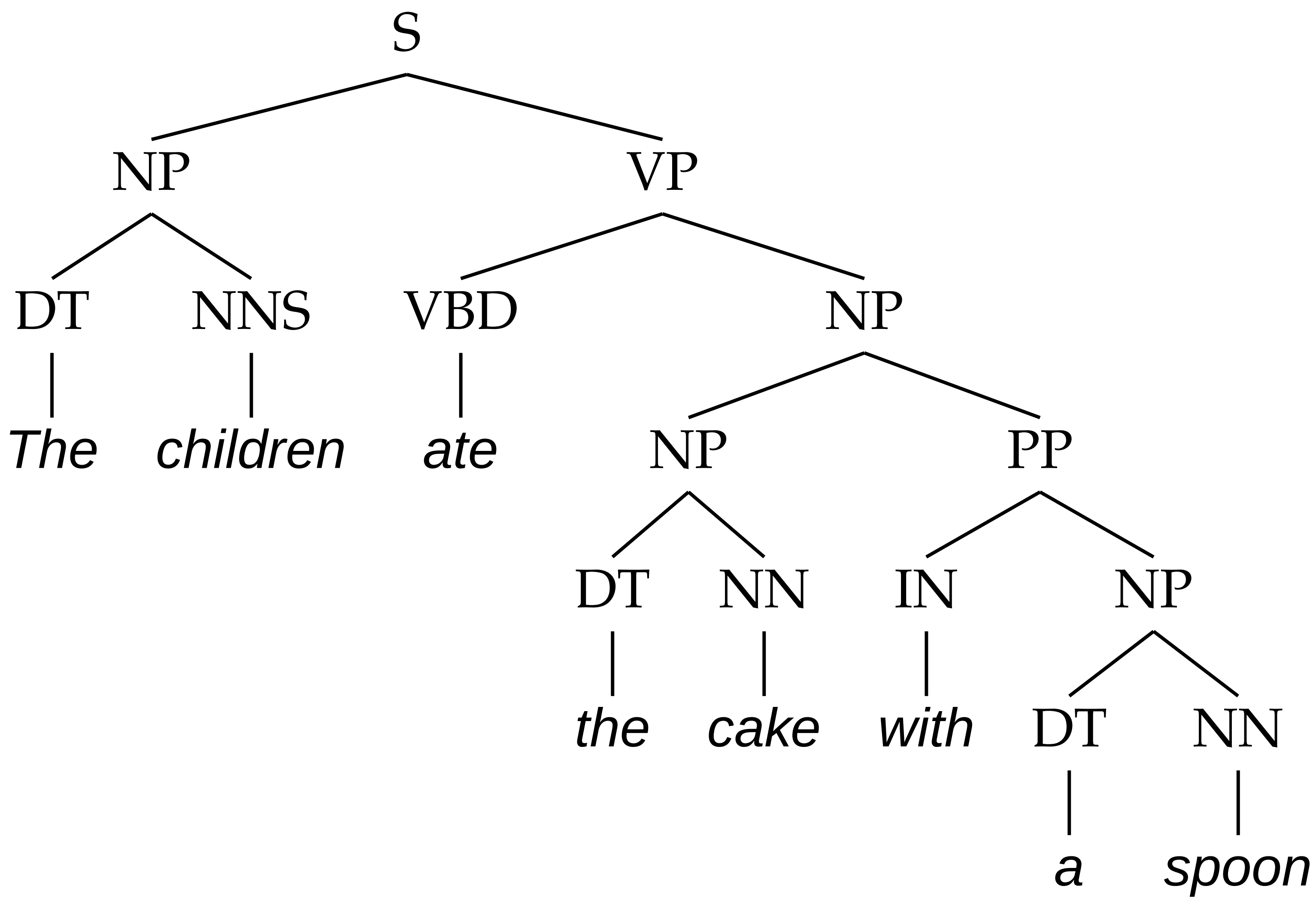
Constituency

- Phrase structure organizes words into nested *constituents*.
- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
  - Distribution: behaves as a unit that appears in different places:
    - *John talked [to the children] [about drugs].*
    - *John talked [about drugs] [to the children].*
    - **John talked drugs to the children about*
  - Substitution/expansion/pro-forms:
    - *I sat [on the box/right on top of the box/there].*
  - Coordination, no intrusion, fragments, semantics, . . .

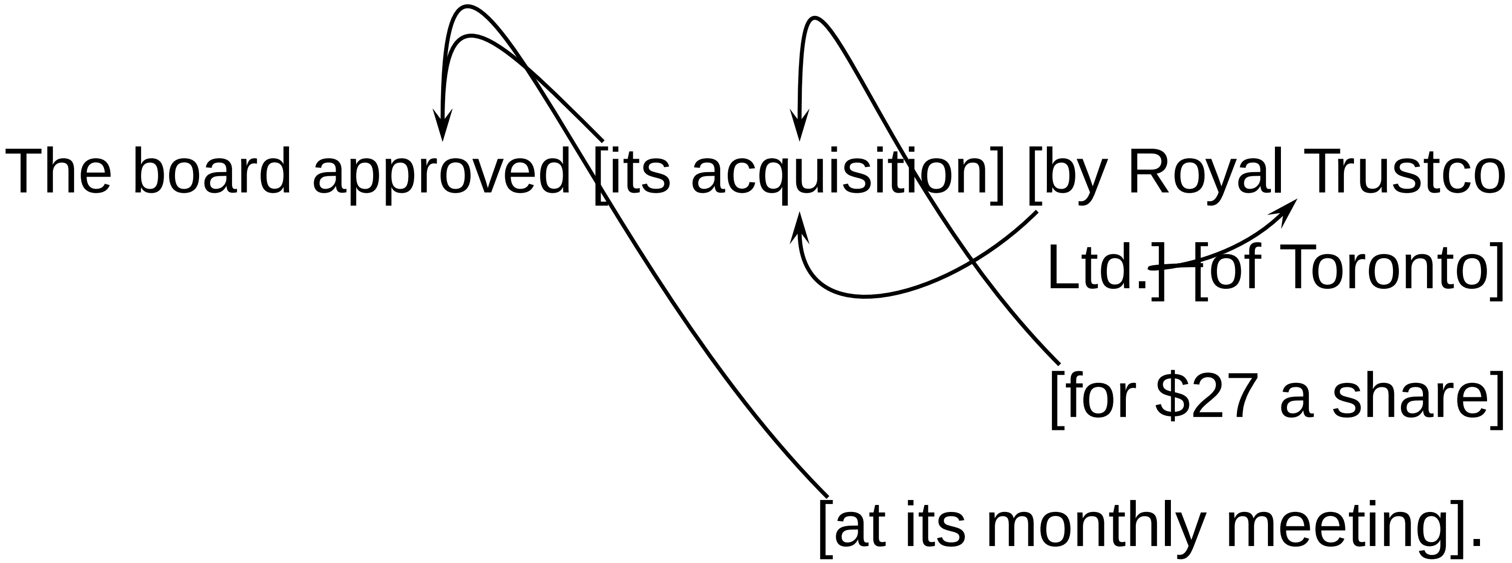
Natural language grammars are ambiguous: Prepositional phrase attaching to verb



Prepositional phrase attaching to noun



Attachment ambiguities in a real sentence



The board approved [its acquisition] [by Royal Trustco
Ltd.] [of Toronto]
[for $27 a share]
[at its monthly meeting].

Penn Treebank Sentences: an example

```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

Ambiguity

- Programming language parsers resolve local ambiguities with lookahead
- Natural languages have global ambiguities:
  - *I saw that gasoline can explode*
- What is the size of embedded NP?

What is parsing?

- We want to run the grammar backwards to find the structures
- Parsing can be viewed as a search problem
- Parsing is a hidden data problem
- We search through the legal rewritings of the grammar
- We want to examine *all* structures for a string of words (for the moment)
- We can do this bottom-up or top-down
    - This distinction is independent of depth-first/bread-first etc. – we can do either both ways
    - Doing this we build a *search tree* which is different from the *parse tree*

Human parsing

- Humans often do ambiguity maintenance
  - *Have the police . . . eaten their supper?*
  - *come in and look around.*
  - *taken out and shot.*
- But humans also commit early and are "garden pathed":
  - *The man who hunts ducks out on weekends.*
  - *The cotton shirts are made from grows in Mississippi.*
  - *The horse raced past the barn fell.*

State space search

- States:
- Operators:
- Start state:
- Goal test:
- *Algorithm*

```
stack = { startState }
solutions = {}
loop
        if stack is empty, return solutions
        state = remove-front(stack)
        if goal(state) push(state, solutions)
        stack = pushAll(expand(state, operators), stack)
end
```

Another phrase structure grammar

S	→	NP VP	N	→	*cats*
VP	→	V NP	N	→	*claws*
VP	→	V NP PP	N	→	*people*
NP	→	NP PP	N	→	*scratch*
NP	→	N	V	→	*scratch*
NP	→	e	P	→	*with*
NP	→	N N	PP	→	P NP

(By linguistic convention, S is the start symbol, but in the PTB, we use the unlabeled node at the top, which can rewrite various ways.)

cats scratch people with claws

S					
NP	VP				
NP	PP	VP			*3 choices*
NP	PP	PP	VP		
oops!					
N	VP				
cats	VP				
cats	V	NP			*2 choices*
cats	scratch	NP			
cats	scratch	N			*3 choices – showing 2nd*
cats	scratch	people			*oops!*
cats	scratch	NP	PP		
cats	scratch	N	PP		*3 choices – showing 2nd* . . .
cats	scratch	people	with	claws	

Phrase Structure (CF) Grammars

$$G = \langle T, N, S, R \rangle$$

- T is set of terminals
- N is set of nonterminals
  - For NLP, we usually distinguish out a set $P \subset N$ of *preterminals* which always rewrite as terminals
- S is start symbol (one of the nonterminals)
- R is rules/productions of the form $X \rightarrow \gamma$, where X is a nonterminal and γ is a sequence of terminals and nonterminals (may be empty)
- A grammar G generates a language L

Recognizers and parsers

- A *recognizer* is a program for which a given grammar and a given sentence returns **yes** if the sentence is accepted by the grammar (i.e., the sentence is in the language) and **no** otherwise
- A *parser* in addition to doing the work of a recognizer also returns the set of parse trees for the string

Soundness and completeness

- A parser is *sound* if every parse it returns is valid/correct
- A parser *terminates* if it is guaranteed to not go off into an infinite loop
- A parser is *complete* if for any given grammar and sentence it is sound, produces every valid parse for that sentence, and terminates
- (For many purposes, we settle for sound but incomplete parsers: e.g., probabilistic parsers that return a k-best list)

Top-down parsing

- Top-down parsing is goal directed
- A top-down parser starts with a list of constituents to be built. The top-down parser rewrites the goals in the goal list by matching one against the LHS of the grammar rules, and expanding it with the RHS, attempting to match the sentence to be derived.
- If a goal can be rewritten in several ways, then there is a choice of which rule to apply (search problem)
- Can use depth-first or breadth-first search, and goal ordering.

Bottom-up parsing

- Bottom-up parsing is data directed
- The initial goal list of a bottom-up parser is the string to be parsed. If a sequence in the goal list matches the RHS of a rule, then this sequence may be replaced by the LHS of the rule.
- Parsing is finished when the goal list contains just the start category.
- If the RHS of several rules match the goal list, then there is a choice of which rule to apply (search problem)
- Can use depth-first or breadth-first search, and goal ordering.
- The standard presentation is as *shift-reduce* parsing.

Problems with top-down parsing

- Left recursive rules
- A top-down parser will do badly if there are many different rules for the same LHS. Consider if there are 600 rules for S, 599 of which start with NP, but one of which starts with V, and the sentence starts with V.
- Useless work: expands things that are possible top-down but not there
- Top-down parsers do well if there is useful grammar-driven control: search is directed by the grammar
- Top-down is hopeless for rewriting parts of speech (preterminals) with words (terminals). In practice that is always done bottom-up as lexical lookup.

- Repeated work: anywhere there is common substructure

Problems with bottom-up parsing

- Unable to deal with empty categories: termination problem, unless rewriting empties as constituents is somehow restricted (but then it's generally incomplete)
- Useless work: locally possible, but globally impossible.
- Inefficient when there is great lexical ambiguity (grammar-driven control might help here)
- Conversely, it is data-directed: it attempts to parse the words that are there.
- Repeated work: anywhere there is common substructure
- Both TD (LL) and BU (LR) parsers can (and frequently do) do work exponential in the sentence length on NLP problems.

Principles for success: what one needs to do

- If you are going to do parsing-as-search with a grammar as is:
  - Left recursive structures must be found, not predicted
  - Empty categories must be predicted, not found
- Doing these things doesn't fix the repeated work problem.

An alternative way to fix things

- Grammar transformations can fix both left-recursion and epsilon productions
- Then you parse the same language but with different trees
- Linguists tend to hate you
  - But this is a misconception: they shouldn’t
  - You can fix the trees post hoc

A second way to fix things

- Rather than doing parsing-as-search, we do parsing as dynamic programming
- This is the most standard way to do things
- It solves the problem of doing repeated work
- But there are also other ways of solving the problem of doing repeated work
  - Memoization (remembering solved subproblems)
  - Doing graph-search rather than tree-search.

Filtering

- Conversion to CNF. First remove ϵ categories.
- Directed vs. Undirected parsers: using the opposite direction for x filtering.

Left corner parsing

- Left corner parsing: Accept word. What is it left-corner of? Parse that constituent top down. Can prune on top-down knowledge. Doesn’t have problem with left recursion except with unaries. Does have problem with empties in left corner, but not while working top down.

n-gram models and statistical estimation

FSNLP, chapter 6

Christopher Manning and Hinrich Schütze

Probabilistic language modeling

- Assigns probability $P(t)$ to a word sequence $t = w_1 w_2 \cdots w_n$
- Chain rule and joint/conditional probabilities for text t:

$$\begin{aligned} P(t) = P(w_1 \cdots w_n) &= P(w_1) \cdots P(w_n | w_1, \cdots w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i | w_1 \cdots w_{i-1}) \end{aligned}$$

where

$$P(w_k | w_1 \ldots w_{k-1}) = \frac{P(w_1 \ldots w_k)}{P(w_1 \ldots w_{k-1})} \approx \frac{C(w_1 \ldots w_k)}{C(w_1 \ldots w_{k-1})}$$

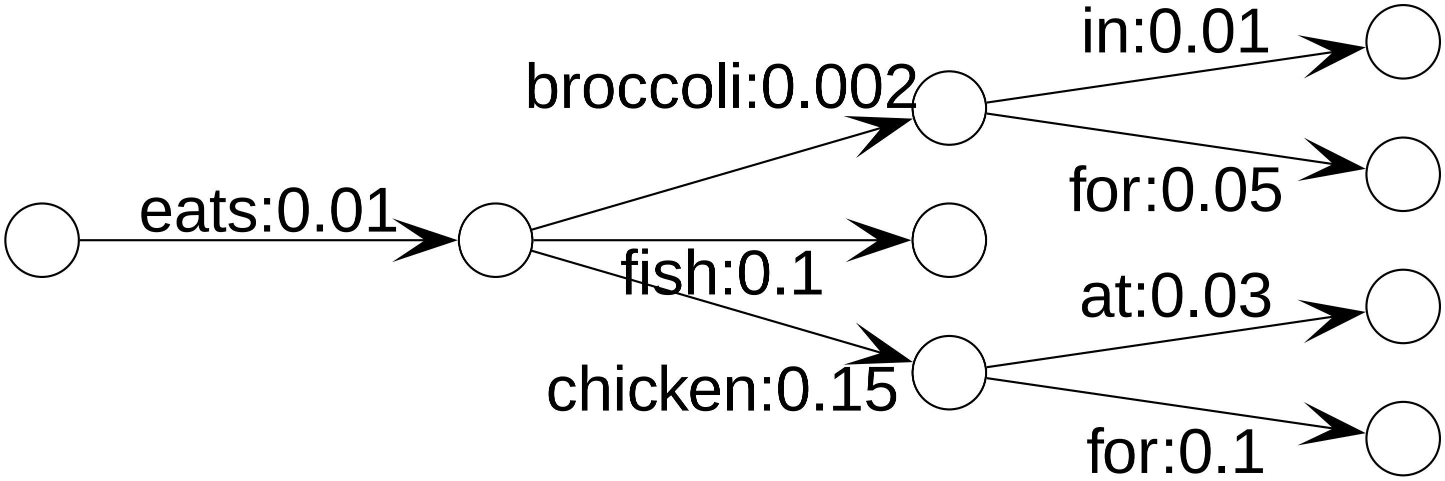
- The chain rule leads to a *history-based* model: we predict following things from past things
- But there are too many histories; we need to cluster histories into equivalence classes

n-gram models: the classic example of a statistical model of language

- Each word is predicted according to a conditional distribution based on a limited prior context
- Conditional Probability Table (CPT): $P(X|\textit{both})$
  - $P(\textit{of}|\textit{both}) = 0.066$
  - $P(\textit{to}|\textit{both}) = 0.041$
  - $P(\textit{in}|\textit{both}) = 0.038$
- From 1940s onward (or even 1910s – Markov 1913)
- a.k.a. Markov (chain) models

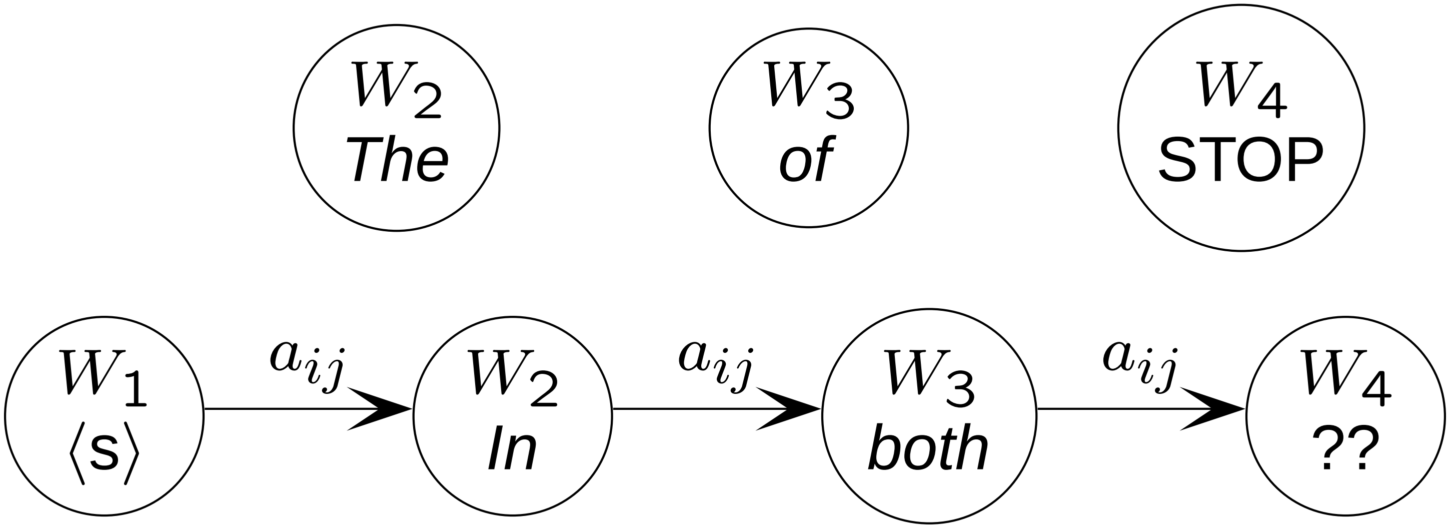
Markov models = n-gram models

- Deterministic FSMs with probabilities



- No long distance dependencies
  - “The future is independent of the past given the present”
- No notion of structure or syntactic dependency
- But lexical
- (And: robust, have frequency information, . . .)

Markov models = n-gram models



- Simplest linear graphical models
- Words are random variables, arrows are direct dependencies between them (CPTs)

n-gram models

- Core language model for the engineering task of better predicting the next word:
  - Speech recognition
  - OCR
  - Context-sensitive spelling correction
- These simple engineering models have just been amazingly successful.
- It is only recently that they have been improved on for these tasks (Chelba and Jelinek 1998; Charniak 2001).
- But linguistically, they are appalling simple and naive

n-th order Markov models

- First order Markov assumption = bigram

$$P(w_k|w_1 \ldots w_{k-1}) \approx P(w_k|w_{k-1}) = \frac{P(w_{k-1}w_k)}{P(w_{k-1})}$$

- Similarly, n-th order Markov assumption
- Most commonly, trigram (2nd order):

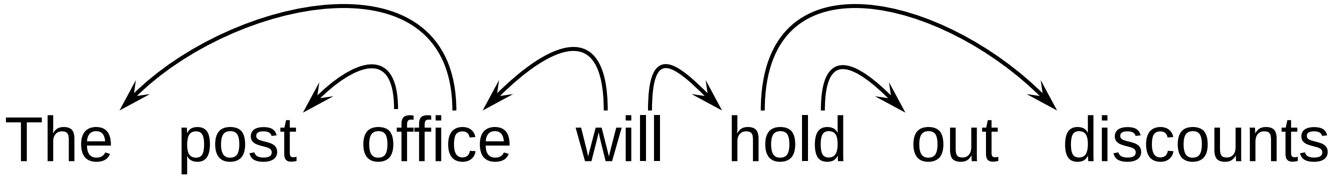
$$P(w_k|w_1 \ldots w_{k-1}) \approx P(w_k|w_{k-2}, w_{k-1}) = \frac{P(w_{k-2}w_{k-1}w_k)}{P(w_{k-2}, w_{k-1})}$$

Why mightn't n-gram models work?

- Relationships (say between subject and verb) can be arbitrarily distant and convoluted, as linguists love to point out:
  - The ***man*** that I was watching without pausing to look at what was happening down the street, and quite oblivious to the situation that was about to befall him confidently ***strode*** into the center of the road.

Why do they work?

- That kind of thing doesn't happen much
- Collins (1997):
  - 74% of dependencies (in the Penn Treebank – WSJ) are with an adjacent word (95% with one $\leq$ 5 words away), once one treats simple NPs as units:
  - Below, 4/6 = 66% based on words



The post office will hold out discounts

Why is that?

Sapir (1921: 14):

> ‘When I say, for instance, “I had a good breakfast this morning,” it is clear that I am not in the throes of laborious thought, that what I have to transmit is hardly more than a pleasurable memory symbolically rendered in the grooves of habitual expression. . . . It is somewhat as though a dynamo capable of generating enough power to run an elevator were operated almost exclusively to feed an electric doorbell.’

Evaluation of language models

- Best evaluation of probability model is task-based
- As substitute for evaluating one component, standardly use corpus per-word cross entropy:

$$H(X, \mathsf{p}) = -\frac{1}{n} \sum_{i=1}^{n} \log_2 P(w_i | w_1, \ldots, w_{i-1})$$

- Shannon game: try to predict next word in discourse
- Or perplexity (measure of uncertainty of predictions):

$$PP(X, \mathsf{p}) = 2^{H(X,\mathsf{p})} = \left[\prod_{i=1}^{n} P(w_i | w_1, \ldots, w_{i-1}) \right]^{-1/n}$$

- *Needs to be assessed on independent, unseen, test data*

Relative frequency = Maximum Likelihood Estimate

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

(or similarly for higher order or joint probabilities)

Makes training data as probable as possible

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Selected bigram counts (Berkeley Restaurant Project – J&M)

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Selected bigram probabilities (Berkeley Restaurant Project – J&M)

Limitations of Maximum Likelihood Estimator

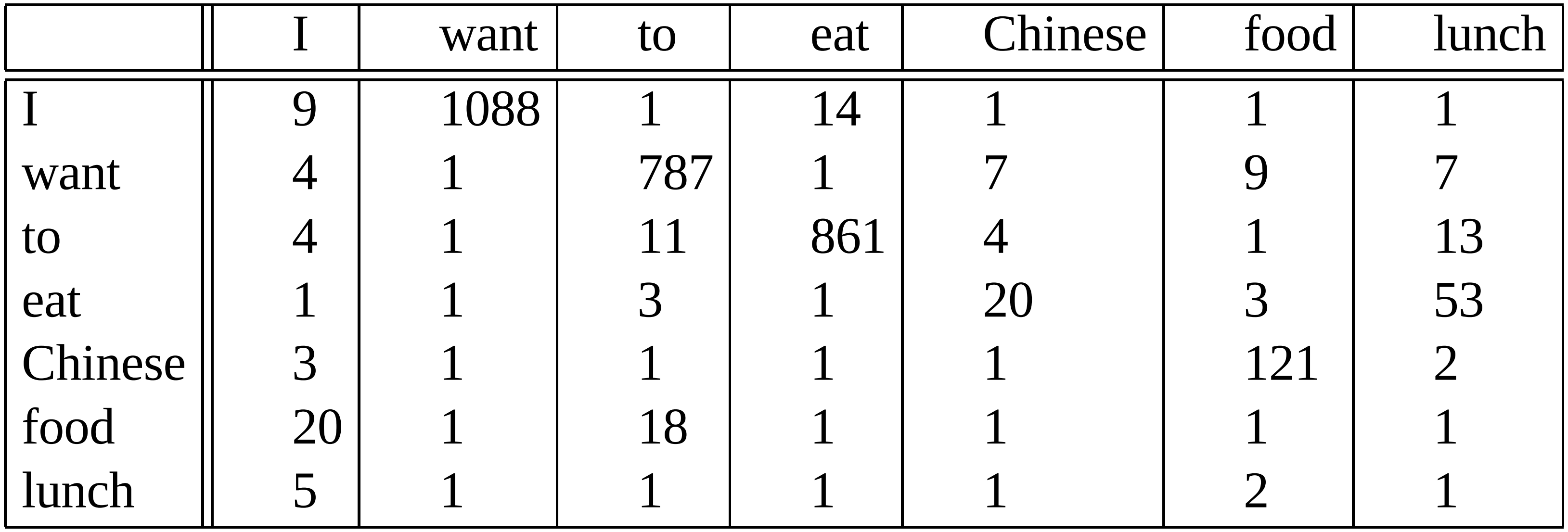
Problem: We are often infinitely surprised when unseen word appears ($P(unseen) = 0$)

- Problem: this happens commonly.
- Probabilities of zero count words are too low
- Probabilities of nonzero count words are too high
- Estimates for high count words are fairly accurate
- Estimates for low count words are mostly inaccurate
- We need smoothing! (We flatten spiky distribution and give shavings to unseen items.)

Adding one = Laplace’s law (1851)

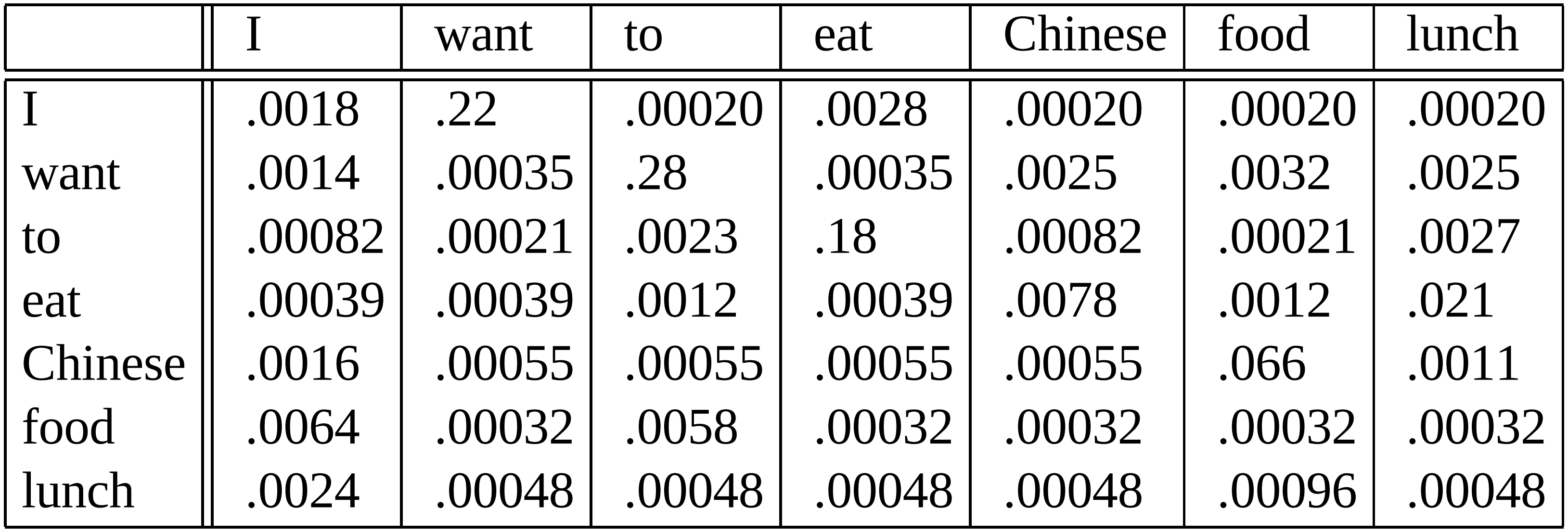
$$P(w_2|w_1) = \frac{C(w_1, w_2) + 1}{C(w_1) + V}$$

- V is the vocabulary size (assume fixed, closed vocabulary)
- This is the Bayesian (MAP) estimator you get by assuming a uniform unit prior on events (= a Dirichlet prior)



	I	want	to	eat	Chinese	food	lunch
I	9	1088	1	14	1	1	1
want	4	1	787	1	7	9	7
to	4	1	11	861	4	1	13
eat	1	1	3	1	20	3	53
Chinese	3	1	1	1	1	121	2
food	20	1	18	1	1	1	1
lunch	5	1	1	1	1	2	1

Add one counts (Berkeley Restaurant Project – J&M)



	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

Add one probabilities (Berkeley Restaurant Project – J&M)

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	want	to	eat	Chinese	food	lunch
I	6	740	.68	10	.68	.68	.68
want	2	.42	331	.42	3	4	3
to	3	.69	8	594	3	.69	9
eat	.37	.37	1	.37	7.4	1	20
Chinese	.36	.12	.12	.12	.12	15	.24
food	10	.48	9	.48	.48	.48	.48
lunch	1.1	.22	.22	.22	.22	.44	.22

Original versus add-one predicted counts

Add one estimator

- Problem: gives too much probability mass to unseens.
- Not good for large vocab, comparatively little data (i.e., NLP)
- e.g 10,000 word vocab, 1,000,000 words of training data, but *comes across* occurs 10 times. Of those, 8 times next word is *as*
  - $P_{\text{MLE}}(\textit{as}|\textit{comes across}) = 0.8$
  - $P_{+1}(\textit{as}|\textit{comes across}) = \frac{8+1}{10+10000} \approx 0.0009$

Partial fixes

- Quick fix: Lidstone's law (Mitchell's (1997) "m-estimate"):

  $$P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda}{C(w_1) + \lambda V}$$

  for $\lambda < 1$, e.g., 1/2 or 0.05.

  – Mitchell's m-estimate sets λV to be m and subdividing it between the words

  – Doesn't correctly estimate difference between things seen 0 and 1 time

- Unigram prior

  – More likely to see next unseen words that are a priori common

  $$P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda P(w_2)}{C(w_1) + \lambda}$$

Absolute discounting

- Idea is that we want to discount counts of seen things a little, and reallocate this probability mass to unseens
- By subtracting a fixed count, probability estimates for commonly seen things are scarcely affected, while probabilities of rare things are greatly affected
- If the discount is around $\delta = 0.75$, then seeing something once is not so different to not having seen it at all

$$P(w_2|w_1) = (C(w_1, w_2) - \delta)/C(w_1) \quad \text{if} \quad C(w_1, w_2) > 0$$

$$P(w_2|w_1) = (V - N_0)\delta/N_0 C(w_1) \quad \text{otherwise}$$

The frequency of previously unseen events

How do you know how likely you are to see a new word type in the future (in a certain context)?

- Examine some further text and find out [empirical held out estimators = validation]
- Use things you've seen once to estimate probability of unseen things:

  $$P(\text{unseen}) = \frac{N_1}{N}$$

  where N_1 is number of things seen once. (Good-Turing: Church and Gale 1991; Gale and Sampson 1995)

Good-Turing smoothing

Derivation reflects leave-one out estimation (Ney et al. 1997):

- For each word token in data, call it the test set; remaining data is training set
- See how often word in test set has r counts in training set
- This will happen every time word left out has $r + 1$ counts in original data
- So total count mass of r count words is assigned from mass of $r + 1$ count words [$= N_{r+1} \times (r + 1)$]
- Doesn't require held out data (which is good!)

Good-Turing smoothing

- r* is corrected frequency estimate for word occurring r times
- There are N_r words with count r in the data
- $N_r \times r* = N_{r+1} \times (r + 1)$ or
- $r* = \frac{N_{r+1} \times (r+1)}{N_r}$
- Or if w had frequency r, $P(w) = (r + 1)N_{r+1}/N_r N$
- All words with same count get same probability
- This reestimation needs smoothing.
- For small r, $N_r > N_{r+1}$. But what of *the*?
- Simple Good Turing: use best-fit power law on low count counts.

Smoothing: Rest of the story (1)

- Other methods: backoff (Katz 1987), cross-validation, Witten-Bell discounting, . . . (Chen and Goodman 1998; Goodman 2001)
- Simple, but surprisingly effective: Simple linear interpolation (deleted interpolation; mixture model; shrinkage):

$$\widehat{P}(w_3|w_1,w_2)=\lambda_3 P_3(w_3|w_1,w_2)+\lambda_2 P_2(w_3|w_2)+\lambda_1 P_1(w_3)$$

- The λ_i can be estimated on held out data
- They can be functions of (equivalence-classed) histories
- For open vocabulary, need to handle words unseen in any context (just use UNK, spelling models, etc.)

Smoothing: Rest of the story (2)

- Recent work emphasizes constraints on the smoothed model
- Kneser and Ney (1995): Backoff n-gram counts not proportional to frequency of n-gram in training data but to expectation of how often it should occur in novel trigram – since one only uses backoff estimate when trigram not found
- (Smoothed) maximum entropy (a.k.a. loglinear) models again place constraints on the distribution (Rosenfeld 1996, 2000)

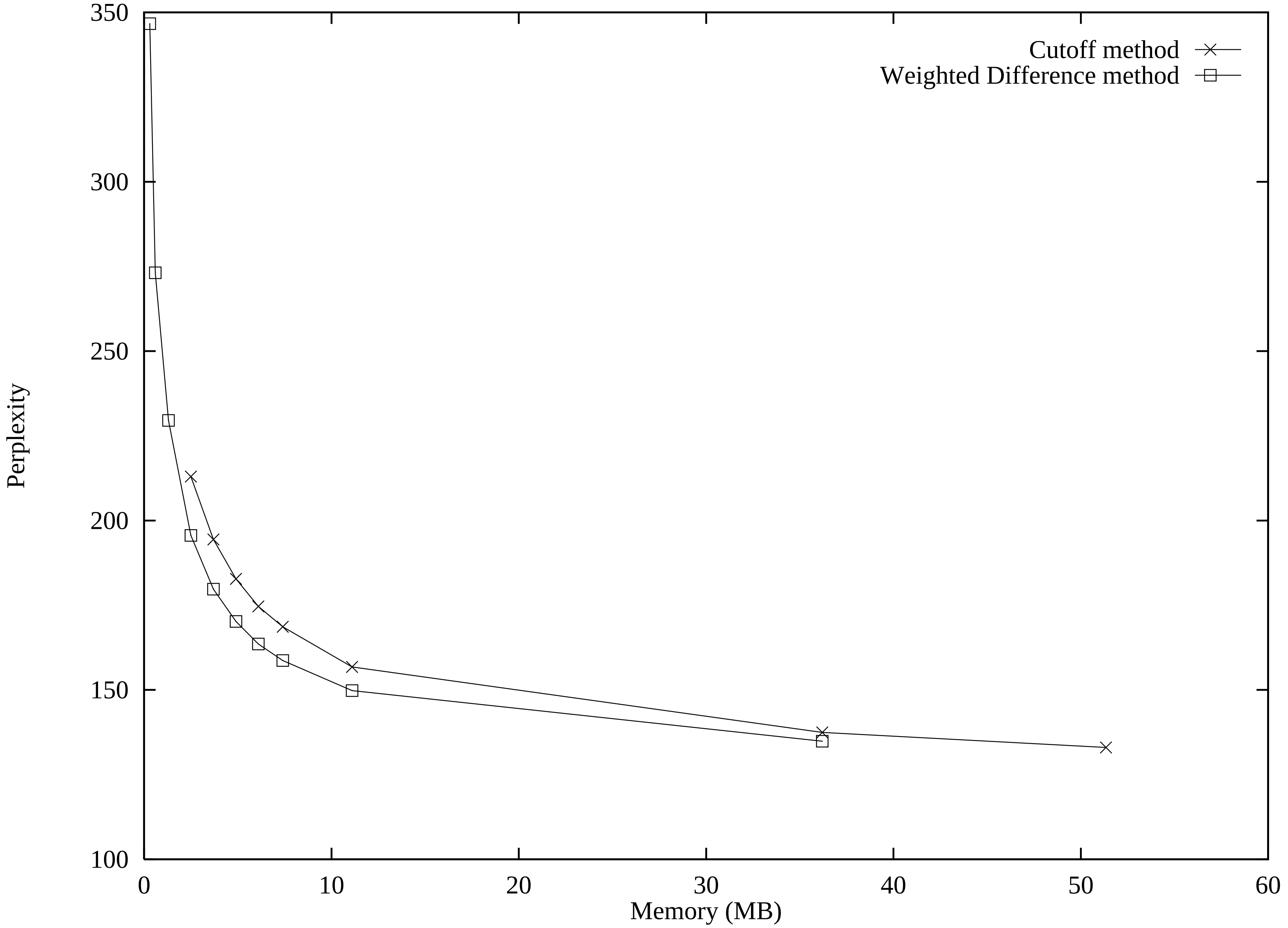
Size of language models with cutoffs

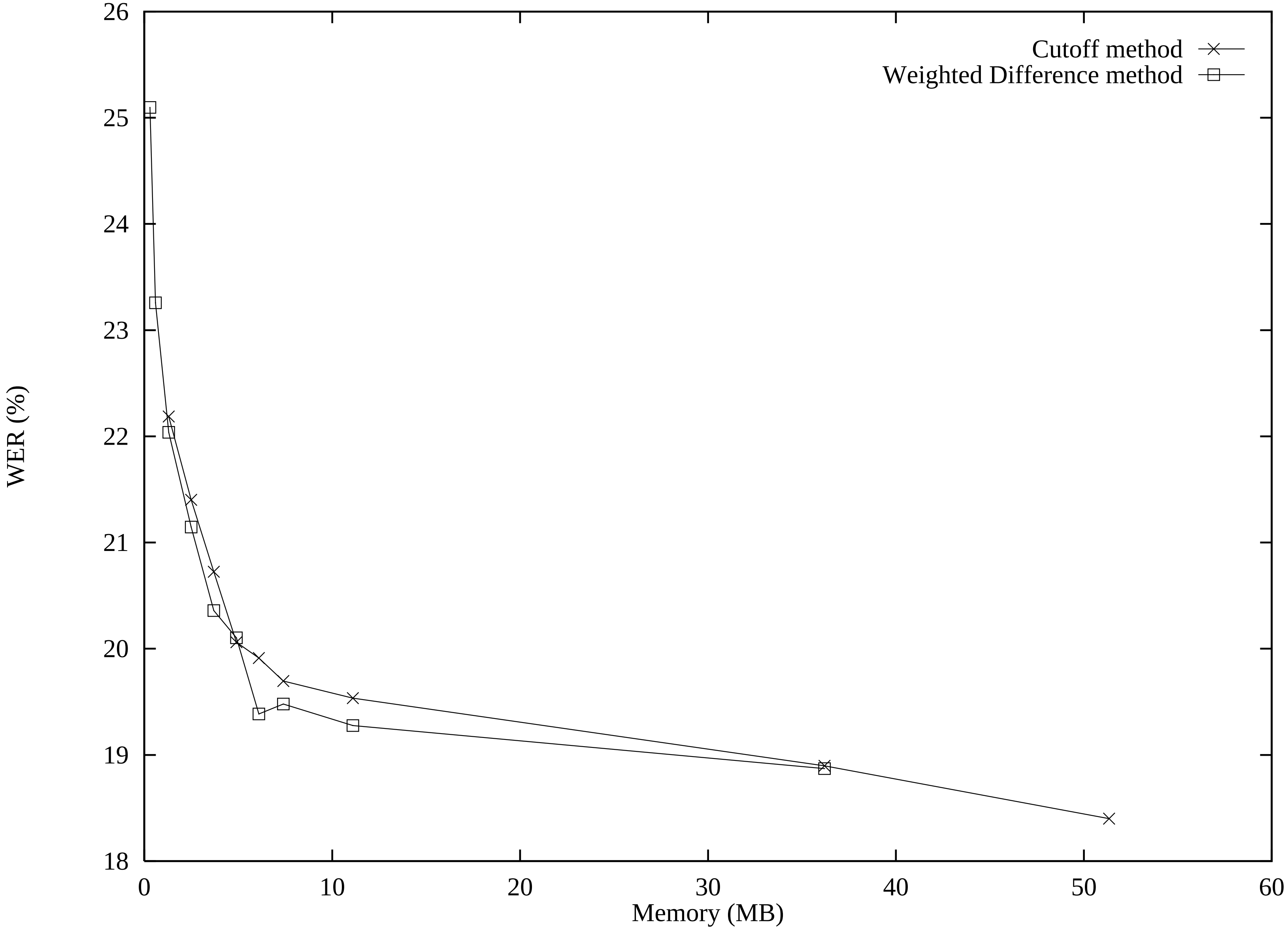
Seymore and Rosenfeld (*ICSLP*, 1996): 58,000 word dictionary, 45 M words of training data, *WSJ*, Sphinx II

Bi/Tri-gram cutoff	# Bigrams	# Trigrams	Memory (MB)
0/0	4,627,551	16,838,937	104
0/1	4,627,551	3,581,187	51
1/1	1,787,935	3,581,187	29
10/10	347,647	367,928	4

80% of unique trigrams occur only once!

- Note the possibilities for compression (if you're confident that you'll be given English text and the encoder/ decoder can use very big tables)


Cutoff method
Weighted Difference method
Perplexity
350
300
250
200
150
100
0
10
20
30
40
50
60
Memory (MB)

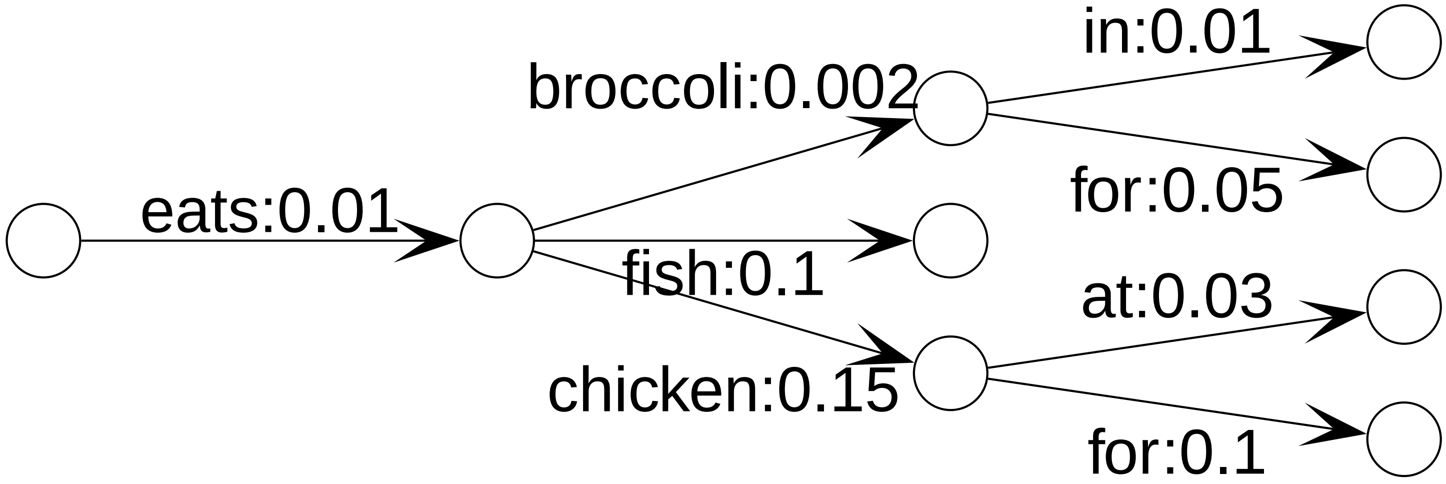

26
25
24
23
22
21
20
19
18
WER (%)
0
10
20
30
40
50
60
Memory (MB)
Cutoff method
Weighted Difference method

More LM facts

- Seymore, Chen, Eskenazi and Rosenfeld (1996)
- HUB-4: Broadcast News 51,000 word vocab, 130M words training. Katz backoff smoothing (1/1 cutoff).
- Perplexity 231
- 0/0 cutoff: 3% perplexity reduction
- 7-grams: 15% perplexity reduction
- Note the possibilities for compression, if you're confident that you'll be given English text (and the encoder/ decoder can use very big tables)

Extra slides

Markov models = n-gram models



Markov models

- Deterministic FSMs with probabilities
- No long distance dependencies
  - “The future is independent of the past given the present”
- No notion of structure or syntactic dependency
- But lexical
- (And: robust, have frequency information, . . .)

Probability estimates for StatNLP

$$P(X_2|X_1 = w_1) = P(w_2|w_1)$$

Relative frequency = MLE

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

Problem: often infinitely surprised when unseen word appears

Problem: this happens commonly

Adding one = Laplace's law

$$P(w_2|w_1) = \frac{C(w_1, w_2) + 1}{C(w_1) + V}$$

(Is Bayesian estimator assuming a uniform prior on events.)

Problem: gives too much probability mass to unseens.

Quick fix: Lidstone's law:

$$P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda}{C(w_1) + \lambda V}$$

for $\lambda < 1$, e.g., 1/2

Better methods

- Lots – in chapter 6. We won't dwell on them.
- Simple but surprisingly effective: mixture models. Simple linear interpolation:

$$P(w_2|w_1) = \lambda_1 P_1(w2) + \lambda_2 P_2(w2|w_1)$$

- Open vocabulary. Unknown words.

Managing data

- Training data
- Validation data
- Final testing data
- Cross-validation
  - One score doesn’t allow system comparison
  - This allows confidence ranges to be computed
  - And systems to be compared with confidence!

Dealing with unseens: Add one estimator = Laplace's law (1814)

- Bayesian estimator assuming a uniform prior on events:

$$P(w|h) = \frac{C(h,w)+1}{C(h)+V}$$

- Problem: gives too much probability mass to unseens.
- Not good for large vocab, comparatively little data
- E.g., word trigrams: 10,000 word vocab; 10,000,000 words of training data; *comes across* occurs 10 times. Of those, 8 times next word is *as*
  - $P_{\text{MLE}}(\textit{as}|\textit{comes across}) = 0.8$
  - $P_{+1}(\textit{as}|\textit{comes across}) = \frac{8+1}{10+10000} \approx 0.0009$

The frequency of previously unseen events

- Quick fix: Lidstone's law: $P(w|h) = \frac{C(h,w)+\lambda}{C(h)+\lambda V}$ for $\lambda < 1$, e.g., 1/2 or 0.05
- How do you know how likely you are to see a new word type in the future (in a certain context)?
  - Examine some further text and find out [held out estimators]
  - Use things you've seen once to estimate probability of unseen things:

    $$P(\text{unseen}) = \frac{N_1}{N}$$

    N_1 is number of things seen once. (Good-Turing: Church and Gale 1991; Gale and Sampson 1995)

Adding one = Laplace’s law

$$P(w_2|w_1) = \frac{C(w_1, w_2) + 1}{C(w_1) + V}$$

- Is Bayesian estimator assuming a uniform prior on events.
- Simplest, but not very good answer to avoiding zero probabilities
- There are much better methods, but I’m putting them aside as ‘technical details’ (Manning and Schütze 1999: ch.6)

Language model topic determination

- Start with some documents labeled for topic (c_i)
- Train an n-gram language model just on documents of each topic, which we regard as a ‘language’
- Testing: Decide which topic/language is most likely to have generated a new document, by calculating the $P(w_1 \cdots w_n | c_i)$
- Choose the most probable one as the topic of the document

Disambiguating using ‘language’ models

- Supervised training from hand-labeled examples
- Train n-gram language model for examples of each sense, treating examples as a ‘language’
  - estimate $P(\text{port}|\text{sailed}, \text{into})$, etc.
  - reduce parameters by backing off where there is insufficient data: $P(\text{port}|\text{into})$ or $P(\text{port})$ [unigram estimate for sense]
- Disambiguate based on in which ‘language’ the sentence would have highest probability
- This gives some of the advantages of wide context bag of words models (Naive Bayes-like) and use of local structural cues around word

Word Sense Disambiguation

FSNLP, chapter 7

Christopher Manning and Hinrich Schütze

Word sense disambiguation

- The task is to determine which of various senses of a word are invoked in context:
  - *the seed companies cut off the tassels of each **plant**, making it male sterile*
  - *Nissan's Tennessee manufacturing **plant** beat back a United Auto Workers organizing effort with aggressive tactics*
- This is an important problem: Most words are ambiguous (have multiple senses)
- Converse: words or senses that mean (almost) the same: *image, likeness, portrait, facsimile, picture*

WSD: Many other cases are harder

- *title:*
  - Name/heading of a book, statute, work of art or music, etc.
  - Material at the start of a film
  - The right of legal ownership (of land)
  - The document that is evidence of this right
  - An appellation of respect attached to a person’s name
  - A written work

WSD: The many meanings of *interest* [n.]

- Readiness to give attention to or to learn about something
- Quality of causing attention to be given
- Activity, subject, etc., which one gives time and attention to
- The advantage, advancement or favor of an individual or group
- A stake or share (in a company, business, etc.)
- Money paid regularly for the use of money

WSD: Many other cases are harder

- *modest:*
  - In evident apprehension that such a prospect might frighten off the young or composers of more modest_1 forms –
  - Tort reform statutes in thirty-nine states have effected modest_9 changes of substantive and remedial law
  - The modest_9 premises are announced with a modest and simple name –
  - In the year before the Nobel Foundation belatedly honoured this modest_0 and unassuming individual,
  - LinkWay is IBM's response to HyperCard, and in Glasgow (its UK launch) it impressed many by providing colour, by its modest_9 memory requirements,
  - In a modest_1 mews opposite TV-AM there is a rumpled hyperactive figure
  - He is also modest_0: the "help to" is a nice touch.

WSD: types of problems

- Homonymy: meanings are unrelated: *bank* of river and *bank* financial institution
- Polysemy: related meanings (as on previous slides)
- Systematic polysemy: standard methods of extending a meaning, such as from an organization to the building where it is housed.
- A word frequently takes on further related meanings through systematic polysemy or metaphor

Word sense disambiguation

- Most early work used semantic networks, frames, logical reasoning, or “expert system” methods for disambiguation based on contexts (e.g., Small 1980, Hirst 1988).
- The problem got quite out of hand:
  - The word expert for ‘throw’ is “currently six pages long, but should be ten times that size” (Small and Rieger 1982)
- Supervised sense disambiguation through use of context is frequently extremely successful – and is a straightforward classification problem
- “You shall know a word by the company it keeps” – Firth
- However, it requires extensive annotated training data

Some issues in WSD

- Supervised vs. unsupervised
  - Or better: What are the knowledge sources used?
- Pseudowords
  - Pain-free creation of training data
  - Not as realistic as real words
- Upper and lower bounds: how hard is the task?
  - Lower bound: go with most common sense (can vary from 20% to 90+% performance)
  - Upper bound: usually taken as human performance

Other (semi-)supervised WSD

- Brown et al. (1991b): used just one key indicating (linguistic) feature (e.g., object of verb) and partitioned its values
- Lesk (1986) used a dictionary; Yarowsky (1992) used a thesaurus
- Use of a parallel corpus (Brown et al. 1991b) or a bilingual dictionary (Dagan and Itai 1994)
- Use of decomposable models (a more complex Markov random field model) (Bruce and Wiebe 1994, 1999)

Unsupervised and semi-supervised WSD

- Really, if you want to be able to do WSD in the large, you need to be able to disambiguate all words as you go.
- You’re unlikely to have a ton of hand-built word sense training data for all words.
- Or you might: the OpenMind Word Expert project:
  - http://teach-computers.org/word-expert.html

Unsupervised and semi-supervised WSD

- Main hope is getting indirect supervision from existing broad coverage resources:
  - Lesk (1986) used a dictionary; Yarowsky (1992) used a thesaurus
  - Use of a parallel corpus (Brown et al. 1991b) or a bilingual dictionary (Dagan and Itai 1994)

  This can be moderately successful. (Still not nearly as good as supervised systems. Interesting research topic.
- There is work on fully unsupervised WSD
  - This is effectively word sense clustering or word sense discrimination (Schütze 1998).

- Usually no outside source of truth
- Can be useful for IR, etc. though

Lesk (1986)

- Words in context can be mutually disambiguated by overlap of their defining words in a dictionary
  - **ash**
    1. the **solid** residue left when **combustible** material is thoroughly **burned** . . .
    2. Something that symbolizes grief or repentence
  - **coal**
    1. a black or brownish black **solid combustible** substances . . .
- We'd go with the first sense of **ash**
- Lesk reports performance of 50%–70% from brief experimentation

Collocations/selectional restrictions

- Sometimes a single feature can give you very good evidence – and avoids need for feature combination
- Traditional version: selectional restrictions
  - Focus on constraints of main syntactic dependencies
  - *I hate washing* ***dishes***
  - *I always enjoy spicy* ***dishes***
  - Selectional restrictions may be weak, made irrelevant by negation or stretched in metaphors or by odd events
- More recent versions: Brown et al. (1991), Resnik (1993)

- Non-standard good indicators: tense, adjacent words for collocations (*mace spray*; *mace* and *parliament*)

Global constraints: Yarowsky (1995)

- One sense per discourse: the sense of a word is highly consistent within a document
  - True for topic dependent words
  - Not so true for other items like adjectives and verbs, e.g. *make, take*
  - Krovetz (1998) “More than One Sense Per Discourse” argues it isn’t true at all once you move to fine-grained senses
- One sense per collocation: A word reoccurring in collocation with the same word will almost surely have the same sense
  - This is why Brown et al.’s (1991b) use of just one disambiguating feature was quite effective

Unsupervised disambiguation

- Word sense discrimination (Schütze 1998) or clustering
- Useful in applied areas where words are usually used in very specific senses (commonly not ones in dictionaries!). E.g., *water table* as bit of wood at bottom of door
- One can use clustering techniques
- Or assume hidden classes and attempt to find them using the EM (Expectation Maximization) algorithm (Schütze 1998)

WSD: Senseval competitions

- Senseval 1: September 1998. Results in *Computers and the Humanities* 34(1–2). OUP Hector corpus.
- Senseval 2: first half of 2001. WordNet senses.
- Senseval 3: first half of 2004. WordNet senses.
- Sense-tagged corpora available:
  - http://www.itri.brighton.ac.uk/events/senseval/
- Comparison of various systems, all the usual suspects (naive Bayes, decision lists, decomposable models, memory-based methods), and of foundational issues

WSD Performance

- Varies widely depending on how difficult the disambiguation task is
- Accuracies of over 90% are commonly reported on some of the classic, often fairly easy, word disambiguation tasks (*pike*, *star*, *interest*, . . .)
- Senseval brought careful evaluation of difficult WSD (many senses, different POS)
- Senseval 1: more fine grained senses, wider range of types:
  - Overall: about 75% accuracy
  - Nouns: about 80% accuracy
  - Verbs: about 70% accuracy

What is a word sense?

- Particular ranges of word senses have to be distinguished in many practical tasks, e.g.:
  - translation
  - IR
- But there generally isn't *one* way to divide the uses of a word into a set of non-overlapping categories. Dictionaries provide neither consisentency nor non-overlapping categories usually.
- Senses depend on the task (Kilgarriff 1997)

Similar ‘disambiguation’ problems

- Sentence boundary detection
- *I live on Palm Dr. Smith lives downtown.*
- Only really ambiguous when:
  - word before the period is an abbreviation (which can end a sentence – not something like a title)
  - word after the period is capitalized (and can be a proper name – otherwise it must be a sentence end)
- Can be treated as ‘disambiguating’ periods (as abbreviation mark, end of sentence, or both simultaneously [haplology])

Similar ‘disambiguation’ problems

- Context-sensitive spelling correction:
- *I know their is a problem with there account.*

Text categorization

- Have some predefined categories for texts
  - Predefined categories for news items on newswires
    – Reuters categories
  - Yahoo! classes (extra complexity: hierarchical)
  - Spam vs. not spam
- Word sense disambiguation can actually be thought of as text (here, context) categorization
  - But many more opportunities to use detailed (semi-) linguistic features

The right features are more important than snazzy models, methods, and objective functions

- Within StatNLP, if a model lets you make use of more linguistic information (i.e., it has better features), then you're likely to do better, even if the model is theoretically rancid
- Example:nnnn
  - Senseval 2: Features for word sense disambiguation

Disambiguating using ‘language’ models

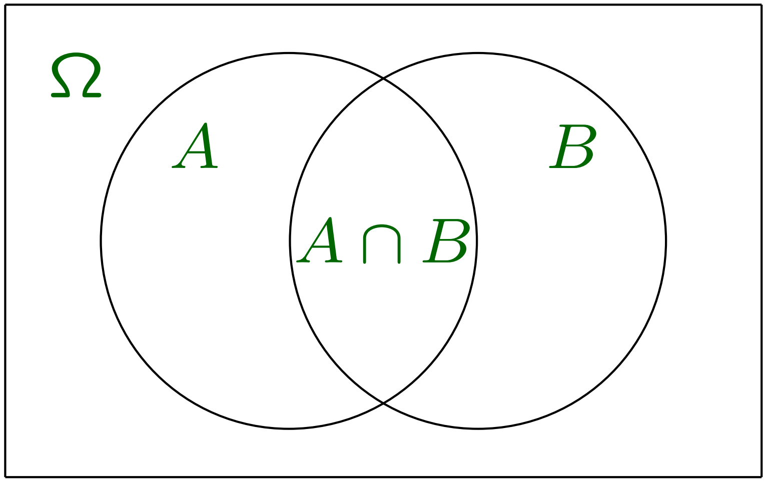
- Supervised training from hand-labeled examples
- Train n-gram language model for examples of each sense, treating examples as a ‘language’
  - estimate $P(\text{frog}|\text{large}, \text{green})$, etc.
  - reduce parameters by backing off where there is insufficient data: use $P(\text{frog}|\text{green})$ or $P(\text{frog})$
- Disambiguate based on in which ‘language’ the sentence would have highest probability
- Multinomial Naive Bayes models are class-conditional unigram language models
- Higher oder models can give some of the advantages of wide context bag of words models (Naive Bayes-like) and use of local structural cues around word

E.g., six senses of *interest* (Aist et al. 1996)

- Senses:
  - Readiness to give attention
  - Quality of causing attention to be given
  - Activity, subject, etc., which one gives time and attention to
  - Advantage, advancement or favor
  - A share (in a company, business, etc.)
  - Money paid for the use of money
- Performance of 91% correct on independent test data
- (This compares well with other methods, e.g., Bruce and Wiebe report 78% using decomposable models on the same data.)

Bayesian classification in WSD

- Vectors $\vec{x}$ are some representation of the context of use of a word
- Problem: We usually don't know $P(c_k|\vec{x})$
- Partial solution: we break it up by Bayes' rule



$$P(B|A) = P(A \cap B)/P(A)$$

$$P(A \cap B) = P(B)P(A|B)$$

Bayes’ theorem

$$P(B|A) = \frac{P(A,B)}{P(A)} = \frac{P(A|B)P(B)}{P(A)}$$

If $A \subseteq \cup_i B_i$ and the B_i are disjoint, then (sum rule):

$$P(A) = \sum_i P(A,B_i) = \sum_i P(A|B_i)P(B_i)$$

Bayes’ theorem: If $A \subseteq \cup_{i=1}^n B_i$, $P(A) > 0$, and $B_i \cap B_j = \emptyset$ for $i \neq j$:

Likelihood Prior

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)} = \frac{P(A|B_j) \times P(B_j)}{\Sigma_{i=1}^n P(A|B_i)P(B_i)}$$

Posterior

Normalizing term

Naive Bayes WSD

- $\vec{x}$ is our context (something like a 100 word window)
- c_k is a sense of the word

$$\begin{aligned} c' &= \operatorname*{arg\,max}_{c_k} P(c_k|\vec{x}) \\ &= \operatorname*{arg\,max}_{c_k} \frac{P(\vec{x}|c_k)}{P(\vec{x})} P(c_k) \\ &= \operatorname*{arg\,max}_{c_k} P(\vec{x}|c_k) P(c_k) \\ &= \operatorname*{arg\,max}_{c_k} [\log P(\vec{x}|c_k) + \log P(c_k)] \end{aligned}$$

(Multinomial) Naive Bayes classifiers for WSD

- $\vec{x}$ is the context (something like a 100 word window)
- c_k is a sense of the word to be disambiguated

$$\begin{aligned}
\text{Choose } c' &= \operatorname*{arg\,max}_{c_k} P(c_k|\vec{x}) \\
&= \operatorname*{arg\,max}_{c_k} \frac{P(\vec{x}|c_k)}{P(\vec{x})} P(c_k) \\
&= \operatorname*{arg\,max}_{c_k} [\log P(\vec{x}|c_k) + \log P(c_k)] \\
&= \operatorname*{arg\,max}_{c_k} \left[\sum_{v_j \text{ in } \vec{x}} \log P(v_j|c_k) + \log P(c_k) \right]
\end{aligned}$$

- An effective method in practice, but also an example of a structure-blind ‘bag of words’ model

WSD methods

- One method: A multinomial naive Bayes classifier, add $\frac{1}{10}$ smoothing. Except words near the ambiguous word are weighted by a strongly peaked function (distance 3–5, $3\times$; distance 2, $5\times$, distance 1, $15\times$)
- Other methods (Senseval 2 entries):
  - Bagged decision trees with unigram, bigram, and long distance bigram features
  - Weighted vote of DT, NB, and kNN classifiers over short and long distance bigram features
  - Hierarchical LazyBoosting over large and small window bag-of-word features, and WordNet features
  - Support vector machine with IDF feature weighting

Senseval 2 results

- The hacked Naive Bayes classifier has no particular theoretical justification. One really cannot make sense of it in terms of the independence assumptions, etc., usually invoked for a Naive Bayes model
- But it is linguistically roughly right: nearby context is often very important for WSD: noun collocations (*complete* ***accident***), verbs (***derive*** *satisfaction*)
- In Senseval 2, it scores an average accuracy of 61.2%
- This model was just a component of a system we entered, but alone it would have come in 6th place out of 27 systems (on English lexical sample data), beating out all the systems on the previous slide

Word Sense Disambiguation

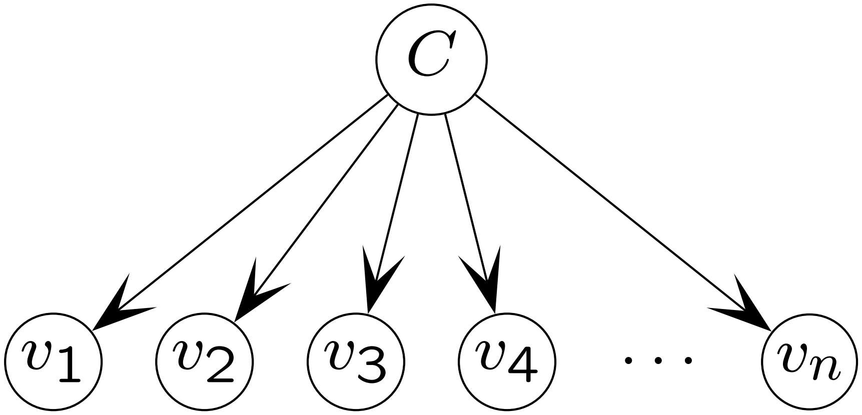
extra or variant slides

Word sense disambiguation

- The problem of assigning the correct sense to a use of a word in context
- *bank:*
  - the rising ground bordering a lake, river, or sea
  - an establishment for the custody, loan exchange, or issue of money
- *Traders said central* ***banks*** *will be waiting in the wings.*
- A straightforward classification problem

(Multinomial) Naive Bayes model

- v_j is word j of the document



- One chooses a category, and then generate words (here a unigram estimate)
- The CPT for each v_i multinomial is identical (tied parameters), and estimated over the whole vocabulary
- This is the simplest Hidden Markov Model

Lexical Acquisition

FSNLP, chapter 8

Christopher Manning and
Hinrich Schütze

Lexical acquisition

- Language acquisition: acquiring the properties of words
- Practical: filling holes in dictionaries
  - Language is productive
  - Lots of stuff isn't in dictionaries anyway
- Claim: most knowledge of language is encoded in words.

Evaluation: The 2$\times$2 contingency matrix

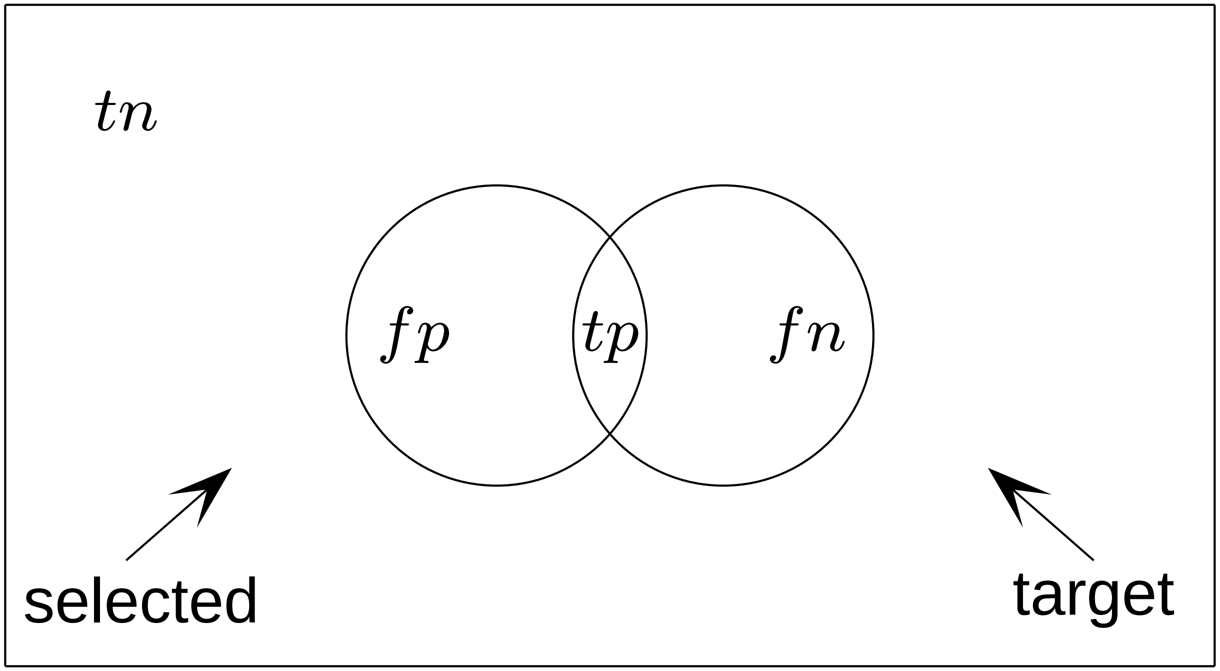
	Actual	
System	target	¬ target
selected	tp	fp
¬selected	fn	tn

Accuracy = $(tp + tn)/N$, where $N = tp + tn + fp + fn$

Error = $(fn + fp)/N$ = 1 $-$ Accuracy

Why is this a bad measure for many NLP tasks?

A diagram motivating the measures of precision and recall.



Accuracy is not a useful measure when the target set is a tiny fraction of the total set.

Precision and recall

- Precision is defined as a measure of the proportion of selected items that the system got right:

$$\text{precision} = \frac{tp}{tp + fp}$$

- Recall is defined as the proportion of the target items that the system selected:

$$\text{recall} = \frac{tp}{tp + fn}$$

These two measures allow us to distinguish between excluding target items and returning irrelevant items.

They still require human-made “gold standard” judgements.

A combined measure

Does one just average them? Bad, because the measures aren’t independent.

What’s a sensible model?

Rijsbergen (1979:174) defines and justifies the usually used alternative.

(see http://www.dcs.gla.ac.uk/Keith/Preface.html)

Assumptions:

- Interested in document proportions not absolute numbers
- Decreasing marginal effectiveness of recall and precision, e.g.:

  $$(R + 1, P - 1) > (R, P)$$

  but

  $$(R + 1, P) > (R + 2, P - 1)$$

  Makes curves convex towards origin.

The F measure

Weighted harmonic mean: The F measure (where $F = 1 - E$):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

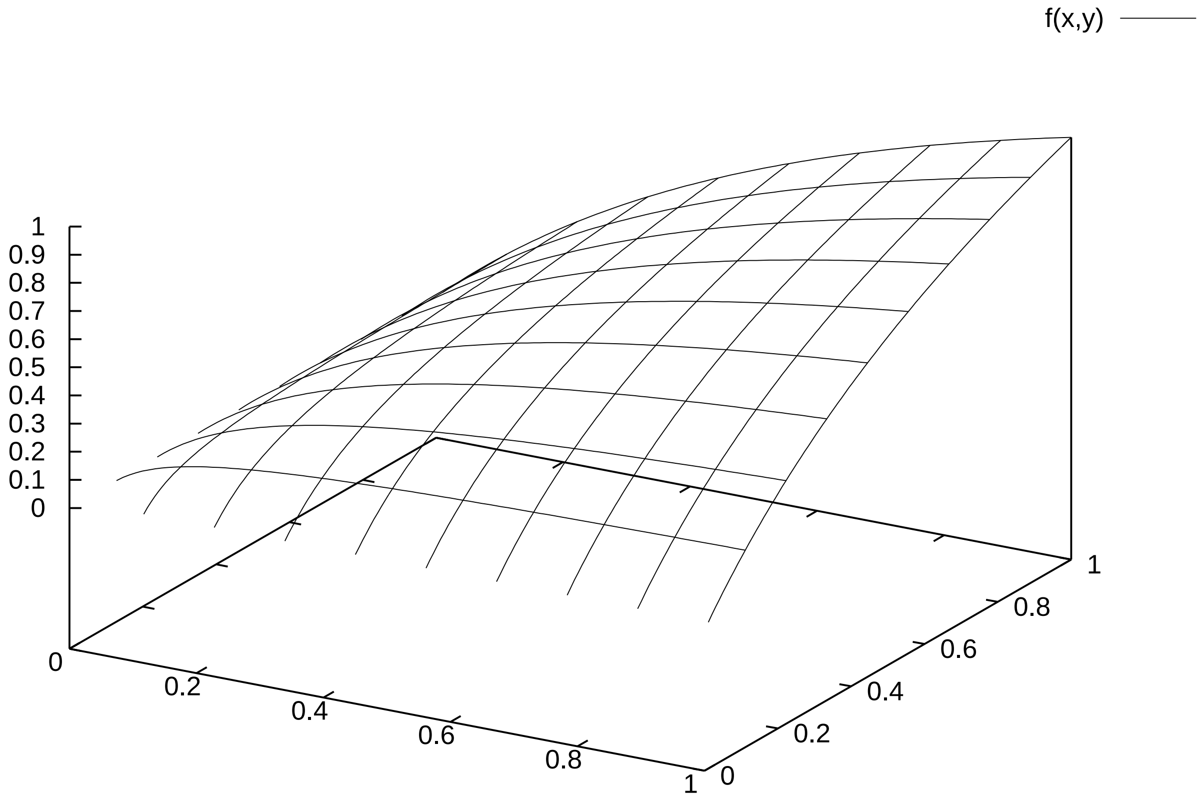
where P is precision, R is recall and α weights precision and recall. (Or in terms of β, where $\alpha = 1/(\beta^2 + 1)$.)

A value of $\alpha = 0.5$ is often chosen.

$$F = \frac{2PR}{R + P}$$

At break-even point, when $R = P$, then $F = R = P$

The F measure ($\alpha = 0.5$)



Ways of averaging

Precision	Recall	Arithmetic	Geometric	Harmonic	Minimum
80	10	45	28.3	17.8	10
80	20	50	40.0	32.0	20
80	30	55	49.0	43.6	30
80	40	60	56.6	53.3	40
80	50	65	63.2	61.5	50
80	60	70	69.3	68.6	60
80	70	75	74.8	74.7	70
80	80	80	80.0	80.0	80
80	90	85	84.9	84.7	80
80	100	90	89.4	88.9	80

Subcategorization frames

Here are some subcategorization frames that are common in English.

- **Intransitive verb.** NP[subject]. *The woman walked*.
- **Transitive verb.** NP[subject], NP[object]. *John loves Mary*.
- **Ditransitive verb.** NP[subject], NP[direct object], NP[indirect object]. *Mary gave Peter flowers*.
- **Intransitive with PP.** NP[subject], PP. *I rent in Paddington*.
- **Transitive with PP.** NP[subject], NP[object], PP. *She put the book on the table*.

- **Sentential complement.** NP[subject], clause. *I know (that) she likes you*.
- **Transitive with sentential complement.** NP[subj], NP[obj], clause. *She told me that Gary is coming on Tuesday*.

(5) a. She told the man where Peter grew up.

b. She found the place where Peter grew up.

(6) a. She told [the man] [where Peter grew up].

b. She found [the place [where Peter grew up]].

(Info in learner’s dictionaries.)

Brent (1993):

- Cues for frames.

  e.g., pronoun or capitalized followed by punctuation
- Hypothesis testing

$$\begin{aligned} p_E &= P(v^i(f^j) = 0 | C(v^i, c^j) \geq m) \\ &= \sum_{r=m}^{n} \binom{n}{r} {\epsilon_j}^r (1 - \epsilon_j)^{n-r} \end{aligned}$$

verb v^i occurs n times; there are $m \leq n$ occurrences with a cue for frame f^j $C(v^i, c^j)$ is the number of times that v^i occurs with cue c^j, and ϵ_j is the error rate for cue f^j,

Manning (1993)

Uses tagger. More errorful, but much more abundant cues.

- *He relies* ***on*** *relatives.*
- *She compared the results* ***with*** *earlier findings.*

Learned subcategorization frames

Verb	Correct	Incorrect	OALD
bridge	1	1	1
burden	2		2
depict	2		3
emanate	1		1
leak	1		5
occupy	1		3
remark	1	1	4
retire	2	1	5
shed	1		2
troop	0		3

Two of the errors are prepositional phrases (PPs): *to bridge between* and *to retire in*.

One could argue that *retire* subcategorizes for the PP *in Malibu* in a sentence like *John retires in Malibu* since the verb and the PP-complement enter into a closer relationship than mere adverbial modification.

The third error in the table is the incorrect assignment of the intransitive frame to *remark*. This is probably due to sentences like (7) which look like *remark* is used without any arguments (except the subject).

(7) “And here we are 10 years later with the same problems,” Mr. Smith remarked.

Attachment ambiguities

- *I saw the man with a telescope*
- What does *with a telescope* modify?
- Is the problem ‘AI-complete’? Yes, but . . .
- Proposed simple structural factors
  - Right association (Kimball 1973) = ‘low’ or ‘near’ attachment = ‘early closure’ (of NP)
  - Minimal attachment (Frazier 1978) [depends on grammar] = ‘high’ or ‘distant’ attachment = ‘late closure’ (of NP)

Attachment ambiguities (2)

- Such simple structural factors dominated in early psycholinguistics, and are still widely invoked.
- In the V NP PP context, right attachment gets right 55–67% of cases.
- But that means it gets wrong 33–45% of cases

Attachment ambiguities (3)

- The children ate the cake with a spoon.
- The children ate the cake with frosting.
- Moscow sent more than 100,000 soldiers into Afghanistan . . .
- Sydney Water breached an agreement with NSW Health . . .

Words are good predictors (even absent understanding).

Importance of lexical factors

Ford, Bresnan and Kaplan (1982) [as part of the promotion of ‘lexicalist’ linguistic theories]

- Order of grammatical rule processing (by human) determines closure effects
- Ordering is jointly determined by strengths of alternative lexical forms, strengths of alternative syntactic rewrite rules, and the sequence of hypotheses in the parsing process

Importance of lexical factors (2)

Ford, Bresnan and Kaplan (1982):

- *Joe included the package for Susan.*
- *Joe carried the package for Susan.*

“It is quite evident, then, that the closure effects in these sentences are induced in some way by the choice of the lexical items.” (Psycholinguistic studies show that this is true *independent* of discourse context.)

Simple model

(Log) Likelihood Ratio [a common and good way of comparing between two exclusive alternatives]

$$\lambda(v, n, p) = \log \frac{P(p|v)}{P(p|n)}$$

Problem: ignores preference for attaching “low”.

Problematic example (*NYT*)

- *Chrysler confirmed that it would end its troubled venture with Maserati.*

-

w	$C(w)$	$C(w, \textit{with})$
end	5156	607
venture	1442	155

- Get wrong answer:

$$P(p|v) = \frac{607}{5156} \approx 0.118$$
$$> P(p|n) = \frac{155}{1442} \approx 0.107$$

Hindle and Rooth (1993 [1991])

- Event space: all V NP PP* sequences, but PP must modify V or first N
- Don't directly decide whether PP modifies V or N
- Rather look at binary RVs:
  - VA_p: Is there a PP headed by p which attaches to v
  - NA_p: Is there a PP headed by p which attaches to n
- Both can be 1:

He put the book on World War II on the table

Independence assumptions:

$$\begin{aligned} P(\text{VA}_p, \text{NA}_p | v, n) &= P(\text{VA}_p | v, n) P(\text{NA}_p | v, n) \\ &= P(\text{VA}_p | v) P(\text{NA}_p | n) \end{aligned}$$

Decision space: first PP after NP. [NB!]

$$\begin{aligned} P(\text{Attach}(p) = n | v, n) &= P(\text{VA}_p = 0 \vee \text{VA}_p = 1 | v) \\ &\quad \times P(\text{NA}_p = 1 | n) \\ &= 1.0 \times P(\text{NA}_p = 1 | n) \\ &= P(\text{NA}_p = 1 | n) \end{aligned}$$

It doesn't matter what VA$_p$ is! If both are true, the first PP after the NP *must* modify the noun (in phrase structure trees, lines don't cross).

But conversely, in order for the first PP headed by the preposition p to attach to the verb, both $\text{VA}_p = 1$ and $\text{NA}_p = 0$ must hold:

$$\begin{aligned} P(\text{Attach}(p) = v|v, n) &= P(\text{VA}_p = 1, \text{NA}_p = 0|v, n) \\ &= P(\text{VA}_p = 1|v)P(\text{NA}_p = 0|n) \end{aligned}$$

We assess which is more likely by a (log) likelihood ratio:

$$\begin{aligned} \lambda(v, n, p) &= \log_2 \frac{P(\text{Attach}(p) = v|v, n)}{P(\text{Attach}(p) = n|v, n)} \\ \\ &= \log_2 \frac{P(\text{VA}_p = 1|v)P(\text{NA}_p = 0|v)}{P(\text{NA}_p = 1|n)} \end{aligned}$$

If large positive, decide verb attachment; if large negative, decide noun attachment.

Building the model

How do we learn probabilities? From (smoothed) MLEs:

$$P(\text{VA}_p = 1|v) = \frac{C(v,p)}{C(v)}$$

$$P(\text{NA}_p = 1|n) = \frac{C(n,p)}{C(n)}$$

How do we get estimates from an unlabelled corpus? Use partial parser, and look for unambiguous cases:

- The road *to London* is long and winding.

- She sent him *into the nursery* to gather up his toys.

Hindle and Rooth heuristically determining $C(v, p)$, $C(n, p)$, and $C(n, \emptyset)$ from unlabeled data:

1. Build an initial model by counting all unambiguous cases.
2. Apply initial model to all ambiguous cases and assign them to the appropriate count if λ exceeds a threshold ($2/-2$).
3. Divide the remaining ambiguous cases evenly between the counts (increase both $C(v, p)$ and $C(n, p)$ by 0.5 for each).

Example

Moscow sent more than 100,000 soldiers into Afghanistan

...

$$
\begin{aligned}
P(\text{VA}_{\textit{into}} = 1|\textit{send}) &= \frac{C(\textit{send}, \textit{into})}{C(\textit{send})} \\
&= \frac{86}{1742.5} \approx 0.049 \\
P(\text{NA}_{\textit{into}} = 1|\textit{soldiers}) &= \frac{C(\textit{soldiers}, \textit{into})}{C(\textit{soldiers})} \\
&= \frac{1}{1478} \approx 0.0007 \\
P(\text{NA}_{\textit{into}} = 0|\textit{soldiers}) &= 1 - P(\text{NA}_{\textit{into}} = 1|\textit{soldiers}) \\
&\approx 0.9993
\end{aligned}
$$

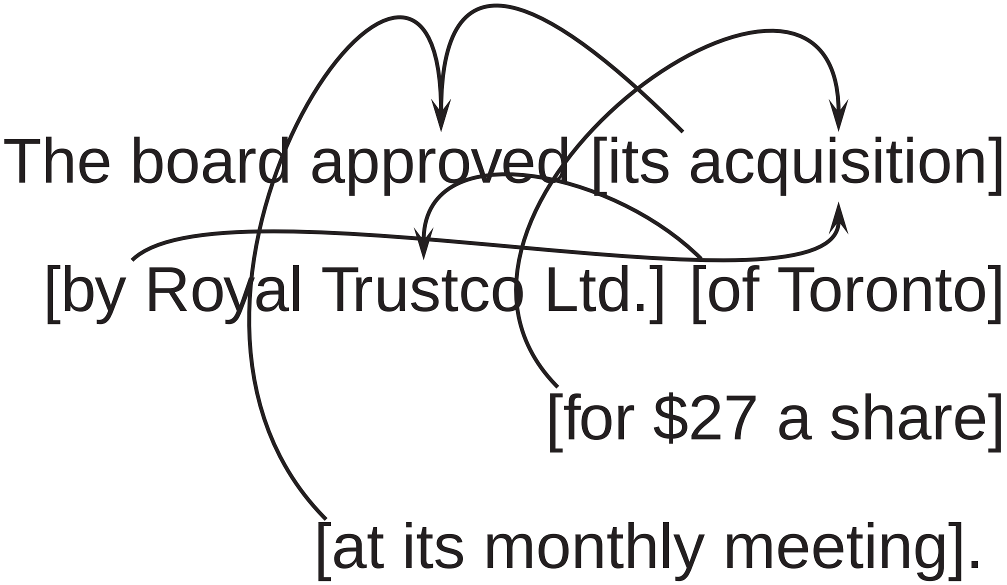
$$\lambda(\textit{send}, \textit{soldiers}, \textit{into}) \approx \log_2 \frac{0.049 \times 0.9993}{0.0007} \approx 6.13$$

Attachment to verb is about 70 times more likely.

Overall accuracy is about 80% (forced choice); 91.7% correct at 55.2% recall ($\lambda = 3.0$).

Final remarks

- Ignores other conditioning factors (noun head in PP, superlative adjective)
- Just doing the simplest V NP PP case
- Gibson and Pearlmutter (1994) argue that overuse of this simple case has greatly biased psycholinguistic studies



The board approved [its acquisition]

[by Royal Trustco Ltd.] [of Toronto]

[for $27 a share]

[at its monthly meeting].

Final remarks (2)

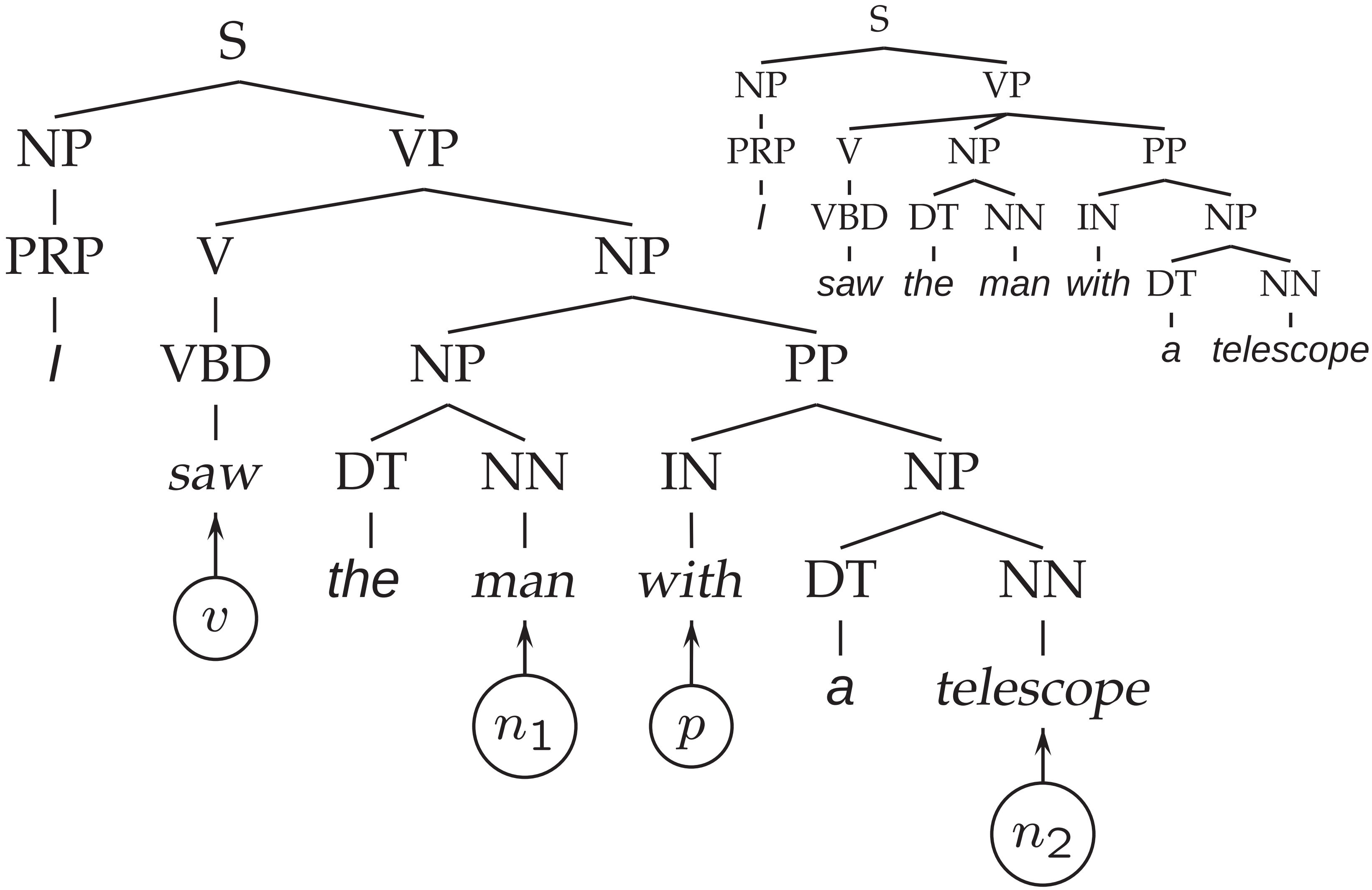
- There are other attachment cases: coordination, adverbial and participial phrases, noun compounds. Data sparseness is a bigger problem with many of these (more open class heads needed).
- In general, indeterminacy is quite common:

  > *We have not signed a settlement agreement with them.*

  Either reading seems equally plausible.

Attachment ambiguities: The key parsing decision

- The main problem in parsing is working out how to ‘attach’ various kinds of constituents – PPs, adverbial or participial phrases, coordinations, and so on
- Prepositional phrase attachment
  - *I saw the man with a telescope*
- What does *with a telescope* modify?
  - The verb *saw*?
  - The noun *man*?
- Is the problem ‘AI-complete’? Yes, but . . .


S
NP
VP
PRP
V
NP
I
VBD
NP
PP
saw
DT
NN
IN
NP
the
man
with
DT
NN
a
telescope
v
n_1
p
n_2
S
NP
VP
PRP
V
NP
PP
I
VBD
DT
NN
IN
NP
saw
the
man
with
DT
NN
a
telescope

Attachment ambiguities (2)

- Proposed simple structural factors
  - Right association (Kimball 1973) = 'low' or 'near' attachment = 'late closure' (of NP) [NP $\longrightarrow$ NP PP]
  - Minimal attachment (Frazier 1978) [depends on grammar] = 'high' or 'distant' attachment = 'early closure' (of NP) [VP $\longrightarrow$ V NP PP]
- Such simple structural factors dominated in early psycholinguistics, and are still widely invoked.
- In the V NP PP context, right attachment gets it right in 55–67% of cases.
- But that means it gets it wrong in 33–45% of cases

Importance of lexical factors

- Words are good predictors (or even inducers) of attachment (even absent understanding):
  - The children ate the cake with a spoon.
  - The children ate the cake with frosting.
  - Moscow sent more than 100,000 soldiers into Afghanistan
  - Sydney Water breached an agreement with NSW Health
- Ford et al. (1982):
  - Ordering is jointly determined by strengths of alternative lexical forms, alternative syntactic rewrite rules, and the sequence of hypotheses in parsing

Simple model

(Log) Likelihood Ratio [a common and good way of comparing between two exclusive alternatives]

$$\lambda(v, n, p) = \log \frac{P(p|v)}{P(p|n)}$$

Problem: ignores preference for attaching “low”

Problematic example (*NYT*)

- *Chrysler confirmed that it would end its troubled venture with Maserati.*

w	$C(w)$	$C(w,\textit{with})$
end	5156	607
venture	1442	155

- Get wrong answer:

$$\begin{aligned} P(p|v) &= \frac{607}{5156} \approx 0.118 \\ > \; P(p|n) &= \frac{155}{1442} \approx 0.107 \end{aligned}$$

Hindle and Rooth (1993) [1991]

- Event space: all V NP PP* sequences, but PP must modify V or first N
- Don't directly decide whether PP modifies V or N
- Rather look at binary RVs:
  - VA_p: Is there a PP headed by p which attaches to v
  - NA_p: Is there a PP headed by p which attaches to n
- Both can be 1:

  He put the book on World War II on the table

Hindle and Rooth (1993)

Independence assumptions:

$$\begin{aligned} P(\text{VA}_p, \text{NA}_p|v,n) &= P(\text{VA}_p|v,n)P(\text{NA}_p|v,n) \\ &= P(\text{VA}_p|v)P(\text{NA}_p|n) \end{aligned}$$

Decision space: first PP after NP. [NB!]

$$\begin{aligned} P(\text{Attach}(p) = n|v,n) &= P(\text{VA}_p = 0 \vee \text{VA}_p = 1|v) \\ &\quad \times P(\text{NA}_p = 1|n) \\ &= 1.0 \times P(\text{NA}_p = 1|n) \\ &= P(\text{NA}_p = 1|n) \end{aligned}$$

Hindle and Rooth (1993)

In the above case, it doesn't matter what VA$_p$ is! If both are true, the first PP after the NP *must* modify the noun (in phrase structure trees, lines don't cross).

But conversely, in order for the first PP headed by the preposition p to attach to the verb, both VA$_p = 1$ and NA$_p = 0$ must hold:

$$\begin{aligned} P(\text{Attach}(p) = v | v, n) &= P(\text{VA}_p = 1, \text{NA}_p = 0 | v, n) \\ &= P(\text{VA}_p = 1 | v) P(\text{NA}_p = 0 | n) \end{aligned}$$

We assess which is more likely by a (log) likelihood ratio:

$$\begin{aligned}\lambda(v,n,p) &= \log_2 \frac{P(\text{Attach}(p) = v|v,n)}{P(\text{Attach}(p) = n|v,n)} \\ &= \log_2 \frac{P(\text{VA}_p = 1|v)P(\text{NA}_p = 0|v)}{P(\text{NA}_p = 1|n)}\end{aligned}$$

If large positive, decide verb attachment; if large negative, decide noun attachment.

Overall accuracy is about 80% (forced choice); 91.7% correct at 55.2% recall ($\lambda = 3.0$).

Building the model

- We learn probabilities from (smoothed) MLEs:

$$
\begin{aligned}
P(\text{VA}_p = 1|v) &= \frac{C(v,p)}{C(v)} \\
P(\text{NA}_p = 1|n) &= \frac{C(n,p)}{C(n)}
\end{aligned}
$$

- Estimates are gotten from an unlabelled corpus by use of a tagger – Church's – and a partial parser – Hindle's Fidditch – by looking for unambiguous cases.

Unambiguous cases

- If a pronoun, assume V attach:
  - She sent him *into the nursery* to gather up his toys.
- If no verb, then N attach:
  - The road *to London* is long and winding.
- If passive then V attach
- etc.

Hindle and Rooth (1993) algorithm

Hindle and Rooth heuristically determining $C(v, p)$, $C(n, p)$, and $C(n, \emptyset)$ from unlabeled data:

1. Build an initial model by counting all unambiguous cases.
2. Apply initial model to all ambiguous cases and assign them to the appropriate count if λ exceeds a threshold ($2 / - 2$).
3. Divide the remaining ambiguous cases evenly between the counts (increase both $C(v, p)$ and $C(n, p)$ by 0.5 for each).

Example

Moscow sent more than 100,000 soldiers into Afghanistan

$$P(\text{VA}_{\textit{into}} = 1|\textit{send}) = \frac{C(\textit{send}, \textit{into})}{C(\textit{send})} = \frac{86}{1742.5} \approx 0.049$$

$$P(\text{NA}_{\textit{into}} = 1|\textit{soldiers}) = \frac{C(\textit{soldiers}, \textit{into})}{C(\textit{soldiers})} = \frac{1}{1478} \approx 0.0007$$

$$P(\text{NA}_{\textit{into}} = 0|\textit{soldiers}) = 1 - P(\text{NA}_{\textit{into}} = 1|\textit{soldiers}) \approx 0.9993$$

$$\lambda(\textit{send}, \textit{soldiers}, \textit{into}) \approx \log_2 \frac{0.049 \times 0.9993}{0.0007} \approx 6.13$$

Attachment to verb is about 70 times more likely.

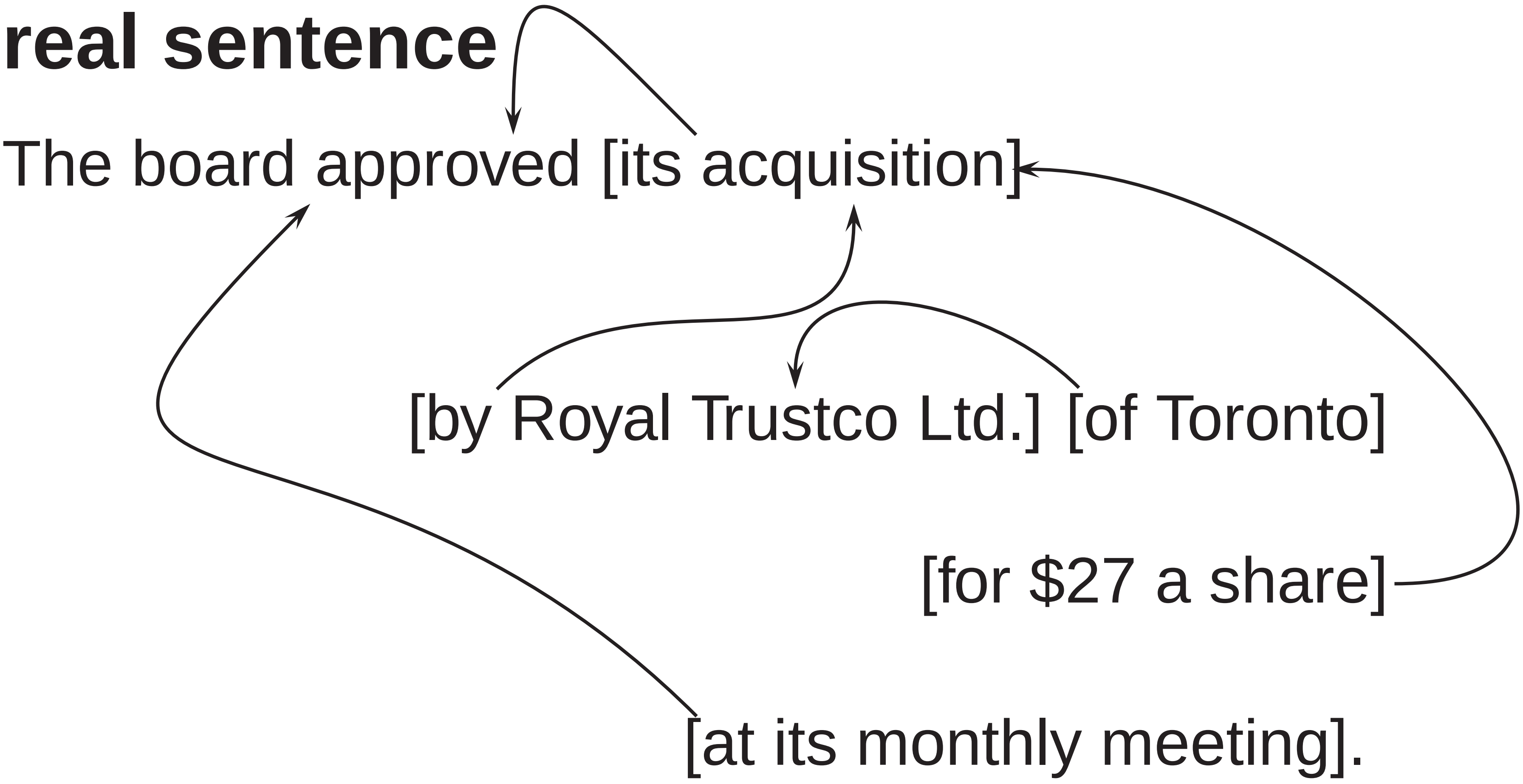
Collins and Brooks (1995)

- Use fully parsed treebank data
- Includes n_2 in model: quadruples $\langle v, n_1, p, n_2 \rangle$
- Uses a back-off model to address data sparseness issues.
- About 95% of test quadruples not seen in training data
- Start with full quadruples, backoff to triples and pairs restricted so as to always contain p
- Accuracy: 86.5%

More complex situations

- Hindle and Rooth (1993) – and Collins and Brooks (1995) – are just doing the simplest V NP PP case
- Gibson and Pearlmutter (1994) argue that overuse of this simple case has greatly biased psycholinguistic studies
- V NP PP PP has 5 possible attachments; V NP PP PP PP has 14
- The problem is much harder, and a dumb algorithm will perform poorly

Example of complex attachments in a real sentence



The board approved [its acquisition]

[by Royal Trustco Ltd.] [of Toronto]

[for $27 a share]

[at its monthly meeting].

Merlo et al. (1997)

- Merlo et al. (1997) attempt some of these more difficult cases
- Basic analysis is a backoff model like Collins and Brooks (1995)
- Results:
  - 1 PP: 84% (61% baseline – most frequent)
  - 2 PPs: 70% (30% baseline)
  - 3 PPs: 44% (19% baseline)

Ratnaparkhi (1998)

- Tagger, simple chunker over unannotated newswire
- Heuristic extraction of exemplars:
  - $\langle v, p, n_2 \rangle$ if p not “of”, v nearest verb and within k of p, no noun between p and n_2, $\dots$
  - $\langle n_1, p, n_2 \rangle$ if n is nearest noun and within k of p, no verb between p and n_2, $\dots$
- Noisy data: c. 69% correct
- Accuracy of system on attachments: about 82%
- Easy to port to other languages (did Spanish)

Final remarks

- Many other similar sorts of attachment decisions: coordination, adverbial and participial phrases, noun compounds
  - I drink [coffee with [cream and sugar]]
  - I drink [[coffee with cream] and sugar]
- Data sparseness is a bigger problem with many of these (more open class heads needed).
- In general, indeterminacy is quite common:
  - *We have not signed a settlement agreement with them.*

Lexical acquisition

- Simple statistical models give same estimate to all unseen events
- Unrealistic – could hope to refine that based on semantic classes of words
- E.g, although never seen, *eating pineapple* should be more likely than *eating holograms* because *pineapple* is similar to *apples*, and we have seen *eating apples*
- It's the same data. Why are classes useful?

An application: selectional preferences

- Verbs take arguments of certain types (usually! – remember metaphor)
- *Bill drove a . . .*
- *Mustang, car, truck, jeep, . . .*
- Resnik (1993) uses KL divergence for verb objects distributions
- Selectional preference strength: how strongly does a verb constrain direct objects
- *see* vs. *unknotted*
- Model via using head words only – a usually correct assumption
- Use a class-based model of nouns – for generalization. Resnik uses WordNet.

Selectional preference strength (how strongly does verb select?)

$$S(v) = D(P(C|v) \| P(C)) = \sum_c P(c|v) \log \frac{P(c|v)}{P(c)}$$

Selectional association between verb and class:

$$A(v,c) = \frac{P(c|v) \log \frac{P(c|v)}{P(c)}}{S(v)}$$

Proportion that its summand contributes to preference strength.

For nouns in multiple classes – disambiguate as most likely sense:

$$A(v,n) = \max_{c \in \text{classes}(n)} A(v,c)$$

SPS example (made-up data)

Noun class c	$P(c)$	$P(c \mid \textit{eat})$	$P(c \mid \textit{see})$	$P(c \mid \textit{find})$
people	0.25	0.01	0.25	0.33
furniture	0.25	0.01	0.25	0.33
food	0.25	0.97	0.25	0.33
action	0.25	0.01	0.25	0.01
SPS $S(v)$		1.76	0.00	0.35

$$
\begin{aligned}
A(\textit{eat}, \text{food}) &= 1.08 \\
A(\textit{find}, \text{action}) &= -0.13
\end{aligned}
$$

SPS example (Resnik, Brown corpus)

Verb v	Noun n	$A(v,n)$	Class	Noun n	$A(v,n)$	Class
answer	*request*	4.49	speech act	*tragedy*	3.88	communication
find	*label*	1.10	abstraction	*fever*	0.22	psych. feature
hear	*story*	1.89	communication	*issue*	1.89	communication
remember	*reply*	1.31	statement	*smoke*	0.20	article of commerce
repeat	*comment*	1.23	communication	*journal*	1.23	communication
read	*article*	6.80	writing	*fashion*	-0.20	activity
see	*friend*	5.79	entity	*method*	-0.01	method
write	*letter*	7.26	writing	*market*	0.00	commerce

But how might we measure word similarity for word classes?

Vector spaces

A document-by-word matrix A.

	cosmonaut	astronaut	moon	car	truck
d_1	1	0	1	1	0
d_2	0	1	1	0	0
d_3	1	0	0	0	0
d_4	0	0	0	1	1
d_5	0	0	0	1	0
d_6	0	0	0	0	1

A word-by-word matrix B

	cosmonaut	astronaut	moon	car	truck
cosmonaut	2	0	1	1	0
astronaut	0	1	1	0	0
moon	1	1	2	1	0
car	1	0	1	3	1
truck	0	0	0	1	2

A modifier-by-head matrix C

	cosmonaut	astronaut	moon	car	truck
Soviet	1	0	0	1	1
American	0	1	0	1	1
spacewalking	1	1	0	0	0
red	0	0	0	1	1
full	0	0	1	0	0
old	0	0	0	1	1

Similarity measures for binary vectors.

Similarity measure	Definition
matching coefficient	$\|X \cap Y\|$
Dice coefficient	$\frac{2\|X\cap Y\|}{\|X\|+\|Y\|}$
Jaccard coefficient	$\frac{\|X\cap Y\|}{\|X\cup Y\|}$
Overlap coefficient	$\frac{\|X\cap Y\|}{\min(\|X\|,\|Y\|)}$
cosine	$\frac{\|X\cap Y\|}{\sqrt{\|X\|\times\|Y\|}}$

Real-valued vector spaces

Vector dot product (how much do they have in common):

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

0 if orthogonal – like matching coefficient, not normalized.

Cosine measure:

$$\cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

maps vectors onto unit circle by dividing through by lengths:

$$|\vec{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Euclidean distance gives same ordering for normalized vectors:

$$|\vec{x}-\vec{y}| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Example: cosine as semantic similarity on *NYT*

Focus word	Nearest neighbors							
garlic	*sauce*	.732	*pepper*	.728	*salt*	.726	*cup*	.726
fallen	*fell*	.932	*decline*	.931	*rise*	.930	*drop*	.929
engineered	*genetically*	.758	*drugs*	.688	*research*	.687	*drug*	.685
Alfred	*named*	.814	*Robert*	.809	*William*	.808	*W*	.808
simple	*something*	.964	*things*	.963	*You*	.963	*always*	.962

Probabilistic measures

(Dis-)similarity measure	Definition
KL divergence	$D(p\|q) = \sum_i p_i \log \frac{p_i}{q_i}$
Skew	$D(p\|\alpha q + (1-\alpha)p)$
Jensen-Shannon (was IRad)	$\frac{1}{2}[D(p\|\frac{p+q}{2}) + D(q\|\frac{p+q}{2})]$
L_1 norm (Manhattan)	$\sum_i \lvert p_i - q_i \rvert$

Generally perform better (Dagan et al. 1999) – perhaps mainly because they lack the L_2 nature of Euclidean distance, which is very sensitive to outliers

Neighbors of *company* (Lee)

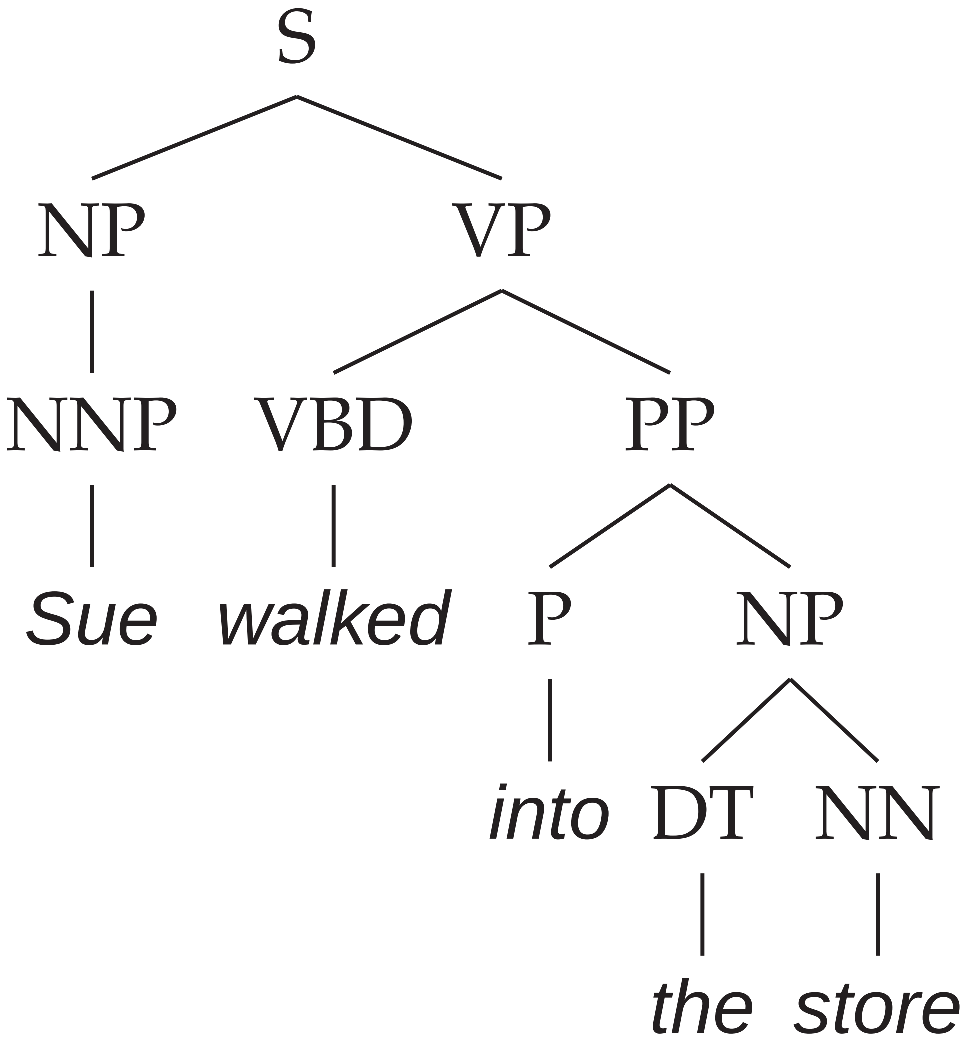
Skew ($\alpha = 0.99$)	J.-S.	Euclidean
airline	business	city
business	airline	airline
bank	firm	industry
agency	bank	program
firm	state	organization
department	agency	bank
manufacturer	group	system
network	govt.	today
industry	city	series
govt.	industry	portion

Evaluation

- Qualitative
- Task-based
  - Language models (Dagan, Pereira, and Lee)
  - Resnik
  - . . .

Final remarks

- In general, indeterminacy is quite common:
  - *We have not signed a settlement agreement with them.*
- Either reading seems equally plausible.
- While later models include n_2, they still ignore other conditioning factors (e.g., superlative adjective in NP_1 favours noun attachment)


S
NP
VP
NNP
VBD
PP
Sue
walked
P
NP
into
DT
NN
the
store

To Do

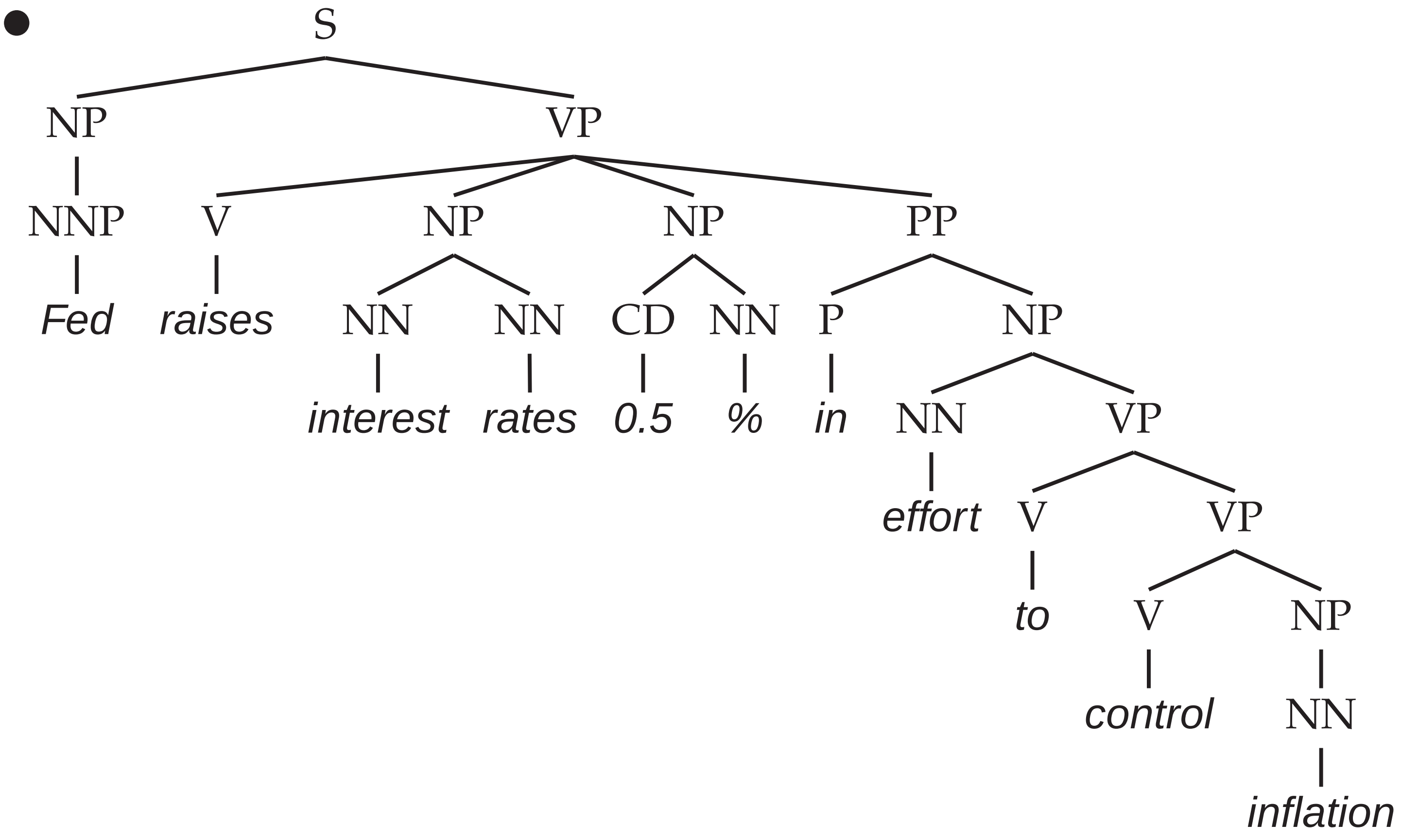
- In the forward probability, it used to be b_{jo_t} but it seems you want i. Need to update the later cases that show the lattice calculations.
- Also, the slide for β probabilities still has a $T+1$ in it, but it seems like you might as well stop at T.
- Discuss in slide state emission vs. arc emission HMMs
- Implement and work through students paying attention or not.

Part of Speech Tagging

FSNLP, chapters 9 and 10

Christopher Manning and
Hinrich Schütze

The problem of POS ambiguity

- Structures for: *Fed raises interest rates 0.5% in effort to control inflation* (*NYT* headline 17 May 2000)
- 

Part-of-speech ambiguities

		VB					
	VBZ	VBP	VBZ				
NNP	NNS	NN	NNS	CD	NN		
Fed	*raises*	*interest*	*rates*	*0.5*	*%*	*in*	*effort*
						to	*control*
							inflation

Part-of-speech examples

NN	noun	baby, toy
VB	verb	see, kiss
JJ	adjective	tall, grateful, alleged
RB	adverb	quickly, frankly, . . .
IN	preposition	in, on, near
DT	determiner	the, a, that
WP	wh-pronoun	who, what, which, . . .
CC	conjunction	and, or

POS ambiguity

- Words often have more than one POS: *back*
  - *The **back** door* = JJ
  - *On my **back*** = NN
  - *Win the voters **back*** = RB
  - *Promised to **back** the bill* = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

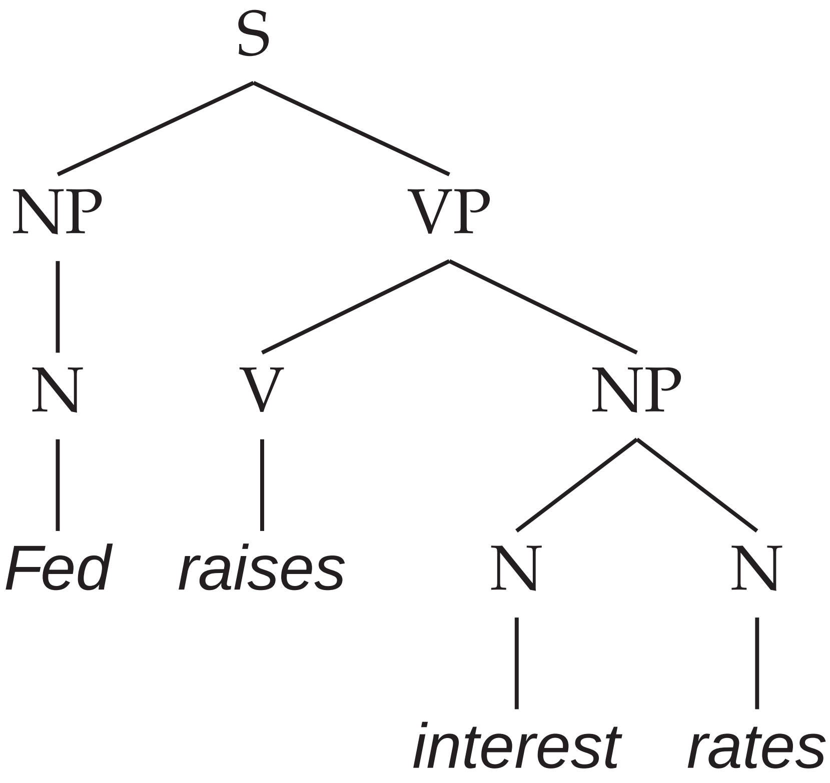
POS Tagging Applications

- Text-to-speech (how do we pronounce “lead”?)
- WSD if the senses are actually different parts of speech. . .
- Information source in all sorts of tasks such as information extraction
- All sorts of simple parsing applications: can write regexps like **Det Adj* N*** over the output (e.g., subcategorization frame learner)
- Preprocessing to speed up a full parser (but a little dangerous)
- If you know the tag, you can back off to it in other tasks, such as parsing (*back-off:* use a simpler representation with denser statistics)

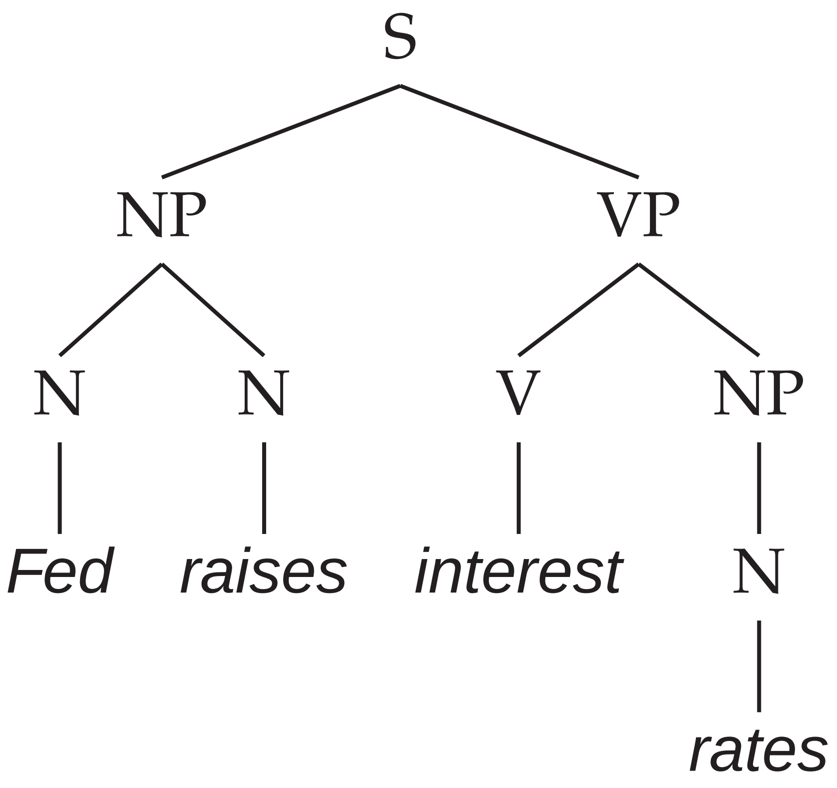
Why should we care?

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical sequence (finite-state model) task
- Can be done well with methods that look at local context
- Though should “really” do it by parsing!
- Fast linear task of considerable value

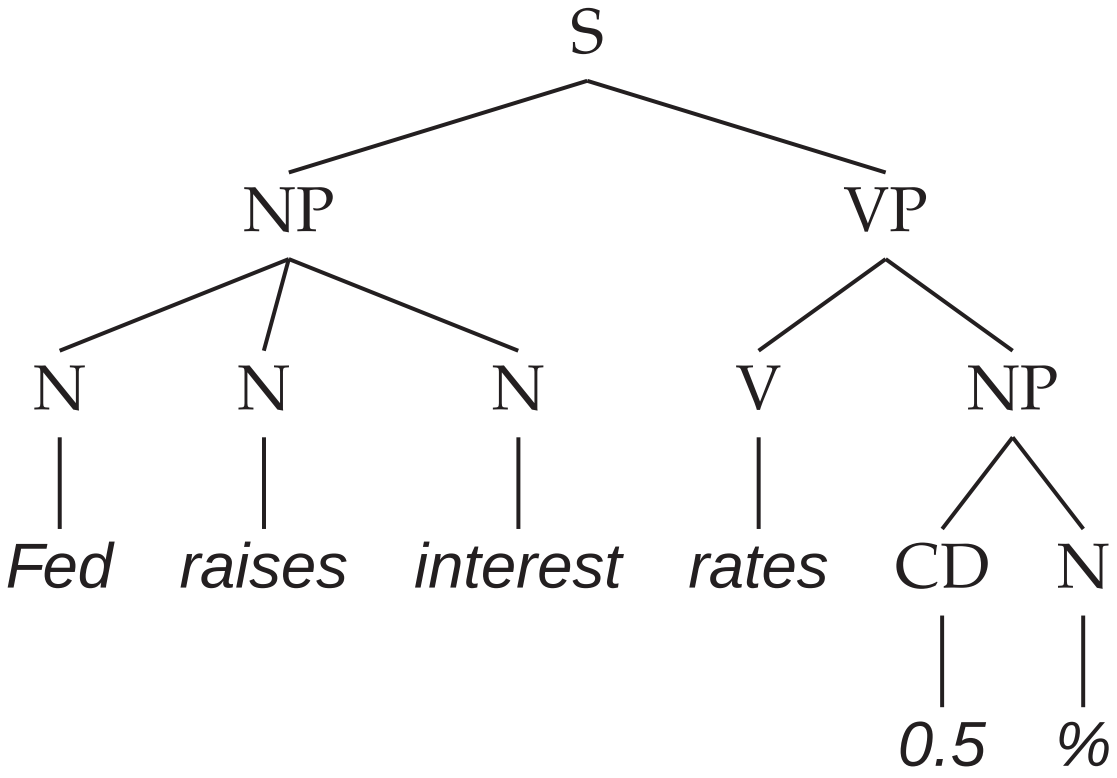
The bad effects of V/N ambiguities (1)



The bad effects of V/N ambiguities (2)



The bad effects of V/N ambiguities (3)



The task of part of speech tagging

- A lightweight (usually linear time) processing task, which can usefully empower other applications:
  - Knowing how to pronounce a word: *récord* [noun] vs. *recórd* [verb]; *lead* as noun vs. verb
  - Matching small phrasal chunks or particular word class patterns for tasks such as information retrieval, information extraction or terminology acquisition (collocation extraction). E.g., just matching nouns, compound nouns, and adjective noun patterns:
    - {A|N}* N
  - POS information can be used to lemmatize a word correctly (i.e., to remove inflections):
    - *saw* [n] → *saw*; *saw* [v] → *see*

The task of part of speech tagging

- - Can differentiate word senses that involve part of speech differences
  - POS can be used as backoff in various class-based models, when too little information is known about a particular word
  - Can be a preprocessor for a parser (speeds up parser; often better, but more expensive, to let the parser do the tagging as well)
  - Tagged text helps linguists find interesting syntactic constructions in texts (*ssh* used as a verb)

The linguistics of parts of speech and tag sets

- We’re not going to substantively discuss parts of speech in class
  - Read section 3.1 to learn about parts of speech, particularly the kind of richer sets of distinctions commonly made by linguists and in NLP applications
  - Read section 4.3.2 for discussion of POS tag sets used in NLP.
    - There’s a handy table explaining tag abbreviations on pp. 141–142

Part of speech tagging

Information sources:

- Sequence of words: syntagmatic information
  - Surprisingly weak information source
  - Many words have various parts of speech – cf. the example above
- Frequency of use of words
  - Surprisingly effective: gets 90+% performance by itself (for English)*
    - This acts as a baseline for performance

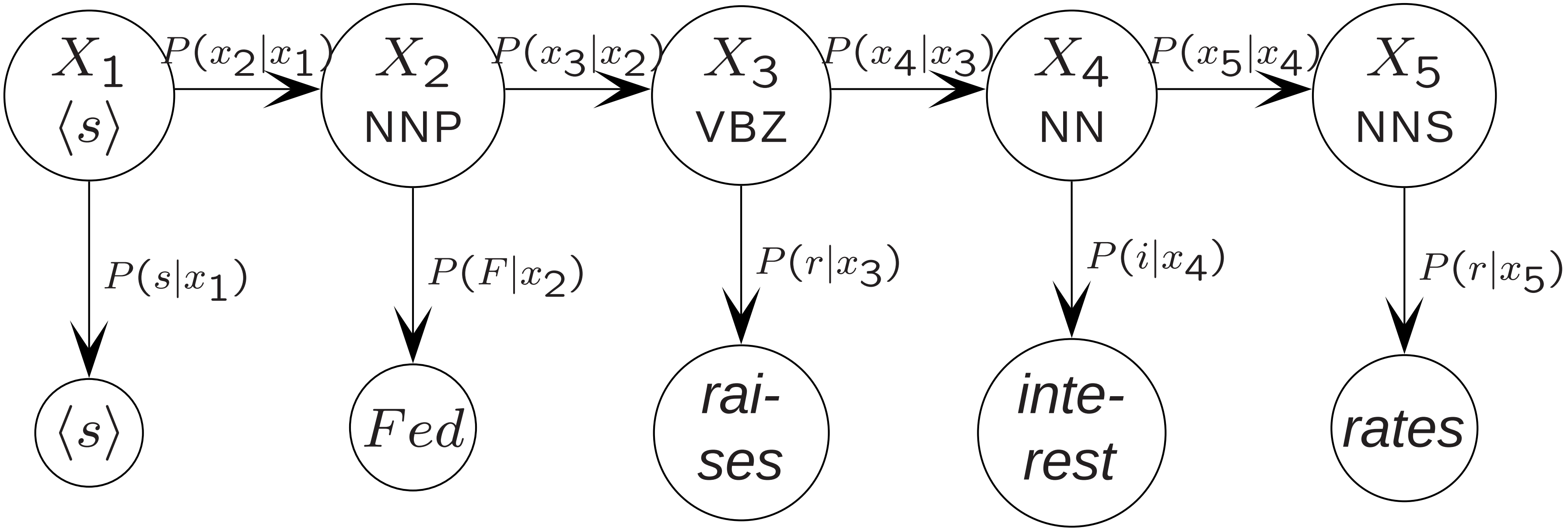
*Even up to 93.7%, based on the results of Toutanova et al. (2003).

(Hidden) Markov model tagger

- View sequence of tags as a Markov chain. Assumptions:
  - **Limited horizon.** $P(X_{i+1} = t^j | X_1, \ldots, X_i) = P(X_{i+1} = t^j | X_i)$
  - **Time invariant (stationary).** $P(X_{i+1} = t^j | X_i) = P(X_2 = t^j | X_1)$

  We assume that a word's tag only depends on the previous tag (limited horizon) and that this dependency does not change over time (time invariance)
- A state (part of speech) generates a word. We assume it depends only on the state

Hidden Markov Models – POS example



- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations
- We normally do supervised training, and then (Bayesian network style) inference to decide POS tags

Standard HMM formalism

- $\langle X, O, \Pi, A, B \rangle$
- X is hidden state sequence; O is observation sequence
- Π is probability of starting in some state
  (can be folded into A: let $A' = [\Pi|A]$, i.e., $a_{0j} = \pi_j$)
- A is matrix of transition probabilities (top row conditional probability tables (CPTs))
- B is matrix of output probabilities (vertical CPTs)

HMM is also a probabilistic (nondeterministic) finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simplest case)

Probabilistic inference in an HMM

Three fundamental questions:

- Given an observation sequence, compute the most likely hidden state sequence
- Compute the probability of a given observation sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

Most likely hidden state sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B, \Pi)$
- We want to find:

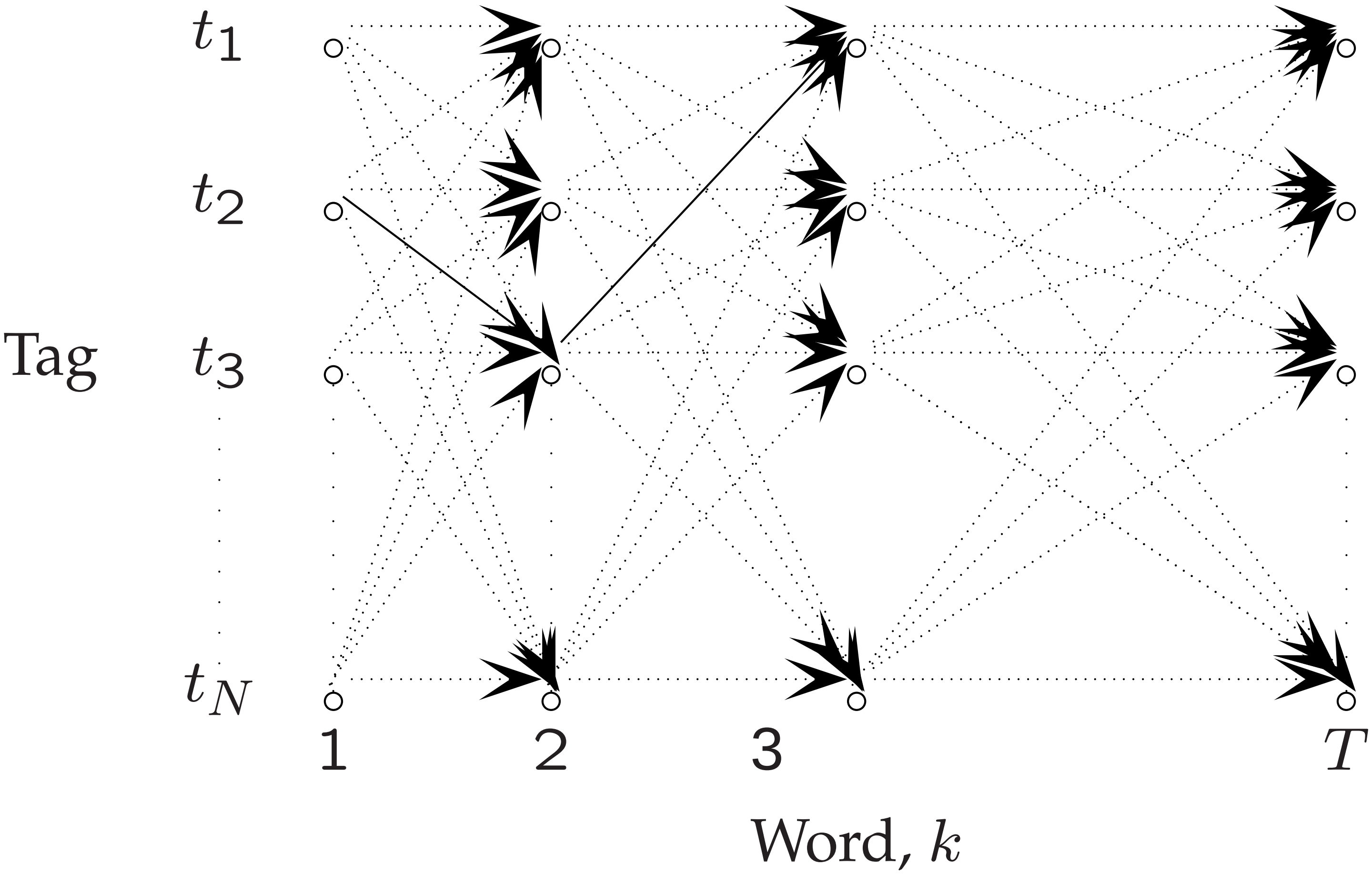
$$\arg\max_X P(X|O,\mu) = \arg\max_X \frac{P(X,O|\mu)}{P(O|\mu)} = \arg\max_X P(X,O|\mu$$

- $P(O|X,\mu) = b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_T o_T}$
- $P(X|\mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{X_{T-1} X_T}$
- $P(O,X|\mu) = P(O|X,\mu) P(X|\mu)$
- $\arg\max_X P(O,X|\mu) = \arg\max_{x_1 \cdots x_T} \Pi_{t=1}^{T} a_{X_{t-1} X_t} b_{x_t o_t}$
- Problem: Exponential in sequence length!

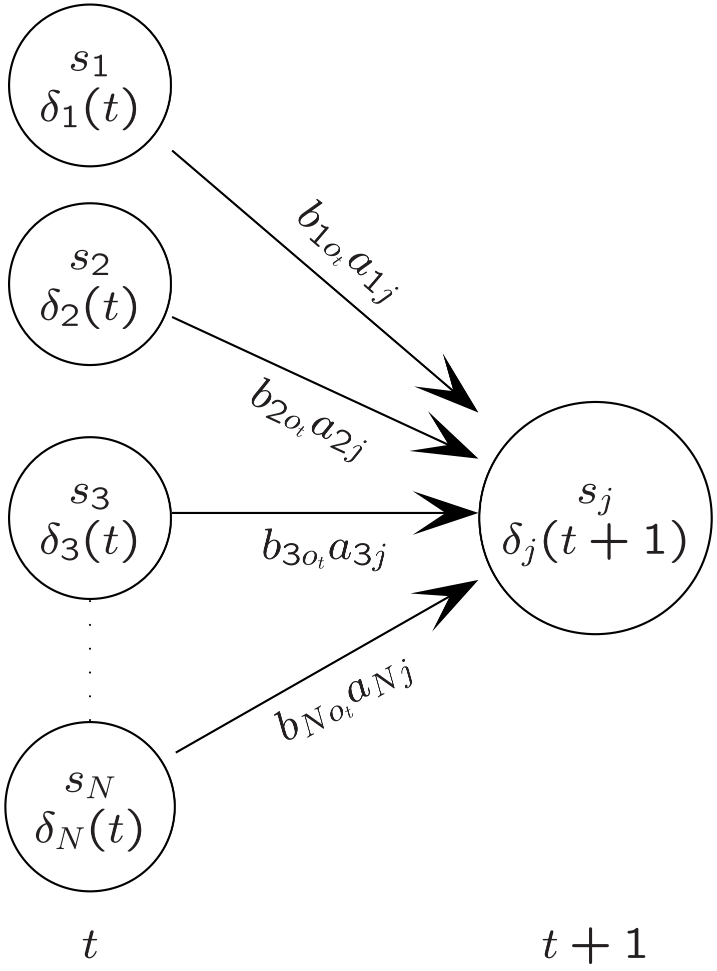
Dynamic Programming

- Efficient computation of this maximum: Viterbi algorithm
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward score $\delta_i(t) = \max_{x_1 \ldots x_{t-1}} P(o_1 o_2 \cdots o_{t-1}, x_1 \cdots x_{t-1}, X_t = i|\mu)$
- $\delta_j(t+1) = \max_{i=1}^{N} \delta_i(t) b_{io_t} a_{ij}$
- Compute it recursively from beginning
- Remember best paths
- A version of Bayes Net most likely state inference

Trellis algorithms



Closeup of the computation at one node



$$\delta_j(t+1) = \max_{i=1}^{N} \delta_i(t) b_{io_t} a_{ij}$$

Viterbi algorithm (Viterbi 1967)

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- A dynamic programming algorithm. Essentially the same except you do a max instead of a summation, and record the path taken:

$$\delta_{i+1}(t^j) = \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(w_i|t^k) \times P(t^j|t^k)]$$

$$\psi_{i+1}(t^j) = \arg\max_{1 \leq k \leq T} [\delta_i(t^k) \times P(w_i|t^k) \times P(t^j|t^k)]$$

- This gives a best tag sequence for POS tagging
- (Note: this is different to finding the most likely tag for each time t!)

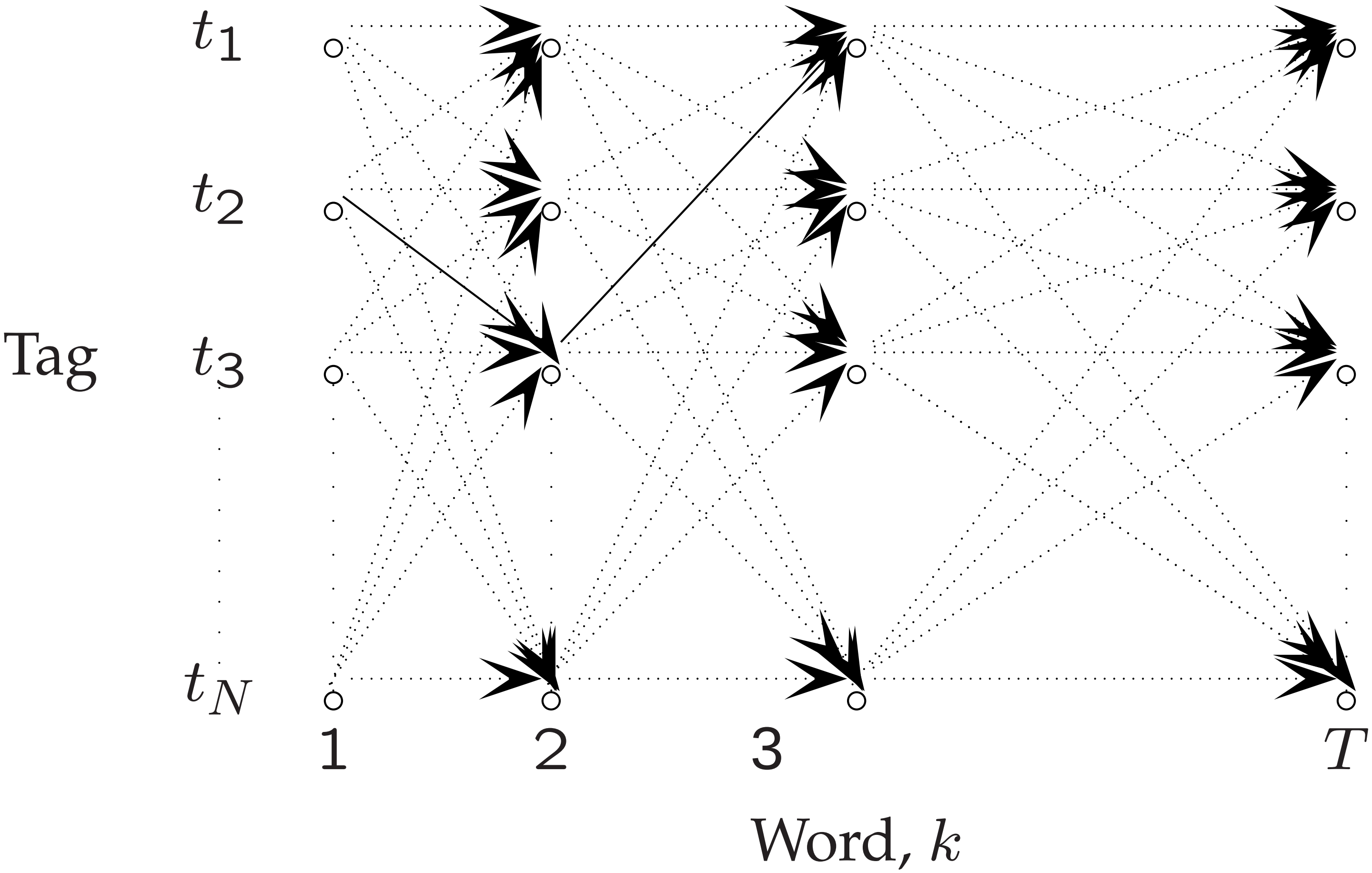
Probability of an observation sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B, \Pi)$
- $P(O|X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_T o_T}$
- $P(X|\mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{X_{T-1} X_T}$
- $P(O, X|\mu) = P(O|X, \mu) P(X|\mu)$
- $P(O|\mu) = \sum_X P(O|X, \mu) P(X|\mu)$ [Marginalization]
- $P(O|\mu) = \sum_{x_1 \cdots x_T} \prod_{t=1}^{T} a_{X_{t-1} X_t} b_{x_t o_t}$
- Difficult to compute like this!

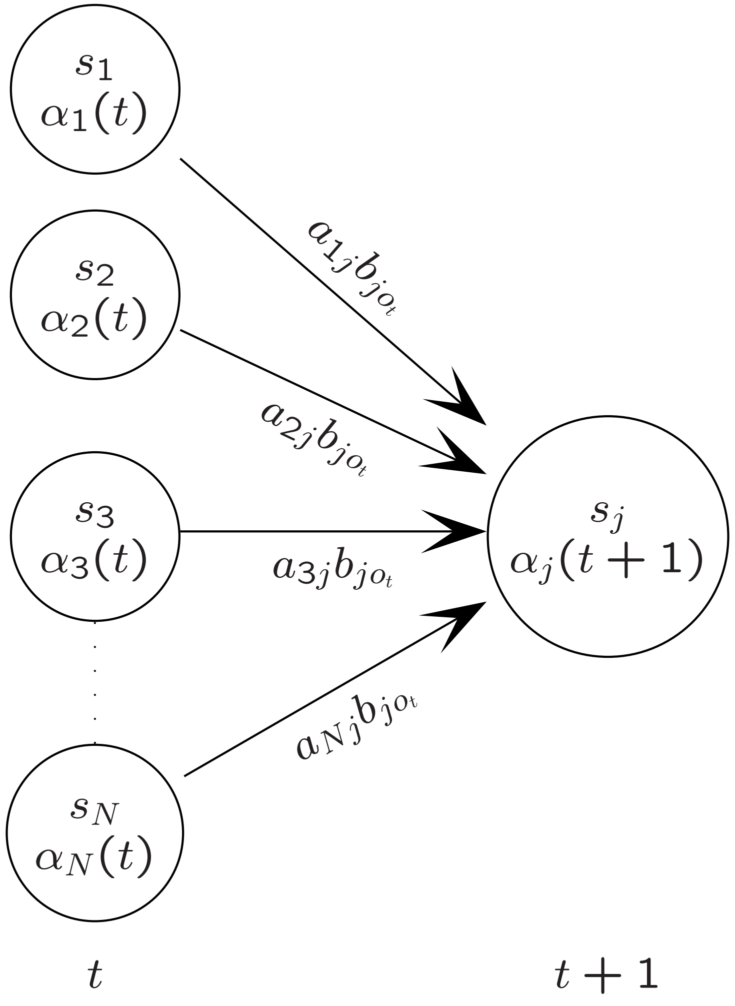
Dynamic Programming

- Efficient computation of this quantity: forward procedure
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward probability
  $\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{io_t}$
- Compute it recursively from beginning
- Similarly calculated backward probability $\beta_i(t)$ from end
- A version of the variable elimination algorithm for Bayes Net inference

Trellis algorithms



Closeup of the computation at one node



$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{jo_t}$$

Dynamic Programming (2)

- Similarly, calculate backward probability $\beta_i(t)$ from end
- The probability of the rest of the observations given a middle state
- $\beta_i(t) = P(o_t \cdots o_T | X_t = i)$
- $\beta_i(T+1) = 1$
- $\beta_i(t) = \sum_{j=1,\ldots,N} a_{ij} b_{io_t} \beta_j(t+1)$
- Forward: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T)$
- Backward: $P(O|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$
- Combination: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t), \quad$ for any t

Viterbi algorithm

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- Another dynamic programming algorithm. Essentially the same except you do a max instead of a summation, and record the path taken:

$$\delta_{i+1}(t^j) = \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

$$\psi_{i+1}(t^j) = \arg\max_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

- This gives a tag sequence for POS tagging

(H)MM POS tagging

- Estimate a and b parameters, preferably by doing counts from tagged training data, or else by using lexicons, Baum-Welch, as best one can
  - i.e., in simplest case, train as visible Markov model, and only tag as hidden Markov model
- Use Viterbi algorithm to tag unseen text
- On Penn tag set, can get accuracy of up to 96.7% (Brants 2000)
- Many other methods have also been used for POS tagging, the best having similar accuracy

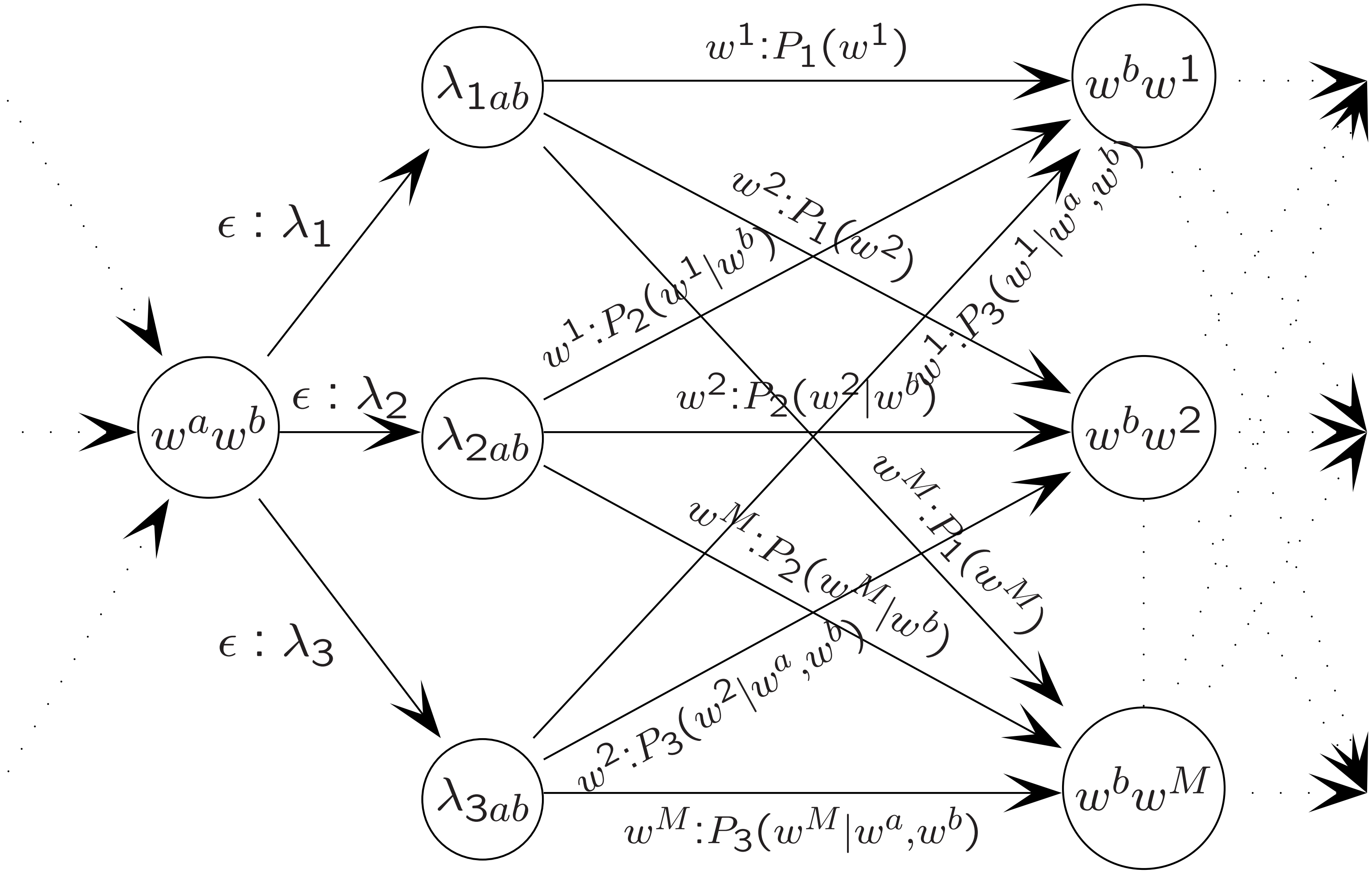
Tagging: Other topics

- Trigram taggers, variable memory models, etc.
- Smoothing
- Unknown words
  - Need to model via word features
- Building taggers without a hand-tagged corpus
- Other methods of tagging:
  - Transformation based learning
  - Decision trees
  - Maximum entropy models
  - . . .

Tagging: Other topics

- Other languages
- Accuracy (factors: training data, tag set, drift, unknown words)
- You gain by having an explicit end symbol/state (making it a probabilistic regular grammar)
- Parameter tying (see following language model example)
- Linguistic foundations of tags: not always determinate
- Applications: IR, IE, QA, terminology extraction

Some of an HMM for an interpolated language model



The third problem: Parameter estimation = Parameter learning

- We want to find the most likely model parameters given the data (using MLE):

$$\arg\max_{\mu} P(O_{\text{training}}|\mu)$$

- This would let us learn model probabilities from raw data
- Can’t determine these probabilities analytically.
- Use iterative hill-climbing algorithm to try to find good model

HMM Training: Baum-Welch reestimation

- Used to automatically estimate parameters of an HMM
- a.k.a. the Forward-Backward algorithm
- A case of the Expectation Maximization (EM) algorithm
- One starts with initial probability estimates
- One computes expectations of how often each transition/emission is used
- One re-estimates the probabilities based on those expectations
- . . . and repeat until convergence

HMM Training: Baum-Welch reestimation

- Needed because the state paths are hidden, and the equations cannot be solved analytically
- Provides a maximum likelihood estimate: attempts to find the model that assigns the training data the highest likelihood
- Hill-climbing algorithm that can get stuck in local maxima
- Not so effective for inductive POS tagging (the ML reestimation procedure doesn't know the meaning we have given to the hidden states)
- But good in many tasks (speech, including information extraction)

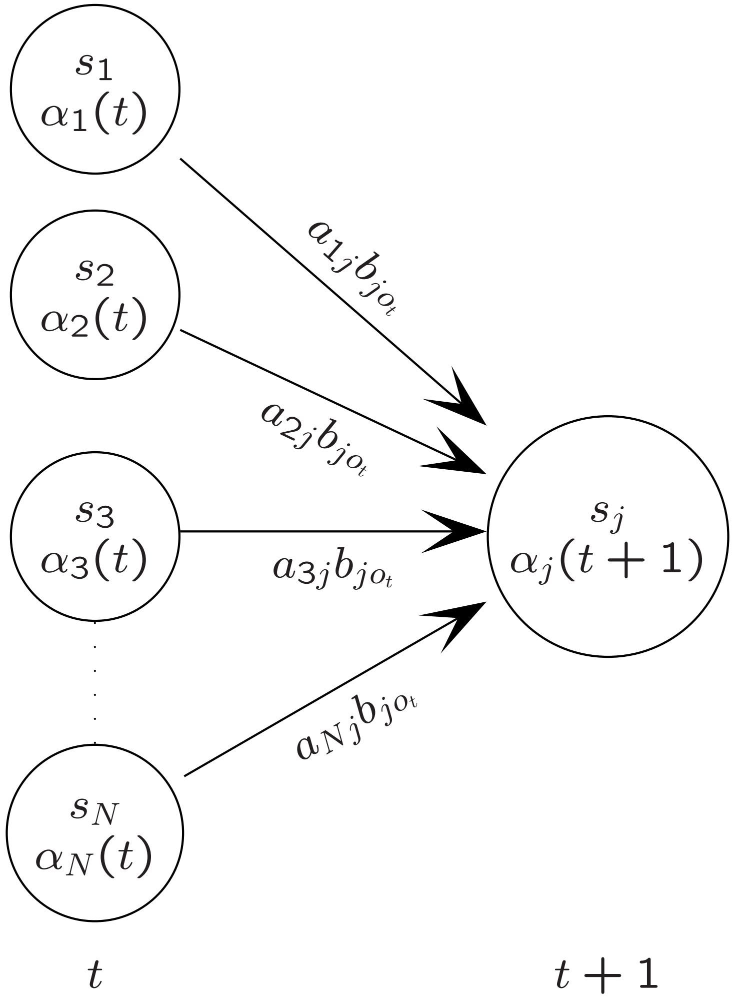
Probability of an observation sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B, \Pi)$
- $P(O|X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_T o_T}$
- $P(X|\mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{x_{T-1} x_T}$
- $P(O, X|\mu) = P(O|X, \mu) P(X|\mu)$
- $P(O|\mu) = \sum_X P(O|X, \mu) P(X|\mu)$ [Marginalization]
- $P(O|\mu) = \sum_{x_1 \cdots x_T} \prod_{t=1}^{T} a_{X_{t-1} X_t} b_{x_t o_t}$
- Again, difficult to compute like this!

Dynamic Programming

- Efficient computation of this quantity: forward procedure
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward probability
  $\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{io_t}$
- Compute it recursively from beginning
- A version of the variable elimination algorithm for Bayes Net inference

Closeup of the computation at one node



$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t)a_{ij}b_{jo_t}$$

Dynamic Programming (2)

- Similarly, calculate backward probability $\beta_i(t)$ from end
- The probability of the rest of the observations given a middle state
- $\beta_i(t) = P(o_t \cdots o_T | X_t = i)$
- $\beta_i(T+1) = 1$
- $\beta_i(t) = \sum_{j=1,\ldots,N} a_{ij} b_{io_t} \beta_j(t+1)$
- Forward: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T)$
- Backward: $P(O|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$
- Combination: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t), \quad$ for any t

Maximum likelihood parameter estimation

- For some observed data $O = \langle o_1 \cdots o_n \rangle$, and a model, here a bigram model, the data likelihood for a particular set of parameters $\Theta = \{P(o^k|o^j)\}$ is:

$$L(O|\Theta) = \prod_{i=1}^{n} P(o_i|o_{i-1}) = \prod_{j=1}^{V} \prod_{k=1}^{V} P(o^k|o^j)^{\#(o^j o^k)}$$

- People often use the log because its easier to manipulate, and the log is monotonic with the likelihood:

$$LL(O|\Theta) = \sum_{i=1}^{n} \log P(o_i|o_{i-1}) = \sum_{j=1}^{V} \sum_{k=1}^{V} \#(o^j o^k) \log P(o^k|o^j)$$

- We can work out how to maximize this likelihood using calculus (assignment)

Maximum likelihood parameter estimation

- For an HMM with observed state data X, and s states, we do the same:

$$\begin{aligned} L(O, X|\Theta) &= \prod_{i=1}^{n} P(x_i|x_{i-1})P(o_i|x_i) \\ &= \prod_{j=1}^{s}\prod_{k=1}^{s} P(x^k|x^j)^{\#(x^j x^k)} \prod_{k=1}^{s}\prod_{m=1}^{V} P(o^m|x^k)^{\#(x} \\ &= a_{x_0x_1}a_{x_1x_2}a_{x_2x_3}\cdots a_{x_{n-1}x_n}b_{x_1o_1}b_{x_2o_2}\cdots b_{x_no_n} \end{aligned}$$

- We can maximize this likelihood by setting the parameters in Θ, and get the same form of relative frequency estimates
- But if our state sequence is *unobserved* we can't do that directly

HMM maximum likelihood parameter estimation

- However, we can work out the likelihood of being in different states at different times, given the current model and the observed data:

$$P(X_t = x^k | O, \Theta) = \frac{\alpha_k(t)\beta_k(t)}{\sum_{j=1}^{s} \alpha_j(t)\beta_j(t)}$$

- Given, these probabilities, something we could do is *sample* from this distribution and generate pseudo-data which is complete.
- From this data $\langle O, \hat{X} \rangle$, we could do ML estimation as before – since it is complete data
- And with sufficient training data, this would work fine.

HMM maximum likelihood parameter estimation

- For the EM algorithm, we do something just slightly subtler. We work out the *expected* number of times we made each state transition and emitted each symbol from each state. This is conceptually just like an observed count, but it'll usually be a non-integer
- We then work out new parameter estimates as relative frequencies just like before.

Parameter reestimation formulae

$$\begin{aligned}
\widehat{\pi}_i &= \text{expected frequency in state } i \text{ at time } t = 1 \\
&= \gamma_i(1)
\end{aligned}$$

$$\begin{aligned}
\widehat{a}_{ij} &= \frac{\text{expected num. transitions from state } i \text{ to } j}{\text{expected num. transitions from state } i} \\
&= \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}
\end{aligned}$$

$$\begin{aligned}
\widehat{b}_{ik} &= \frac{\text{expected num. times } k \text{ observed in state } i}{\text{expected num. transitions from } i} \\
&= \frac{\sum_{\{t: o_t = k, 1 \le t \le T\}} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}
\end{aligned}$$

EM Algorithm

- Changing the parameters in this way *must* have increased (or at any rate not decreased) the likelihood of this completion of the data: we're setting the parameters on the pseudo-observed data to maximize the likelihood of this pseudo-observed data
- But, then, we use these parameter estimates to compute new expectations (or, to sample new complete data)
- Since this new data completion is directly based on the current parameter settings, it is at least intuitively reasonable to think that the model should assign it higher likelihood than the old completion (which was based on different parameter settings)

We’re guaranteed to get no worse

- Repeating these two steps iteratively gives us the EM algorithm
- One can prove rigorously that iterating it changes the parameters in such a way that the data likelihood is non-decreasing (Baum et al. 1970; Dempster et al. 1977)
- But we can get stuck in local maxima or on saddle points, though
  - For a lot of NLP problems with a lot of hidden structure, this is actually a *big* problem

Information extraction evaluation

- Example text for IE:

  Australian Tom Moody took six for 82 but Chris Adams , 123 , and Tim O’Gorman , 109 , took Derbyshire to 471 and a first innings lead of 233 .

- Boxes shows attempt to extract person names (correct ones in purple)
- What score should this attempt get?
- A stringent criterion is exact match precision/recall/F_1

Precision and recall

- Precision is defined as a measure of the proportion of selected items that the system got right:

$$\text{precision} = \frac{tp}{tp + fp}$$

- Recall is defined as the proportion of the target items that the system selected:

$$\text{recall} = \frac{tp}{tp + fn}$$

These two measures allow us to distinguish between excluding target items and returning irrelevant items.

They still require human-made “gold standard” judgements.

Combining them: The F measure

Weighted harmonic mean: The F measure (where $F = 1 - E$):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

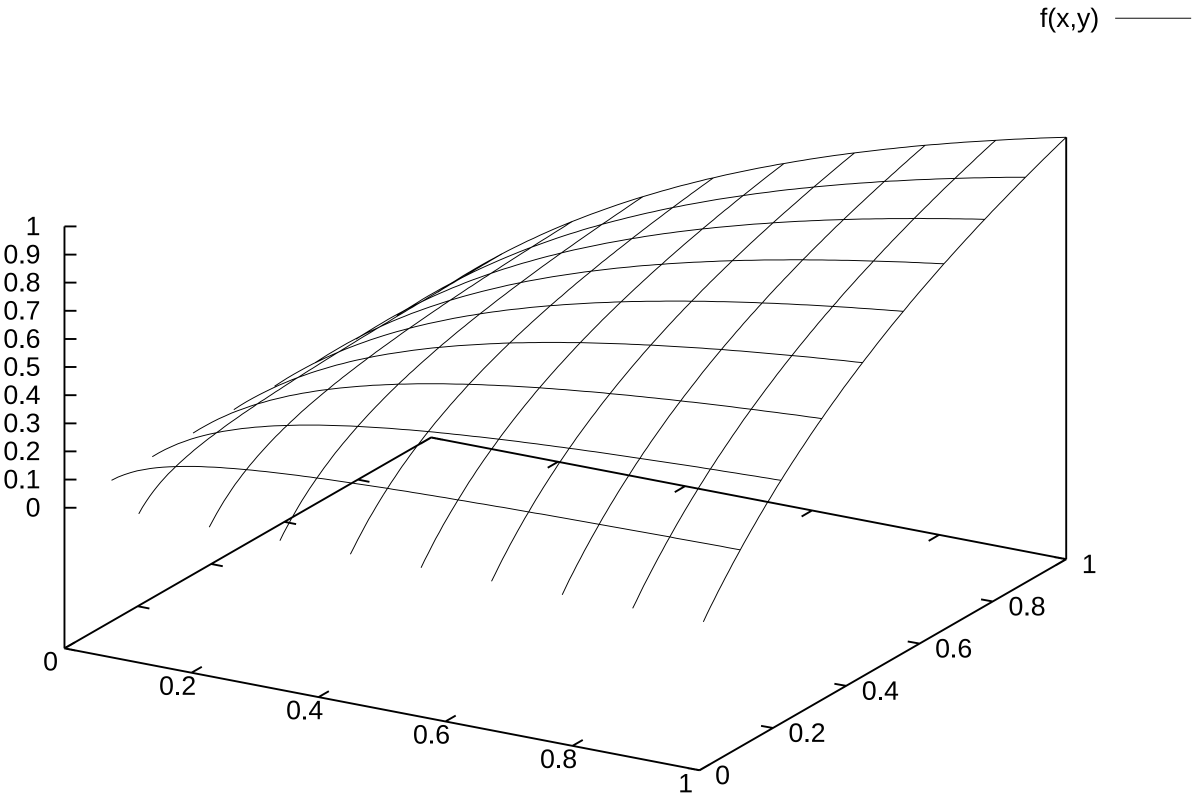
where P is precision, R is recall and α weights precision and recall. (Or in terms of β, where $\alpha = 1/(\beta^2 + 1)$.)

A value of $\alpha = 0.5$ is often chosen.

$$F = \frac{2PR}{R + P}$$

At break-even point, when $R = P$, then $F = R = P$

The F measure ($\alpha = 0.5$)



Ways of averaging

Precision	Recall	Arithmetic	Geometric	Harmonic	Minimum
80	10	45	28.3	17.8	10
80	20	50	40.0	32.0	20
80	30	55	49.0	43.6	30
80	40	60	56.6	53.3	40
80	50	65	63.2	61.5	50
80	60	70	69.3	68.6	60
80	70	75	74.8	74.7	70
80	80	80	80.0	80.0	80
80	90	85	84.9	84.7	80
80	100	90	89.4	88.9	80

EM Algorithm Intuition

- We don’t know what the model is.
- But we can work out the probability of the observation sequence using some (perhaps random) model
- Looking at that calculation, we can see which state transitions and symbol emissions were probably used the most
- By increasing the probability of those, we can choose a revised model which gives a higher probability to the observation sequence

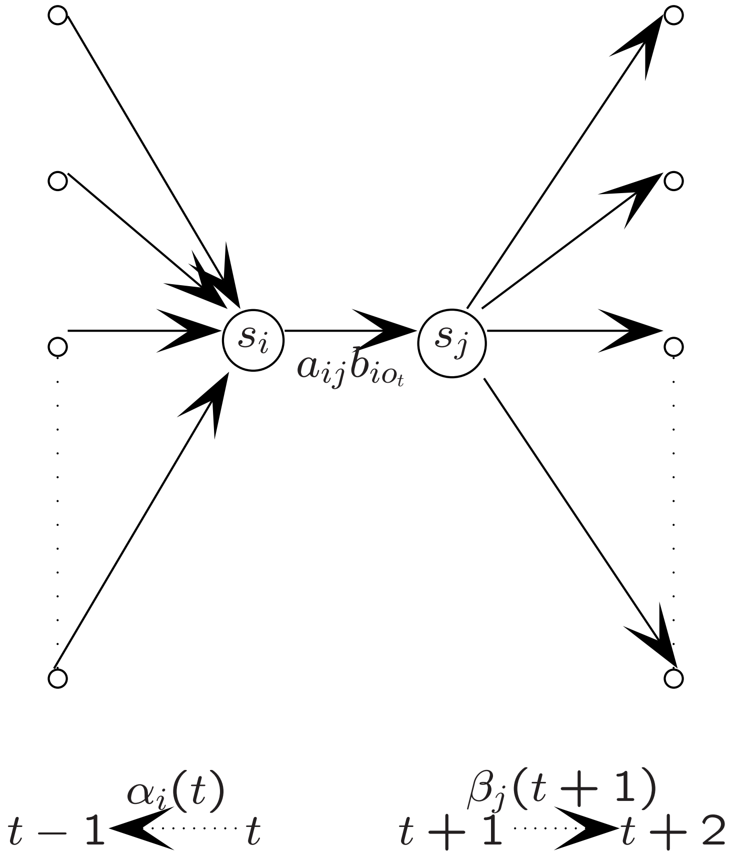
The likelihood of being in state i at time t

Using forward and backward variables, we can calculate $\gamma_i(t)$, the probability of being in state i at time t:

$$\begin{aligned}\gamma_i(t) &= P(X_t = i|O, \mu) \\ &= \frac{P(X_t = i, O|\mu)}{P(O|\mu)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}\end{aligned}$$

Chance of moving from state i to j at time t

Define $p_t(i, j)$, the probability of traversing an arc $i - j$ at time t given observations O.



Computing probability of traversing arc

$$\begin{aligned}
p_t(i,j) &= \quad P(X_t = i, X_{t+1} = j | O, \mu) \\
&= \frac{P(X_t{=}i, X_{t+1}{=}j, O|\mu)}{P(O|\mu)} \\
&= \frac{\alpha_i(t) a_{ij} b_{io_t} \beta_j(t{+}1)}{\sum_{m=1}^{N} \alpha_m(t) \beta_m(t)} \\
&= \frac{\alpha_i(t) a_{ij} b_{io_t} \beta_j(t{+}1)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m(t) a_{mn} b_{mo_t} \beta_n(t{+}1)}
\end{aligned}$$

Note that $\gamma_i(t) = \sum_{j=1}^{N} p_t(i,j)$.

Expectations

Now, if we sum over the time index, this gives us expectations (counts):

$\sum_{t=1}^{T} \gamma_i(t)$ = expected number of transitions from state i in O

$\sum_{t=1}^{T} p_t(i,j)$ = expected number of transitions from state i to j in O

Parameter reestimation formulae

$$\begin{aligned}
\widehat{\pi}_i &= \text{expected frequency in state } i \text{ at time } t = 1 \\
&= \gamma_i(1)
\end{aligned}$$

$$\begin{aligned}
\widehat{a}_{ij} &= \frac{\text{expected num. transitions from state } i \text{ to } j}{\text{expected num. transitions from state } i} \\
&= \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}
\end{aligned}$$

$$\begin{aligned}
\widehat{b}_{ik} &= \frac{\text{expected num. times } k \text{ observed in state } i}{\text{expected num. transitions from } i} \\
&= \frac{\sum_{\{t:o_t=k,1\le t\le T\}} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}
\end{aligned}$$

Baum-Welch training algorithm

- Begin with some model μ (perhaps random, perhaps preselected)
- Run O through the current model to estimate the expectations of each model parameter
- Change the model to maximize the values of the paths that are used a lot (while still respecting the stochastic constraints)
- Repeat, hoping to converge on optimal values for the model parameters μ.

We’re guaranteed to get no worse

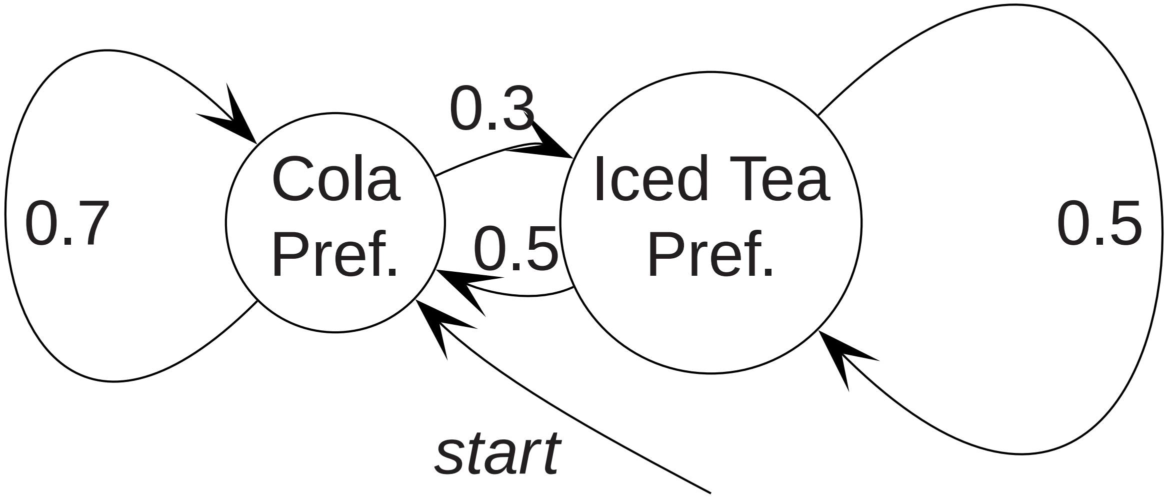
From $\mu = (A, B, \Pi)$, one iteration derives $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$.

Baum initially proved for HMMs, and the EM framework in general gives us that:

$$P(O|\hat{\mu}) \geq P(O|\mu)$$

But we may stay in a local optimum.

HMMs: The crazy soft drink machine



Output probability given state

	cola	iced tea (ice_t)	lemonade (lem)
CP	0.6	0.1	0.3
IP	0.1	0.7	0.2

Always starts off in cola-preferring state.

Values for $p_t(i, j)$ on O = (lem, ice_t, cola):

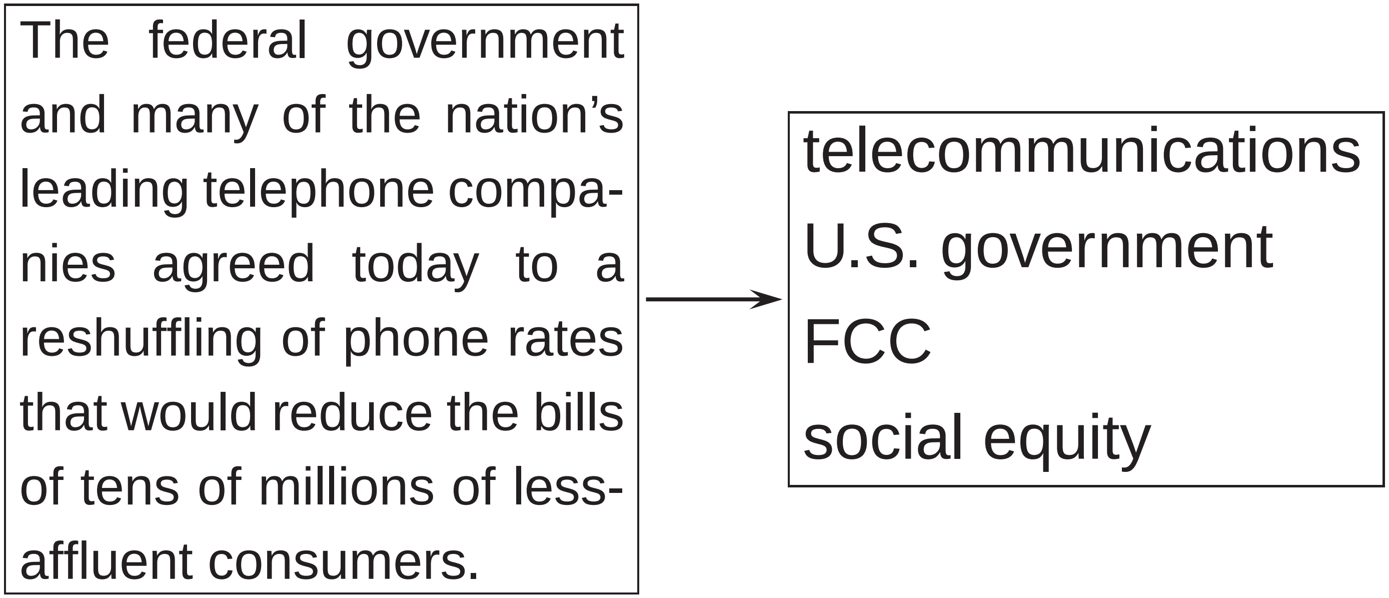
Time (and j)	**1**	CP	IP	γ_1
i	CP	0.3	0.7	1.0
	IP	0.0	0.0	0.0
	2	CP	IP	γ_2
i	CP	0.28	0.02	0.3
	IP	0.6	0.1	0.7
	3	CP	IP	γ_3
i	CP	0.616	0.264	0.88
	IP	0.06	0.06	0.12

The parameters will be reestimated as follows by EM:

		Original			Reestimated		
Π	CP	1.0			1.0		
	IP	0.0			0.0		
		CP	IP		CP	IP	
A	CP	0.7	0.3		0.548	0.451	
	IP	0.5	0.5		0.804	0.195	
		cola	ice_t	lem	cola	ice_t	lem
B	CP	0.6	0.1	0.3	0.404	0.138	0.459
	IP	0.1	0.7	0.2	0.146	0.854	0.0

Other uses of HMMs: Topic extraction

- More fine-grained IR: a text can be about multiple topics, some words of it generated by each
- Train classifiers for individual topics
- Commonly use hierarchical classification, so smaller classifiers can be trained for individual decisions



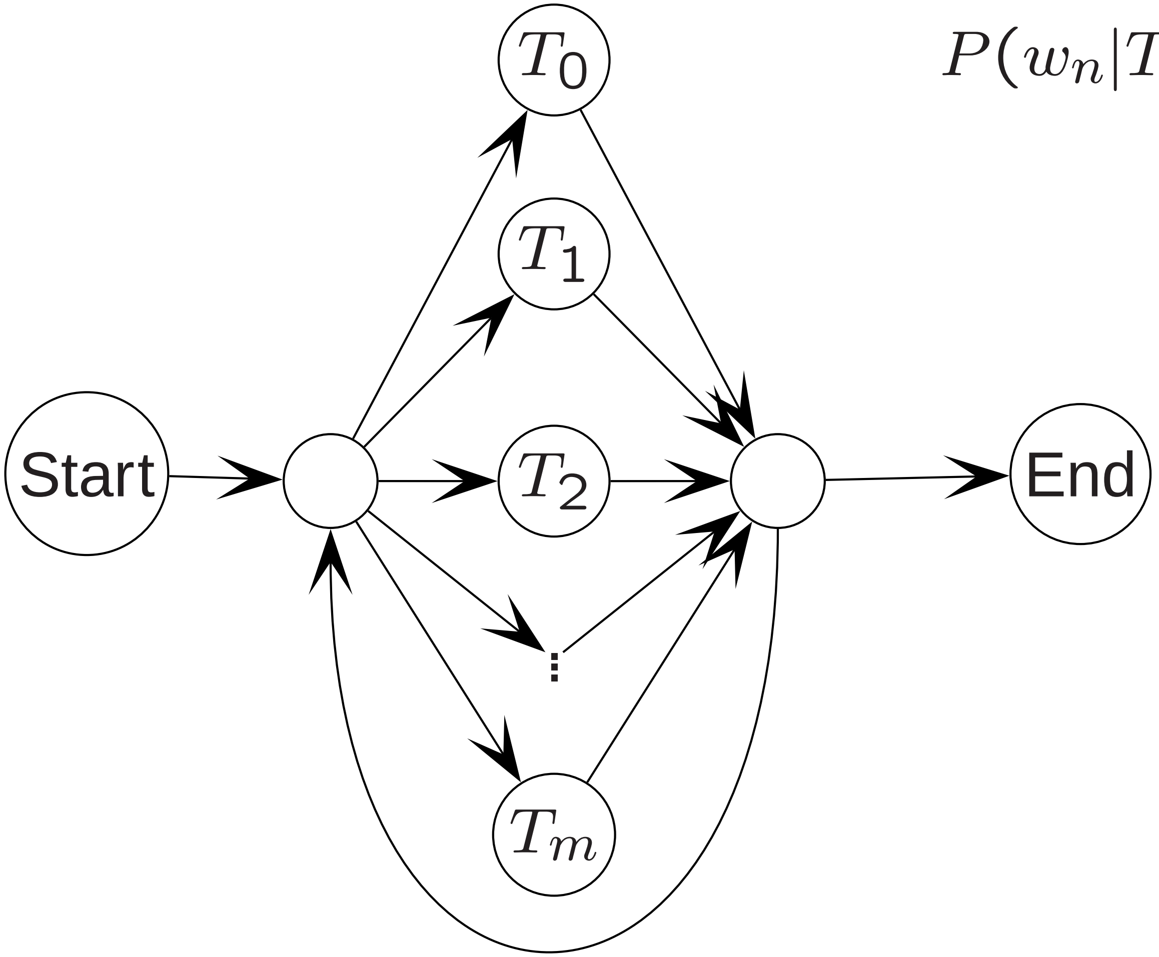
The federal government and many of the nation's leading telephone companies agreed today to a reshuffling of phone rates that would reduce the bills of tens of millions of less-affluent consumers.

telecommunications
U.S. government
FCC
social equity

Generative HMM for topic extraction

- Choose a set of topics $T = \{T_1, \ldots, T_m\}$
- Put a distribution over how many topics are in the set – disfavoring a lot of topics
- For each word in the story, choose a topic according to $P(T_j|T)$
- Choose a word according to $P(w_n|T_j)$
- Proceed to decode by finding out the use of which set of topics would give highest probability to the observed story [often restrict when topic can change. . .]
- BBN: 4600 topics, 76% of first choices in annotated labels (vs. 45% for IR-style tf.idf)

Generative HMM for topic extraction



$P(w_n|T_j)$

Other uses of HMMs: Extracting facts

- *The problem with IR:* You search for “soldiers attacking rebels” and the top matches are:
  - Hutu rebels attacked soldiers in one of Bujumbura’s suburbs (Google 2000/10/03)
  - Sudanese rebels say they have killed or wounded more than 300 government soldiers (Hotbot 2000/10/03)
  - [Altavista: a Confederate soldier’s memorial!]
- We need to be able to match relationships like:
  - attack(soldiers, rebels)
- Models that see sentence structure, like HMMs or dependency parsers, let us capture these relations, though we still need to deal with synonymy and polysemy

Other uses of HMMs: Information Extraction (Freitag and McCallum 1999)

- IE: extracting instance of a relation from text snippets
- States correspond to fields one wishes to extract, token sequences in the context that are good for identifying the fields to be extracted, and a background “noise” state
- Estimation is from tagged data (perhaps supplemented by EM reestimation over a bigger training set)
- The Viterbi algorithm is used to tag new text
- Things tagged as fields to be extracted are returned

Information Extraction (Freitag and McCallum 1999)

- State topology is set by hand. Not fully connected
- Use simpler and more complex models, but generally:
  - Background state
  - Preceding context state(s)
  - Target state(s)
  - Following context state(s)
- Preceding context states connect only to target state, etc.

Information Extraction (Freitag and McCallum 1999)

- Each HMM is for only one field type (e.g., “speaker”)
- Use different HMMs for each field (*bad:* no real notion of multi-slot structure)
- Semi-supervised training: target words (generated only by target states) are marked
- Shrinkage/deleted interpolation is used to generalize parameter estimates to give more robustness in the face of data sparseness
- Some other work has done multi-field extraction over more structured data (Borkar et al. 2001)

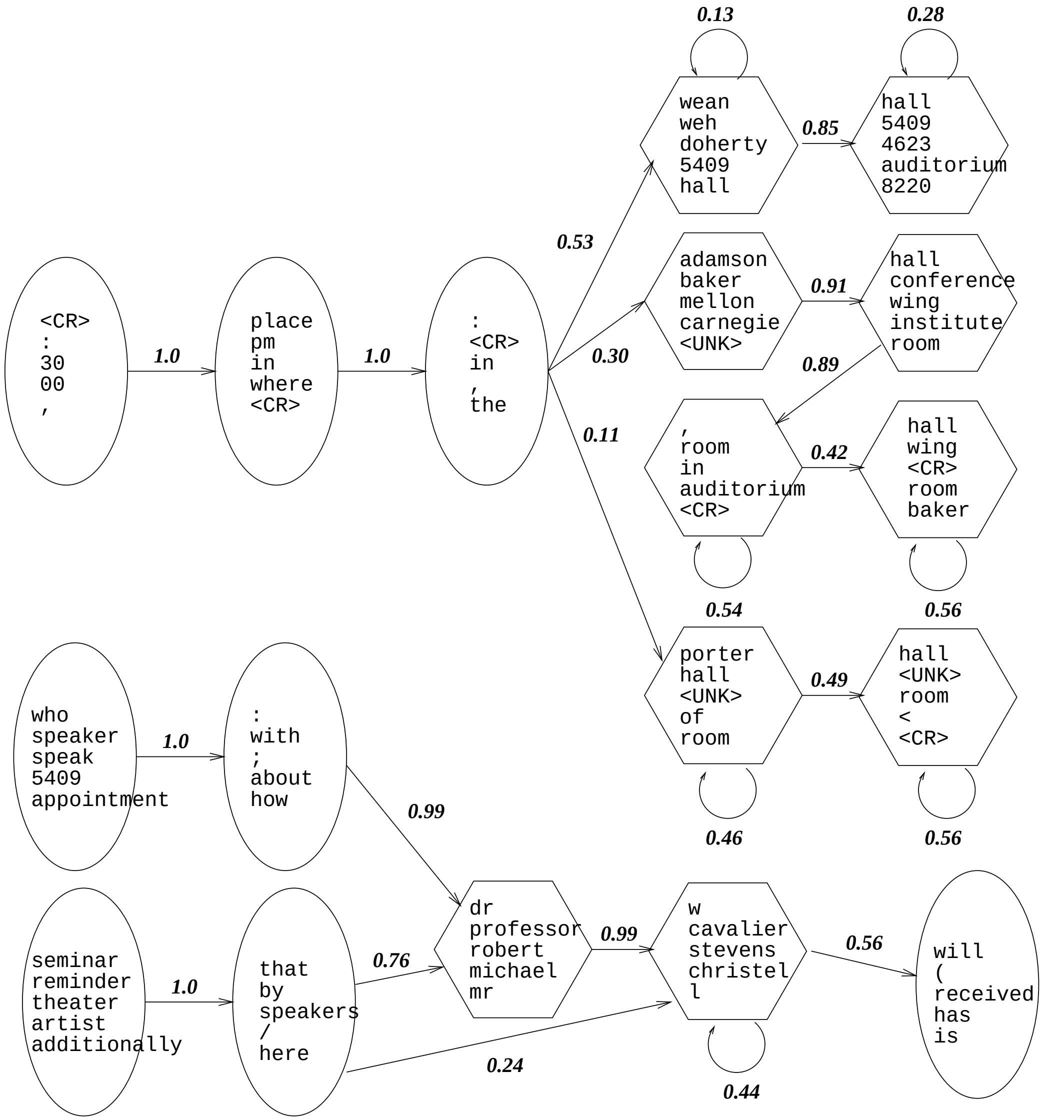
Information Extraction (Freitag and McCallum 1999)

- Tested on seminar announcements and corporate acquisitions data sets
- Performance is generally equal to or better than that of other information extraction methods
- Though probably more suited to semi-structured text with clear semantic sorts, than strongly NLP-oriented problems
- HMMs tend to be especially good for robustness and high recall

Information extraction

- Getting particular fixed semantic relations out of text (e.g., buyer, sell, goods) for DB filling
- Statistical approaches have been explored recently, particularly use of HMMs (Freitag and McCallum 2000)
- States correspond to elements of fields to extract, token sequences in the context that identify the fields to be extracted, and background “noise” states
- Estimation is from labeled data (perhaps supplemented by EM reestimation over a bigger training set)
- Structure learning used to find a good HMM structure
- The Viterbi algorithm is used to tag new text
- Things tagged as within fields are returned

Information extraction: locations and speakers


<CR> : 30 00 ,
1.0
place pm in where <CR>
1.0
: <CR> in , the
0.53
0.30
0.11
0.13
wean weh doherty 5409 hall
0.85
0.28
hall 5409 4623 auditorium 8220
adamson baker mellon carnegie <UNK>
0.91
hall conference wing institute room
0.89
, room in auditorium <CR>
0.42
hall wing <CR> room baker
0.54
0.56
porter hall <UNK> of room
0.49
hall <UNK> room < <CR>
0.46
0.56
who speaker speak 5409 appointment
1.0
: with ; about how
0.99
seminar reminder theater artist additionally
1.0
that by speakers / here
0.76
0.24
dr professor robert michael mr
0.99
w cavalier stevens christel l
0.44
0.56
will (received has is

HMMs in Computational Biology

- Determining coding/non-coding regions of DNA
- Determining protein-binding sites of DNA
- Aligning and analyzing protein families
- Aligning DNA regions
- Unlike human languages, observation alphabets are normally small, and so clever smoothing/shrinkage methods are less necessary

Identifying G-protein-coupled receptors (Baldi and Chauvin 1994)

- Long but very variable length chains (200–1200) of amino acids
- HMM trained on known examples, with states equal to average length
- Effectively HMM is computing a kind of probabilistic ‘edit distance’, measuring changes, repeats, and deletions
- Model can effectively discriminate GPCR sequences from random or non-GPCR sequences based on calculated log-likelihood of sequence

Gene finding in DNA sequences (Krogh, Mian and Haussler 1994)

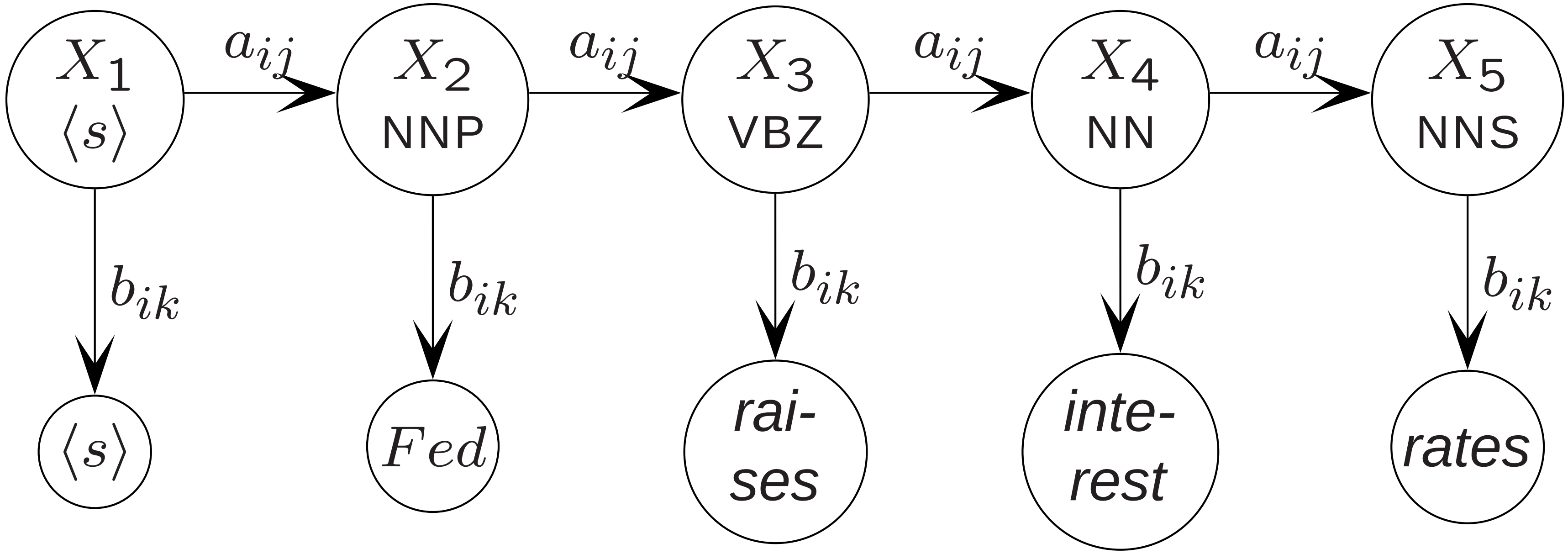
- Finds protein-coding genes in *E. coli* DNA.
- HMM states model codons, intergenic regions (incl. palindromic sequences, etc.)
- Allows for insertions and deletions
- HMM parameters estimated from 1 million nucleotides of annotated DNA
- On independent data, model finds exact location of about 80% of known *E. coli* genes, and approximate location of a further 10%.
- It also potentially finds several new genes. . . .

Summary

- HMMs are one simple, well-understood, and generally effective form of probabilistic model
- Their math is well understood
- Suitable for problems with a linear (or finite-state or time-series) character
- Widely used in speech, language, vision & movement, and DNA alignment applications
- Probabilistic context-free grammar and richer models also exist when richer recursive structure is required.

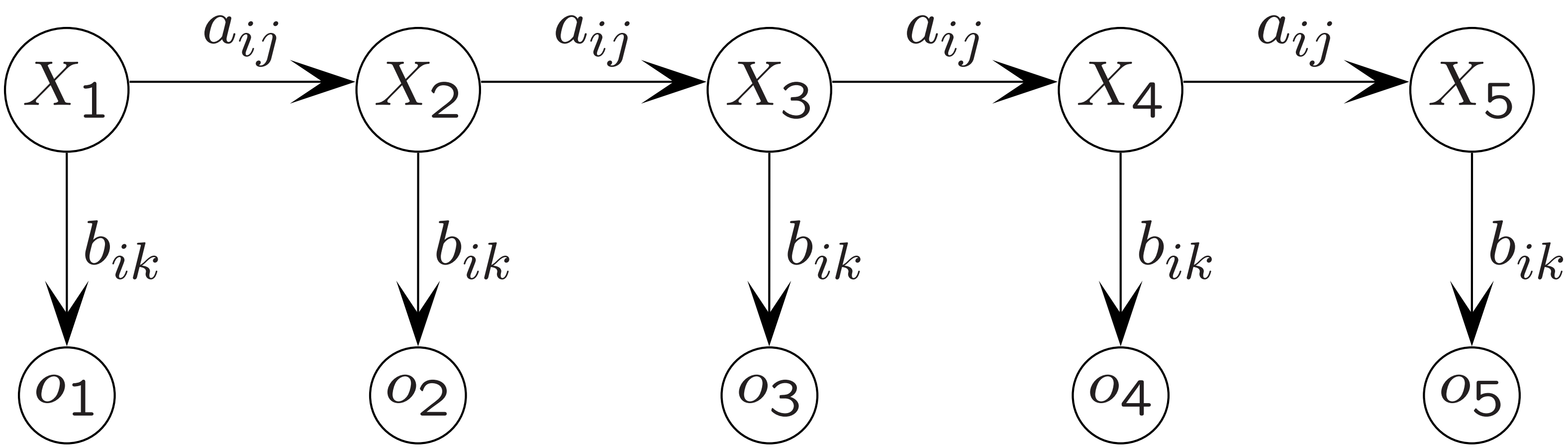
Extra variant slides

Hidden Markov Models – POS example



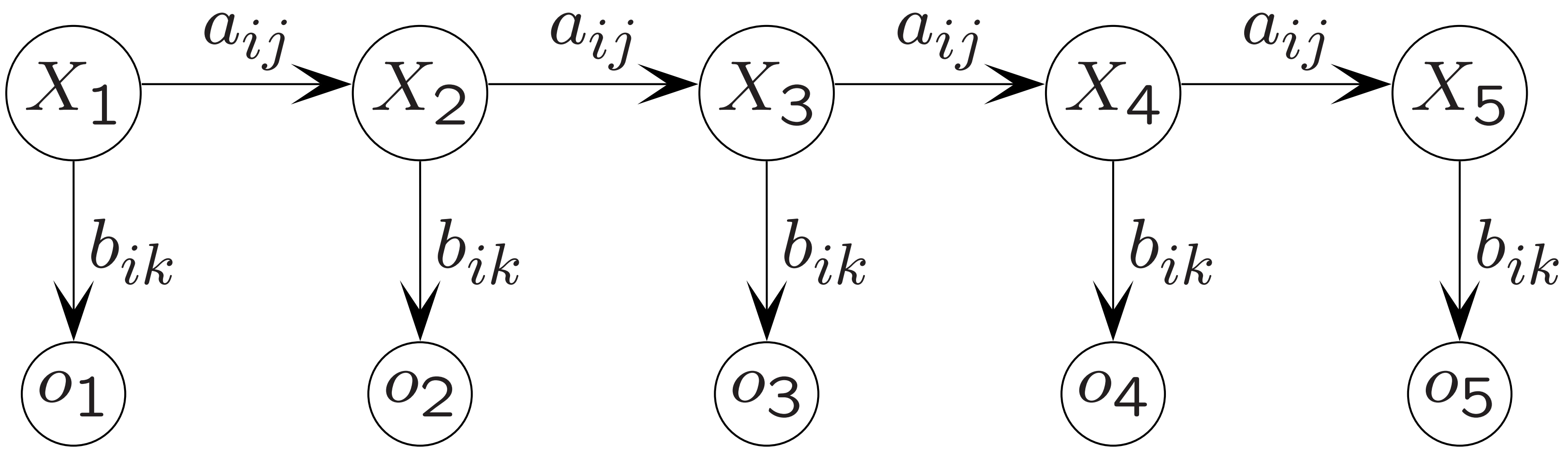
- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations (i.e., words)

Hidden Markov Models (HMMs)



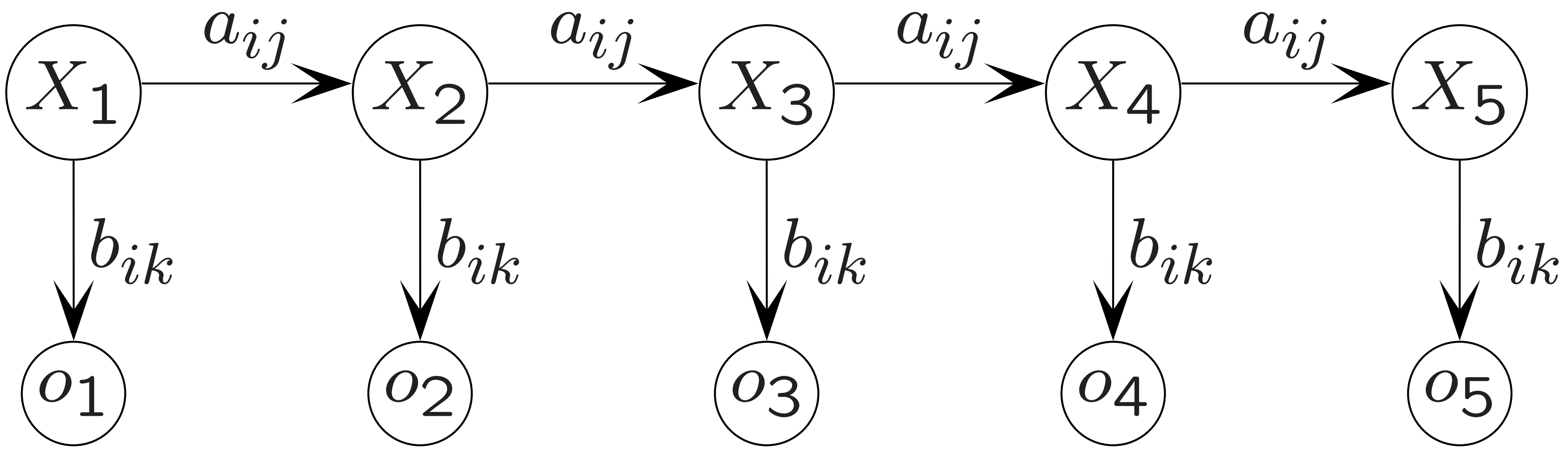
- HMMs are dynamic Bayes Nets (graphical models) with a simple linear structure
- Vertices (circles) are variables; arrows show direct probabilistic dependencies

Hidden Markov Models (HMMs)



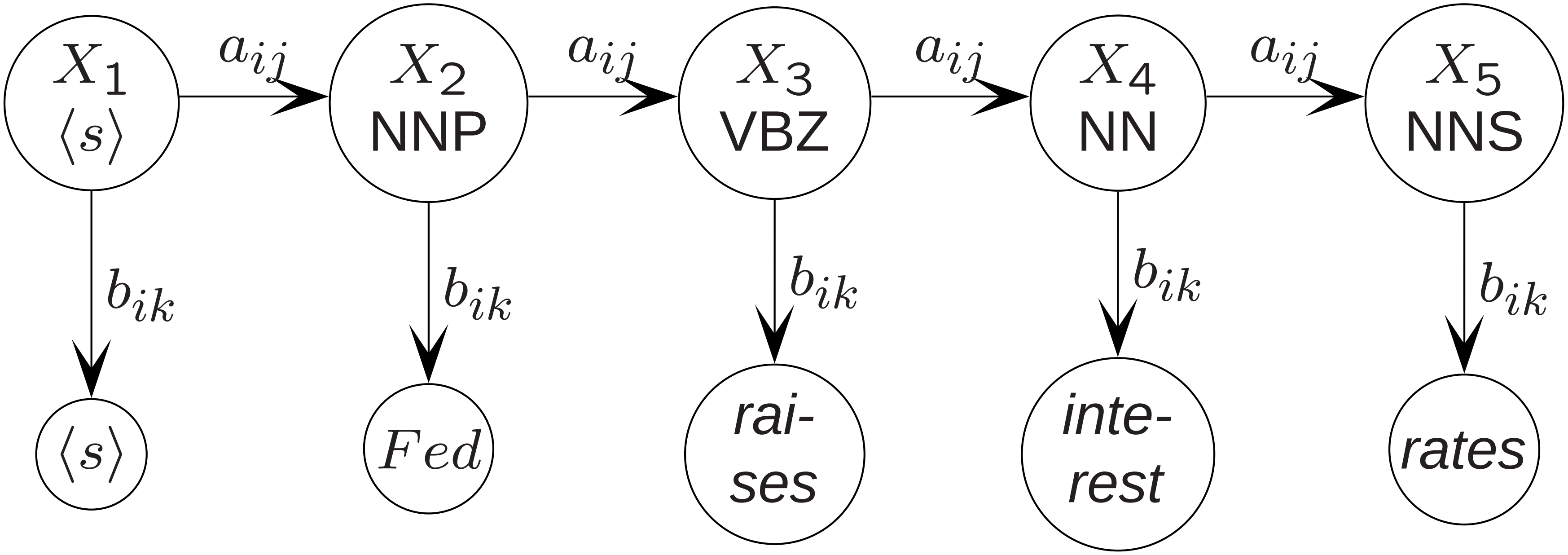
- Top row circles are unobserved hidden variables, which can have various values (states)
- Dependent only on previous state (Markov assumption)

Hidden Markov Models (HMMs)



- Bottom row circles are observed variables/output observations.
- Directly dependent only on hidden state above them

HMMs – POS example



- $a_{ij} = P(X_n = j | X_{n-1} = i)$
- $b_{ik} = P(O_n = k | X_n = i)$

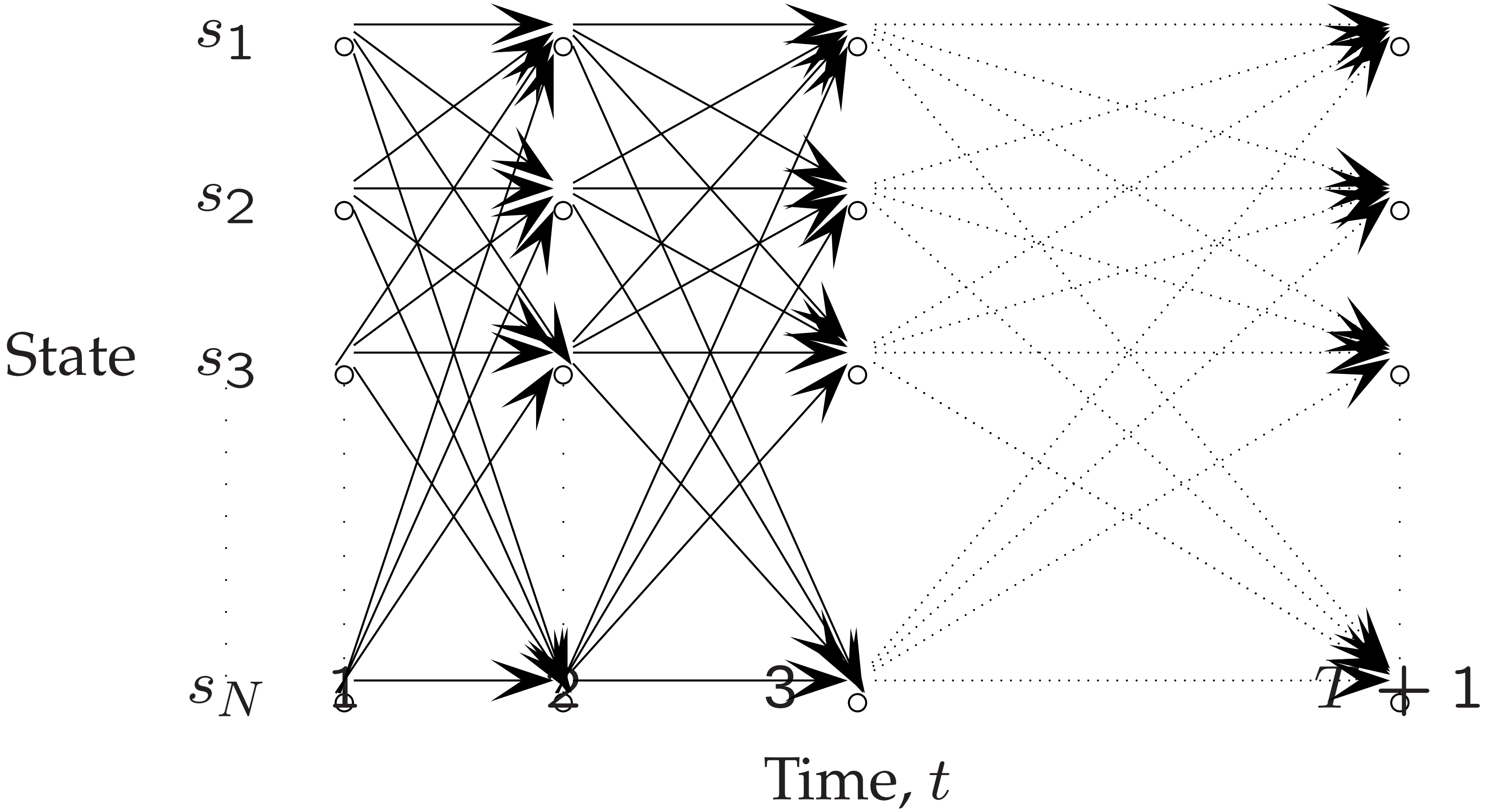
Dynamic Programming

- Efficient computation of this quantity: forward procedure
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward probability
  $\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}$
- Compute it recursively from beginning
- A version of the polytree algorithm for Bayes Net inference

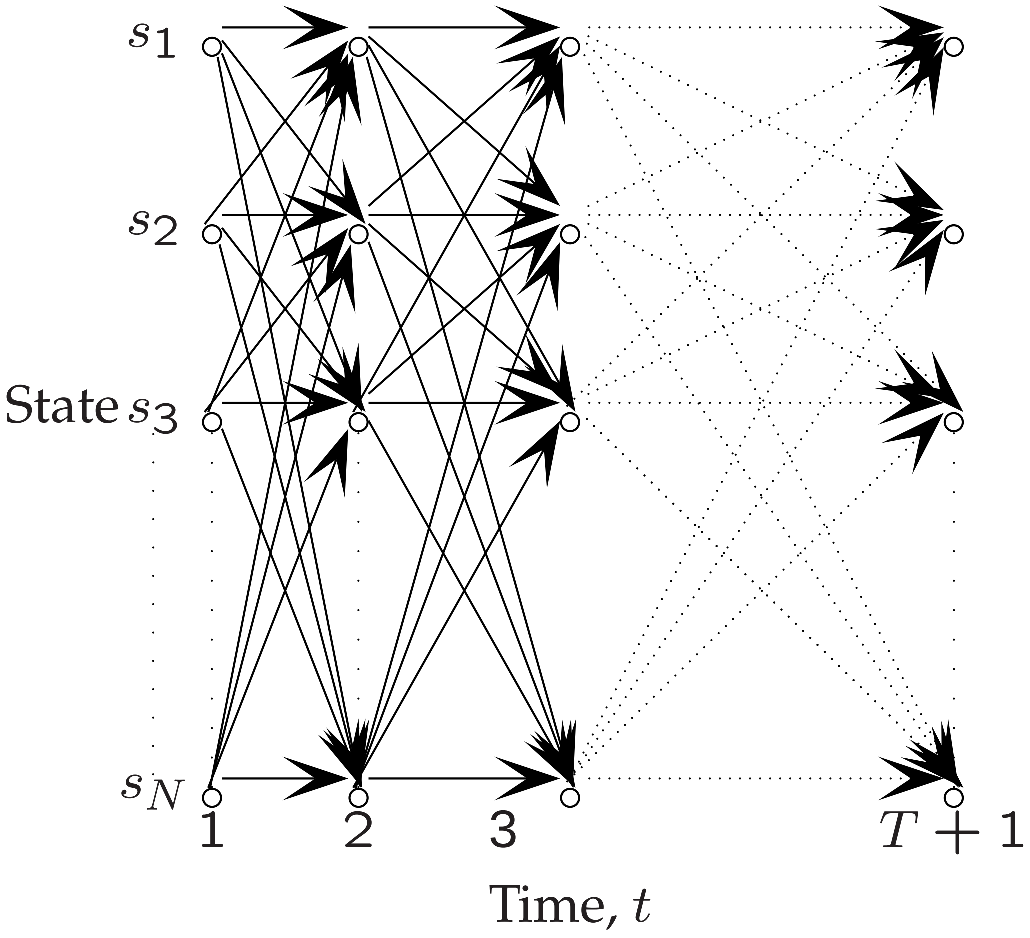
Dynamic Programming

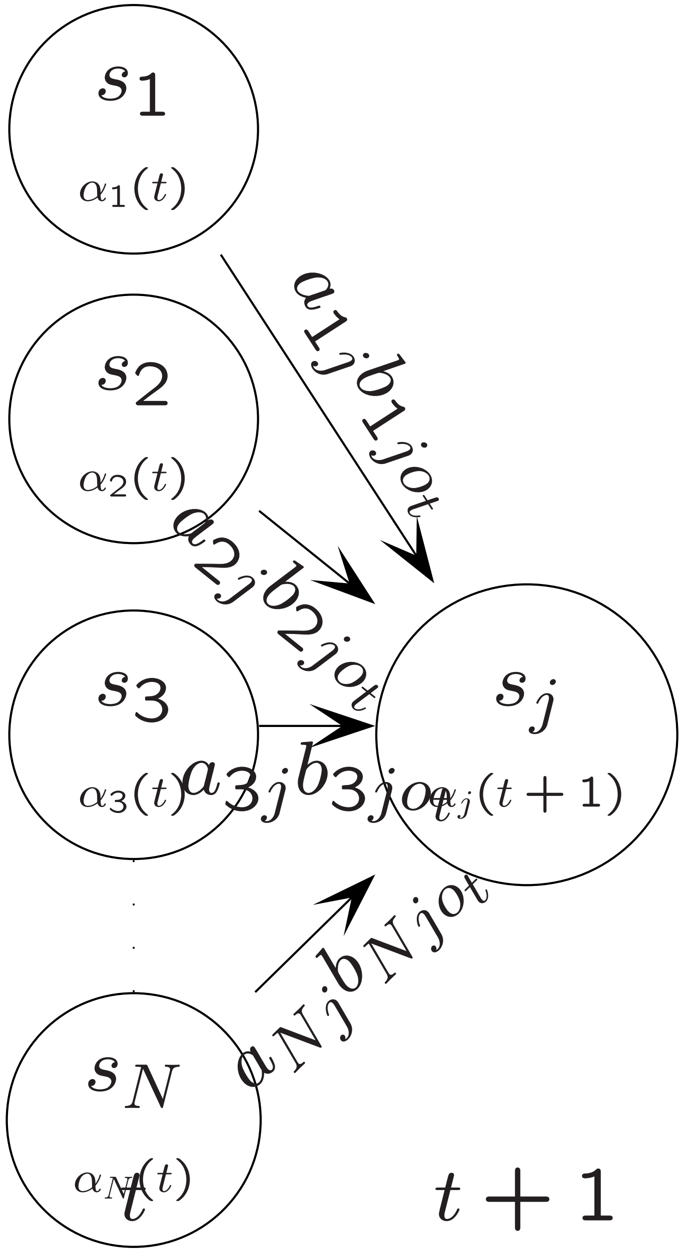
- The **forward procedure** efficiently computes this probability
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward prob. $\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}$
- Compute it recursively from beginning
- Similarly calculated backward probability $\beta_i(t)$ from end
- A version of the variable elimination algorithm for Bayes Net inference

Trellis algorithms



‘Trellis algorithms’




s_1
$\alpha_1(t)$
s_2
$\alpha_2(t)$
s_3
$\alpha_3(t)$
s_N
$\alpha_N(t)$
s_j
$\alpha_j(t+1)$
$a_{1j}b_{1jo_t}$
$a_{2j}b_{2jo_t}$
$a_{3j}b_{3jo_t}$
$a_{Nj}b_{Njo_t}$
t
$t+1$

Viterbi algorithm

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- Want: $\arg\max_X P(X|O,\mu)$
- Sufficient to maximize for a fixed O:

$$\arg\max_X P(X,O|\mu)$$

- This gives a maximum likelihood tag *sequence* for POS tagging

Viterbi algorithm

- Another dynamic programming algorithm. Essentially the same except you do a max instead of a summation, and record the path taken.
- δ variables record maximum probability to (t, i)
- ψ variables record a backpointer

$$\delta_{i+1}(t^j) = \max_{1 \le k \le T} [\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

$$\psi_{i+1}(t^j) = \arg\max_{1 \le k \le T} [\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

Where do HMM POS taggers make errors?

- Inconsistent labeling: is *chief* or *maximum* an NN or a JJ?
- Difficulties with unknown words (NN/NNP/NNPS/NNS)
- Systematic tag ambiguities, for which there are good structural cues:
  - *John* **has** *quite clearly* **made** *great progress . . .*
- Subtle semantic distinctions, for which there aren't good distributional syntactic cues, e.g., RP/IN/RB ambiguity, but where semantic annotation might help

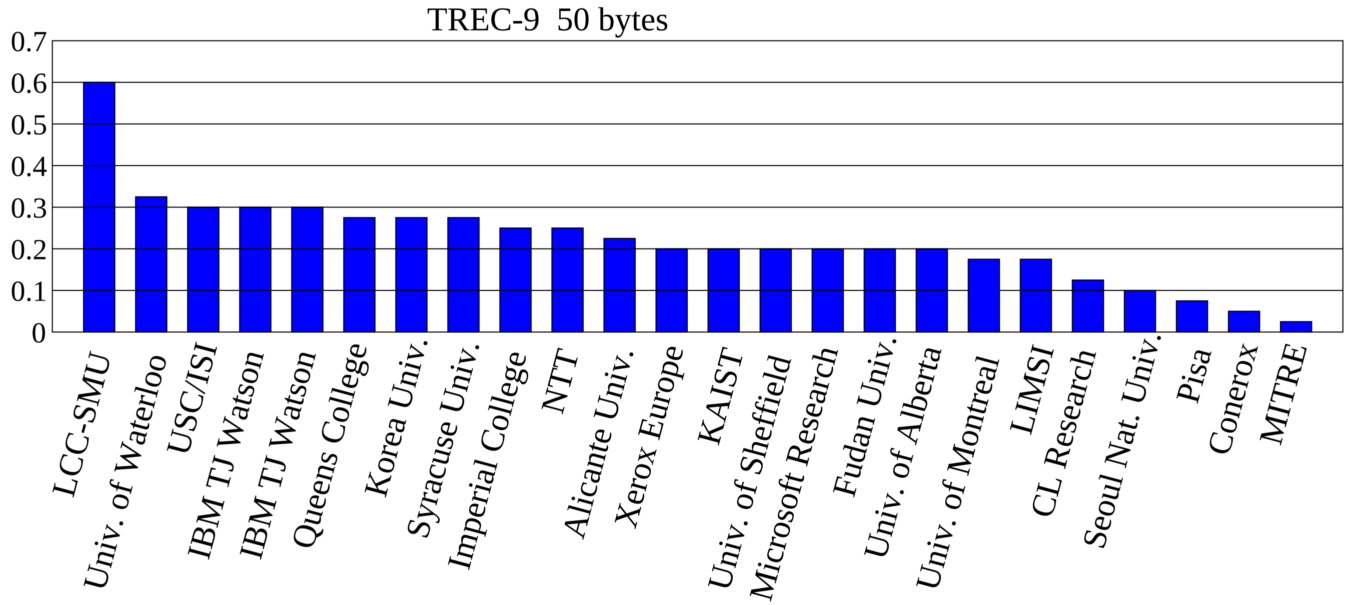
{Probabilistic|Stochastic} Context-Free Grammars (PCFGs)

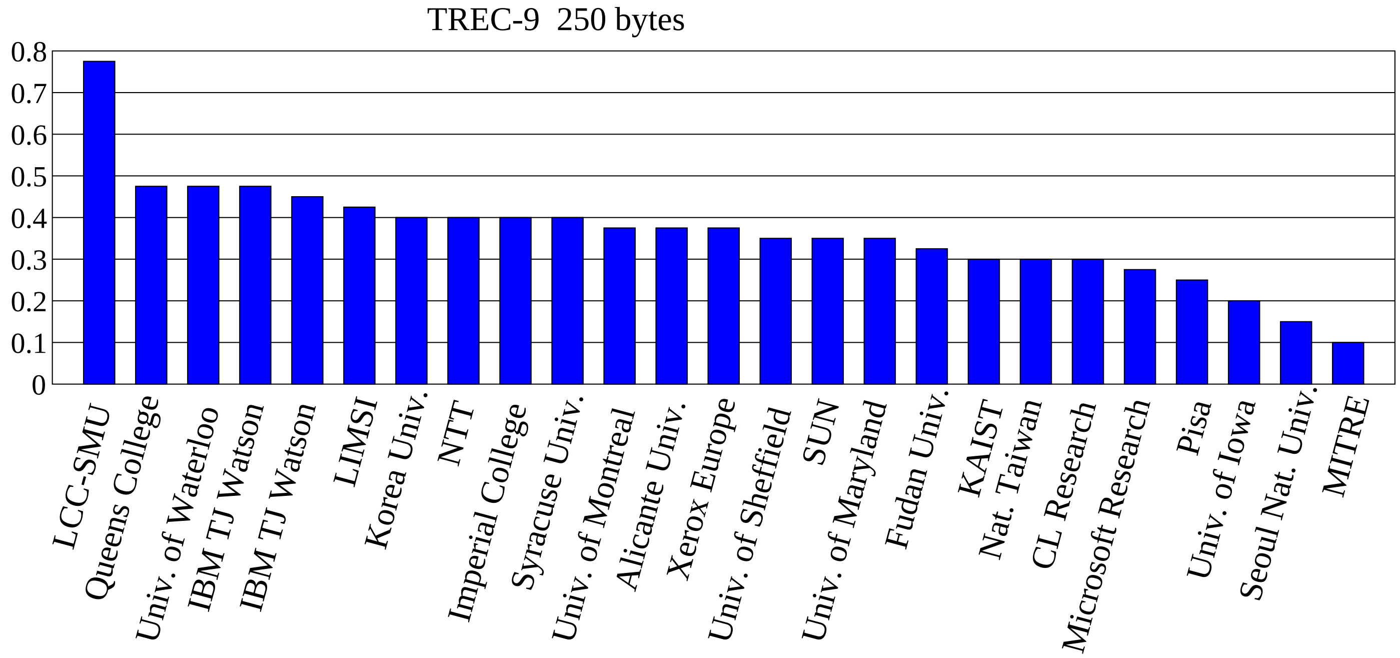
***FSNLP*, chapter 11**

Christopher Manning and Hinrich Schütze

Question answering from text

- TREC 8+ QA competition (1999–; it's ongoing): an idea originating from the IR community
- With massive collections of on-line documents, manual translation of textual information into knowledge bases covering large numbers of domains is impractical: We want to answer questions from textbases
- Evaluated output is 5 answers of 50/250 byte snippets of text drawn from a 3 GB text collection, and required to contain at least one concept of the semantic category of the expected answer type. (Until 2002. IR think: Suggests the use of named entity recognizers.)
- Get reciprocal points for highest correct answer.


TREC-9 50 bytes
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
LCC-SMU
Univ. of Waterloo
USC/ISI
IBM TJ Watson
IBM TJ Watson
Queens College
Korea Univ.
Syracuse Univ.
Imperial College
NTT
Alicante Univ.
Xerox Europe
KAIST
Univ. of Sheffield
Microsoft Research
Fudan Univ.
Univ. of Alberta
Univ. of Montreal
LIMSI
CL Research
Seoul Nat. Univ.
Pisa
Conerox
MITRE


TREC-9 250 bytes
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
LCC-SMU
Queens College
Univ. of Waterloo
IBM TJ Watson
IBM TJ Watson
LIMSI
Korea Univ.
NTT
Imperial College
Syracuse Univ.
Univ. of Montreal
Alicante Univ.
Xerox Europe
Univ. of Sheffield
SUN
Univ. of Maryland
Fudan Univ.
KAIST
Nat. Taiwan
CL Research
Microsoft Research
Pisa
Univ. of Iowa
Seoul Nat. Univ.
MITRE

Pasca and Harabagiu (2001) demonstrates the value of sophisticated NLP processing

- Good IR is needed: paragraph retrieval based on SMART
- Large taxonomy of question types and expected answer types is crucial
- Parsing: A statistical parser (modeled on Collins 1997) is used to parse questions, relevant text for answers, *and* WordNet to build a knowledge base for reasoning
- Controlled query expansion loops (morphological, lexical synonyms, and semantic relations) are all important in retrieving the correct answer.
- Answer ranking by ML method based on this information surpasses IR-style empirical methods.

Question Answering Example (1)

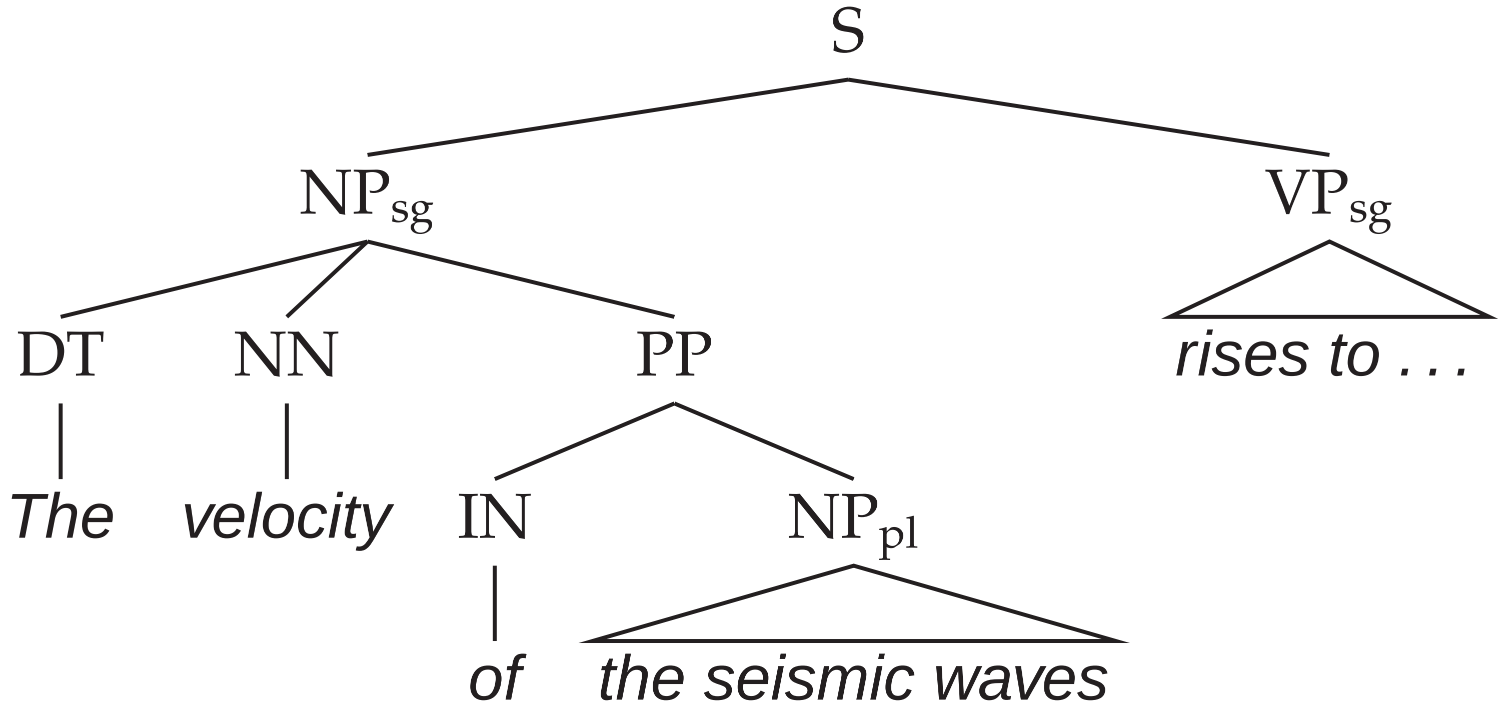
- Q261: What company sells most greetings cards ?
- sells(ORGANIZATION, cards(greeting), most)
- “Hallmark remains the largest maker of greeting cards”
  maker(ORGANIZATION(Hallmark), cards(greeting), largest)
- Need an entailment between *producing*, or *making* and *selling* goods
- Derived from WordNet, since synset *make, produce, create* has the genus **manufacture**, defined in the gloss of its nominalization as (*for*) *sale*
- Also, need *most* $\approx$ *largest*
- Therefore the semantic form of question Q261 and its illustrated answer are similar

Question Answering Example (2)

- How hot does the inside of an active volcano get ?
- get(TEMPERATURE, inside(volcano(active)))
- “lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit”
- fragments(lava, TEMPERATURE(degrees(300)),
  belched(out, mountain))
- – volcano ISA mountain
  - lava ISPARTOF volcano □ lava inside volcano
  - fragments of lava HAVEPROPERTIESOF lava
- The needed semantic information is available in WordNet definitions, and was successfully translated into a form that can be used for rough ‘proofs’

Why we need recursive phrase structure

- *The velocity of the seismic waves rises to . . .*
- Kupiec (1992): HMM tagger goes awry: *waves* → verb



PCFGs

A PCFG G consists of the usual parts of a CFG

- A set of terminals, $\{w^k\}, k = 1, \dots, V$
- A set of nonterminals, $\{N^i\}, i = 1, \dots, n$
- A designated start symbol, N^1
- A set of rules, $\{N^i \rightarrow \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)

and

- A corresponding set of probabilities on rules such that:

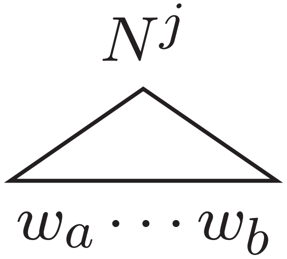
$$\forall i \quad \sum_j P(N^i \rightarrow \zeta^j) = 1$$

PCFG notation

Sentence: sequence of words $w_1 \cdots w_m$

w_{ab}: the subsequence $w_a \cdots w_b$

N^i_{ab}: nonterminal N^i dominates $w_a \cdots w_b$



$N^i \overset{*}{\Longrightarrow} \zeta$: Repeated derivation from N^i gives ζ.

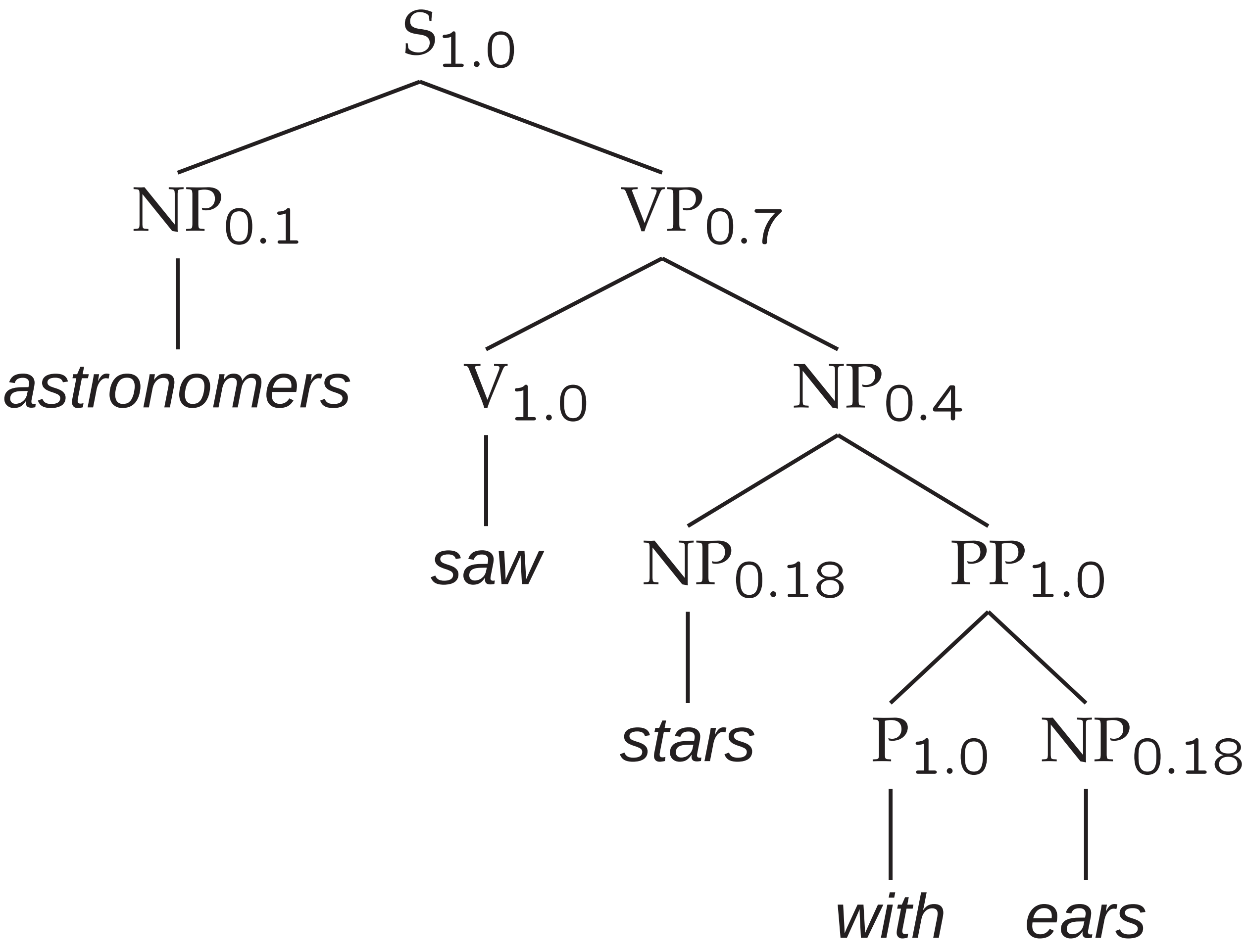
PCFG probability of a string

$$\begin{aligned} P(w_{1n}) &= \sum_t P(w_{1n}, t) \quad t \text{ a parse of } w_{1n} \\ &= \sum_{\{t:\mathsf{yield(t)=w_{1n}}\}} P(t) \end{aligned}$$

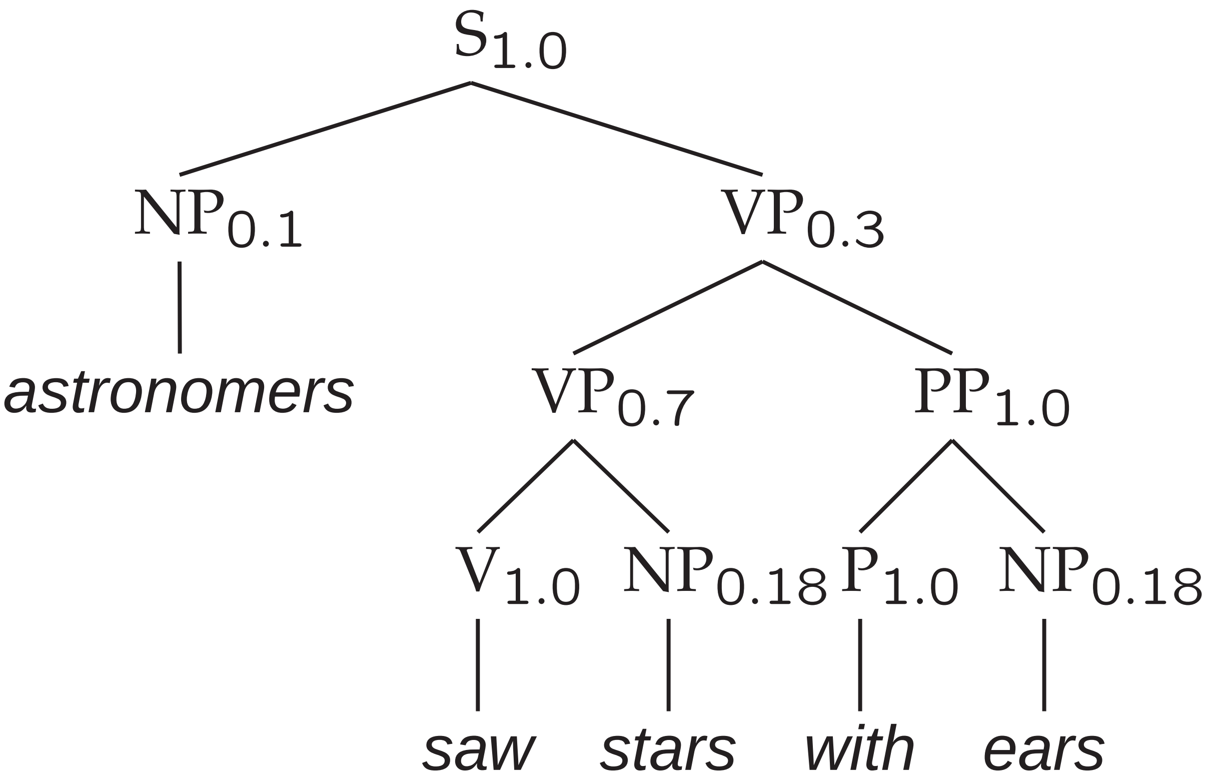
A simple PCFG (in CNF)

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

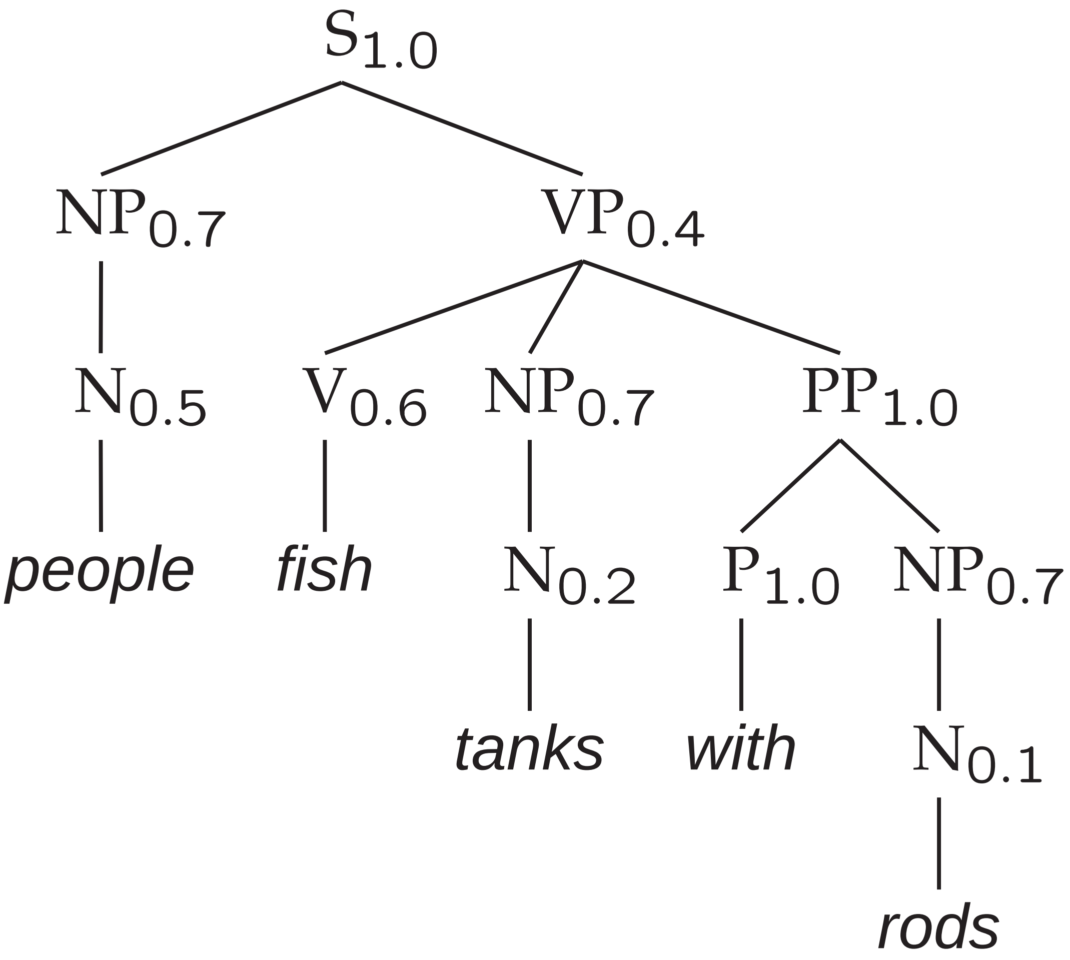
t_1:


$S_{1.0}$
$NP_{0.1}$
$VP_{0.7}$
astronomers
$V_{1.0}$
$NP_{0.4}$
saw
$NP_{0.18}$
$PP_{1.0}$
stars
$P_{1.0}$
$NP_{0.18}$
with
ears

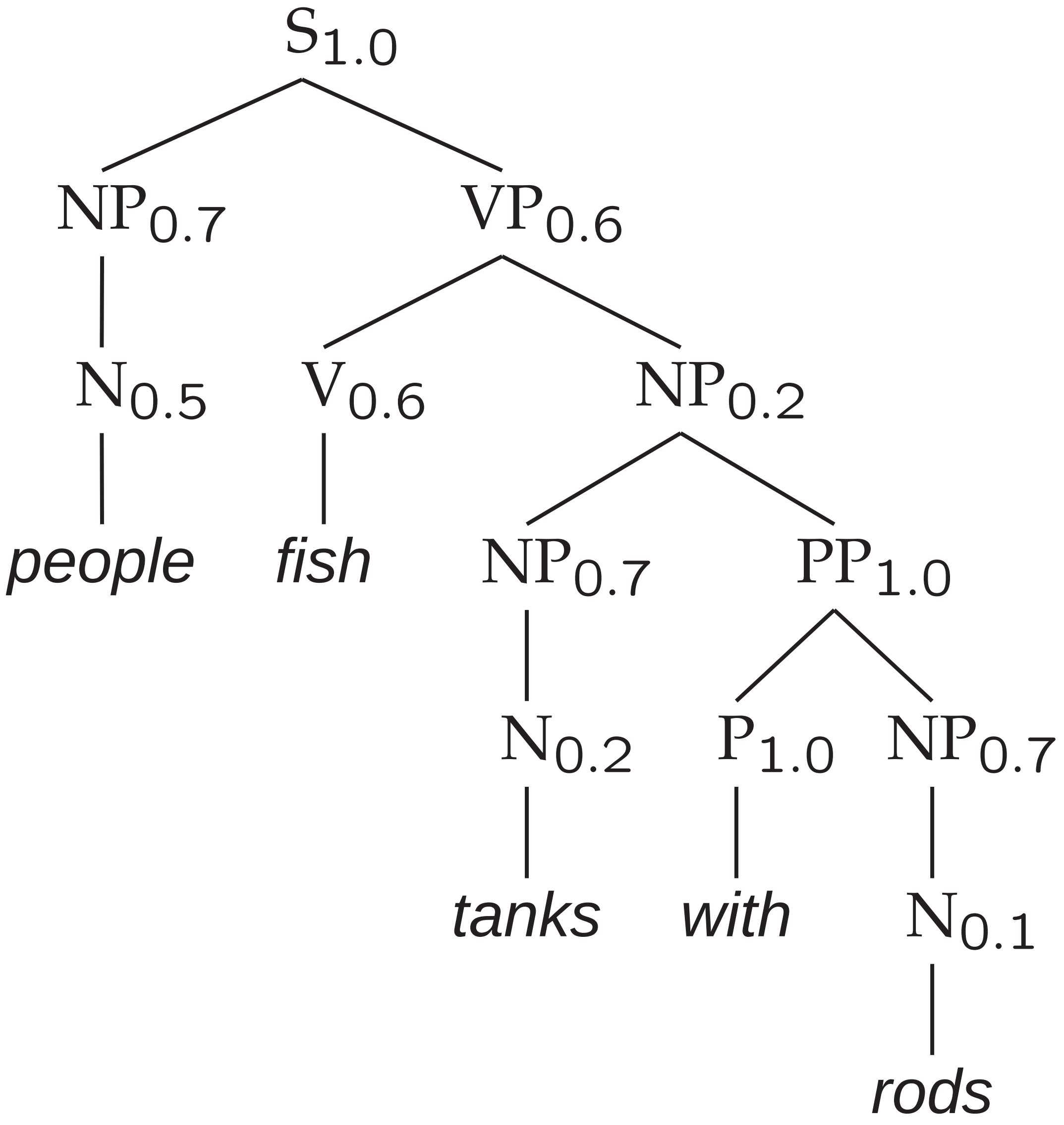
t_2:


$S_{1.0}$
$NP_{0.1}$
$VP_{0.3}$
astronomers
$VP_{0.7}$
$PP_{1.0}$
$V_{1.0}$
$NP_{0.18}$
$P_{1.0}$
$NP_{0.18}$
saw
stars
with
ears

t_1:


$S_{1.0}$
$NP_{0.7}$
$VP_{0.4}$
$N_{0.5}$
$V_{0.6}$
$NP_{0.7}$
$PP_{1.0}$
people
fish
$N_{0.2}$
$P_{1.0}$
$NP_{0.7}$
tanks
with
$N_{0.1}$
rods

t_2:


$S_{1.0}$
$NP_{0.7}$
$VP_{0.6}$
$N_{0.5}$
$V_{0.6}$
$NP_{0.2}$
people
fish
$NP_{0.7}$
$PP_{1.0}$
$N_{0.2}$
$P_{1.0}$
$NP_{0.7}$
tanks
with
$N_{0.1}$
rods

The two parse trees' probabilities and the sentence probability

$$\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) = 0.0015876
\end{aligned}$$

Attachment ambiguities: A key parsing decision

- The main problem in parsing is working out how to 'attach' various kinds of constituents – PPs, adverbial or participial phrases, coordinations, and so on
- Prepositional phrase attachment
  - *I saw the man with a telescope*
- What does *with a telescope* modify?
  - The verb *saw*?
  - The noun *man*?
- Is the problem 'AI-complete'? Yes, but . . .

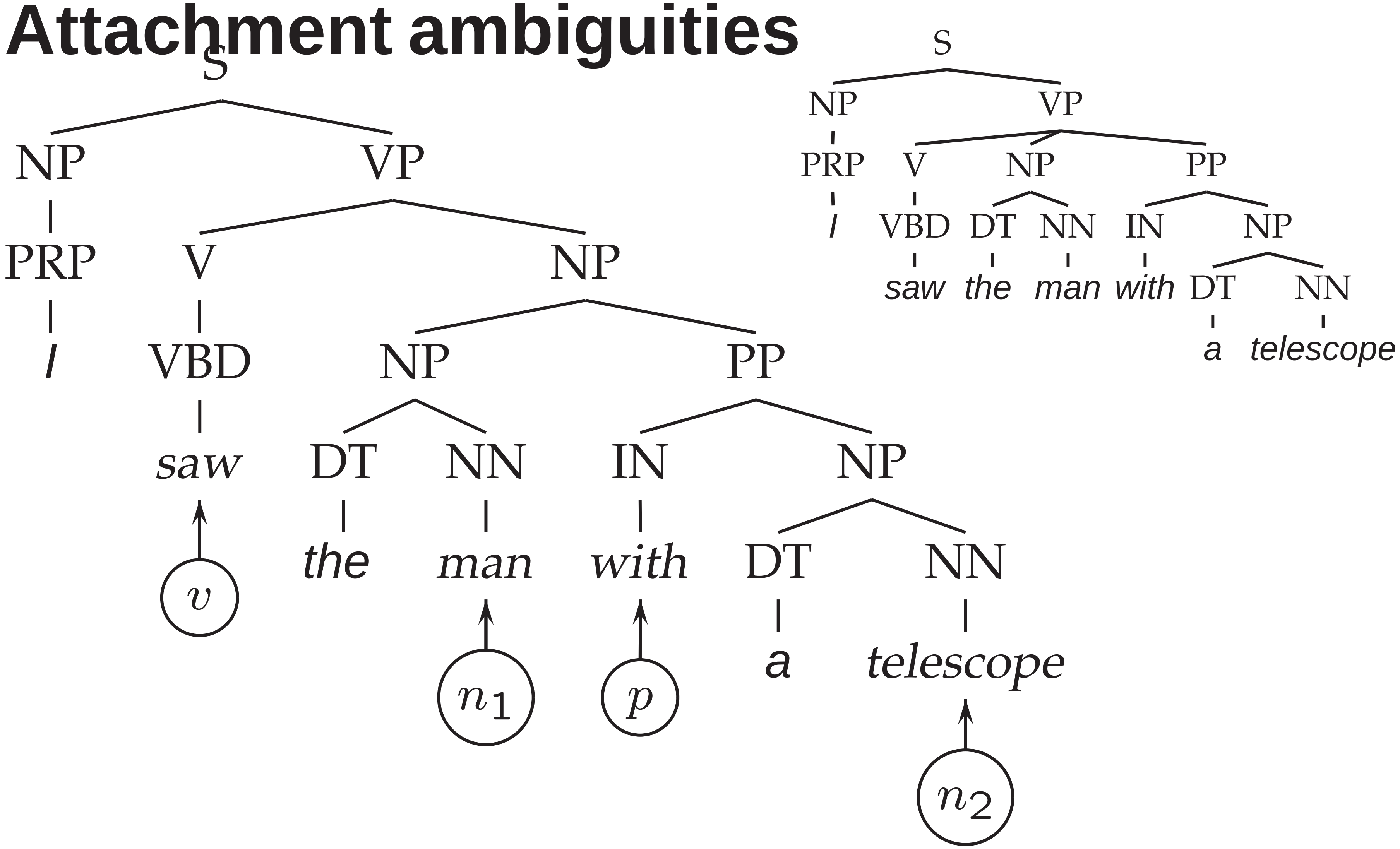
Attachment ambiguities (2)

- Proposed simple structural factors
  - Right association (Kimball 1973) = ‘low’ or ‘near’ attachment = ‘late closure’ (of NP) [NP → NP PP]
  - Minimal attachment (Frazier 1978) [depends on grammar] = ‘high’ or ‘distant’ attachment = ‘early closure’ (of NP) [VP → V NP PP]
- Such simple structural factors dominated in early psycholinguistics, and are still widely invoked.
- In the V NP PP context, right attachment gets it right in 55–67% of cases.
- But that means it gets it wrong in 33–45% of cases

Importance of lexical factors

- Words are good predictors (or even inducers) of attachment (even absent understanding):
  - The children ate the cake with a spoon.
  - The children ate the cake with frosting.
  - Moscow sent more than 100,000 soldiers into Afghanistan
  - Sydney Water breached an agreement with NSW Health
- Ford et al. (1982):
  - Ordering is jointly determined by strengths of alternative lexical forms, alternative syntactic rewrite rules, and the sequence of hypotheses in parsing

Attachment ambiguities



Assumptions of PCFGs

1. Place invariance (like time invariance in HMM):

$$\forall k \quad P(N^j_{k(k+c)} \rightarrow \zeta)\text{is the same}$$

2. Context-free:

$$P(N^j_{kl} \rightarrow \zeta|\text{words outside } w_k \dots w_l) = P(N^j_{kl} \rightarrow \zeta)$$

3. Ancestor-free:

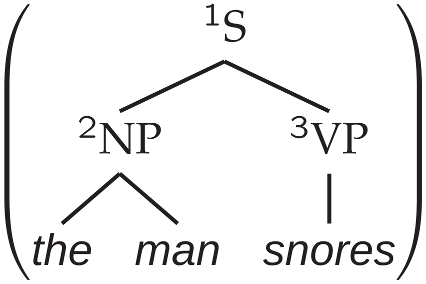
$$P(N^j_{kl} \rightarrow \zeta|\text{ancestor nodes of } N^j_{kl}) = P(N^j_{kl} \rightarrow \zeta)$$

The sufficient statistics of a PCFG are thus simply counts of how often different local tree configurations occurred (= counts of which grammar rules were applied).

Let the upper left index in ${}^iN^j$ be an arbitrary identifying index for a particular token of a nonterminal.

Then,

$P\Bigg($



$\Bigg)$

$$
\begin{aligned}
&= P({}^1S_{13} \to {}^2NP_{12}\,{}^3VP_{33}, {}^2NP_{12} \to the_1\; man_2, {}^3VP_{33}\\
&= \ldots\\
&= P(S \to NP\; VP)P(NP \to the\; man)P(VP \to snores)
\end{aligned}
$$

Some features of PCFGs

Reasons to use a PCFG, and some idea of their limitations:

- Partial solution for grammar ambiguity: a PCFG gives some idea of the plausibility of a sentence.
- But, in the simple case, not a very good idea, as independence assumptions are two strong (e.g., not lexicalized).
- Gives a probabilistic language model for English.
- In the simple case, a PCFG is a worse language model for English than a trigram model.
- Better for grammar induction (Gold 1967 vs. Horning 1969)
- Robustness. (Admit everything with low probability.)

Some features of PCFGs

- A PCFG encodes certain biases, e.g., that smaller trees are normally more probable.
- One can hope to combine the strengths of a PCFG and a trigram model.

We'll look at simple PCFGs first. They have certain inadequacies, but we'll see that most of the state-of-the-art probabilistic parsers are fundamentally PCFG models, just with various enrichments to the grammar

Improper (inconsistent) distributions

- S → *rhubarb* $P = \frac{1}{3}$

  S → S S $P = \frac{2}{3}$

- *rhubarb* $\frac{1}{3}$

  rhubarb rhubarb $\frac{2}{3} \times \frac{1}{3} \times \frac{1}{3} = \frac{2}{27}$

  rhubarb rhubarb rhubarb $\left(\frac{2}{3}\right)^2 \times \left(\frac{1}{3}\right)^3 \times 2 = \frac{8}{243}$

  $\ldots$

- $P(\mathcal{L}) = \frac{1}{3} + \frac{2}{27} + \frac{8}{243} + \ldots = \frac{1}{2}$
- Improper/inconsistent distribution
- Not a problem if you estimate from parsed treebank: Chi and Geman (1998).

Questions for PCFGs

Just as for HMMs, there are three basic questions we wish to answer:

- Language modeling: $P(w_{1m}|G)$
- Parsing: $\arg\max_t P(t|w_{1m}, G)$
- Learning algorithm: Find G such that $P(w_{1m}|G)$ is maximized.

Chomsky Normal Form grammars

We'll do the case of Chomsky Normal Form grammars, which only have rules of the form:

$$\begin{aligned} N^i &\rightarrow N^j N^k \\ N^i &\rightarrow w^j \end{aligned}$$

Any CFG can be represented by a weakly equivalent CFG in Chomsky Normal Form. It's straightforward to generalize the algorithm (recall chart parsing).

PCFG parameters

We'll do the case of Chomsky Normal Form grammars, which only have rules of the form:

$$\begin{aligned} N^i &\rightarrow N^j N^k \\ N^i &\rightarrow w^j \end{aligned}$$

The parameters of a CNF PCFG are:

$P(N^j \rightarrow N^r N^s \vert G)$	A n^3 matrix of parameters
$P(N^j \rightarrow w^k \vert G)$	An nt matrix of parameters

For $j = 1, \ldots, n$,

$$\sum_{r,s} P(N^j \rightarrow N^r N^s) + \sum_k P(N^j \rightarrow w^k) = 1$$

Probabilistic Regular Grammar:

$N^i \rightarrow w^j N^k$

$N^i \rightarrow w^j$

Start state, N^1

HMM:

$$\sum_{w_{1n}} P(w_{1n}) = 1 \quad \forall n$$

whereas in a PCFG or a PRG:

$$\sum_{w \in L} P(w) = 1$$

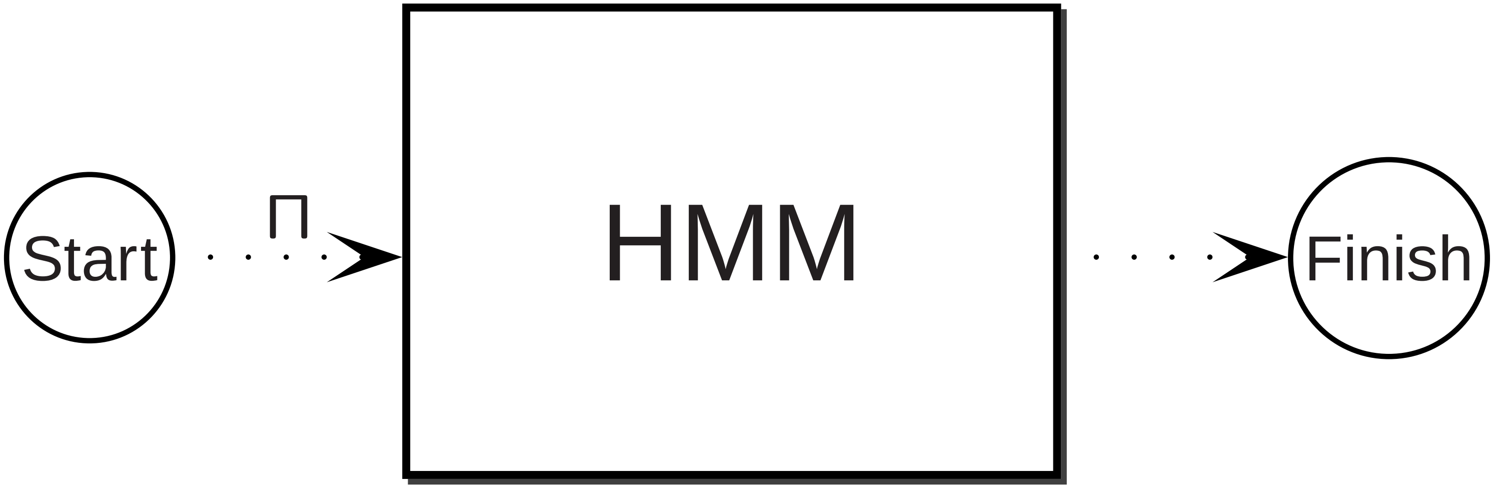
Probabilistic Regular Grammar

Consider:

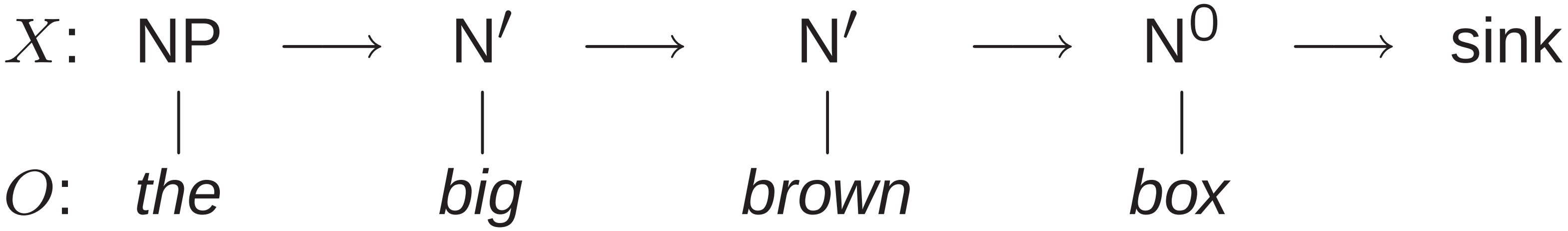
$$P(\textit{John decided to bake a})$$

High probability in HMM, low probability in a PRG or a PCFG. Implement via sink (end) state.

A PRG

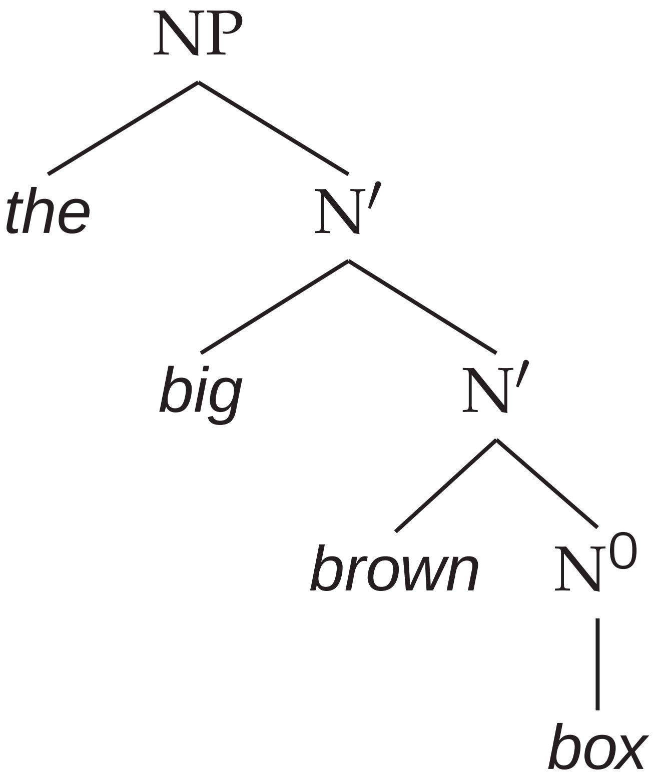


Comparison of HMMs (PRGs) and PCFGs



X: NP ⟶ N′ ⟶ N′ ⟶ N^0 ⟶ sink

O: *the* *big* *brown* *box*



Inside and outside probabilities

This suggests: whereas for an HMM we have:

Forwards = $\alpha_i(t) = P(w_{1(t-1)}, X_t = i)$

Backwards = $\beta_i(t) = P(w_{tT}|X_t = i)$

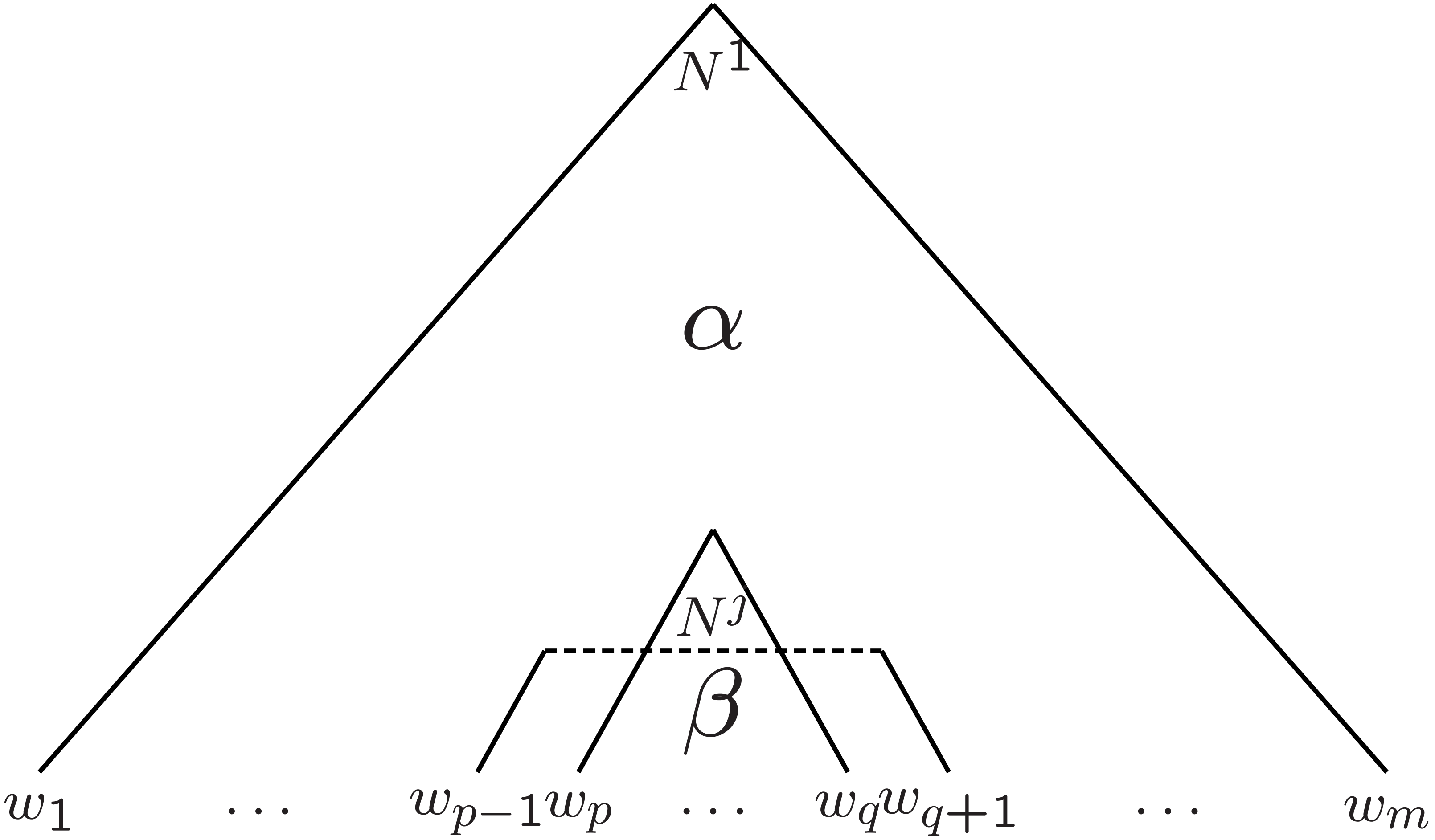
for a PCFG we make use of Inside and Outside probabilities, defined as follows:

Outside = $\alpha_j(p, q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$

Inside = $\beta_j(p, q) = P(w_{pq}|N^j_{pq}, G)$

A slight generalization of dynamic Bayes Nets covers PCFG inference by the inside-outside algorithm (and-or tree of conjunctive daughters disjunctively chosen)

Inside and outside probabilities in PCFGs.



Probability of a string

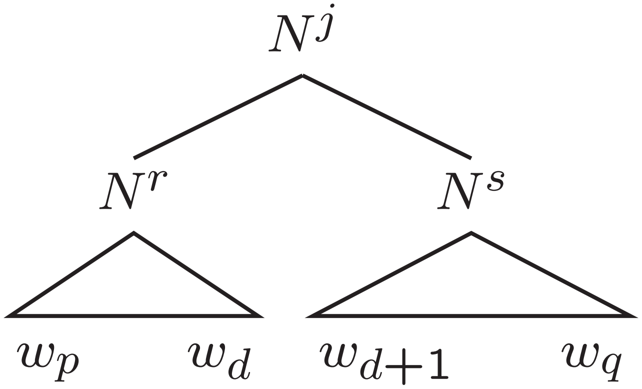
Inside probability

$$\begin{aligned} P(w_{1m}|G) &= P(N^1 \Rightarrow w_{1m}|G) \\ &= P(w_{1m}, N^1_{1m}, G) \ = \beta_1(1,m) \end{aligned}$$

Base case: We want to find $\beta_j(k,k)$ (the probability of a rule $N^j \to w_k$):

$$\begin{aligned} \beta_j(k,k) &= P(w_k|N^j_{kk}, G) \\ &= P(N^j \to w_k|G) \end{aligned}$$

Induction: We want to find $\beta_j(p, q)$, for $p < q$. As this is the inductive step using a Chomsky Normal Form grammar, the first rule must be of the form $N^j \rightarrow N^r \quad N^s$, so we can proceed by induction, dividing the string in two in various places and summing the result:



These inside probabilities can be calculated bottom up.

For all j,

$$\begin{aligned}
\beta_j(p,q) &= P(w_{pq}|N^j_{pq},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(w_{pd},N^r_{pd},w_{(d+1)q},N^s_{(d+1)q}|N^j_{pq},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{pd},N^s_{(d+1)q}|N^j_{pq},G) \\
&\qquad P(w_{pd}|N^j_{pq},N^r_{pd},N^s_{(d+1)q},G) \\
&\qquad P(w_{(d+1)q}|N^j_{pq},N^r_{pd},N^s_{(d+1)q},w_{pd},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{pd},N^s_{(d+1)q}|N^j_{pq},G) \\
&\qquad P(w_{pd}|N^r_{pd},G)P(w_{(d+1)q}|N^s_{(d+1)q},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s)\beta_r(p,d)\beta_s(d+1,q)
\end{aligned}$$

Calculation of inside probabilities (CKY algorithm)

	1	2	3	4	5
1	$\beta_{\mathrm{NP}} =$ 0.1		$\beta_{\mathrm{S}} =$ 0.0126		$\beta_{\mathrm{S}} =$ 0.0015876
2		$\beta_{\mathrm{NP}} =$ 0.04 $\beta_{\mathrm{V}} =$ 1.0	$\beta_{\mathrm{VP}} =$ 0.126		$\beta_{\mathrm{VP}} =$ 0.015876
3			$\beta_{\mathrm{NP}} =$ 0.18		$\beta_{\mathrm{NP}} =$ 0.01296
4				$\beta_{\mathrm{P}} =$ 1.0	$\beta_{\mathrm{PP}} =$ 0.18
5					$\beta_{\mathrm{NP}} =$ 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Outside probabilities

Probability of a string: For any k, $1 \leq k \leq m$,

$$\begin{aligned}
P(w_{1m}|G) &= \sum_j P(w_{1(k-1)}, w_k, w_{(k+1)m}, N^j_{kk}|G) \\
&= \sum_j P(w_{1(k-1)}, N^j_{kk}, w_{(k+1)m}|G) \\
&\qquad \times P(w_k|w_{1(k-1)}, N^j_{kk}, w_{(k+1)n}, G) \\
&= \sum_j \alpha_j(k,k) P(N^j \rightarrow w_k)
\end{aligned}$$

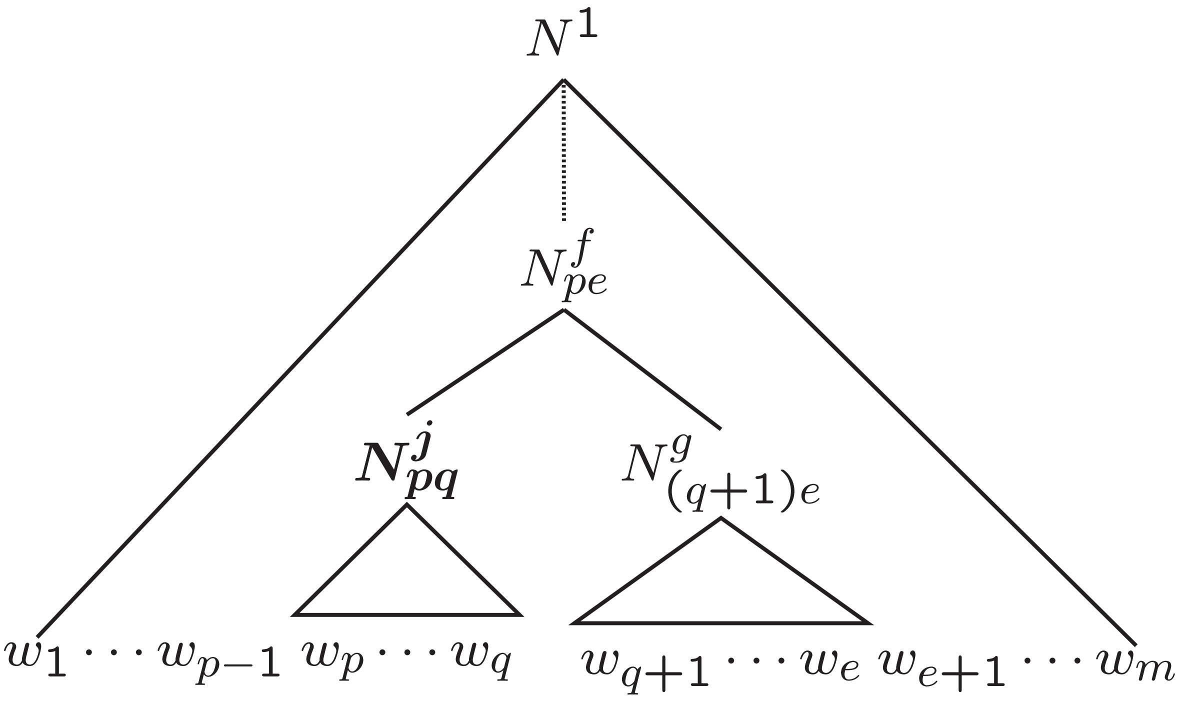
Inductive (DP) calculation: One calculates the outside probabilities top down (after determining the inside probabilities).

Outside probabilities

Base Case:

$$\alpha_1(1, m) = 1$$
$$\alpha_j(1, m) = 0, \text{ for } j \neq 1$$

Inductive Case:



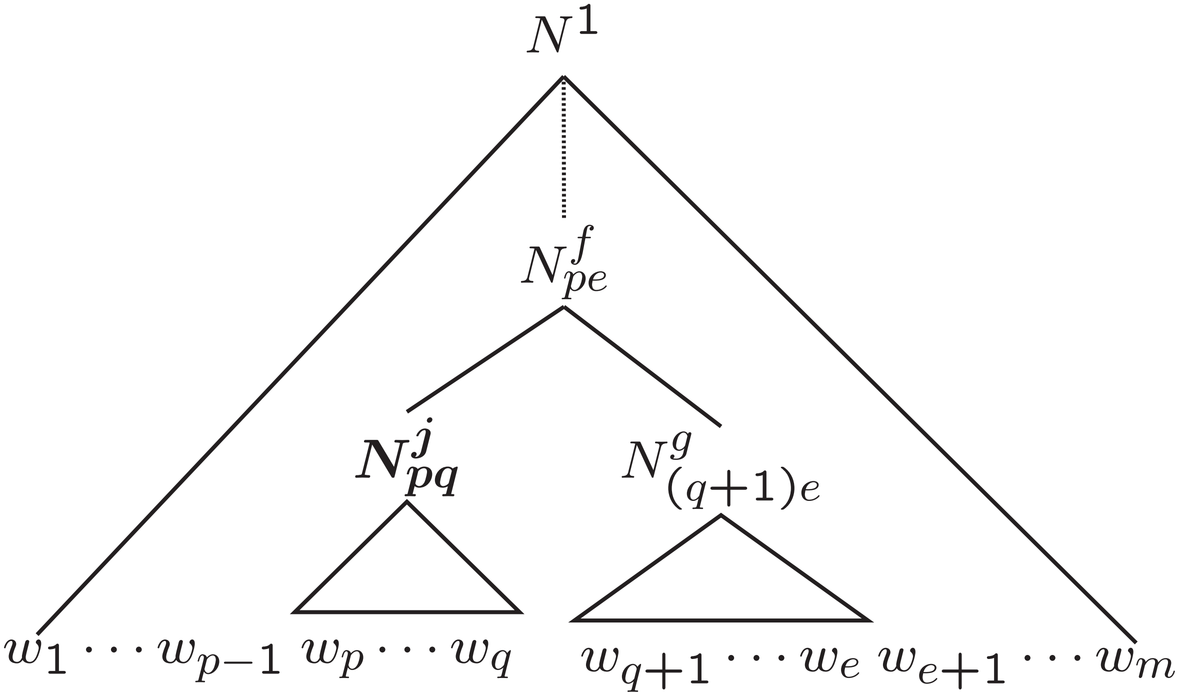
N^1
N^f_{pe}
$\boldsymbol{N^j_{pq}}$
$N^g_{(q+1)e}$
$w_1 \cdots w_{p-1}$ $w_p \cdots w_q$ $w_{q+1} \cdots w_e$ $w_{e+1} \cdots w_m$

Outside probabilities

Base Case:

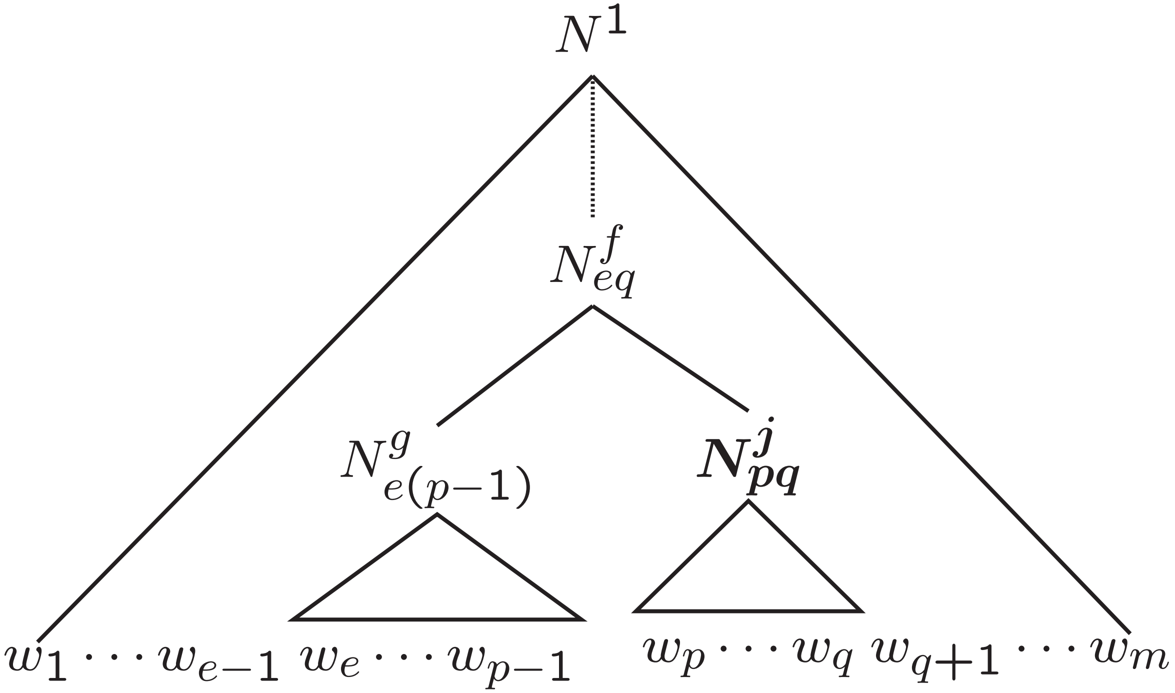
$$\alpha_1(1,m) = 1$$
$$\alpha_j(1,m) = 0\text{, for } j \neq 1$$

Inductive Case: it's either a left or right branch – we will some over both possibilities and calculate using outside *and* inside probabilities


N^1
N^f_{pe}
$\boldsymbol{N^j_{pq}}$
$N^g_{(q+1)e}$
$w_1 \cdots w_{p-1}$ $w_p \cdots w_q$ $w_{q+1} \cdots w_e$ $w_{e+1} \cdots w_m$

Outside probabilities – inductive case

A node N^j_{pq} might be the left or right branch of the parent node. We sum over both possibilities.


N^1
N^f_{eq}
$N^g_{e(p-1)}$
$\boldsymbol{N^j_{pq}}$
$w_1 \cdots w_{e-1}$ $w_e \cdots w_{p-1}$ $w_p \cdots w_q$ $w_{q+1} \cdots w_m$

Inductive Case:

$$\begin{aligned}
\alpha_j(p,q) &= \Big[\sum_{f,g}\sum_{e=q+1}^{m} P(w_{1(p-1)}, w_{(q+1)m}, N^f_{pe}, N^j_{pq}, N^g_{(q+1)e})\Big] \\
&\quad + \Big[\sum_{f,g}\sum_{e=1}^{p-1} P(w_{1(p-1)}, w_{(q+1)m}, N^f_{eq}, N^g_{e(p-1)}, N^j_{pq})\Big] \\
&= \Big[\sum_{f,g}\sum_{e=q+1}^{m} P(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}) P(N^j_{pq}, N^g_{(q+1)e} | N^f_{pe}) \\
&\quad \times P(w_{(q+1)e} | N^g_{(q+1)e})\Big] + \Big[\sum_{f,g}\sum_{e=1}^{p-1} P(w_{1(e-1)}, w_{(q+1)m}, N^f_{eq}) \\
&\quad \times P(N^g_{e(p-1)}, N^j_{pq} | N^f_{eq}) P(w_{e(p-1)} | N^g_{e(p-1})\Big] \\
&= \Big[\sum_{f,g}\sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \rightarrow N^j\ N^g) \beta_g(q+1,e)\Big] \\
&\quad + \Big[\sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g\ N^j) \beta_g(e,p-1)\Big]
\end{aligned}$$

Overall probability of a node existing

As with a HMM, we can form a product of the inside and outside probabilities. This time:

$$\begin{aligned}
&\alpha_j(p,q)\beta_j(p,q) \\
&= P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)P(w_{pq}|N^j_{pq}, G) \\
&= P(w_{1m}, N^j_{pq}|G)
\end{aligned}$$

Therefore,

$$p(w_{1m}, N_{pq}|G) = \sum_j \alpha_j(p,q)\beta_j(p,q)$$

Just in the cases of the root node and the preterminals, we know there will always be some such constituent.

Finding the most likely parse (Viterbi algorithm)

Like inside algorithm, but find maximum rather than sum

Record which rule gave this maximum

$\delta_i(p,q) =$ the highest inside probability parse of a subtree N^i_{pq}

1. Initialization: $\delta_i(p,p) = P(N^i \to w_p)$
2. Induction

$$\delta_i(p,q) = \max_{\substack{1 \le j,k \le n \\ p \le r < q}} P(N^i \to N^j \ N^k)\delta_j(p,r)\delta_k(r+1,q)$$

3. Store backtrace

$$\psi_i(p,q) = \arg\max_{(j,k,r)} P(N^i \to N^j \ N^k)\delta_j(p,r)\delta_k(r+1,q)$$

4. From start symbol N^1, most likely parse t is:

   t begins with $\psi_1(1,m)$. $\qquad P(\hat{t}) = \delta_1(1,m)$

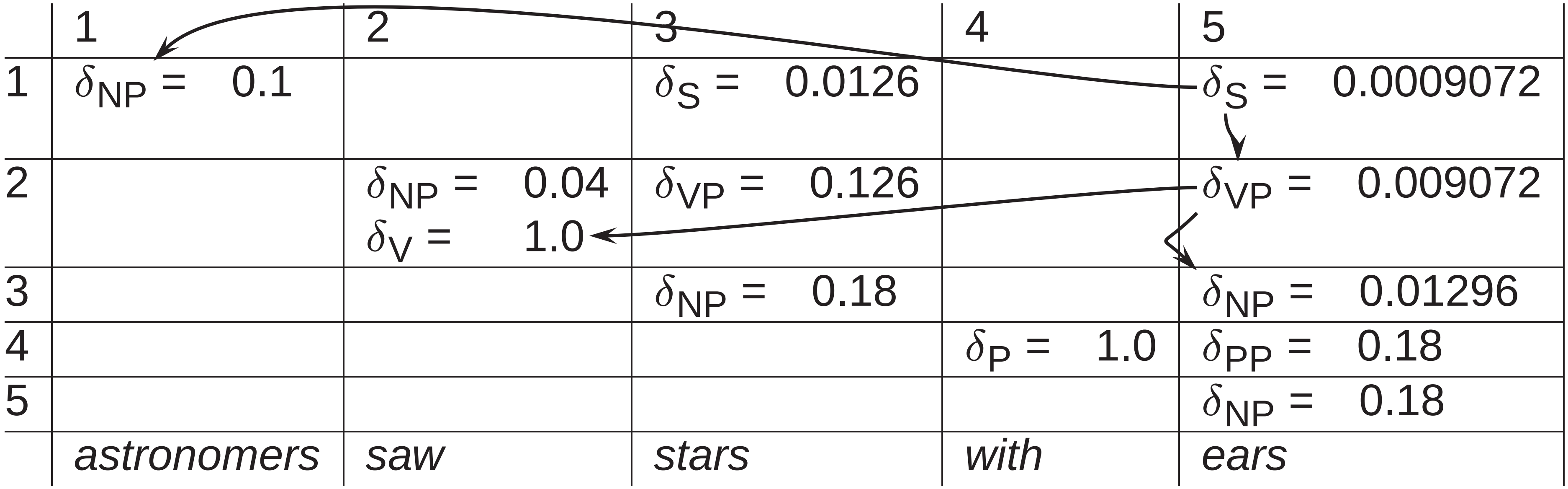
(Probabilistic) CKY algorithm

```
function CKY(words, grammar) returns most probable parse/probability
    score = new double[#(words)+1][#(words)+1][#(nonterms)];
    back = new Pair[#(words)+1][#(words)+1][#(nonterms)];
    for i = 0; i $<$ #(words); i++
       for A in nonterms
          if A $\rightarrow$ words[i] in grammar
             score[i][i+1][A] = P(A $\rightarrow$ words[i])
       // handle unaries
       boolean added = true
       while added
          added = false
          for A, B in nonterms
             if score[i][i+1][B] > 0 && A $\rightarrow$ B in grammar
                prob = P(A $\rightarrow$ B) $\times$ score[i][i+1][B]
                if (prob $>$ score[i][i+1][A])
                   score[i][i+1][A] = prob
                   back[i][i+1][A] = B
                   added = true
```

(Probabilistic) CKY algorithm [continued]

```
for span = 2 to #(words)
    for begin = 0 to #words $-$ span
        end = begin + span
        for split = begin + 1 to end $-$ 1
            for A, B, C in nonterms
                prob = score[begin][split][B] * score[split][end][C] * P(A $\rightarrow$ B C)
                if (prob $>$ score[begin][end][A]
                    score[begin][end][A] = prob
                    back[begin][end][A] = new Triple(split,B,C)
        // handle unaries
        boolean added = true
        while added
            added = false
            for A, B in nonterms
                prob = P(A $\rightarrow$ B) $\times$ score[begin][end][B]
                if (prob $>$ score[begin][end][A])
                    score[begin][end][A] = prob
                    back[begin][end][A] = B
                    added = true
return buildTree(score, back)
```

Calculation of Viterbi probabilities (CKY algorithm)



	1	2	3	4	5
1	δ_{NP} = 0.1		δ_{S} = 0.0126		δ_{S} = 0.0009072
2		δ_{NP} = 0.04 δ_{V} = 1.0	δ_{VP} = 0.126		δ_{VP} = 0.009072
3			δ_{NP} = 0.18		δ_{NP} = 0.01296
4				δ_{P} = 1.0	δ_{PP} = 0.18
5					δ_{NP} = 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Training a PCFG

We construct an EM training algorithm, as for HMMs. We would like to calculate how often each rule is used:

$$\hat{P}(N^j \to \zeta) = \frac{C(N^j \to \zeta)}{\sum_\gamma C(N^j \to \gamma)}$$

Have data $\Rightarrow$ count; else work iteratively from expectations of current model.
Consider:

$$\begin{aligned} \alpha_j(p,q)\beta_j(p,q) &= P(N^1 \overset{*}{\Longrightarrow} w_{1m}, N^j \overset{*}{\Longrightarrow} w_{pq}|G) \\ &= P(N^1 \overset{*}{\Longrightarrow} w_{1m}|G)P(N^j \overset{*}{\Longrightarrow} w_{pq}|N^1 \overset{*}{\Longrightarrow} w_{1m}, G) \end{aligned}$$

We have already solved how to calculate $P(N^1 \Rightarrow w_{1m})$; let us call this probability π. Then:

$$P(N^j \overset{*}{\Longrightarrow} w_{pq}|N^1 \overset{*}{\Longrightarrow} w_{1m}, G) = \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

and

$$E(N^j \text{ is used in the derivation}) = \sum_{p=1}^{m}\sum_{q=p}^{m} \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

In the case where we are not dealing with a preterminal, we substitute the inductive definition of β, and $\forall r, s, p > q$:

$$P(N^j \rightarrow N^r\ N^s \Rightarrow w_{pq} | N^1 \Rightarrow w_{1n}, G) =$$

$$\frac{\sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r\ N^s) \beta_r(p,d) \beta_s(d+1,q)}{\pi}$$

Therefore the expectation is:

$E(N^j \rightarrow N^r\ N^s, N^j \text{ used})$

$$\frac{\sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r\ N^s) \beta_r(p,d) \beta_s(d+1,q)}{\pi}$$

Now for the maximization step, we want:

$$P(N^j \rightarrow N^r\ N^s) = \frac{E(N^j \rightarrow N^r\ N^s, N^j \text{ used})}{E(N^j \text{ used})}$$

Therefore, the reestimation formula, $\widehat{P}(N^j \rightarrow N^r\ N^s)$ is the quotient:

$\widehat{P}(N^j \rightarrow N^r\ N^s) =$

$$\frac{\sum_{p=1}^{m-1}\sum_{q=p+1}^{m}\sum_{d=p}^{q-1}\alpha_j(p,q)P(N^j \rightarrow N^r\ N^s)\beta_r(p,d)\beta_s(d+1,q)}{\sum_{p=1}^{m}\sum_{q=1}^{m}\alpha_j(p,q)\beta_j(p,q)}$$

Similarly,

$E(N^j \rightarrow w^k | N^1 \Rightarrow w_{1m}, G) =$

$$\frac{\sum_{h=1}^{m}\alpha_j(h,h)P(N^j \rightarrow w_h, w_h = w^k)}{\pi}$$

Therefore,

$$\widehat{P}(N^j \rightarrow w^k) = \frac{\sum_{h=1}^{m}\alpha_j(h,h)P(N^j \rightarrow w_h, w_h = w^k)}{\sum_{p=1}^{m}\sum_{q=1}^{m}\alpha_j(p,q)\beta_j(p,q)}$$

Inside-Outside algorithm: repeat this process until the estimated probability change is small.

Multiple training instances: if we have training sentences $W = (W_1, \dots W_\omega)$, with $W_i = (w_1, \dots, w_{m_i})$ and we let u and v bet the common subterms from before:

$$u_i(p,q,j,r,s) =$$

$$\frac{\sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)}{P(N^1 \Rightarrow W_i|G)}$$

and

$$v_i(p,q,j) = \frac{\alpha_j(p,q)\beta_j(p,q)}{P(N^1 \Rightarrow W_i|G)}$$

Assuming the observations are independent, we can sum contributions:

$$\hat{P}(N^j \rightarrow N^r\ N^s) = \frac{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i-1} \sum_{q=p+1}^{m_i} u_i(p,q,j,r,s)}{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i} \sum_{q=p}^{m_i} v_i(p,q,j)}$$

and

$$\hat{P}(N^j \to w^k) = \frac{\sum_{i=1}^{\omega} \sum_{\{h: w_h = w^k\}} v_i(h, h, j)}{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i} \sum_{q=p}^{m_i} v_i(p, q, j)}$$

Variant slides

Probabilistic context-free grammars (PCFGs)

A PCFG G consists of:

- A set of terminals, $\{w^k\}$
- A set of nonterminals, $\{N^i\}$, with a start symbol, N^1
- A set of rules, $\{N^i \rightarrow \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)
- A set of probabilities on rules such that:

$$\forall i \quad \sum_j P(N^i \rightarrow \zeta^j) = 1$$

- A generalization of HMMs to tree structures (branching processes)
- A similar DP algorithm to the Viterbi algorithm is used for finding the most probable parse

PCFGs

- Like a regular CFG but put probability on each rule
- Probabilities are also completely context-free, depending just on parent node
- Just as for HMMs, there are three basic questions we wish to answer:
  - $P(w_{1m}|G)$
  - $\arg\max_t P(t|w_{1m}, G)$
  - Learning algorithm. Find G such that $P(w_{1m}|G)$ is maximized.

Learning PCFGs (1)

- We would like to calculate how often each rule is used:

$$\widehat{P}(N^j \to \zeta) = \frac{C(N^j \to \zeta)}{\sum_\gamma C(N^j \to \gamma)}$$

- If we have labeled data, we count and find out
- Relative frequency again gives maximum likelihood probability estimates
- This is the motivation for building *Treebanks* of hand-parsed sentences

Learning PCFGs (2): the Inside-Outside algorithm (Baker 1979)

- Otherwise we work iteratively from expectations of current model.
- We construct an EM training algorithm, as for HMMs
- For each sentence, at each iteration, we work out expectation of how often each rule is used using inside and outside probabilities
- We assume sentences are independent and sum expectations over parses of each
- We re-estimate rules based on these ‘counts’

The EM algorithm

- Estimates the values of the hidden parameters of a model.
- If we have seen data X; we can estimate $P(X|\,\mathrm{p}(\Theta))$, the prob. of X according to some model p with parameters Θ.
- We want to find the model which maximizes the likelihood of the data
- This point is a maximum in the parameter space $\Rightarrow$ the probability surface is flat there.
- So for each model parameter θ_i, we want to set $\frac{\partial}{\partial\theta_i} \log P(\ldots) = 0$ and solve for the θ_i.

The EM algorithm

- These parameters have to be adjusted subject to stochastic constraints (constrained optimization)
- In general, one can't do these differentiations analytically.
- So we attempt to find the maximum using the (iterative, hill-climbing) EM algorithm.
- It's roughly like gradient-descent, but specialized for learning likelihood functions

EM algorithm

- Initalize all parameters (randomly or intelligently)
- E-step: We work out the expectation of the hidden variables, given the current parameters for the model
- M-step: (Assuming these expectations are right), we calculate the maximum likelihood estimates for the parameters
- Repeat until convergence

(Dempster, Laird, and Rubin 1977). Guaranteed to not diminish the likelihood of the data according to the model. Feasible for a large range of models (where E and M step can be computed).

Problems with the Inside-Outside algorithm

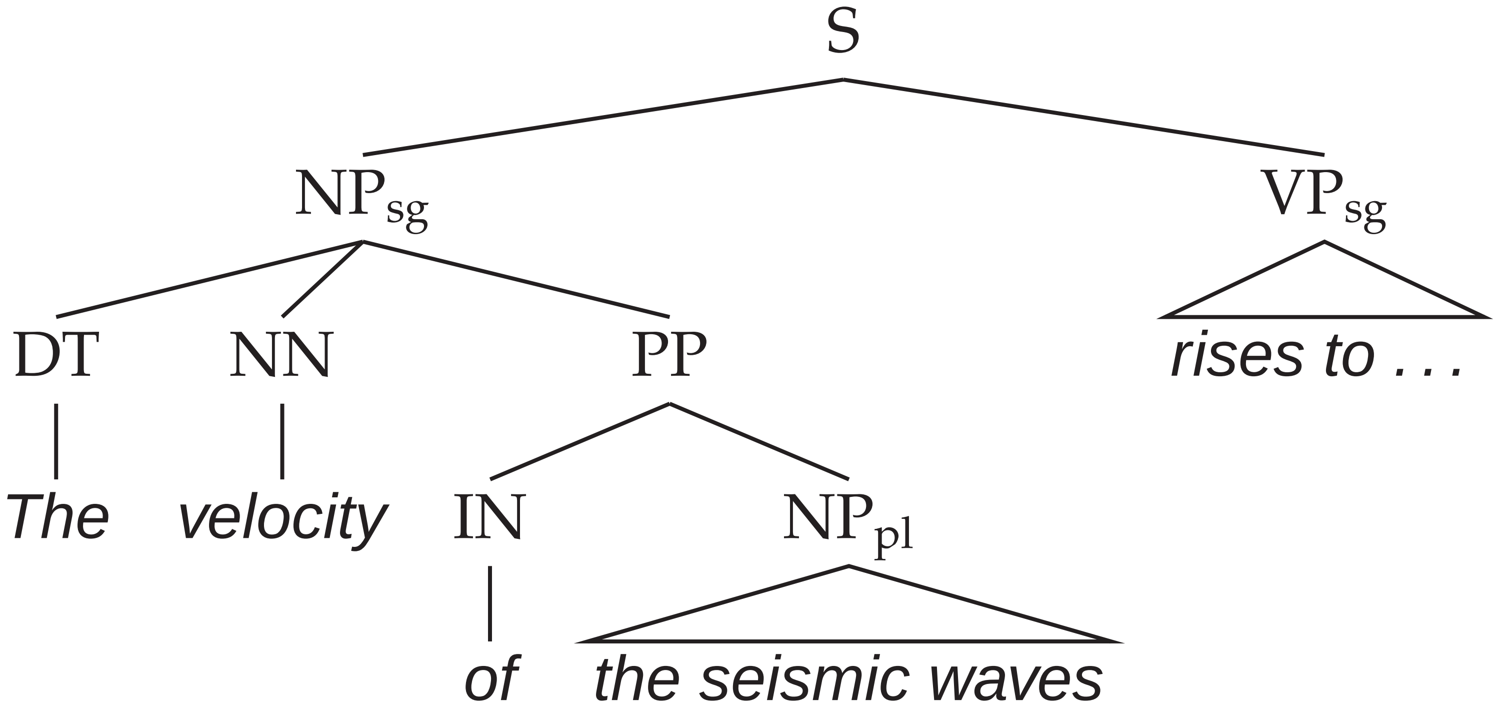
1. Slow. Each iteration is $O(m^3n^3)$, where $m = \sum_{i=1}^{\omega} m_i$, and n is the number of nonterminals in the grammar.
2. Local maxima are much more of a problem. Charniak (1993) reports that on each of 300 trials a different local maximum was found. Use simulated annealing? Restrict rules by initializing some parameters to zero? Or HMM initialization?

Problems with the Inside-Outside algorithm

3. Lari and Young (1990) suggest that you need many more nonterminals available than are theoretically necessary to get good grammar learning (about a threefold increase?). This compounds the first problem.
4. There is no guarantee that the nonterminals that the algorithm learns will have any satisfactory resemblance to the kinds of nonterminals normally motivated in linguistic analysis (NP, VP, etc.).

Why we need recursive phrase structure

- *The velocity of the seismic waves rises to . . .*
- Kupiec (1992): HMM tagger goes awry: *waves* → verb



- Or you can use dependency grammar representations

DP: Inside and outside probabilities

Whereas for an HMM we cache:

Forwards = $\alpha_i(t) = P(w_{1(t-1)}, X_t = i)$

Backwards = $\beta_i(t) = P(w_{tT} | X_t = i)$

for a PCFG we make use of Inside and Outside probabilities, defined as follows:

Outside = $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G)$

Inside = $\beta_j(p,q) = P(w_{pq} | N^j_{pq}, G)$

A slight generalization of Dynamic Bayes Nets covers probabilistic context free grammars and inference by the inside-outside algorithm (and-or tree of conjunctive daughters disjunctively chosen)

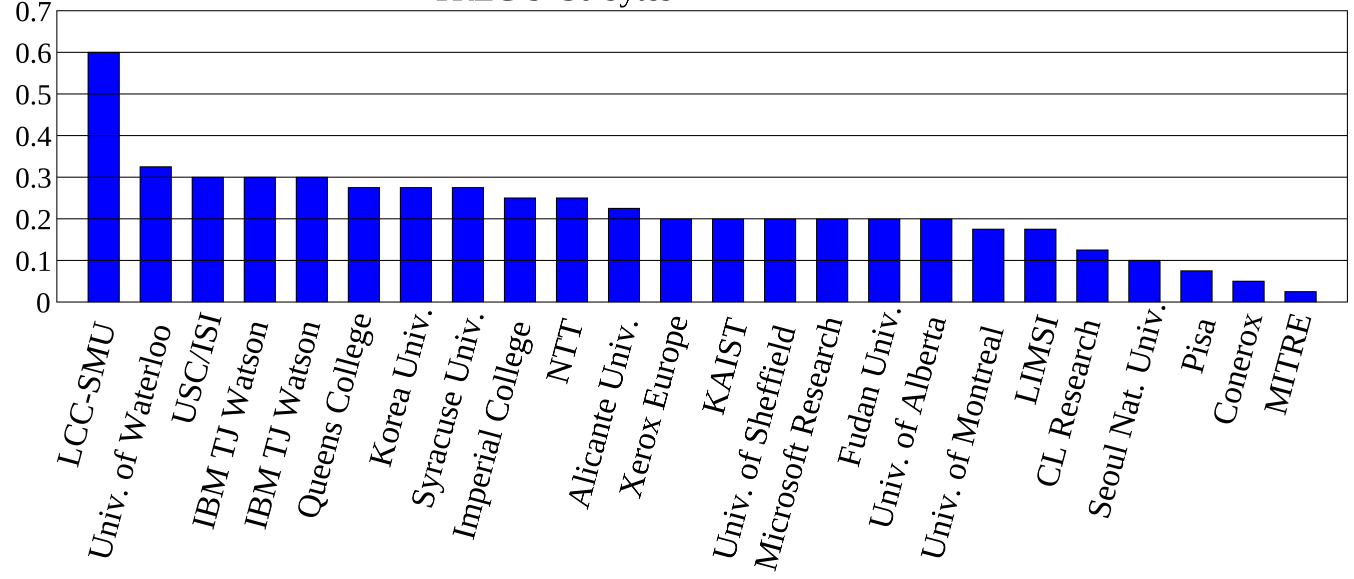
2. Complex NLP Case Study: Question Answering

- Task is a long pipeline of many subtasks
- Task seems AI complete
- NLP offers compelling performance advantages over IR-style techniques
- But still lots of room for improvement

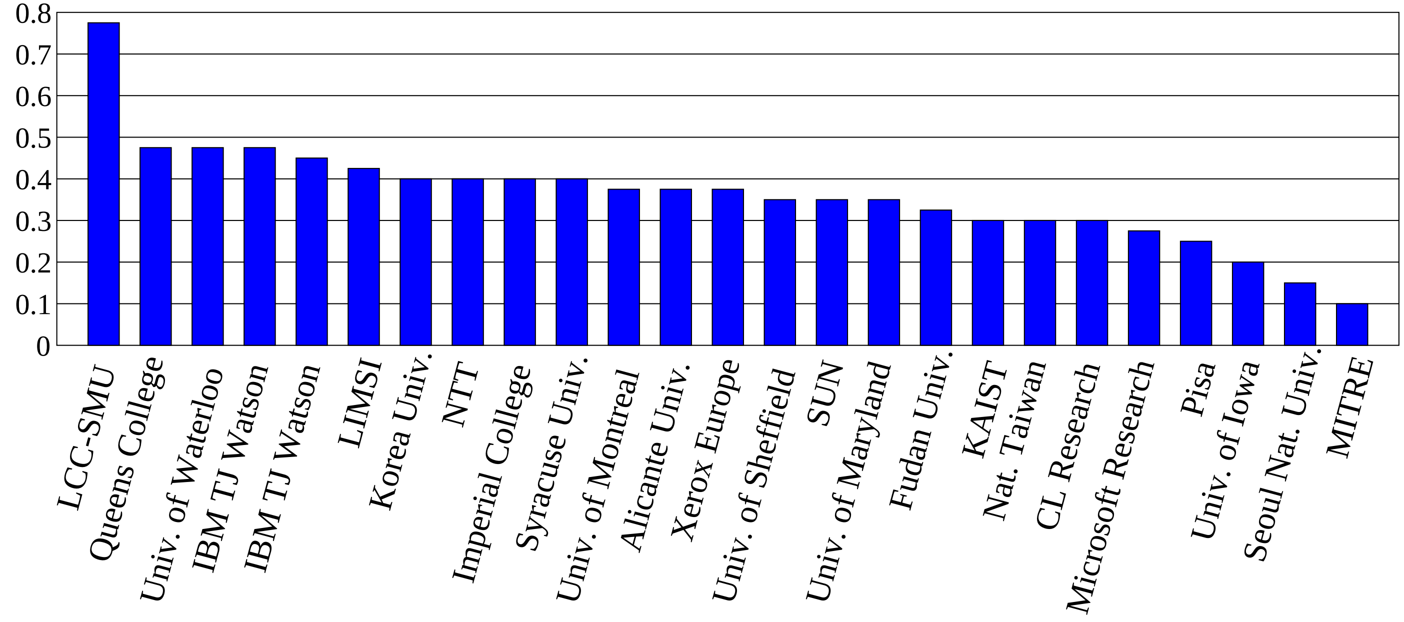
Question answering from text

- TREC 8/9 QA competition: an idea originating from the IR community
- With massive collections of on-line documents, manual translation of textual information into knowledge bases covering large numbers of domains is impractical: We want to answer questions from textbases
- Evaluated output is 5 answers of 50/250 byte snippets of text drawn from a 3 GB text collection, and required to contain at least one concept of the semantic category of the expected answer type. (IR think. Suggests the use of named entity recognizers.)
- Get reciprocal points for highest correct answer.

TREC-9 50 bytes



TREC-9 250 bytes



Pasca and Harabagiu (2001) demonstrates the value of sophisticated NLP processing

- Good IR is needed: paragraph retrieval based on SMART
- Large taxonomy of question types and expected answer types is crucial
- Statistical parser (modeled on Collins 1997) used to parse questions and relevant text for answers, and to build knowledge base
- Controlled query expansion loops (morphological, lexical synonyms, and semantic relations) are all important in retrieving the correct answer.
- Answer ranking by ML method based on this information surpasses IR-style empirical methods.

Question Answering Example (1)

- Q261: What company sells most greetings cards ?
- sells(ORGANIZATION, cards(greeting), most)
- “Hallmark remains the largest maker of greeting cards”
  maker(ORGANIZATION(Hallmark), cards(greeting), largest)
- Need an entailment between *producing*, or *making* and *selling* goods
- Derived from WordNet, since synset *make, produce, create* has the genus **manufacture**, defined in the gloss of its nominalization as (*for*) *sale*
- Also, need *most* $\approx$ *largest*
- Therefore the semantic form of question Q261 and its illustrated answer are similar

Question Answering Example (2)

- How hot does the inside of an active volcano get ?
- get(TEMPERATURE, inside(volcano(active)))
- “lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit”
- fragments(lava, TEMPERATURE(degrees(300)),
belched(out, mountain))
- - volcano ISA mountain
  - lava ISPARTOF volcano □ lava inside volcano
  - fragments of lava HAVEPROPERTIESOF lava
- The needed semantic information is available in WordNet definitions, and was successfully translated into a form that can be used for rough ‘proofs’

What’s used where? QA lessons learned

- IR: is not probabilistic, but is thoroughly quantitative, with well-honed, if ad hoc, ranking schemes
- Parsing: probabilistic parser (Collins 1997 clone) – the one big practical success of probabilistic parsing [speech wins (Charniak 2001) are still theoretical].
- Semantic interpretation: all loosely logical (though probabilistic parser is used to build knowledge base from WordNet) – succeeded by aiming low?
- Result ranking: ordering of paragraphs done by a perceptron classifier (7 features: question word matches, distance from words to answer type word, punctuation after answer, etc.). Statistical, but not high tech.

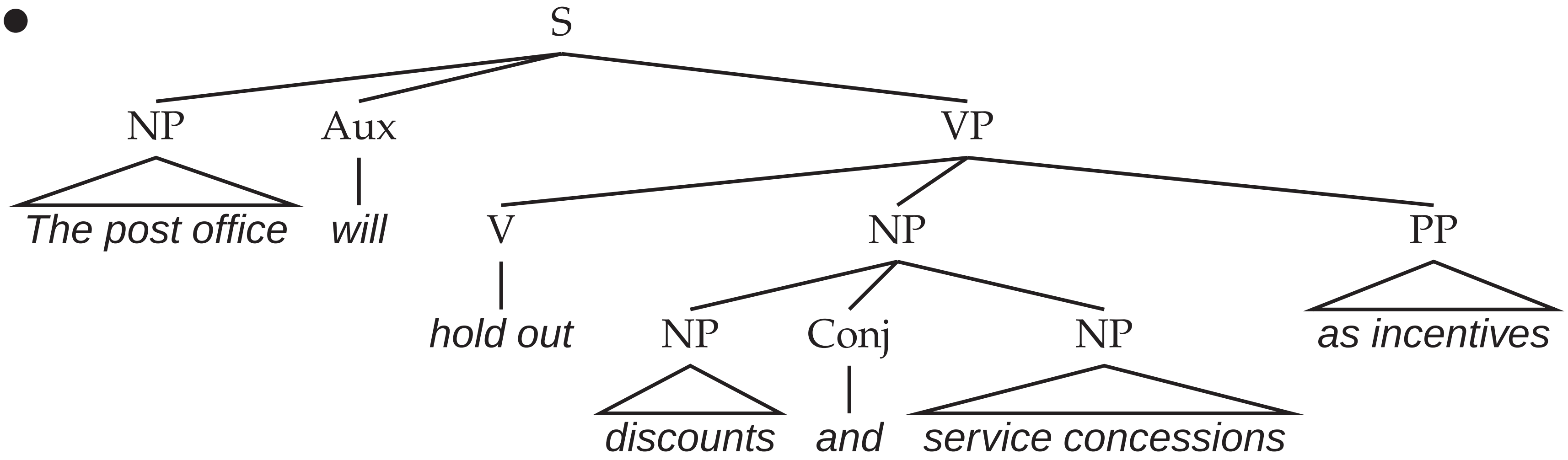
PCFGs

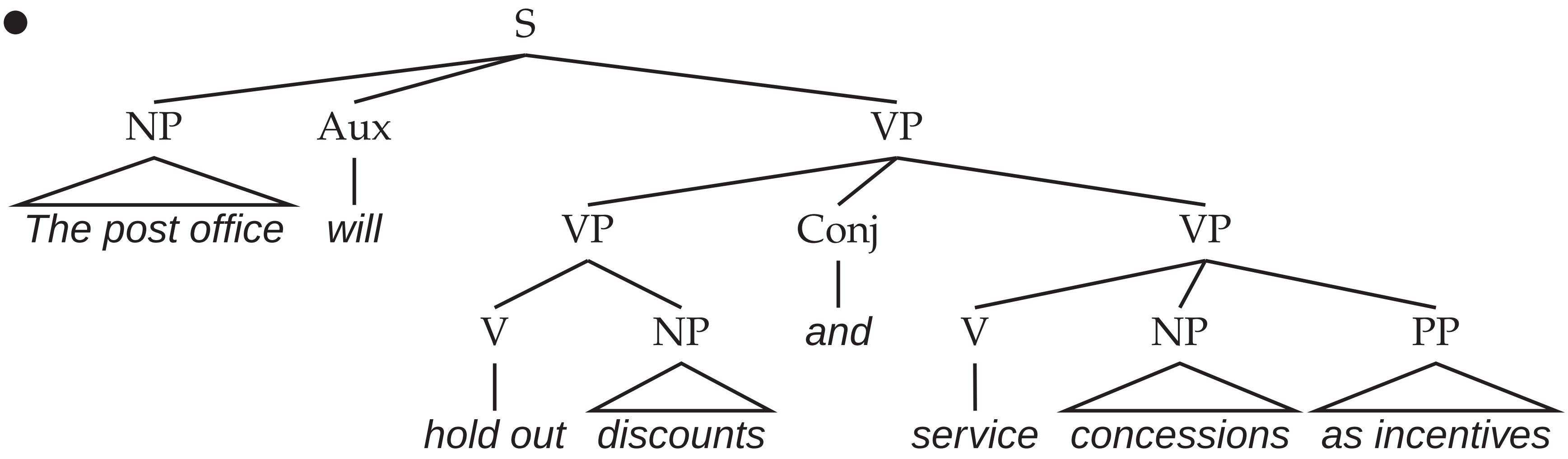
- Like a regular CFG but put probability on each rule
- *Key independence assumption:* Probabilities are also completely context-free, depending just on parent node
- Allow probabilistic inference:
  - $P(w_1 \cdots w_m | G)$
  - $\arg\max_t P(t | w_1 \cdots w_m, G)$
  - Find G such that $P(w_1 \cdots w_m | G)$ is maximized
- Give a partial solution for resolving grammar ambiguities – but not too good, as not lexicalized
- Better for grammar induction (Gold (1967) vs.Horning (1969)) and robustness

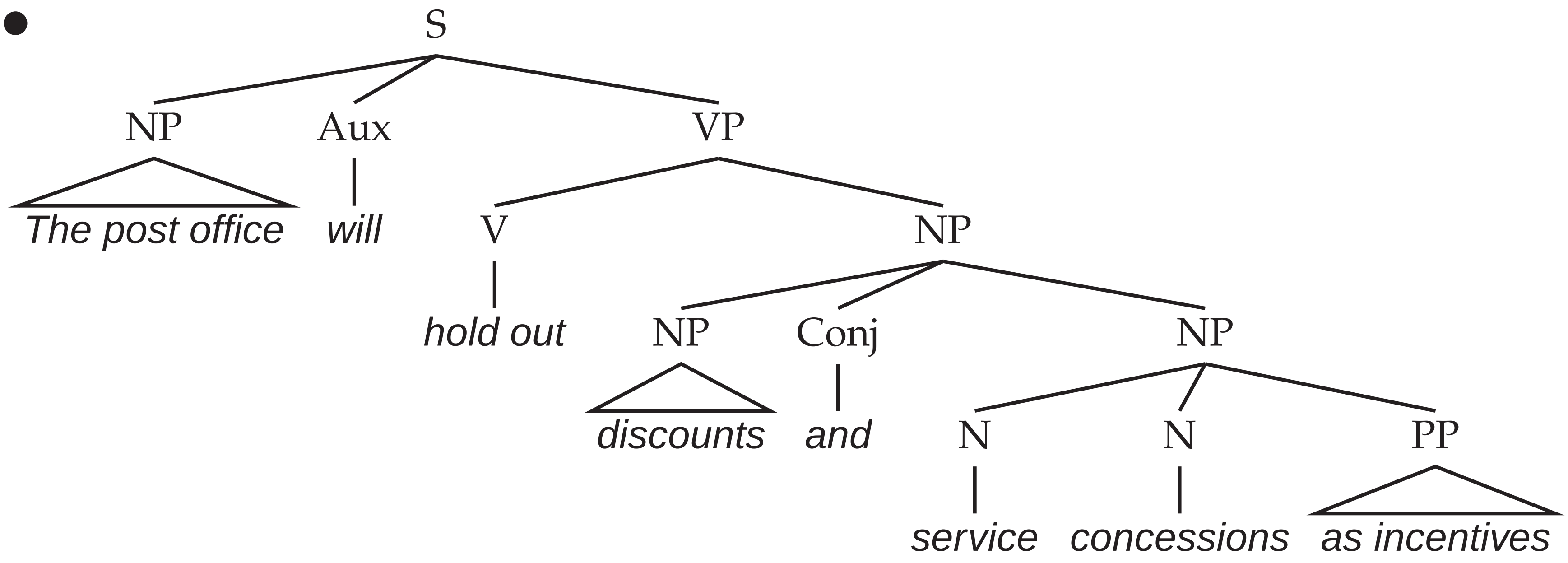
Probabilistic Parsing

FSNLP, chapter 12

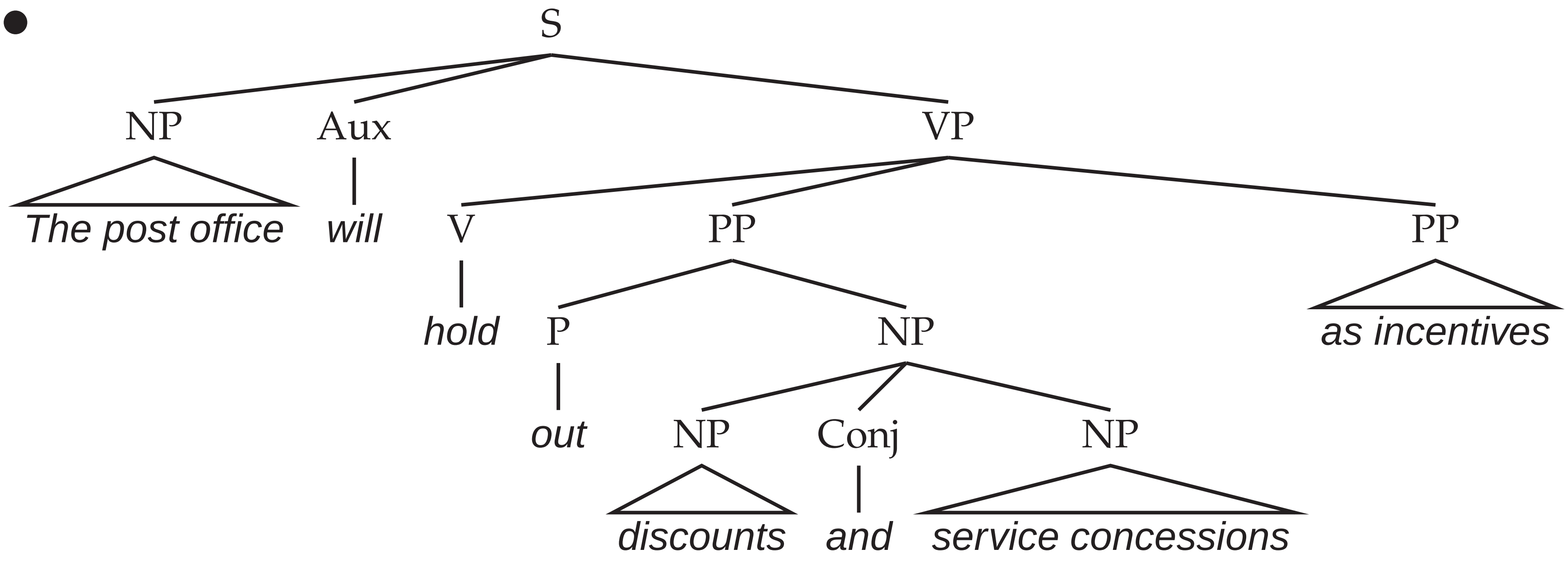
Christopher Manning and
Hinrich Schütze

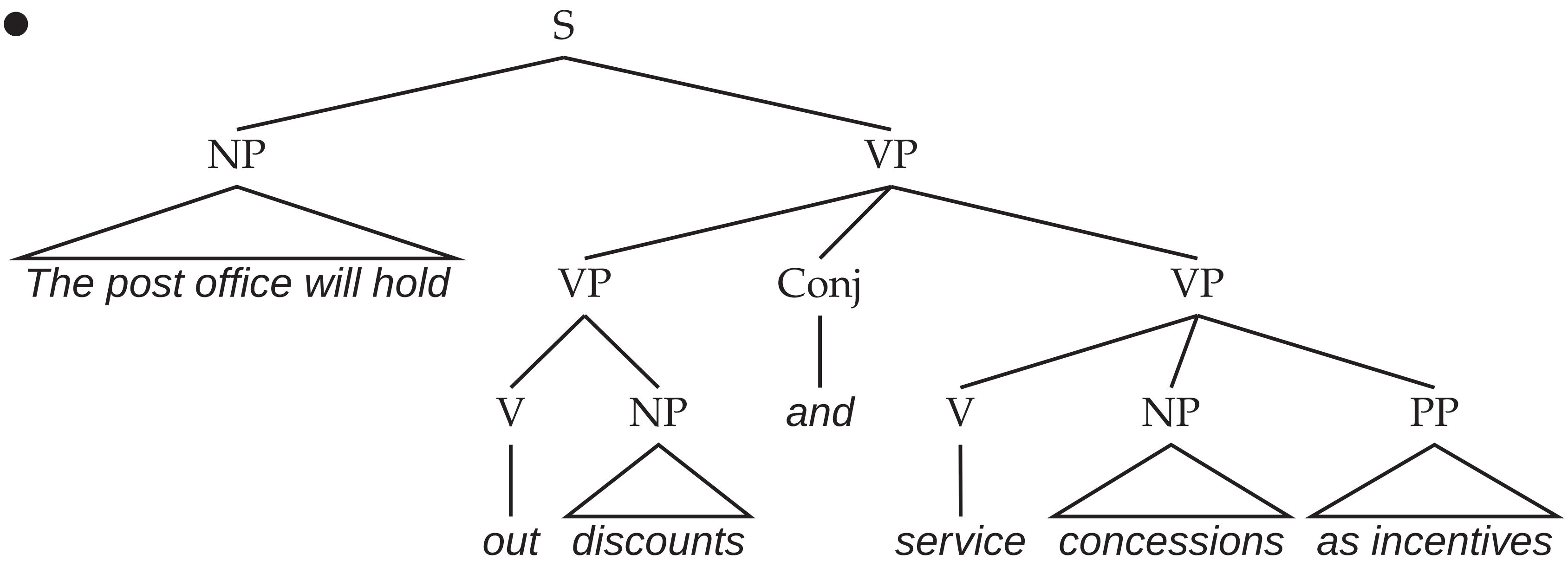
- 
S
NP
Aux
VP
The post office
will
V
NP
PP
hold out
NP
Conj
NP
as incentives
discounts
and
service concessions

- 
S
NP
Aux
VP
The post office
will
VP
Conj
VP
V
NP
and
V
NP
PP
hold out
discounts
service
concessions
as incentives

- 
S
NP
Aux
VP
The post office
will
V
NP
hold out
NP
Conj
NP
discounts
and
N
N
PP
service
concessions
as incentives

●


S
NP
Aux
VP
The post office
will
V
PP
PP
hold
P
NP
as incentives
out
NP
Conj
NP
discounts
and
service concessions

- 
S
NP
VP
The post office will hold
VP
Conj
VP
V
NP
and
V
NP
PP
out
discounts
service
concessions
as incentives

Modern Statistical Parsers

- A greatly increased ability to do accurate, robust, broad coverage parsing (Charniak 1997; Collins 1997; Ratnaparkhi 1997b; Charniak 2000; Charniak and Johnson 2005)
- Achieved by converting parsing into a classification task and using statistical/machine learning methods
- Statistical methods (fairly) accurately resolve structural and real world ambiguities
- Much faster: rather than being cubic in the sentence length or worse, many statistical parsers parse in linear time (by using beam search)
- Provide probabilistic language models that can be integrated with speech recognition systems.

Supervised ML parsing

- The crucial resource has been treebanks of parses, especially the Penn Treebank (Marcus et al. 1993)
- From these, we train classifiers:
  - Mainly probabilistic models, but also:
  - Conventional decision trees
  - Decision lists/transformation-based learning
- Possible only when extensive resources exist
- Somewhat uninteresting from Cog. Sci. viewpoint – which would prefer bootstrapping from minimal supervision

Penn Treebank Sentences: an example

```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

A Penn Treebank tree (POS tags not shown)

```
( (S (NP-SBJ The move)
     (VP followed
         (NP (NP a round)
             (PP of
                 (NP (NP similar increases)
                     (PP by
                         (NP other lenders))
                     (PP against
                         (NP Arizona real estate loans)))))
         ,
         (S-ADV (NP-SBJ *)
                (VP reflecting
                    (NP (NP a continuing decline)
                        (PP-LOC in
                                (NP that market))))))
     .))
```

Probabilistic models for parsing

- **Conditional/Parsing model:** We estimate directly the probability of parses of a sentence

  $\hat{t} = \arg\max_t P(t|s, G)$ where $\sum_t P(t|s, G) = 1$

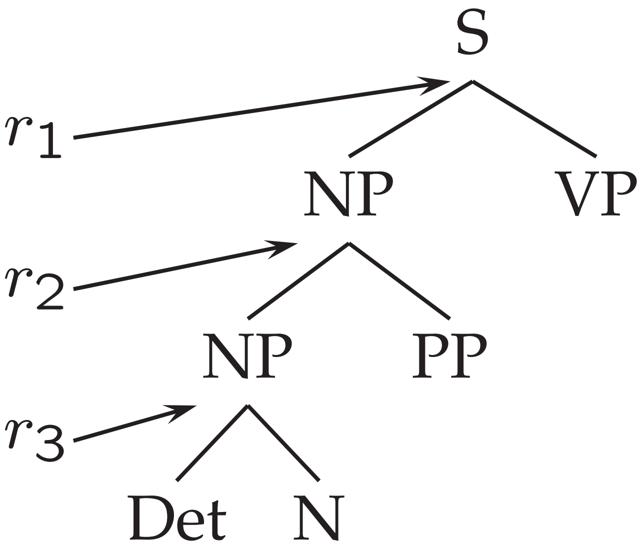
- We don't learn from the distribution of sentences we see (but nor do we assume some distribution for them)
  - (Magerman 1995; Ratnaparkhi 1999)
- **Generative/Joint/Language model:**

$$\sum_{\{t:\ \text{yield}(t) \in \mathcal{L}\}} P(t) = 1$$

- Most likely tree

  $\hat{t} = \arg\max_t P(t|s) = \arg\max_t \frac{P(t,s)}{P(s)} = \arg\max_t P(t, s)$
  - (Collins 1997; Charniak 1997, 2000)

Generative/Derivational model = Chain rule



$$P(t) = \sum_{\{d:\ d \text{ is a derivation of } t\}} P(d)$$

Or: $P(t) = P(d)$ where d is the canonical derivation of t

$$d = P(\mathsf{S} \xrightarrow{r_1} \alpha_1 \xrightarrow{r_2} \ldots \xrightarrow{r_m} \alpha_m = s) = \prod_{i=1}^{m} P(r_i | r_1, \ldots r_{i-1})$$

- History-based grammars

$$P(d) = \prod_{i=1}^{m} P(r_i | \pi(h_i))$$

Enriching a PCFG

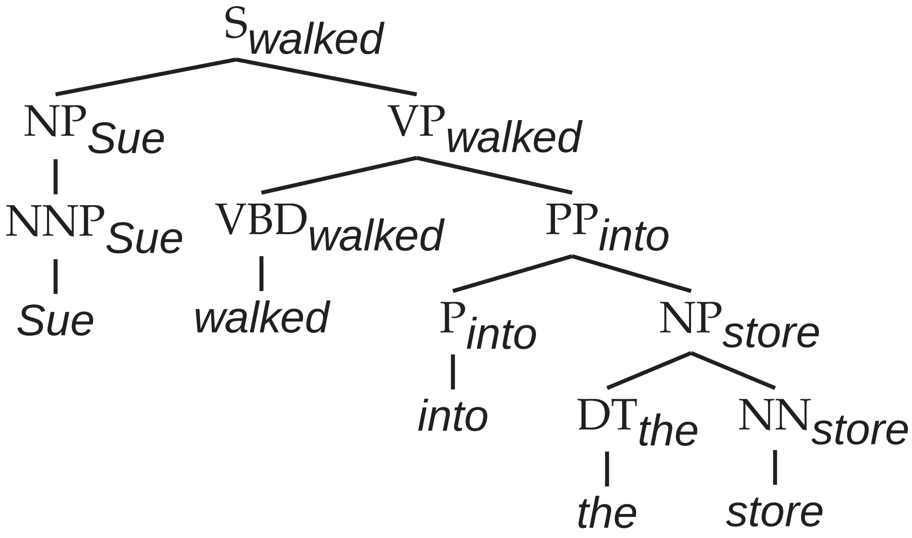
- A naive PCFG with traditional nonterminals (NP, PP, etc.) works quite poorly due to the independence assumptions it embodies (Charniak 1996)
- Fix: encode more information into the nonterminal space
  - Structure sensitivity (Manning and Carpenter 1997; Johnson 1998b; Klein and Manning 2003)
    - ∗ Expansion of nodes depends on their tree context. E.g., enrich nodes by recording their parents: ${}^{\mathrm{S}}\mathrm{NP}$ is different to ${}^{\mathrm{VP}}\mathrm{NP}$
  - (Head) Lexicalization (Collins 1997; Charniak 1997)
    - ∗ The head word of a phrase gives a good representation of the phrase's structure and meaning
    - ∗ Puts the properties of words back into a PCFG

Enriching a PCFG

- A naive PCFG with traditional nonterminals (NP, PP, etc.) works quite poorly due to the independence assumptions it embodies (Charniak 1996)
- Fix: encode more information into the nonterminal space
  - Structure sensitivity (Manning and Carpenter 1997; Johnson 1998b)
    - ∗ Expansion of nodes depends a lot on their position in the tree (independent of lexical content)
    - ∗ E.g., enrich nodes by also recording their parents: ${}^{\mathrm{S}}\mathrm{NP}$ is different to ${}^{\mathrm{VP}}\mathrm{NP}$

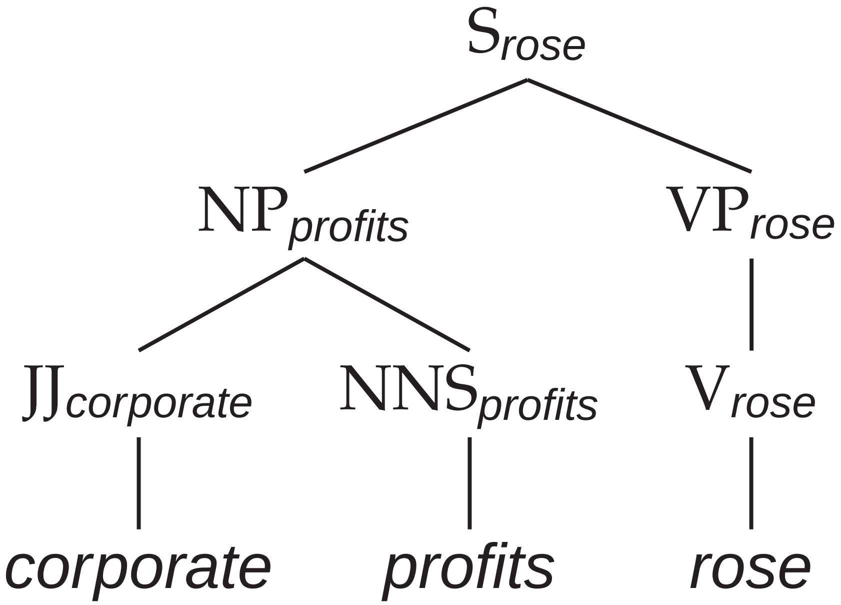
Enriching a PCFG (2)

- (Head) Lexicalization (Collins 1997; Charniak 1997)
  - ∗ The head word of a phrase gives a good representation of the phrase's structure and meaning
  - ∗ Puts the properties of words back into a PCFG

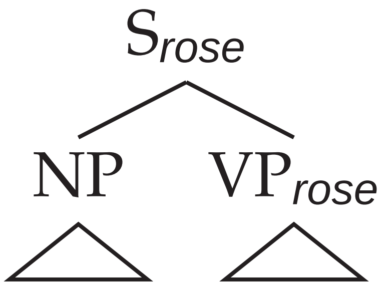


Parsing via classification decisions: Charniak (1997)

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is “top-down” (but actual computation is bottom-up)



Charniak (1997) example

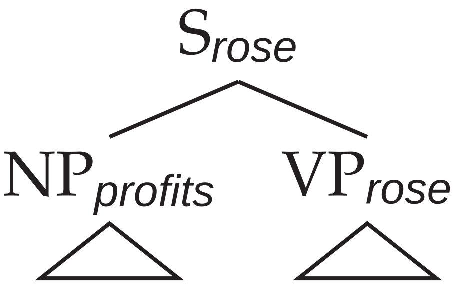


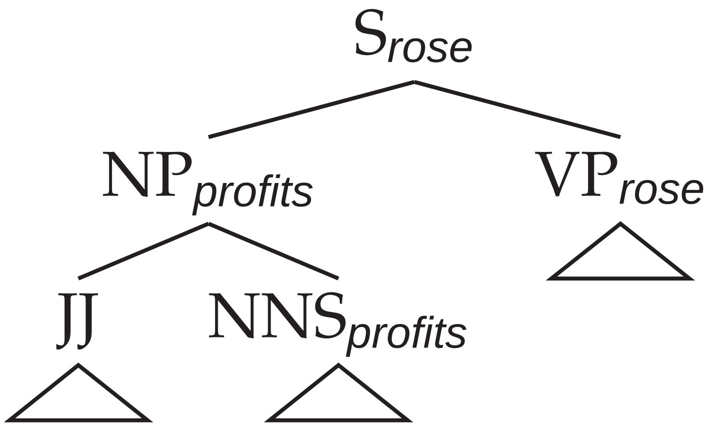
a. h = *profits*; c = NP

b. ph = *rose*; pc = S

c. $P(h|ph, c, pc)$

d. $P(r|h, c, pc)$





Charniak (1997) linear interpolation/shrinkage

$$\begin{aligned}\widehat{P}(h|ph,c,pc) &= \lambda_1(e)P_{\text{MLE}}(h|ph,c,pc)\\ &\quad+\lambda_2(e)P_{\text{MLE}}(h|C(ph),c,pc)\\ &\quad+\lambda_3(e)P_{\text{MLE}}(h|c,pc)+\lambda_4(e)P_{\text{MLE}}(h|c)\end{aligned}$$

- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

	$P(\text{prft}\|\text{rose}, \text{NP}, \text{S})$	$P(\text{corp}\|\text{prft}, \text{JJ}, \text{NP})$
$P(h\|ph, c, pc)$	0	0.245
$P(h\|C(ph), c, pc)$	0.00352	0.0150
$P(h\|c, pc)$	0.000627	0.00533
$P(h\|c)$	0.000557	0.00418

- Allows utilization of rich highly conditioned estimates, but smoothes when sufficient data is unavailable
- One can’t just use MLEs: one commonly sees previously unseen events, which would have probability 0.

Sparseness & the Penn Treebank

- The Penn Treebank – 1 million words of parsed English *WSJ* – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*, but there is an infinite space of these (*how clever/original/incompetent* (*at risk assessment and evaluation*))
- Most of the probabilities that you would like to compute, you can't compute

Sparseness & the Penn Treebank (2)

- Most intelligent processing depends on bilexical statistics: likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the *WSJ*:

–	stocks plummeted	2 occurrences
–	stocks stabilized	1 occurrence
–	stocks skyrocketed	0 occurrences
–	$^{\#}$stocks discussed	0 occurrences

- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

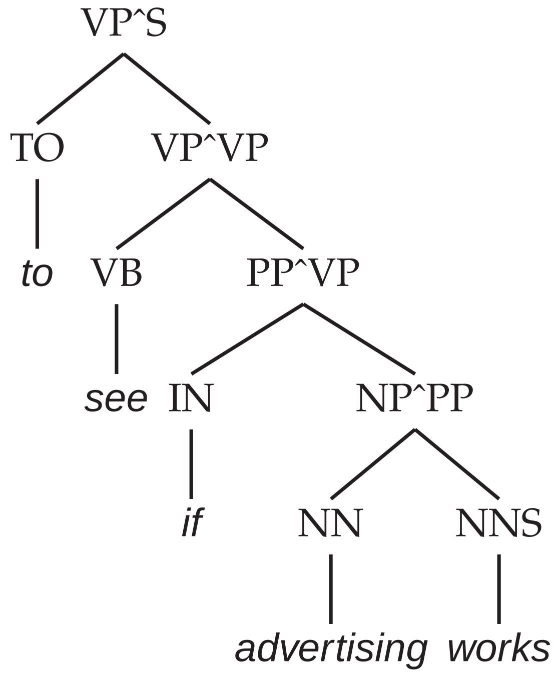
Probabilistic parsing

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes
- But it is bad because of data sparseness
- A pure dependency, one child at a time, model is worse
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

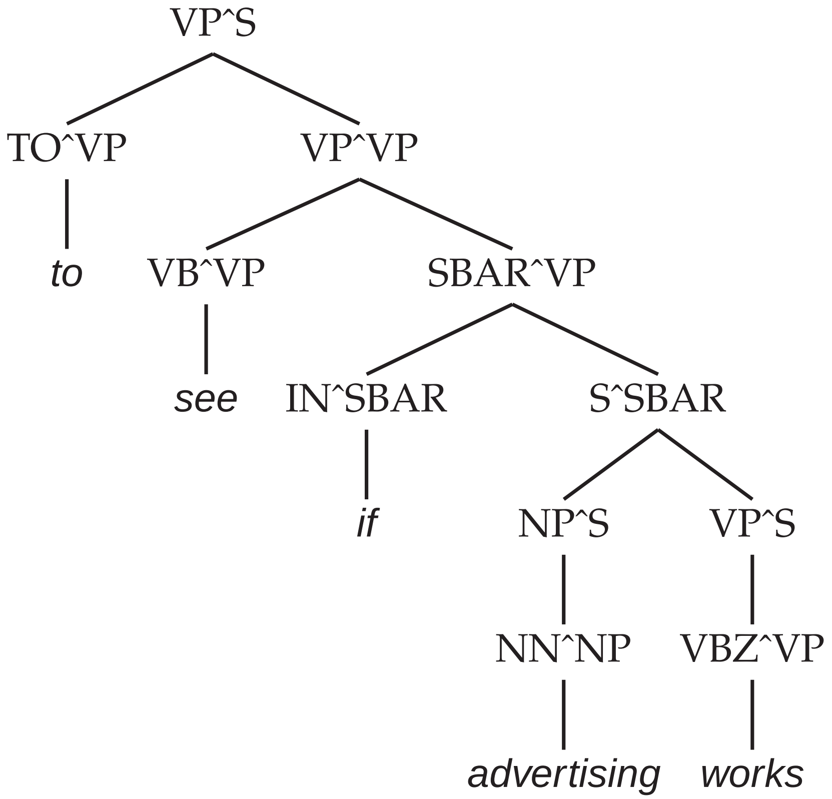
Correcting wrong context-freedom assumptions

		Horizonal Markov Order				
Vertical Order		$h = 0$	$h = 1$	$h \leq 2$	$h = 2$	$h = \infty$
$v = 0$	No annotation	71.27	72.5	73.46	72.96	72.62
		(854)	(3119)	(3863)	(6207)	(9657)
$v \leq 1$	Sel. Parents	74.75	77.42	77.77	77.50	76.91
		(2285)	(6564)	(7619)	(11398)	(14247)
$v = 1$	All Parents	74.68	77.42	77.81	77.50	76.81
		(2984)	(7312)	(8367)	(12132)	(14666)
$v \leq 2$	Sel. GParents	76.50	78.59	79.07	78.97	78.54
		(4943)	(12374)	(13627)	(19545)	(20123)
$v = 2$	All GParents	76.74	79.18	79.74	79.07	78.72
		(7797)	(15740)	(16994)	(22886)	(22002)

Correcting wrong context-freedom assumptions



(a)



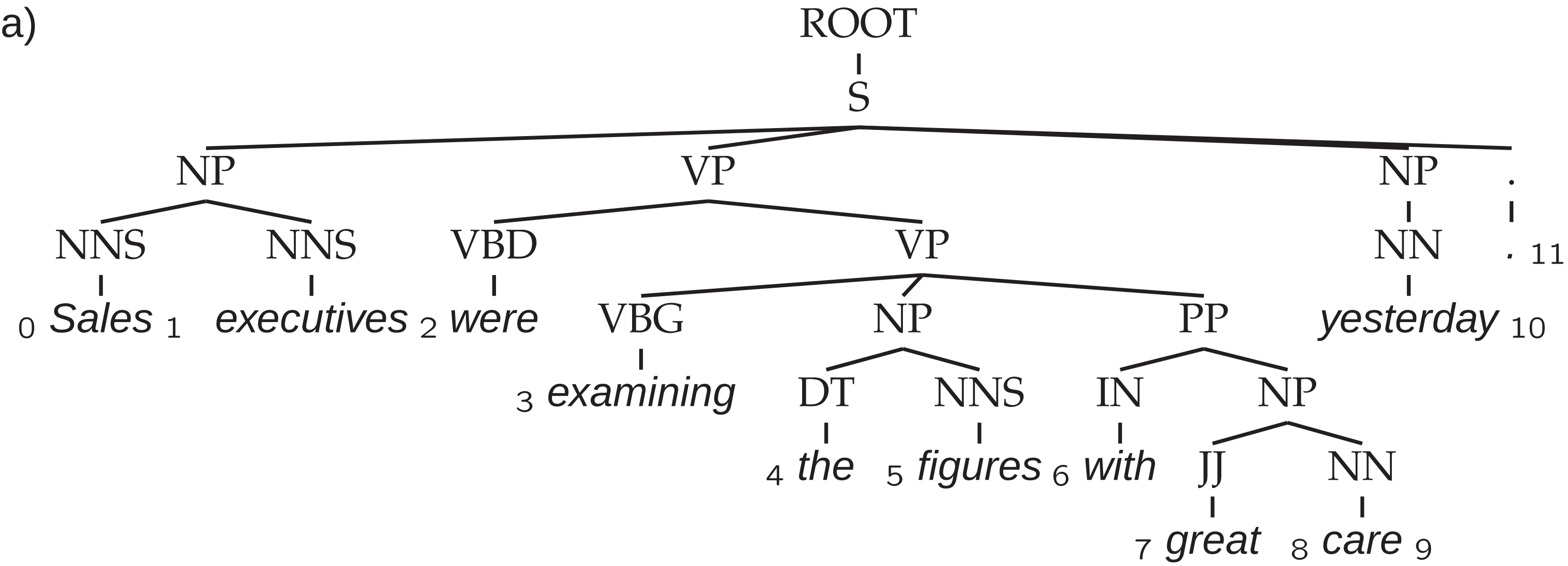
(b)

Correcting wrong context-freedom assumptions

	Cumulative			Indiv.
Annotation	Size	F_1	Δ F_1	Δ F_1
Baseline	7619	77.72	0.00	0.00
UNARY-INTERNAL	8065	78.15	0.43	0.43
UNARY-DT	8078	80.09	2.37	0.22
UNARY-RB	8081	80.25	2.53	0.48
TAG-PA	8520	80.62	2.90	2.57
SPLIT-IN	8541	81.19	3.47	2.17
SPLIT-AUX	9034	81.66	3.94	0.62
SPLIT-CC	9190	81.69	3.97	0.17
SPLIT-%	9255	81.81	4.09	0.20
TMP-NP	9594	82.25	4.53	1.12
GAPPED-S	9741	82.28	4.56	0.22
POSS-NP	9820	83.06	5.34	0.33
SPLIT-VP	10499	85.72	8.00	1.41
BASE-NP	11660	86.04	8.32	0.78
DOMINATES-V	14097	86.91	9.19	1.47
RIGHT-REC-NP	15276	87.04	9.32	1.99

Evaluation

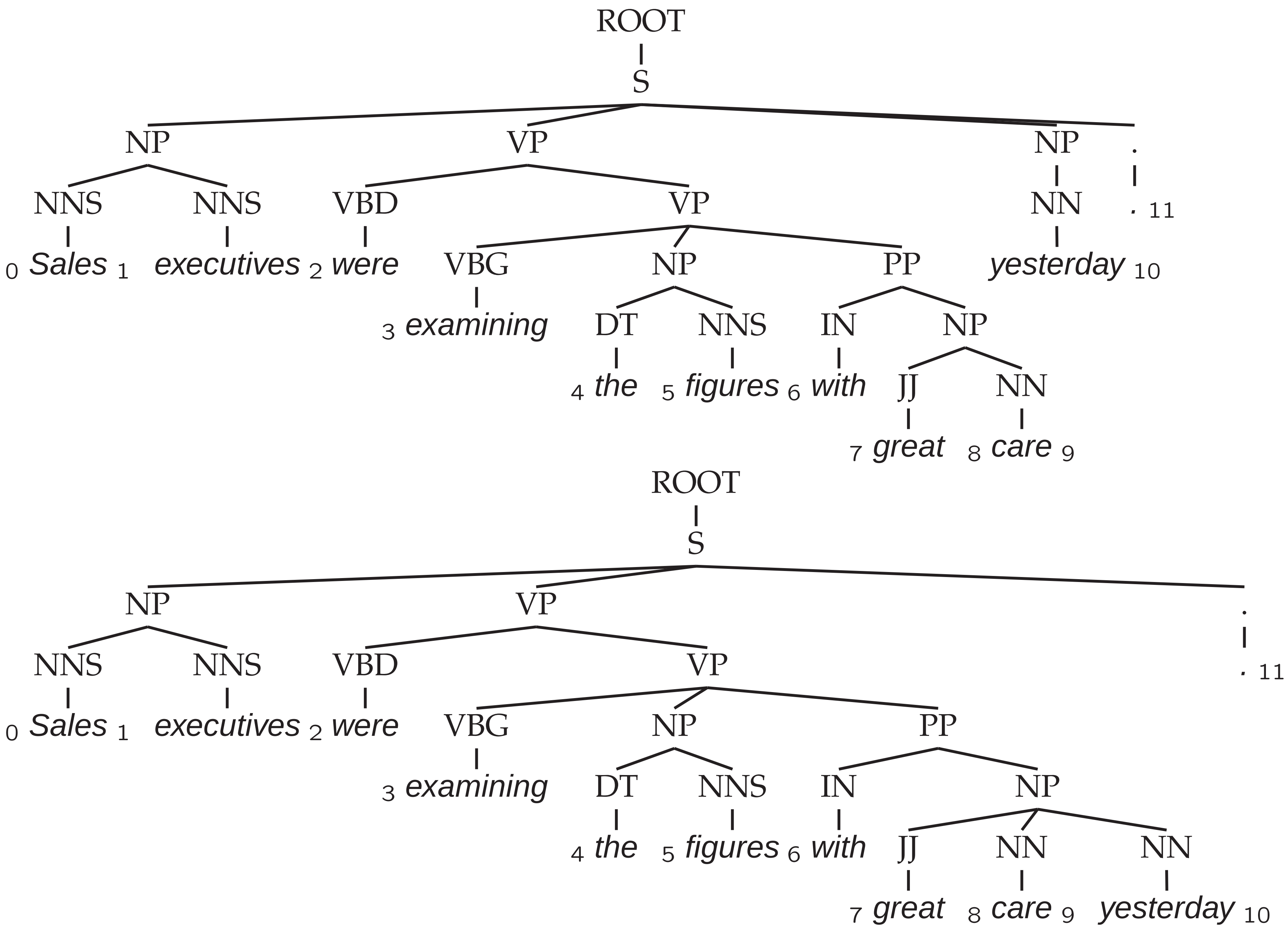
(a)



(b) Brackets in gold standard tree (a.):
S-(0:11), **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), *NP-(9:10)

(c) Brackets in candidate parse:
S-(0:11), **NP-(0:2)**, VP-(2:10), VP-(3:10), NP-(4:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

(d)	Precision:	3/8 = 37.5%	Crossing Brackets:	0
	Recall:	3/8 = 37.5%	Crossing Accuracy:	100%
	Labeled Precision:	3/8 = 37.5%	Tagging Accuracy:	10/11 = 90.9%
	Labeled Recall:	3/8 = 37.5%		



Gold standard brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), NP-(9

Candidate brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:10), VP-(3:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

Gold standard brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), NP-(9

Candidate brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:10), VP-(3:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

Labeled Precision:	3/7 = 42.9%
Labeled Recall:	3/8 = 37.5%
LP/LR F_1	40.0%
Tagging Accuracy:	11/11 = 100.0%

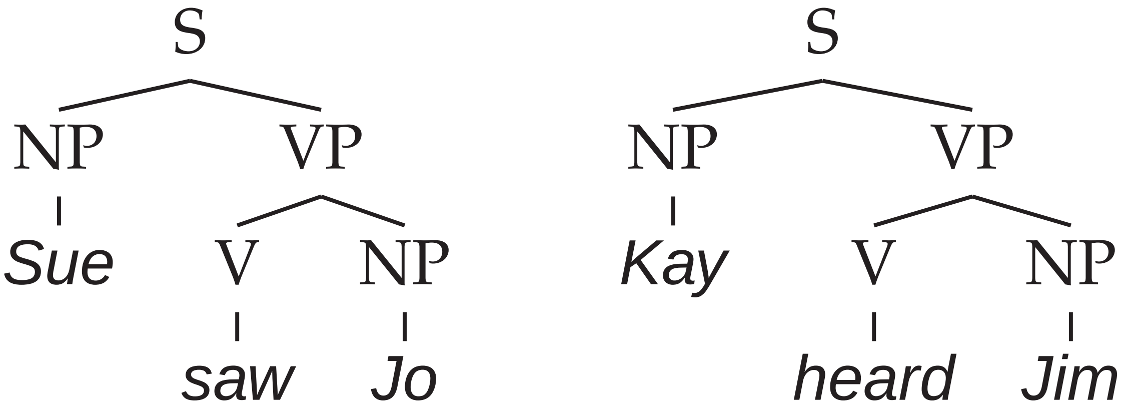
Parser results

- Parsers are normally evaluated on the relation between *individual postulated nodes* and ones in the gold standard tree (Penn Treebank, section 23)
- Normally people make systems balanced for precision/recall
- Normally evaluate on sentences of 40 words or less
- Magerman (1995): about 85% labeled precision and recall
- Charniak (2000) gets 90.1% labeled precision and recall
- Good performance. Steady progress in error reduction
- At some point size of and errors in treebank must become the limiting factor

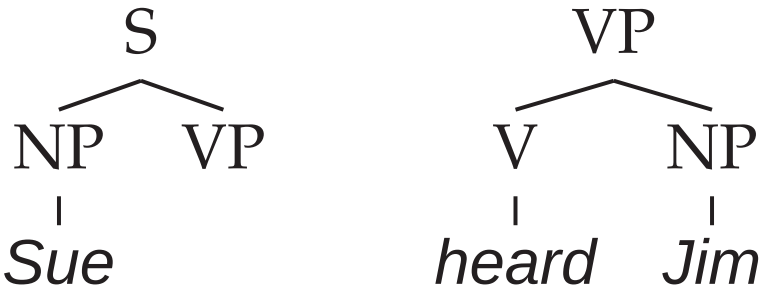
– (Some thought that was in 1997, when several systems were getting 87.x%, but apparently not.)

Bod’s (1998) Data Oriented Parsing (DOP)

- A memory-based approach using linguistic experiences
- Suppose we have a corpus of two sentences:



- To parse the new sentence *Sue heard Jim*, we put together tree fragments that we have already seen:

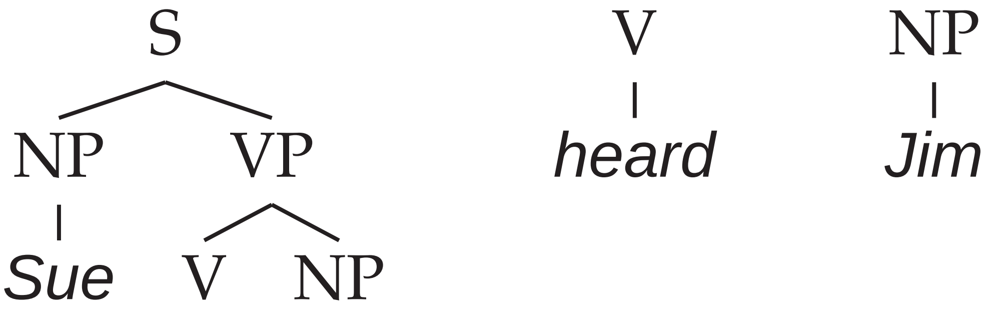


Bod's (1998) Data Oriented Parsing (DOP)

- Probability of fragment F expanding mother M:

$$P(F|M) = \#(F)/\#(M)$$

- We assume independence across fragments
- So, we just multiply to get a derivation probability
  - E.g., 8 fragments with VP as mother, 1 *heard Jim*, so P([VP [V *heard*] [NP *Jim*]]) = 1/8.
- I gave one DOP derivation of a tree. There are many:



- Need to sum over derivations to find tree probability

Data Oriented Parsing results (Bod 2000)

- The model has a lot of really bad problems:
  - The estimator is biased and inconsistent
  - Big trees are greatly over-represented in estimates
    - # of fragments is exponential in the tree size
  - There is little explicit modeling of linguistic structure, as in the head percolation of lexicalized PCFGs
- But DOP learns nonhead dependencies, topical correlations and idioms, that no other approach treats.
- Because of this, it works well:
  - 89.5% LP, 89.3% LR – better than Charniak (1997), Collins (1997); a smidgen behind Charniak (2000)

Extra/variant slides

Statistical Parsing

- Recent work on statistical parsing, including (Magerman 1995; Charniak 1997; Collins 1996, 1997; Ratnaparkhi 1997b; Charniak 2000) has greatly raised the bar of parsing performance
- It's redefined the ability to do robust, rapid broad coverage parsing
  - Around 1990: Probably no parser could do better than around 65% labeled precision/recall over broad coverage text (evaluation was less rigorous)
  - 2000: Over 90% labeled precision/recall
- How? By converting parsing into a classification task and using machine learning methods

Probabilistic models for parsing

- Parsing model: We estimate directly the probability of parses of a sentence

  $$\hat{t} = \arg\max_t P(t|s, G) \quad \text{where} \quad \sum_t P(t|s, G) = 1$$

  (Magerman 1995; Collins 1996)
- We don't learn from the distribution of sentences we see (but nor do we assume some distribution for them)
- But in effect we're always generalizing over sentences in estimating rules

Language model

- Language model:

$$\sum_{\{t\colon \text{yield}(t) \in \mathcal{L}\}} P(t) = 1$$

- Sentence probability

$$P(s) = \sum_t P(s,t) = \sum_{\{t\colon \text{yield}(t) = s\}} P(t)$$

- Most likely tree

$$\hat{t} = \arg\max_t P(t|s) = \arg\max_t \frac{P(t,s)}{P(s)} = \arg\max_t P(t,s)$$

- (Collins 1997; Charniak 1997, 2000)

Derivational model

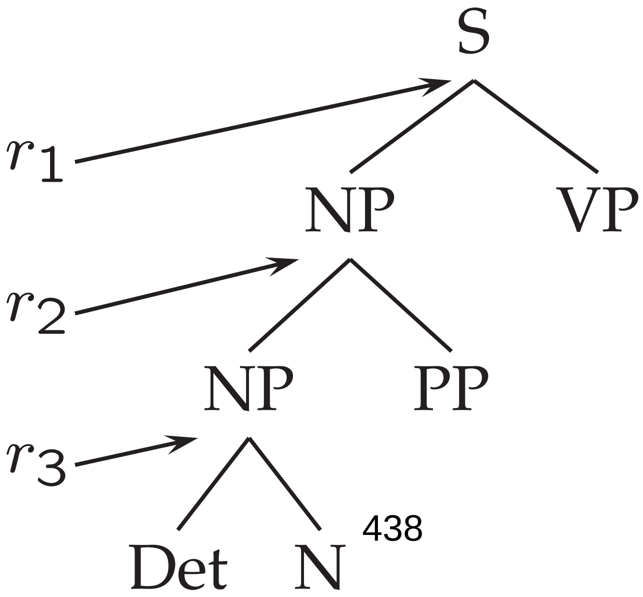
$$P(t) = \sum_{\{d:\ d \text{ is a derivation of } t\}} P(d)$$

Or: $P(t) = P(d)$ where d is the canonical derivation of t

$$d = P(\mathsf{S} \xrightarrow{r_1} \alpha_1 \xrightarrow{r_2} \ldots \xrightarrow{r_m} \alpha_m = s) = \prod_{i=1}^{m} P(r_i | r_1, \ldots r_{i-1})$$

- History-based grammars:

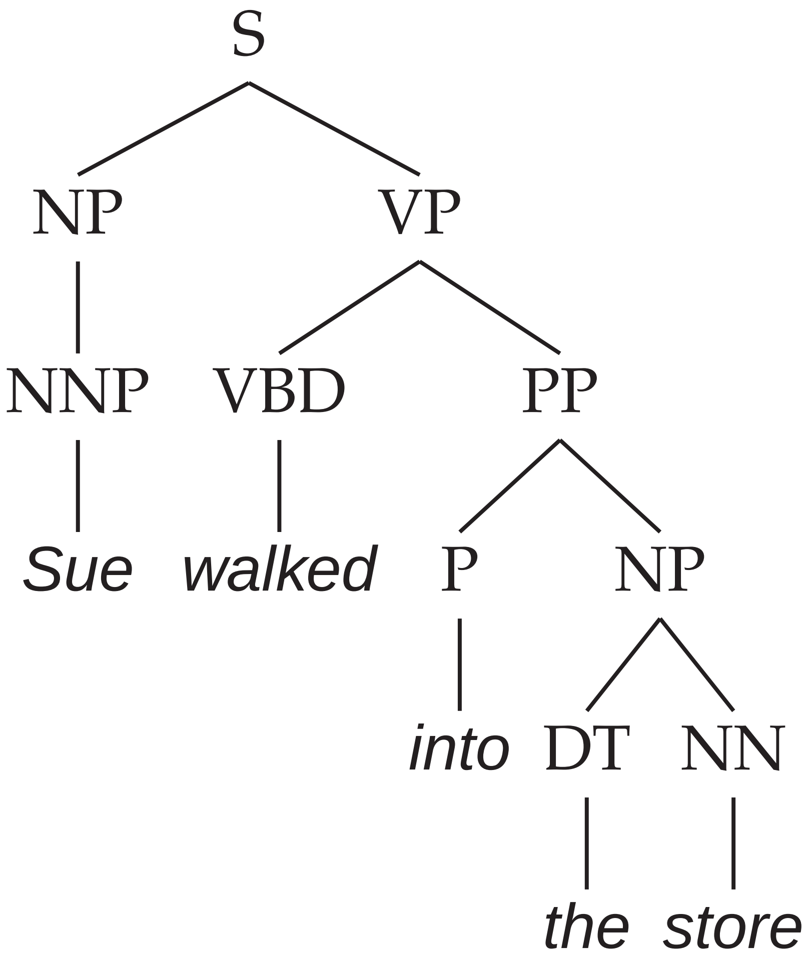
$$P(d) = \prod_{i=1}^{m} P(r_i | \pi(h_i))$$



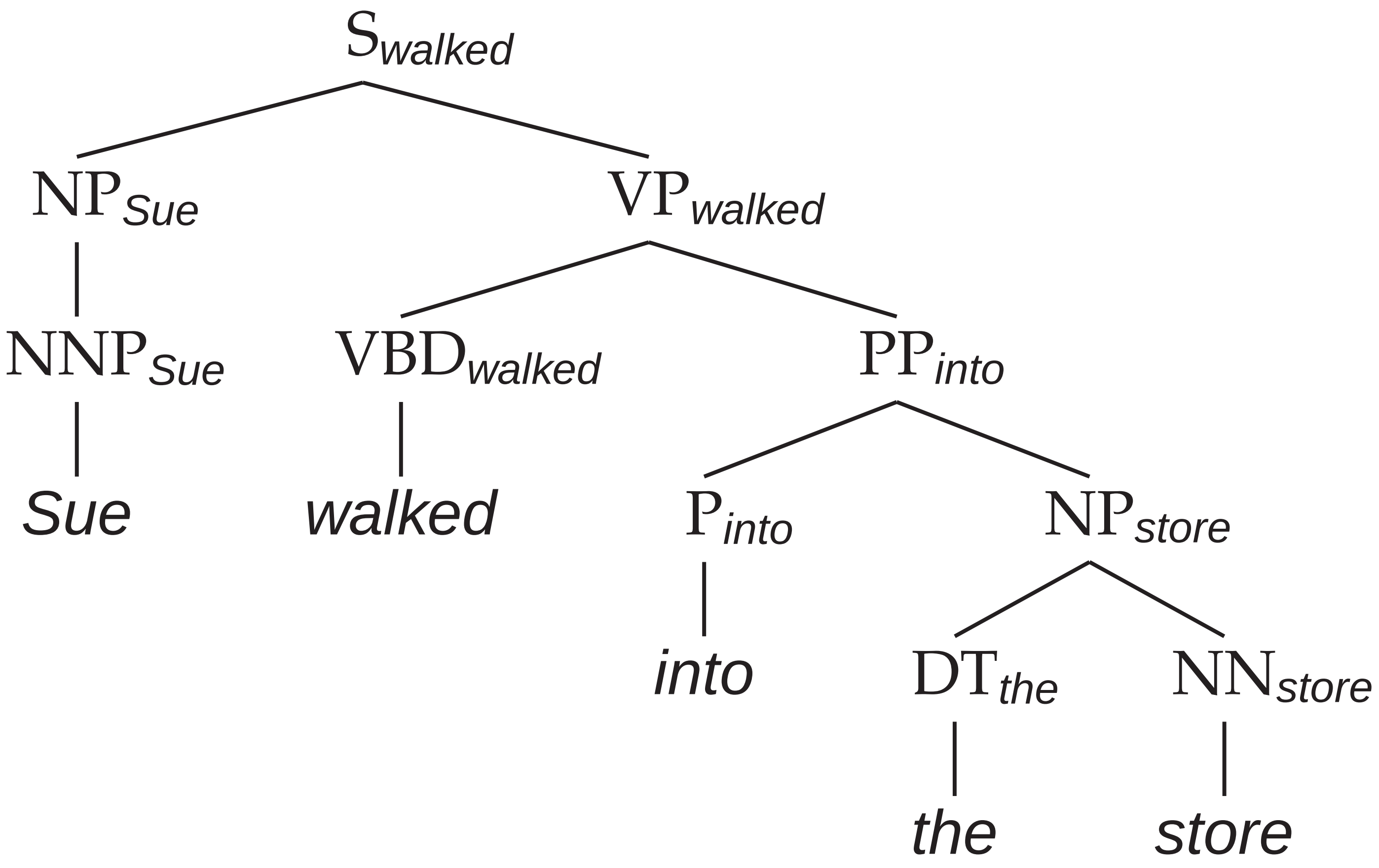
Enrichments

- Structure sensitivity
  - Expansions of nodes depends a lot on their position in the tree (independent of lexical content)
  - (Manning and Carpenter 1997; Johnson 1998a)
- Lexicalization – getting back properties of words

A standard CFG phrase structure tree

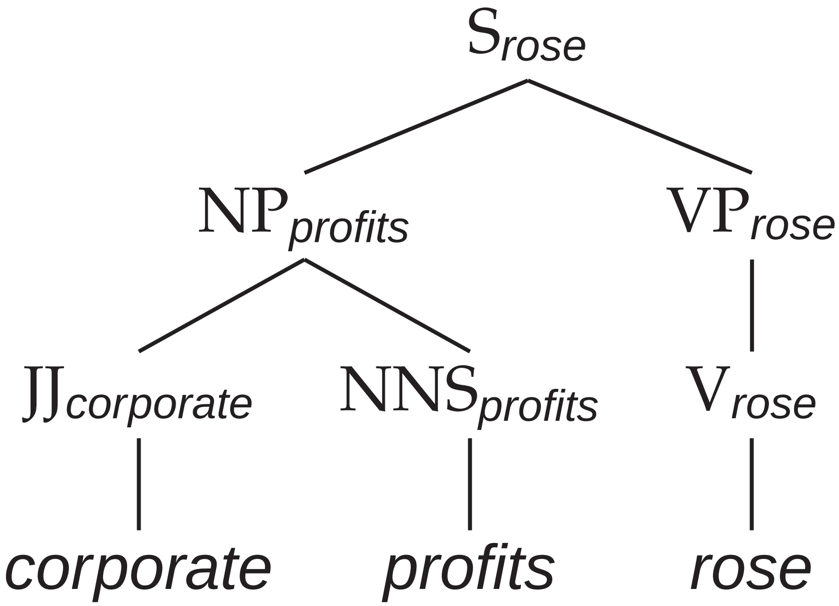


Lexicalization: From this we form a lexicalized tree



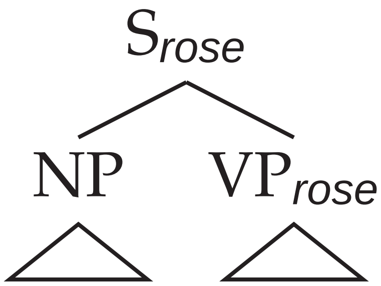
Modern enhanced PCFG parsing
E.g., Charniak (1997)

- A very simple, conservative model of a lexicalized PCFG



- Probabilistic conditioning is "top-down" (but actual computation is bottom-up)

Charniak (1997) example

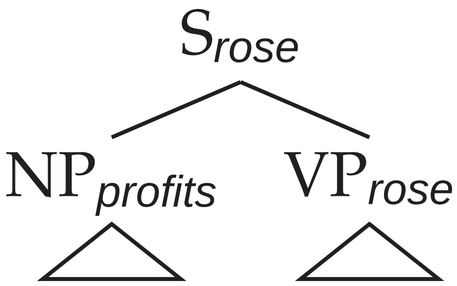


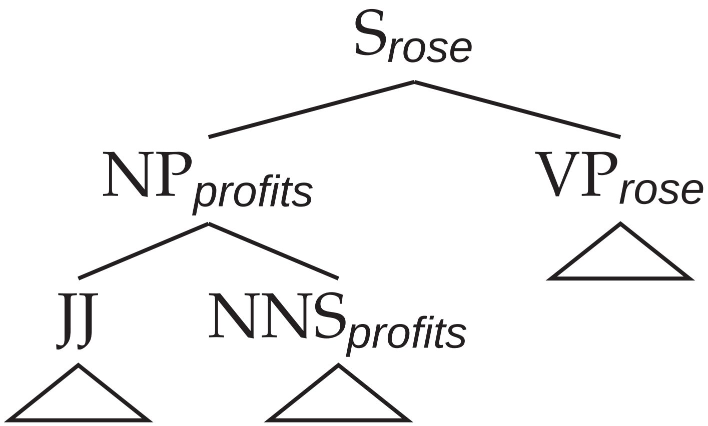
a. h = *profits*; c = NP

b. ph = *rose*; pc = S

c. $P(h|ph, c, pc)$

d. $P(r|h, c, pc)$





Charniak (1997) linear interpolation/shrinkage

$$\begin{aligned}\widehat{P}(h|ph,c,pc) \;=\; & \lambda_1(e)P_{\mathrm{MLE}}(h|ph,c,pc) \\ & +\lambda_2(e)P_{\mathrm{MLE}}(h|C(ph),c,pc) \\ & +\lambda_3(e)P_{\mathrm{MLE}}(h|c,pc)+\lambda_4(e)P_{\mathrm{MLE}}(h|c)\end{aligned}$$

- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

	$P(\text{prft}\|\text{rose}, \text{NP}, \text{S})$	$P(\text{corp}\|\text{prft}, \text{JJ}, \text{NP})$
$P(h\|ph, c, pc)$	0	0.245
$P(h\|C(ph), c, pc)$	0.00352	0.0150
$P(h\|c, pc)$	0.000627	0.00533
$P(h\|c)$	0.000557	0.00418

- Allows utilization of rich highly conditioned estimates, but smoothes when sufficient data is unavailable
- In practice, one is frequently having to back off to simple conditioning which is purely syntactic

Topics in Information Retrieval

***FSNLP*, chapter 15**

Christopher Manning and

Hinrich Schütze

Information Retrieval

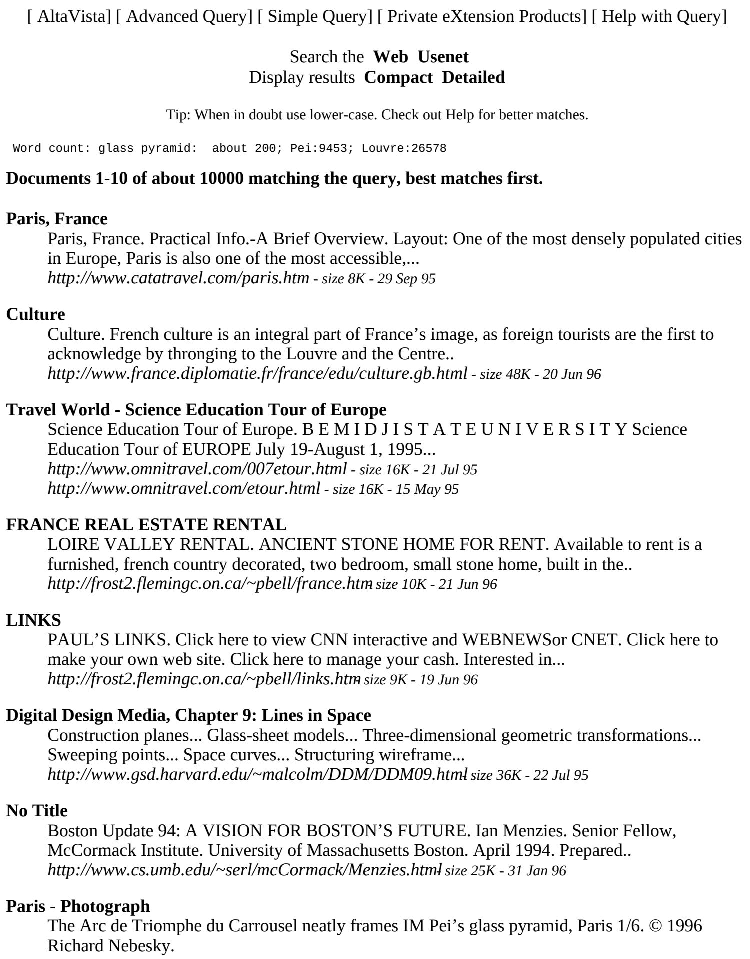
- Getting information from document repositories
- Normally text (though spoken, image, and video data are all becoming more important)
- Traditionally a rather separate field from NLP, and always very empirically based
- A field of some antiquity: the famous SMART IR system (Salton) predates the relational model in databases
- New directions: the Web, email, multimedia, . . .
- There is much scope for greater profitable interaction between IR and Statistical NLP

Tasks

- “Ad hoc retrieval”: the user enters query terms which describe the desired information; the system returns a set of (sometimes ranked) documents.
- Document categorization: assign a document to one or more categories (e.g., subject codes) [chapter 16]
  - Filtering: categorization with binary choice about the relevance of a document (e.g., screen for junk email).
  - Routing: categorization for the purpose of transmitting a document to one or more users (e.g., customer service by product)

Tasks (continued)

- Document clustering: group similar documents into clusters (e.g., for making sense of ad hoc retrieval results) [chapter 14]
- Text segmentation: identify semantically coherent units within a text (e.g., for retrieval below the document level) [section 15.4]
- Text summarization: create a shorter version of a document containing just the relevant information
  - Knowledge-based: generate new text
  - Selection-based: extract the n most important summary sentences from the orginal document



[AltaVista] [Advanced Query] [Simple Query] [Private eXtension Products] [Help with Query]

Search the **Web** **Usenet**
Display results **Compact** **Detailed**

Tip: When in doubt use lower-case. Check out Help for better matches.

```
Word count: glass pyramid:  about 200; Pei:9453; Louvre:26578
```

Documents 1-10 of about 10000 matching the query, best matches first.

Paris, France
Paris, France. Practical Info.-A Brief Overview. Layout: One of the most densely populated cities in Europe, Paris is also one of the most accessible,...
http://www.catatravel.com/paris.htm - size 8K - 29 Sep 95

Culture
Culture. French culture is an integral part of France’s image, as foreign tourists are the first to acknowledge by thronging to the Louvre and the Centre..
http://www.france.diplomatie.fr/france/edu/culture.gb.html - size 48K - 20 Jun 96

Travel World - Science Education Tour of Europe
Science Education Tour of Europe. B E M I D J I S T A T E U N I V E R S I T Y Science Education Tour of EUROPE July 19-August 1, 1995...
http://www.omnitravel.com/007etour.html - size 16K - 21 Jul 95
http://www.omnitravel.com/etour.html - size 16K - 15 May 95

FRANCE REAL ESTATE RENTAL
LOIRE VALLEY RENTAL. ANCIENT STONE HOME FOR RENT. Available to rent is a furnished, french country decorated, two bedroom, small stone home, built in the..
http://frost2.flemingc.on.ca/~pbell/france.htm - size 10K - 21 Jun 96

LINKS
PAUL’S LINKS. Click here to view CNN interactive and WEBNEWSor CNET. Click here to make your own web site. Click here to manage your cash. Interested in...
http://frost2.flemingc.on.ca/~pbell/links.htm - size 9K - 19 Jun 96

Digital Design Media, Chapter 9: Lines in Space
Construction planes... Glass-sheet models... Three-dimensional geometric transformations... Sweeping points... Space curves... Structuring wireframe...
http://www.gsd.harvard.edu/~malcolm/DDM/DDM09.html - size 36K - 22 Jul 95

No Title
Boston Update 94: A VISION FOR BOSTON’S FUTURE. Ian Menzies. Senior Fellow, McCormack Institute. University of Massachusetts Boston. April 1994. Prepared..
http://www.cs.umb.edu/~serl/mcCormack/Menzies.html - size 25K - 31 Jan 96

Paris - Photograph
The Arc de Triomphe du Carrousel neatly frames IM Pei’s glass pyramid, Paris 1/6. © 1996 Richard Nebesky.

Results of the search ‘ “glass pyramid” Pei Louvre’ on AltaVista

IR system design

- Unlike databases, IR systems index *everything*
- Usually by an *inverted index* that contains *postings* of all word occurrences in documents
- Having position-in-file information enables *phrase matching* (where an IR “phrase” is just contiguous words)
- A *stop list* of common, meaningless words is often not indexed
- This greatly cuts the inverted index size (given Zipf’s Law)
- *Stemming* means indexing only truncated morphological roots. This sometimes helps (but not always).

Stop words: A small stop list for English

a	also	an	and	as	at	be	but
by	can	could	do	for	from	go	
have	he	her	here	his	how		
i	if	in	into	it	its		
my	of	on	or	our	say	she	
that	the	their	there	therefore	they		
this	these	those	through	to	until		
we	what	when	where	which	while	who	with
would	you	your					

The probability ranking principle (PRP)

IR fundamentally addresses this problem: Given a query W_1 and a document W_2 attempt to decide relevance of W_2 to W_1, where relevance is meant to be computed with respect to their hidden meanings M_1 and M_2.

The model underlying most IR systems (van Rijsbergen 1979: 113):

- PRP: Rank documents in order of decreasing probability of relevance is optimal.

Problems: documents that aren't independent. Any that don't give additional information (especially, duplicates!). Implies not doing word-sense disambiguation.

The Vector Space Model (Salton, TREC)

Represents terms and documents as vectors in k-dimen. space based on the bag of words they contain:

d = The man said that a space age man appeared

d' = Those men appeared to say their age

$$\vec{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

	$\vec{d}$	$\vec{d'}$
age	1	1
appeared	1	1
man	2	0
men	0	1
said	1	0
say	0	1
space	1	0

Real-valued vector spaces

Vector dot product (how much do they have in common?):

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

0 if orthogonal (no words in common)

Length of a vector:

$$|\vec{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Normalized vectors

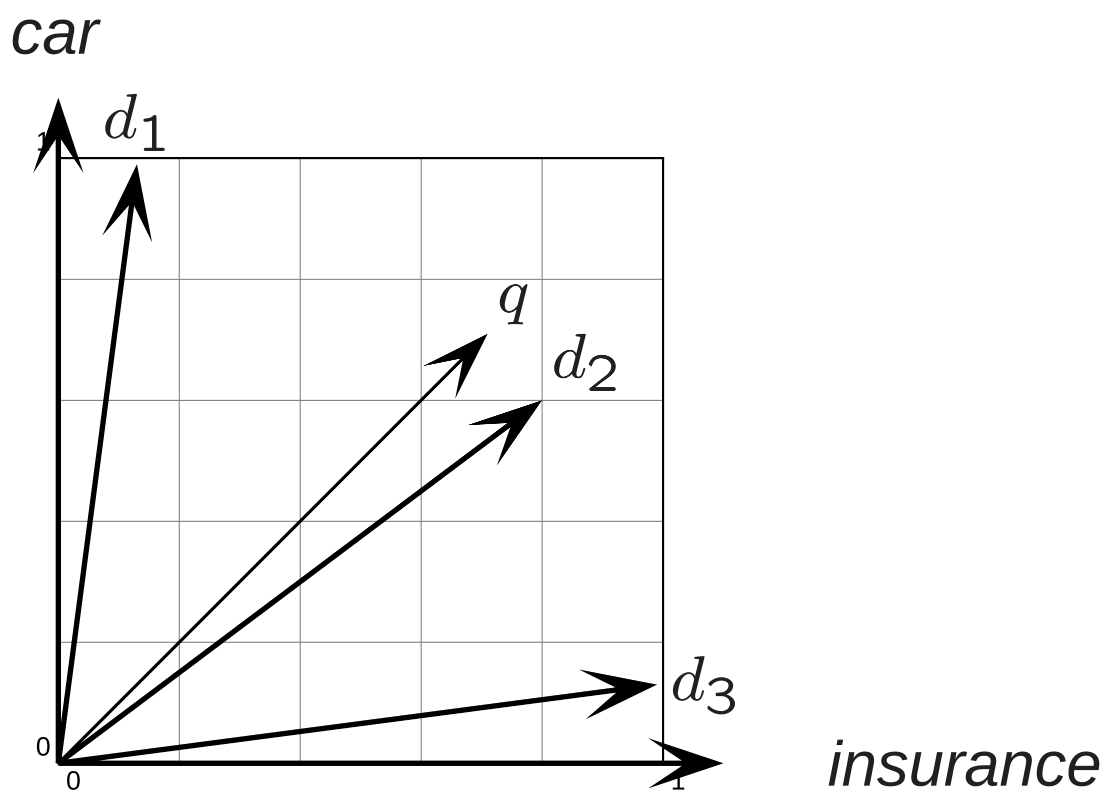
A vector can be normalized (i.e., given a length of 1) by dividing each of its components by the vector's length

This maps vectors onto the unit circle by dividing through by lengths:

Then, $|\vec{x}| = \sqrt{\sum_{i=1}^{n} x_i^2} = 1$

If we didn't normalize vectors, long documents would be more similar to each other! (By the dot product measure.)

The Vector Space Model (normalized vectors)



Cosine measure of similarity (angle between two vectors)

$$\cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

For normalized vectors, the cosine is simply the dot product: $\cos(\vec{x},\vec{y}) = \vec{x}\cdot\vec{y}$

Developed in SMART system (Salton) and standardly used by TREC participants

Euclidean distance between vectors

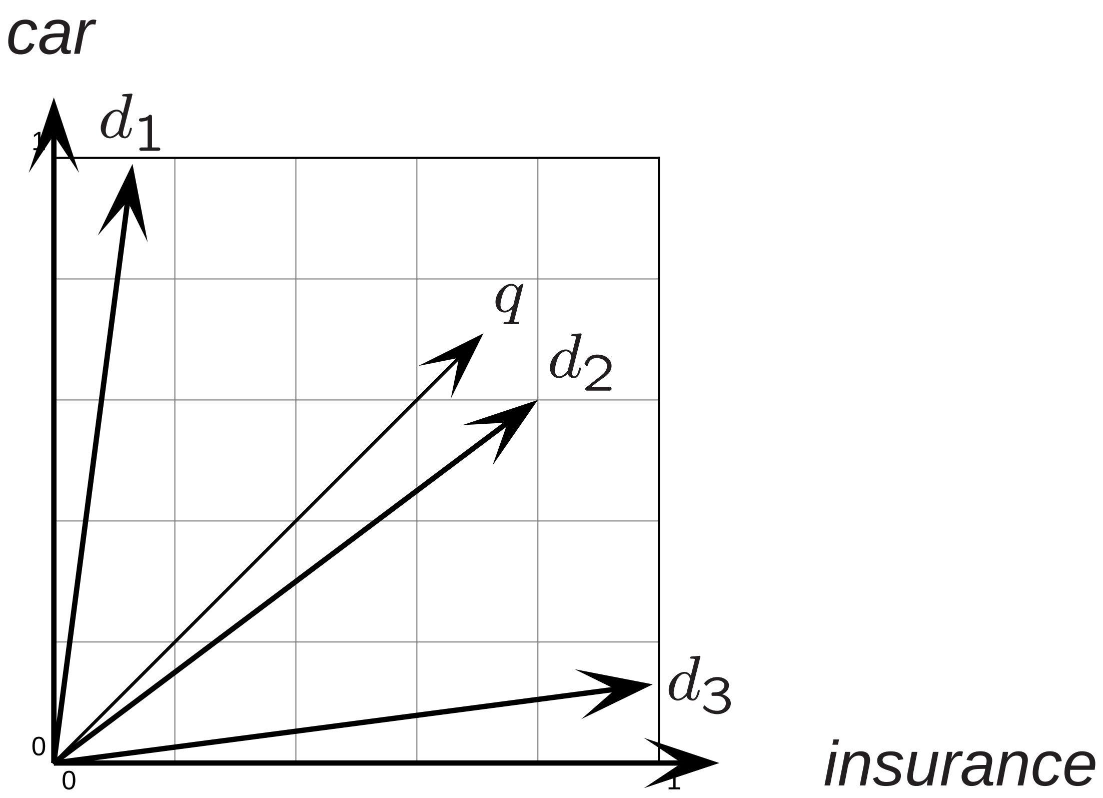
Euclidean distance:

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

For normalized vectors, Euclidean distance gives the same closeness ordering as the cosine measure (simple exercise).

The Vector Space Model: Doing a query

We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.



Measuring performance: The 2×2 contingency matrix

Black-box or “end-to-end” system performance

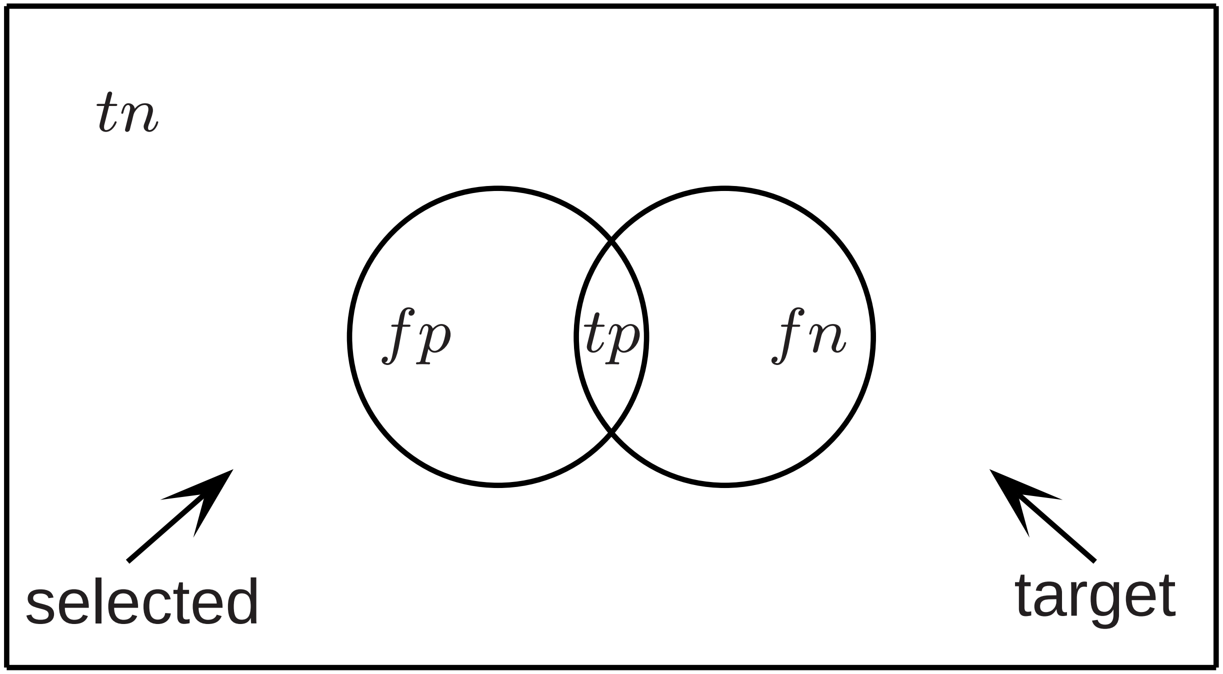
	Actual	
System	target	¬ target
selected	tp	fp
¬selected	fn	tn

Accuracy = $(tp + tn)/N$

Error = $(fn + fp)/N$ = 1 $-$ Accuracy

Why is this measure inadequate for IR?

The motivation for precision and recall



Accuracy is not a useful measure when the target set is a tiny fraction of the total set.

Precision is defined as a measure of the proportion of selected items that the system got right:

$$\text{precision } P = \frac{tp}{tp + fp}$$

Recall is defined as the proportion of the target items that the system selected:

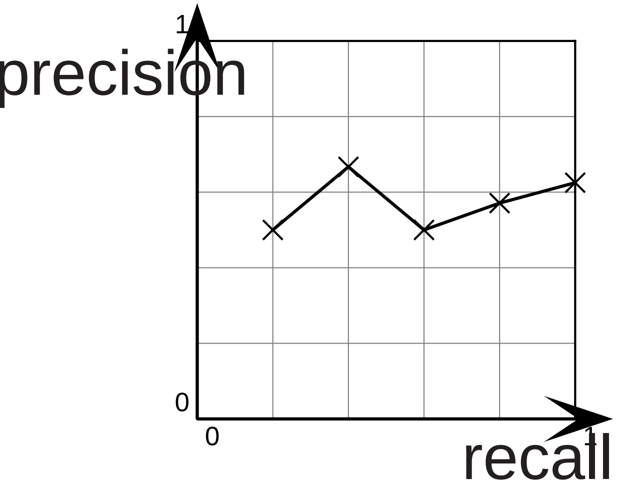
$$\text{recall } R = \frac{tp}{tp + fn}$$

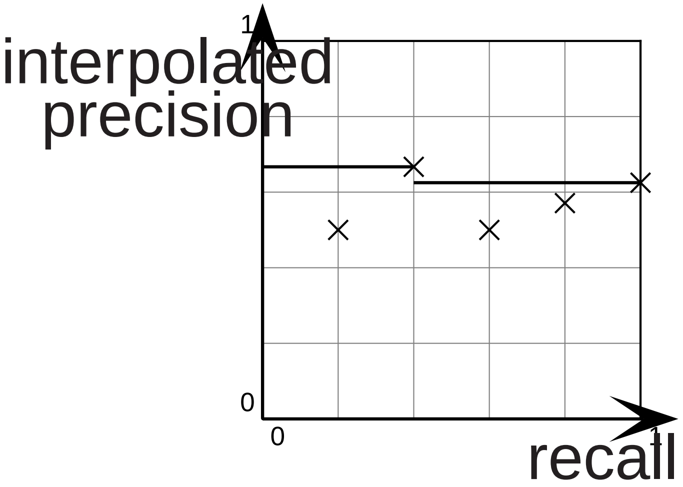
These two measures allow us to distinguish between excluding target items and returning irrelevant items.

They still require human-made “gold standard” judgements.

Evaluation of *ranked*	Ranking 1	Ranking 2	Ranking 3
results	d1: ✓	d10: ×	d6: ×
	d2: ✓	d9: ×	d1: ✓
	d3: ✓	d8: ×	d2: ✓
	d4: ✓	d7: ×	d10: ×
	d5: ✓	d6: ×	d9: ×
	d6: ×	d1: ✓	d3: ✓
	d7: ×	d2: ✓	d5: ✓
	d8: ×	d3: ✓	d4: ✓
	d9: ×	d4: ✓	d7: ×
	d10: ×	d5: ✓	d8: ×
precision at 5	1.0	0.0	0.4
precision at 10	0.5	0.5	0.5
uninterpolated av. prec.	1.0	0.3544	0.5726
interpolated av. prec. (11-point)	1.0	0.5	0.6440

Interpolated average precision





Combined measures

If we can decide on the relative importance of precision and recall, then they can be combined into a single measure.

Does one just average them? Bad, because the measures aren't independent.

What's a sensible model?

Rijsbergen (1979:174) defines and justifies the usually used alternative, the F measure
(see http://www.dcs.gla.ac.uk/Keith/Preface.html).

Assumptions:

- Interested in document proportions not absolute numbers
- Decreasing marginal effectiveness of recall and precision, e.g.:

  $$(R+1, P-1) > (R, P)$$

  but

  $$(R+1, P) > (R+2, P-1)$$

  Makes curves convex towards origin.

The F measure (where $F = 1 - E$)

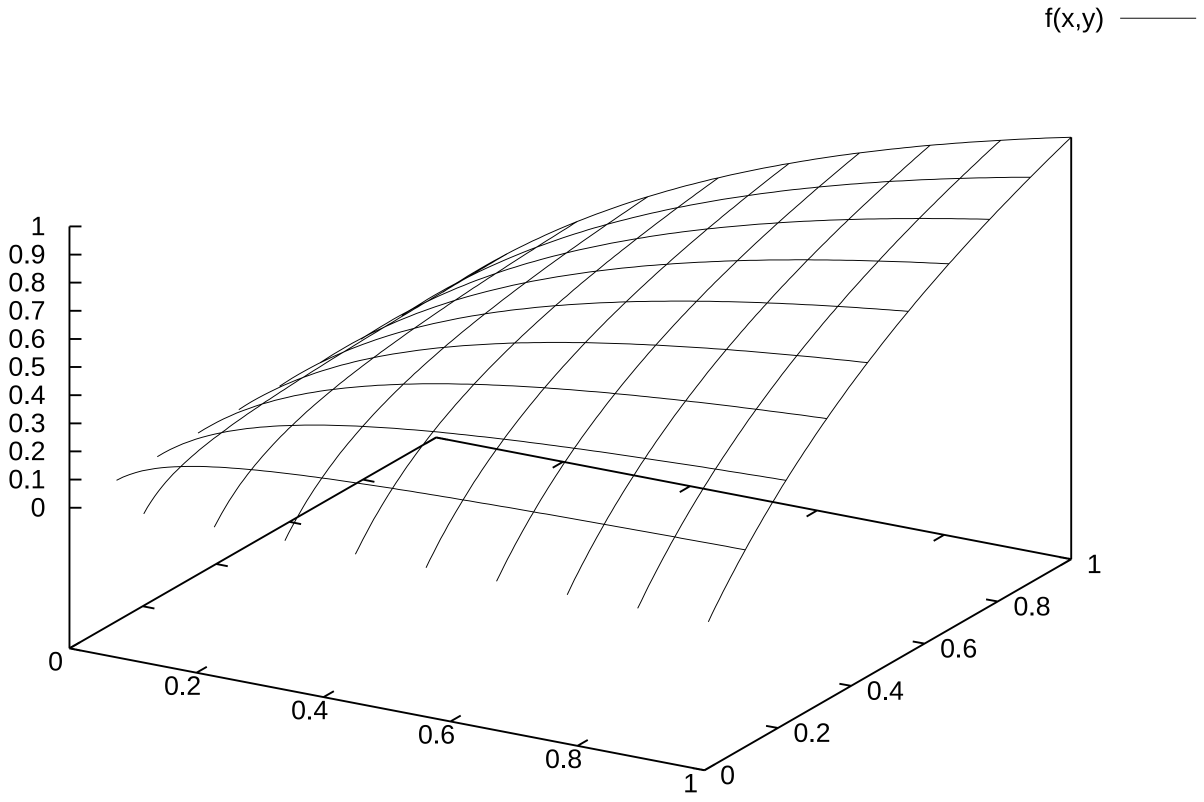
$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)\frac{1}{R}}$$

where P is precision, R is recall and α weights precision and recall. (Or in terms of β, where $\alpha = 1/(\beta^2 + 1)$.)

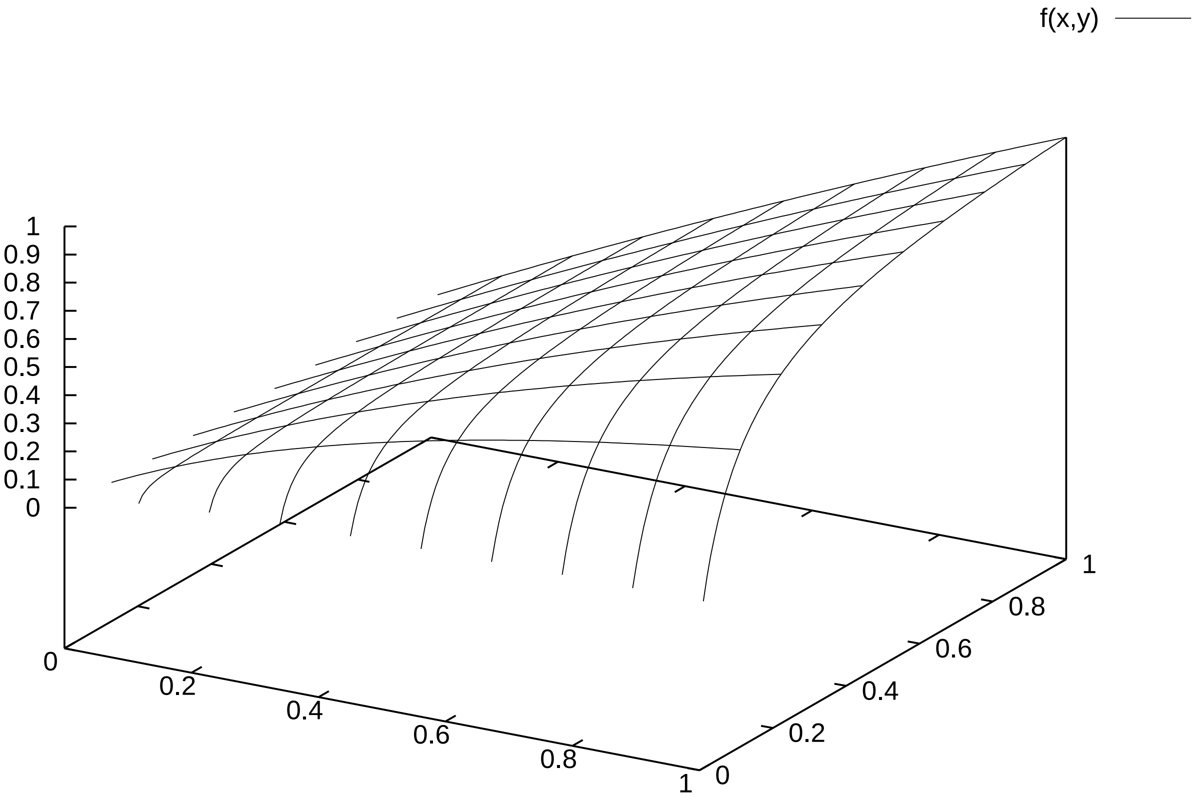
A value of $\alpha = 0.5$ is often chosen.

$$F = \frac{2PR}{R + P}$$

The F measure ($\alpha = 0.5$)



The F measure ($\alpha = 0.9$)



Term weighting

- Simplest term (vector component) weightings are:
  - count of number of times word occurs in document
  - binary: word does or doesn't occur in document
- However, general experience is that a document is a better match if a word occurs three times than once, but not a three times better match.
- This leads to a series of weighting functions that damp the term weighting, e.g., $1 + \log(x)$, $x > 0$, or $\sqrt{x}$.
- This is a good thing to do, but still imperfect: it doesn't capture that the occurrence of a term in a document is more important if that term does not occur in many other documents.

Example of term frequency (from Steven Bird)

- Documents: Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*
- Terms: affection, jealous, gossip
- SAS: (115, 10, 2); PAP: (58, 7, 0); WH: (20, 11, 6)
- SAS: (0.996, 0.087, 0.017); PAP: (0.993, 0.120, 0.0); WH: (0.847, 0.466, 0.254)

$$\begin{aligned}\text{cos}(SAS, PAP) &= .996 \times .993 + .087 \times .120 + .017 \times 0.0 = 0 \\ \text{cos}(SAS, WH) &= .996 \times .847 + .087 \times .466 + .017 \times .254 =\end{aligned}$$

Document frequency: indicates informativeness

Word	Collection Frequency	Document Frequency
insurance	10440	3997
try	10422	8760

Adding this in (one of many ways):

$$\text{weight}(i,j) = \begin{cases} (1+\log(\mathtt{tf}_{i,j}))\log\frac{N}{\mathtt{df}_i} & \text{if } \mathtt{tf}_{i,j} \geq 1 \\ 0 & \text{if } \mathtt{tf}_{i,j} = 0 \end{cases}$$

Document frequency weighting is only possible if we have a static collection. Sometimes we don't – it's dynamically created.

Term weighting summary

term frequency $\mathtt{tf}_{i,j}$ number of occurrences of w_i in d_j

document frequency $\mathtt{df}_i$ number of documents in the collection that w_i occurs in

collection frequency $\mathtt{cf}_i$ total number of occurrences of w_i in the collection

Note that $\mathtt{df}_i \leq \mathtt{cf}_i$ and that $\sum_j \mathtt{tf}_{i,j} = \mathtt{cf}_i$.

- $tf.idf$ weighting: term frequency times inverse document frequency. This is the standard in IR (but it is really a family of methods depending on how each figure is scaled)

Language and implementation problems

- Traditional IR relies on word matching. There are two fundamental query matching problems:
  - synonymy (image, likeness, portrait, facsimile, icon)
  - polysemy (port: harbor, fortified wine, computer jack, . . .)
- Effective indexing needs scale, and accuracy
- Dimensionality reduction techniques address part of the first problem, while remaining fairly efficient

Latent Semantic Indexing (LSI)

- *Approach:* Treat word-to-document association data as an unreliable estimate of a larger set of applicable words lying on ‘latent’ dimensions.
- *Goal:* Cluster similar documents which may share no terms in a low-dimensional subspace (improve recall).
- *Preprocessing:* Compute low-rank approximation to the original term-by-document (sparse) matrix
- *Vector Space Model:* Encode terms and documents using factors derived from SVD
- *Evaluation:* Rank similarity of terms and docs to query via Euclidean distances or cosines

Singular Value Decomposition Encoding

- Computes a truncated SVD of the document-term matrix, using the singlular vectors as axes of the lower dimensional space
- A_k is the best rank-k approximation to the term-by-document matrix A
- Want minimum number of factors (k) that discriminates most concepts
- In practice, k ranges between 100 and 300 but could be much larger.
- Choosing optimal k for different collections is challenging.

Strengths and weaknesses of LSI

- Strong formal framework. Completely automatic. No stemming required. Allows misspellings
- Can be used for multilingual search (Flournoy & Peters Stanford, Landauer Colorado, Littman Duke)
- ‘Conceptual IR’ recall improvement: one can retrieve relevant documents that do not contain any search terms

- Calculation of LSI is expensive
- Continuous normal-distribution-based methods not really appropriate for count data
- Often improving precision is more important: need query and word sense disambiguation

Cosine, Euclidean, and Assignment 6

- Just so there are no misunderstandings. . .
- In class on Monday, I suggested that one might pre-normalize the vectors in a vector space model
- However, for the assignment
  - The data structures store integer counts and lengths, they're not set up for pre-normalizing
  - We want you to compare the cosine measure and Euclidean distance, and this distinction only makes sense if you don't pre-normalize (as we discussed)

Similarity measures for binary vectors

- Rather than working with weighted term frequencies, a simpler model just uses binary (0/1) vectors
- Matching coefficient (like dot product): $|X \cap Y|$
- Jaccard coefficient: $\frac{|X\cap Y|}{|X\cup Y|}$

- Jaccard is one of several coefficients that normalize the degree of matching (others are Dice, overlap, . . .).
- In practice it is often a good one because it penalizes matches of a small percentage of shared entries relative to the document size more than other methods

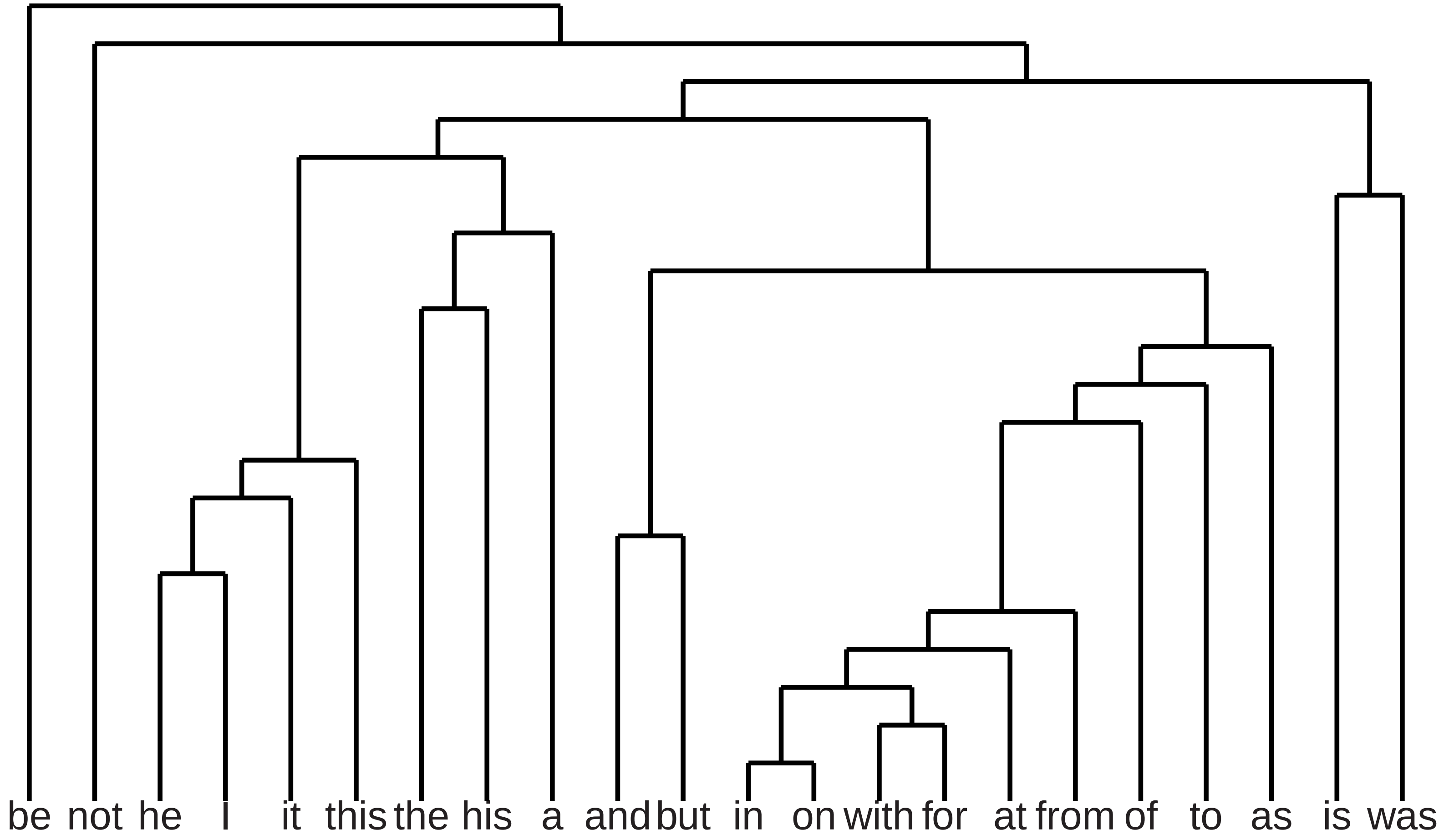
Bored. Anything else I might try?

- The L_1 metric (absolute distance not squared)

$$L_1(x,y) = \sum_i \left| \frac{x_i}{\sum_j x_j} - \frac{y_i}{\sum_j y_j} \right|$$

- I'm not sure how well it'd work, but might be worth a shot. . . .

A single-link clustering of 22 frequent English words represented as a dendrogram



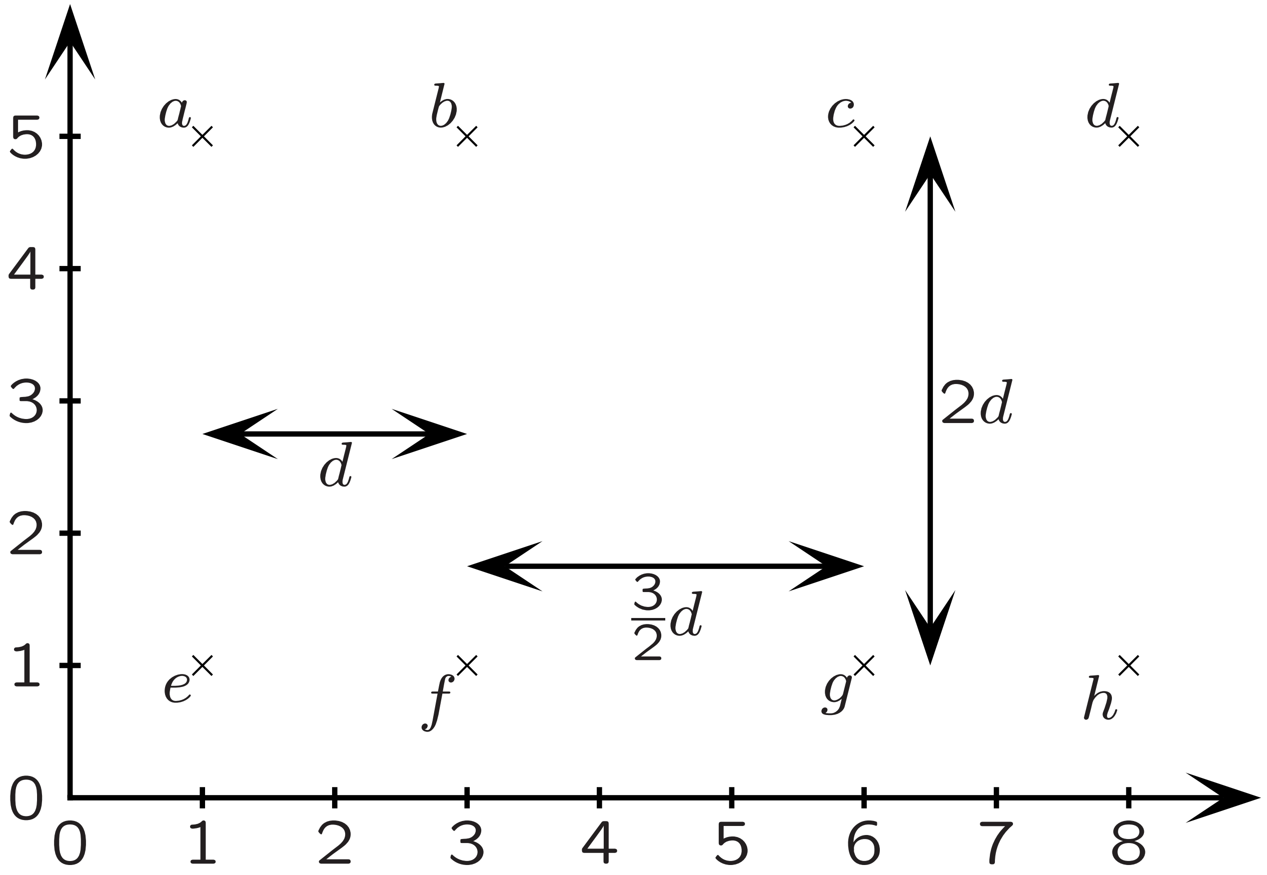
Bottom-up hierarchical clustering

```
Given:  a set $\mathcal{X} = \{x_1, \ldots x_n\}$ of objects
        a function sim: $\mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \rightarrow \mathbb{R}$
for $i := 1$ to $n$ do
    $c_i := \{x_i\}$
end
$C := \{c_1, \ldots , c_n\}$
$j := n + 1$
while $|C| > 1$
        $(c_{n_1}, c_{n_2}) := \text{arg max}_{(c_u,c_v) \in C \times C} \text{sim}(c_u, c_v)$
        $c_j = c_{n_1} \cup c_{n_2}$
        $C := C \backslash \{c_{n_1}, c_{n_2}\} \cup \{c_j\}$
        $j := j + 1$
```

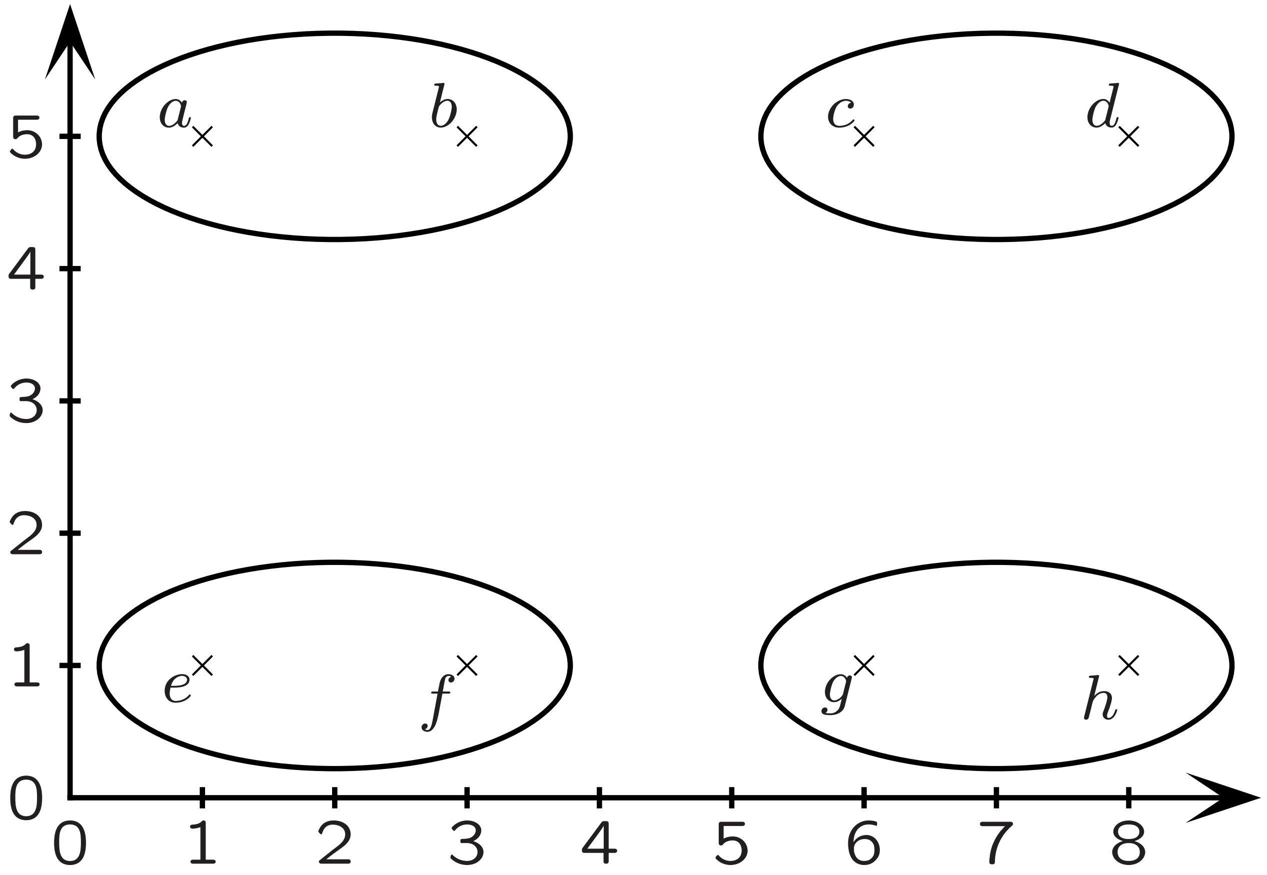
Similarity functions used in clustering

Function	Definition
single link	similarity of two most similar members
complete link	similarity of two least similar members
group-average	average similarity between members

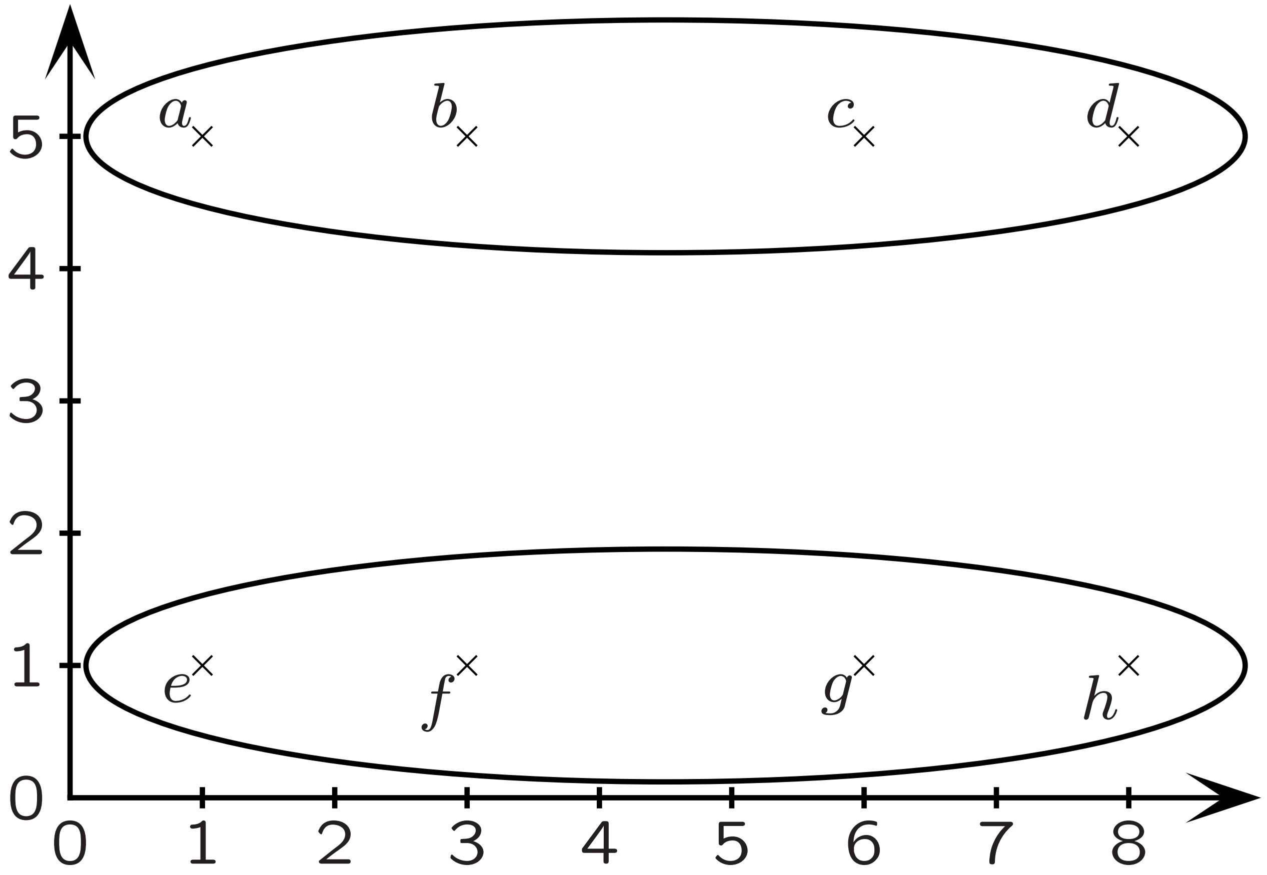
A cloud of points in a plane



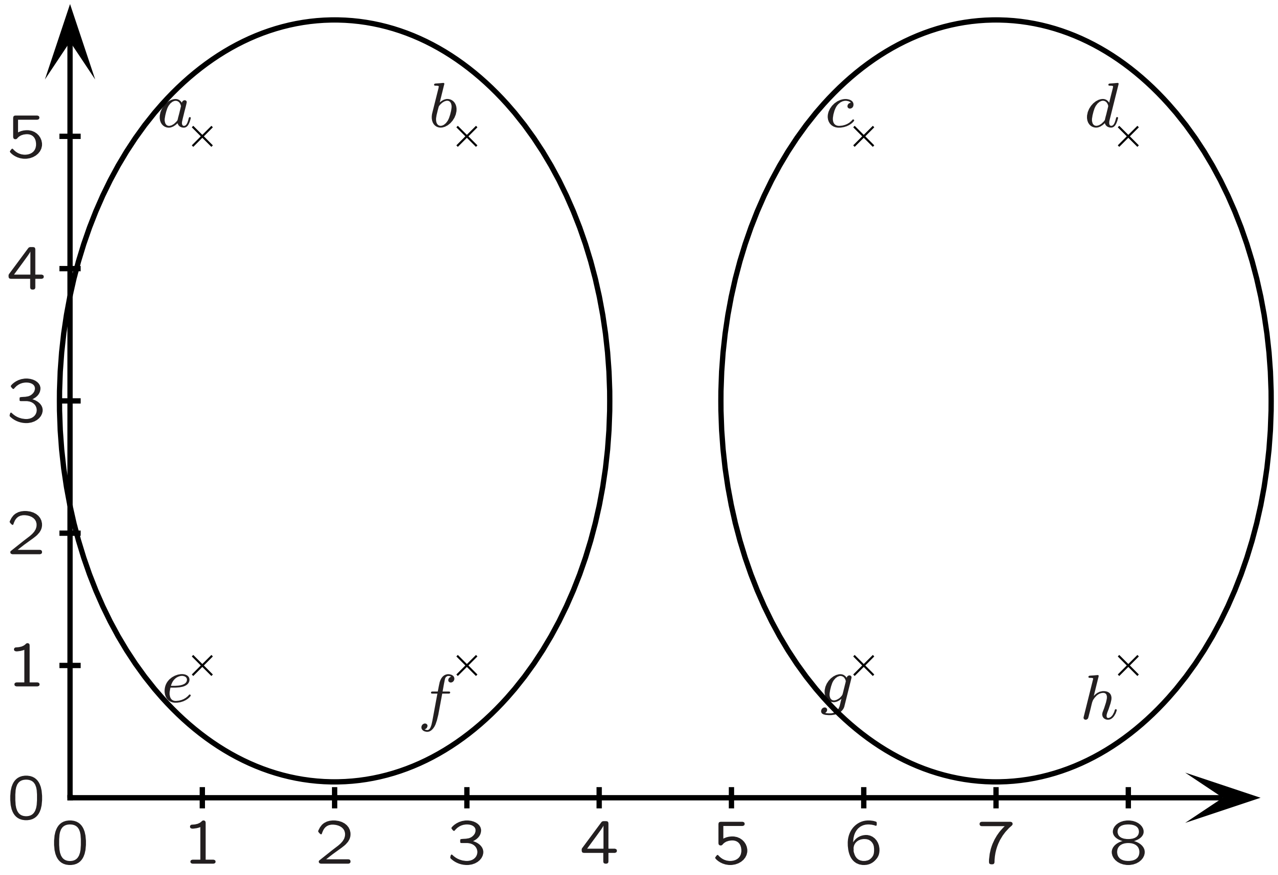
Intermediate clustering of the points



Single-link clustering of the points



Complete-link clustering of the points



The EM algorithm

based on a presentation by Dan Klein

- A very general and well-studied algorithm
- I cover only the specific case we use in this course: maximum-likelihood estimation for models with discrete hidden variables
- (For continuous case, sums go to integrals; for MAP estimation, changes to accommodate prior)
- As an easy example we estimate parameters of an n-gram mixture model
- For all details of EM, try McLachlan and Krishnan (1996)

Maximum-Likelihood Estimation

- We have some data X and a probabilistic model $P(X|\Theta)$ for that data
- X is a collection of individual data items x
- Θ is a collection of individual parameters θ.
- The *maximum-likelihood estimation* problem is, given a model $P(X|\Theta)$ and some actual data X, find the Θ which makes the data most likely:

$$\Theta' = \operatorname*{arg\,max}_{\Theta} P(X|\Theta)$$

- This problem is just an optimization problem, which we could use any imaginable tool to solve

Maximum-Likelihood Estimation

- In practice, it’s often hard to get expressions for the derivatives needed by gradient methods
- EM is one popular and powerful way of proceeding, but not the only way.
- Remember, EM is doing MLE

Finding parameters of a n-gram mixture model

- P may be a mixture of k pre-existing multinomials:

$$P(x_i|\Theta) = \sum_{j=1}^{k} \theta_j P_j(x_i)$$

$$\widehat{P}(w_3|w_1, w_2) = \theta_3 P_3(w_3|w_1, w_2) + \theta_2 P_2(w_3|w_2) + \theta_1 P_1(w_3)$$

- We treat the P_j as **fixed**. We learn by EM *only* the θ_j.

$$\begin{aligned} P(X|\Theta) &= \prod_{i=1}^{n} P(x_i|\Theta) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{k} P_j(x_i|\Theta_j) \end{aligned}$$

- $X = [x_1 \ldots x_n]$ is a sequence of n words drawn from a vocabulary V, and $\Theta = [\theta_1 \ldots \theta_k]$ are the mixing weights

EM

- EM applies when your data is incomplete in some way
- For each data item x there is some extra information y (which we don't know)
- The vector X is referred to as the the *observed data* or *incomplete data*
- X along with the completions Y is referred to as the *complete data*.
- There are two reasons why observed data might be incomplete:
  - It's really incomplete: Some or all of the instances really have missing values.
  - It's artificially incomplete: It simplifies the math to pretend there's extra data.

EM and Hidden Structure

- In the first case you might be using EM to “fill in the blanks” where you have missing measurements.
- The second case is strange but standard. In our mixture model, viewed generatively, if each data point x is assigned to a *single* mixture component y, then the probability expression becomes:

$$\begin{aligned} P(X,Y|\Theta) &= \prod_{i=1}^{n} P(x_i, y_i|\Theta) \\ &= \prod_{i=1}^{n} P_{y_i}(x_i|\Theta) \end{aligned}$$

Where $y_i \in \{1, ..., k\}$. $P(X,Y|\Theta)$ is called the *complete-data likelihood*.

EM and Hidden Structure

- Note:
  - the sum over components is gone, since y_i tells us which single component x_i came from. We just don't know what the y_i are.
  - our model for the *observed* data X involved the "unobserved" structures – the component indexes – all along. When we wanted the observed-data likelihood we summed out over indexes.
  - there are two likelihoods floating around: the observed-data likelihood $P(X|\Theta)$ and the complete-data likelihood $P(X,Y|\Theta)$. EM is a method for maximizing $P(X|\Theta)$.

EM and Hidden Structure

- Looking at completions is useful because finding

$$\Theta = \operatorname*{arg\,max}_{\Theta} P(X|\Theta)$$

  is hard (it's our original problem – maximizing products of sums is hard)

- On the other hand, finding

$$\Theta = \operatorname*{arg\,max}_{\Theta} P(X, Y|\Theta)$$

  would be easy – if we knew Y.

- The general idea behind EM is to alternate between maximizing Θ with Y fixed and "filling in" the completions Y based on our best guesses given Θ.

The EM algorithm

- The actual algorithm is as follows:

  Initialize Start with a guess at Θ – it may be a very bad guess

  Until tired

  E-Step Given a current, fixed Θ', calculate completions: $P(Y|X, \Theta')$

  M-Step Given fixed completions $P(Y|X, \Theta')$, maximize $\sum_Y P(Y|X, \Theta') \log P(X, Y|\Theta)$ with respect to Θ.

The EM algorithm

- In the E-step we calculate the likelihood of the various completions with our fixed Θ'.
- In the M-stem we maximize the expected log-likelihood of the complete data. That's not the same thing as the likelihood of the observed data, but it's close
- The hope is that even relatively poor guesses at Θ, when constrained by the actual data X, will still produce decent completions
- Note that "the complete data" changes with each iteration

EM made easy

- Want: Θ which maximizes the data likelihood

$$\begin{aligned} L(\Theta) &= P(X|\Theta) \\ &= \textstyle\sum_Y P(X,Y|\Theta) \end{aligned}$$

- The Y ranges over all possible completions of X. Since X and Y are vectors of independent data items,

$$L(\Theta) = \prod_x \sum_y P(x,y|\Theta)$$

- We don't want a product of sums. It'd be easy to maximize if we had a product of products.
- Each x is a data item, which is broken into a sum of sub-possibilities, one for each completion y. We want to make each completion be like a mini data item, all multiplied together with other data items.

EM made easy

- Want: a product of products
- Arithmetic-mean-geometric-mean (AMGM) inequality says that, if $\sum_i w_i = 1$,

$$\prod_i z_i^{w_i} \leq \sum w_i z_i$$

- In other words, arithmetic means are larger than geometric means (for 1 and 9, arithmetic mean is 5, geometric mean is 3)
- This equality is promising, since we have a sum and want a product
- We can use $P(x, y|\Theta)$ as the z_i, but where do the w_i come from?

EM made easy

- The answer is to bring our previous guess at Θ into the picture.
- Let's assume our old guess was Θ'. Then the old likelihood was

$$L(\Theta') = \prod_x P(x|\Theta')$$

- This is just **a constant**. So rather than trying to make $L(\Theta)$ large, we could try to make the relative change in likelihood

$$R(\Theta|\Theta') = \frac{L(\Theta)}{L(\Theta')}$$

large.

EM made easy

- Then, we would have

$$\begin{aligned}
R(\Theta|\Theta') &= \frac{\prod_x \sum_y P(x,y|\Theta)}{\prod_x P(x|\Theta')} \\
&= \prod_x \frac{\sum_y P(x,y|\Theta)}{P(x|\Theta')} \\
&= \prod_x \sum_y \frac{P(x,y|\Theta)}{P(x|\Theta')} \\
&= \prod_x \sum_y \frac{P(x,y|\Theta)}{P(x|\Theta')} \frac{P(y|x,\Theta')}{P(y|x,\Theta')} \\
&= \prod_x \sum_y P(y|x,\Theta') \frac{P(x,y|\Theta)}{P(x,y|\Theta')}
\end{aligned}$$

- Now that's promising: we've got a sum of relative likelihoods $P(x,y|\Theta)/P(x,y|\Theta')$ weighted by $P(y|x,\Theta')$.

EM made easy

- We can use our identity to turn the sum into a product:

$$\begin{aligned} R(\Theta|\Theta') &= \prod_x \sum_y P(y|x,\Theta')\frac{P(x,y|\Theta)}{P(x,y|\Theta')} \\ &\geq \prod_x \prod_y \left[\frac{P(x,y|\Theta)}{P(x,y|\Theta')}\right]^{P(y|x,\Theta')} \end{aligned}$$

- Θ, which we're maximizing, is a variable, but Θ' is just a constant. So we can just maximize

$$Q(\Theta|\Theta') = \prod_x \prod_y P(x,y|\Theta)^{P(y|x,\Theta')}$$

EM made easy

- We started trying to maximize the likelihood $L(\Theta)$ and saw that we could just as well maximize the relative likelihood $R(\Theta|\Theta') = L(\Theta)/L(\Theta')$. But $R(\Theta|\Theta')$ was still a product of sums, so we used the AMGM inequality and found a quantity $Q(\Theta|\Theta')$ which was (proportional to) a lower bound on R. That's useful because Q is something that is easy to maximize, if we know $P(y|x, \Theta')$.

The EM Algorithm

- So here's EM, again:
  - Start with an initial guess Θ'.
  - Iteratively do

    E-Step Calculate $P(y|x, \Theta')$

    M-Step Maximize $Q(\Theta|\Theta')$ to find a new Θ'
- In practice, maximizing Q is just setting parameters as relative frequencies in the complete data – these are the maximum likelihood estimates of Θ

The EM Algorithm

- The first step is called the E-Step because we calculate the expected likelihoods of the completions.
- The second step is called the M-Step because, using those completion likelihoods, we maximize Q, which hopefully increases R and hence our original goal L
- The expectations give the shape of a simple Q function for that iteration, which is a lower bound on L (because of AMGM). At each M-Step, we maximize that lower bound
- This procedure increases L at every iteration until Θ' reaches a local extreme of L.
- This is because successive Q functions are better approximations, until you get to a (local) maxima

EM algorithm

- We initalize all parameters (randomly or intelligently)
- E-step: We work out the expectation of the hidden variables, given the current parameters for the model
- M-step: (Assuming these expectations are right), we calculate the maximum likelihood estimates for the parameters
- Repeat until convergence

(Dempster, Laird, and Rubin 1977). Guaranteed to not diminish the likelihood of the data according to the model. Feasible for a large range of models (where E and M step can be computed).

Text Categorization

FSNLP, chapter 16

Christopher Manning and
Hinrich Schütze

Text categorization/classification

- Take a document and assign it a label representing its topic
- Have some predefined categories for texts
  - Classic example: decide if a newspaper article is about politics, business, or sports
  - Predefined categories for news items on newswires – Reuters categories
  - ACM keywords
  - Yahoo! or Open Directory classes (extra complexity: hierarchical)
  - UMLS (Unified Medical Language System) MeSH codes
- Generally approached using a bag of words model

Text categorization/classification

- There are many other uses for the same technology:
  - Is this page a laser printer product page?
  - Does this company accept overseas orders?
  - What kind of job does this job posting describe?
  - What kind of position does this list of responsibilities describe?
  - What position does this this list of skills best fit?
  - Word sense disambiguation can actually be thought of as text (here, context) categorization:
    - Is this the “computer” or “harbor” sense of port?

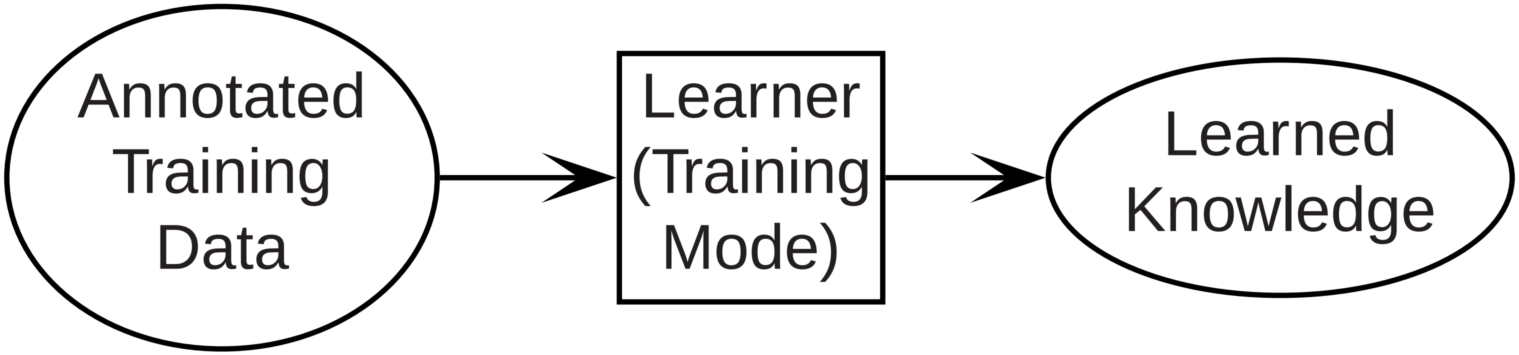
Text categorization/classification

- This can be thought of as working out the “meaning” of documents in a very primitive sense: for instance, a news article “means” *Europe* and *politics*
- Most of the companies that claim to be understanding your document, doing eCRM processing of your users complaining emails, etc., are essentially offering text categorization at the moment:
  - Kana
  - Autonomy
  - Bhasha
  - Banter

NLP as a classification problem

- Central to recent advances in NLP has been reconceptualizing NLP as a machine learning classification problem
- We – preferably someone else – hand-annotate data, and then learn using standard ML methods
- Annotated data items are feature vectors $\vec{x}_i$ with a classification c_i.
- Our job is to assign an unannotated data item $\vec{x}$ to one of the classes c_k (or possibly to the doubt $\mathcal{D}$ or outlier $\mathcal{O}$ categories – though in practice rarely used).

Training procedure

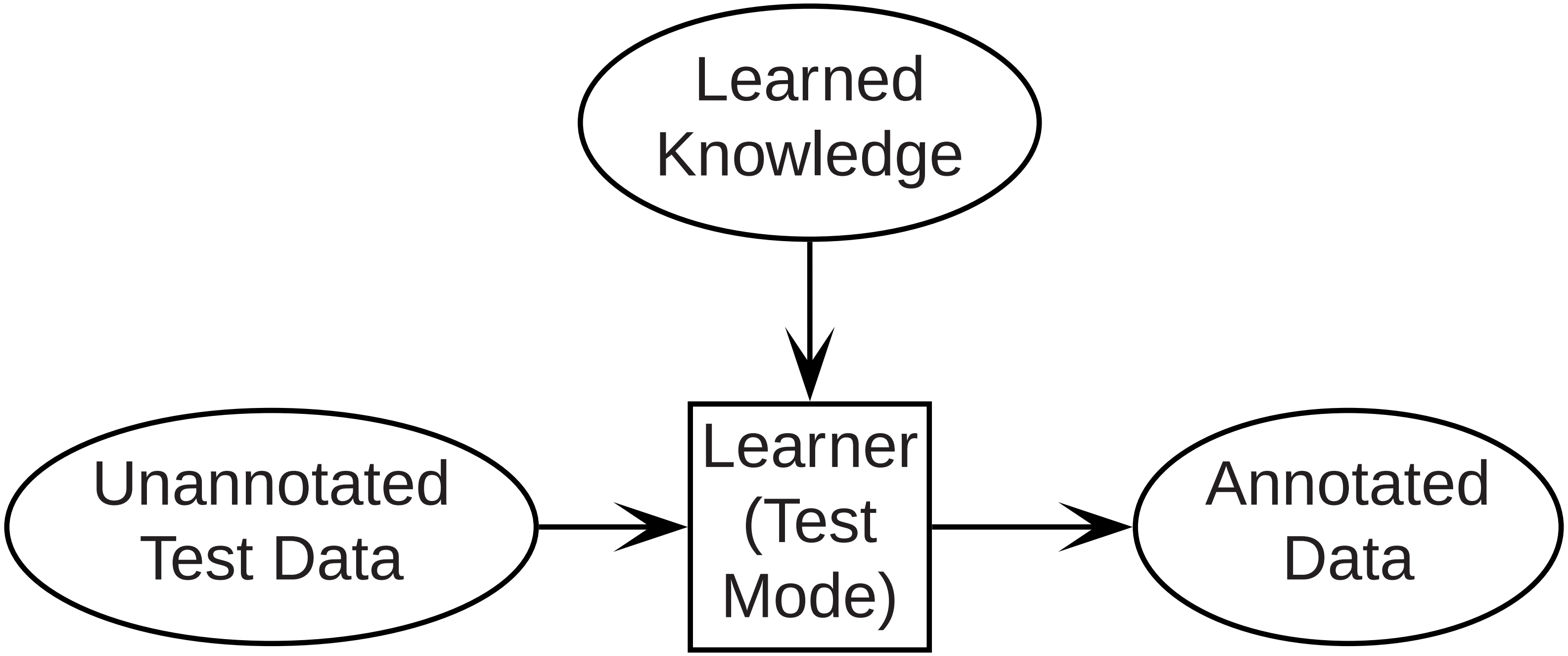


Annotated Training Data

Learner (Training Mode)

Learned Knowledge

Testing = deployed procedure



Learned Knowledge

Unannotated Test Data

Learner (Test Mode)

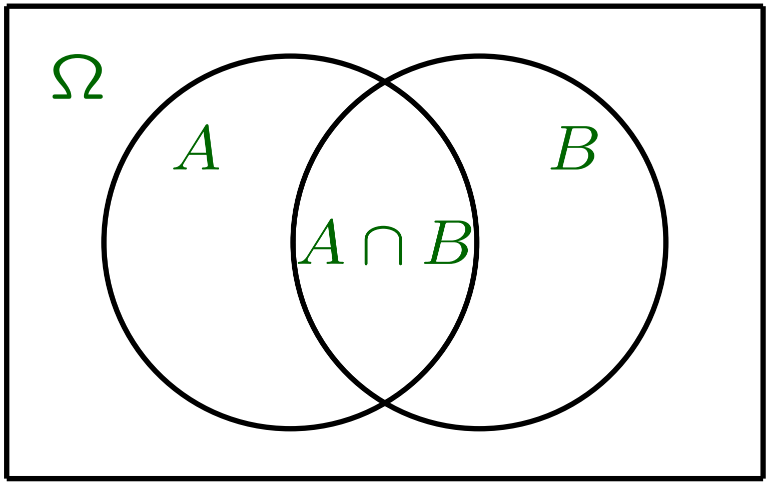
Annotated Data

Statistical decision theory

- The proportion of items in each class c_k has a prior probability π_k.
- Feature vectors $\vec{x}$ have distribution $P_k(\vec{x})$ for class c_k
- Assume (falsely) that these are known
- **Bayes optimal classifier / Bayes (decision) rule**:
  - Simple version: For an item $\vec{x}$, assign class $\boldsymbol{c = \arg\max_{c_k} P(c_k|\vec{x})}$
  - More complex versions put in explicit loss functions for making different kinds of mistakes, and decide $\mathcal{D}$ if no class is likely.

Bayesian classification in WSD

- Vectors $\vec{x}$ are some representation of the context of use of a word
- Problem: We usually don't know $P(c_k|\vec{x})$
- Partial solution: we break it up by Bayes' rule



$$P(B|A) = P(A \cap B)/P(A)$$

$$P(A \cap B) = P(B)P(A|B)$$

Bayes' theorem

$$P(B|A) = \frac{P(A,B)}{P(A)} = \frac{P(A|B)P(B)}{P(A)}$$

If $A \subseteq \cup_i B_i$ and the B_i are disjoint, then (sum rule):

$$P(A) = \sum_i P(A,B_i) = \sum_i P(A|B_i)P(B_i)$$

Bayes' theorem: If $A \subseteq \cup_{i=1}^{n} B_i$, $P(A) > 0$, and $B_i \cap B_j = \emptyset$ for $i \neq j$:

Likelihood Prior

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)} = \frac{P(A|B_j) \times P(B_j)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)}$$

Posterior Normalizing term

A simple but OK approach: Naive Bayes WSD

- $\vec{x}$ is our context (something like a 100 word window)
- c_k is a sense of the word

$$\begin{aligned}
\text{Choose } c' &= \arg\max_{c_k} P(c_k|\vec{x}) \\
&= \arg\max_{c_k} \frac{P(\vec{x}|c_k)}{P(\vec{x})} P(c_k) \\
&= \arg\max_{c_k} P(\vec{x}|c_k) P(c_k) \\
&= \arg\max_{c_k} [\log P(\vec{x}|c_k) + \log P(c_k)] \\
&= \arg\max_{c_k} \left[\sum_{v_j \text{ in } \vec{x}} \log P(v_j|c_k) + \log P(c_k) \right]
\end{aligned}$$

Naive Bayes Text Categorization

- $\vec{x}$ is our document, a list of words
- c_k is the hidden class

$$\begin{aligned} c' &= \operatorname*{arg\,max}_{c_k} P(c_k|\vec{x}) \\ &= \operatorname*{arg\,max}_{c_k} \frac{P(\vec{x}|c_k)}{P(\vec{x})} P(c_k) \\ &= \operatorname*{arg\,max}_{c_k} P(\vec{x}|c_k) P(c_k) \\ &= \operatorname*{arg\,max}_{c_k} [\log P(\vec{x}|c_k) + \log P(c_k)] \end{aligned}$$

Naive Bayes models

- The *Naive Bayes assumption* is that the attributes used for description are all conditionally independent:

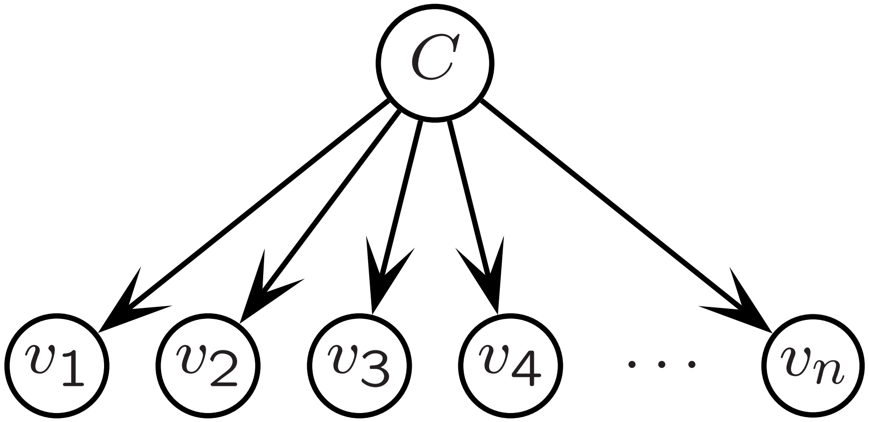
  Naive Bayes assumption

  $P(\vec{x}|c_k) = P(\{v_j|v_j \text{ in } \vec{x}\}|c_k) = \Pi_{v_j \text{ in } \vec{x}} P(v_j|c_k)$
- This is commonly referred to as the *bag of words* assumption
- **Decision rule for Naive Bayes**

  Decide c' if $c' = \arg\max_{c_k} [\log P(c_k) + \sum_{v_j \text{ in } \vec{x}} \log P(v_j|c_k)]$
- Note that there are two Naive Bayes models (McCallum and Nigam 1998)

Two Naive Bayes models: Multinomial

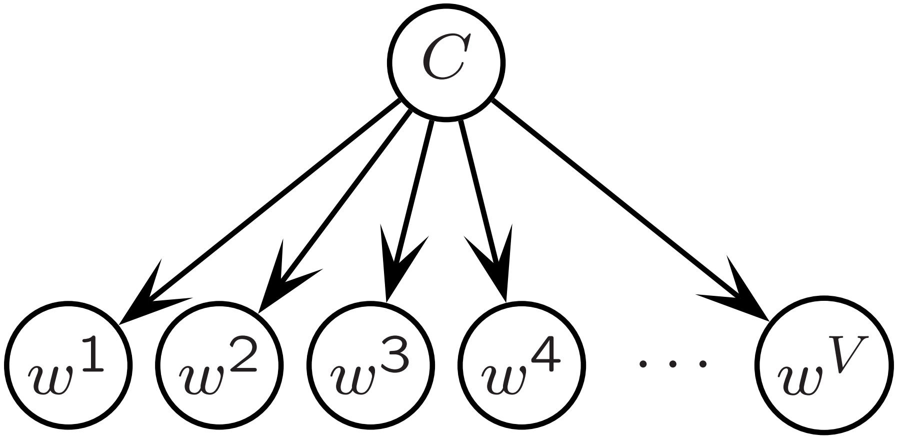
- v_j is word j of the context



- Model of Gale et al. (1992) (for WSD). Usual in StatNLP.
- The CPT for each multinomial is identical (tied parameters)
- The multinomial is estimated over the whole vocabulary.

Two Naive Bayes models: Bernoulli

- w^j is word (type) j of the vocabulary of features

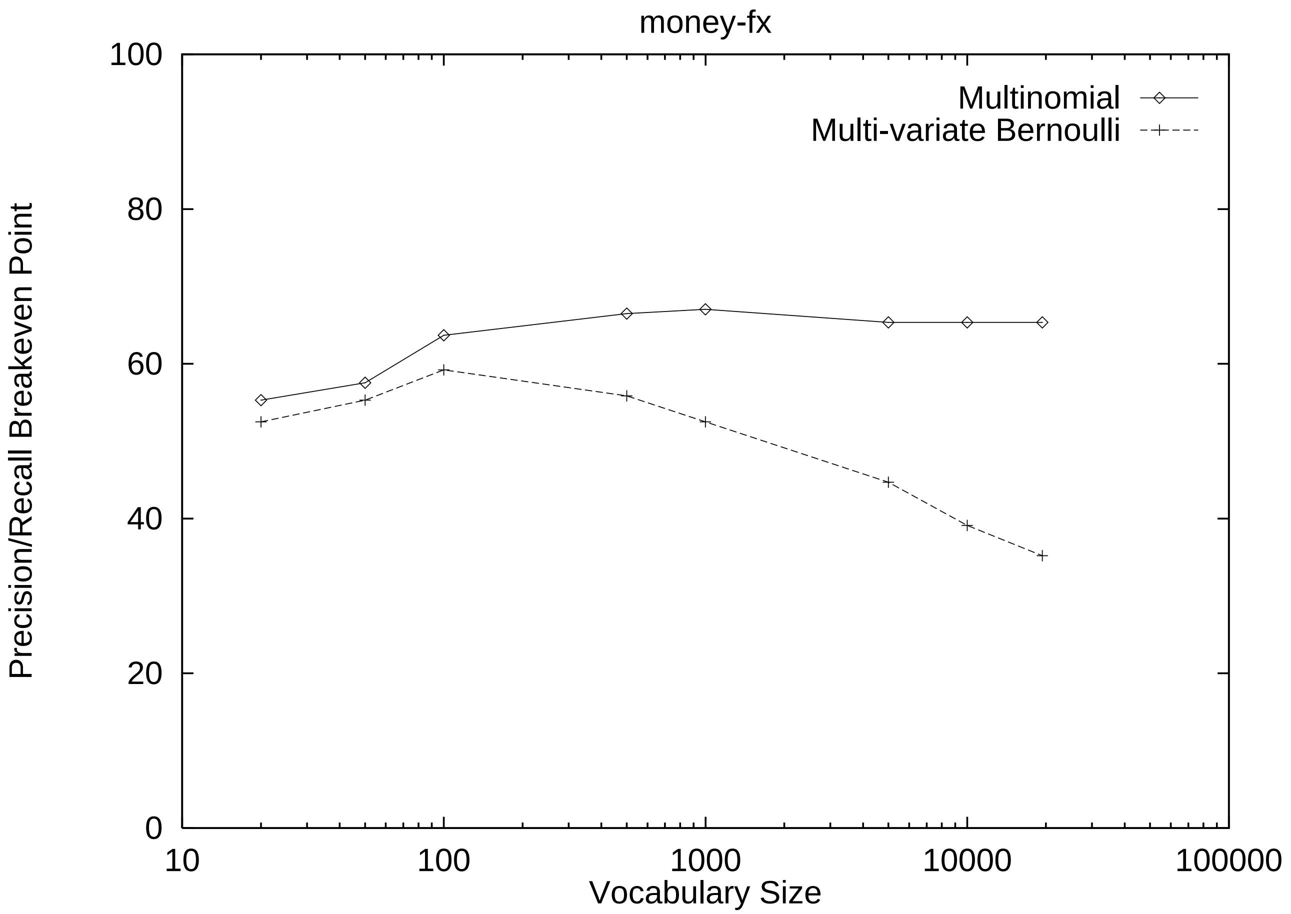


- Each feature is binary yes/no (though could be count/range)
- Model normally presented in the graphical models literature
- Generally (but not always) performs worse
- Requires careful and aggressive feature selection

Naive Bayes models

- Feature selection: commonly count, χ^2 or mutual information, but there are methods to find non-overlapping features (Koller and Sahami 1996). Only important/relevant in Bernoulli model.
- Naive Bayes is simple, but often about as good as there is (Friedman 1997; Domingos and Pazzani 1997)
- There are successful more complex probabilistic classifiers, particularly TAN – Tree Augmented Naive Bayes (van Rijsbergen 1979; Friedman and Goldszmidt 1996)
- One can get value from varying context size according to type of word being disambiguated (commonly: noun is big context, verb is small context)

‘Typical’ McCallum and Nigam (1998) result: Reuters Money-FX category



Text classification results

- You’ve got to be careful that you’re comparing apples and apples
- One source of careful recent comparative studies is Yang (1999a *Information Retrieval*, 1999b *SIGIR*)
- Results on Reuters for which several versions have been prepared commonly differed by as great a percentage between *versions* as between *methods*.
- Many comparisons were invalidated by data differences.
- (Should do statistical significance tests, confidence intervals, ...)

Averaging

- Two ways of computing an average give very different results!
- *Micro-averaging* make a single contingency table for all the data by summing the scores in each cell for all categories, and evaluate it
  - Gives equal weight to each object in the collection
  - Score is dominated by large categories
- *Macro-averaging:* compute measure for each category contingency table separately and then average the evaluation measure over categories
  - Gives equal weight to each category
  - Gives a better sense of the quality of classification across all categories.

Yang SIGIR results

- Micro-averaging (dominated by performance on large categories):
  - $SVM > kNN \gg \{LLSF, NNet\} \gg NB$
  - F: 0.860, 0.857, 0.850, 0.829, 0.80
- Macro-averaging
  - $\{SVM, kNN, LLSF\} \gg \{NNet, NB\}$
  - F: 0.525, 0.524, 0.501, 0.377, 0.389
- Yang doesn't test MaxEnt (loglinear) models, but other results (Nigam et al. 1999) show it doing much better than NB, so it appears to belong in the top group.

Memory-Based Learning (MBL)/kNN

- Case-based, exemplar-based or instance-based learning
- A non-parametric method: no abstract model is created
- Training: just store the data (indexed!)
- Generalization: target function is estimated locally.
- The method is not aware of the global shape of the data, but does not make any assumptions about its shape
- Non-linear classifier
- “The cost for computation is decreasing roughly by a factor of two every year, whereas the price paid for incorrect assumptions is remaining the same.” (Breiman and Friedman 1988)

MBL training and testing

- “We should never underestimate the power of simply remembering some or all of the examples and comparing test examples with our memory” (Ripley 1996) *Pattern Recognition and Neural Networks* ← a good book!
- Asymptotically, under various reasonable assumptions, risk converges to twice the Bayes risk as sample grows – independent of distance metric (Duda and Hart 1973; Ripley 1996)
  - Bayes risk = optimal error rate, assuming that probability distribution is known

MBL: similarity function

- Need distance function to calculate similarity
- In practice the metric is important
- Usually use Euclidean metric (alternatives Mahalanobis distance, L_1 distance = features different, etc.)
- Appropriate scaling of the variables first is important: normalize values, sometimes log-scale etc. Can put knowledge into metric.

MBL

- For 1NN decision surface is Voronoi diagram.
- Commonly distance-weight the classes of the neighbors in reaching a decision
  - e.g., information gain weighting used in TiMBL (Tilburg Memory-based Learner http://ilk.kub.nl/software.html)
- Asymptotically this doesn't help, but can allow use of larger k for fixed size training set
- $k > 1$ increases resistance to noise (decreases variance)

Feature selection for MBL

- Excluding features which have little or no relevance is very important
- Wrapper methods generate candidate features by testing performance of the classifier with those features
- Filter methods decide features independently of algorithm that will use them
- Can use decision tree techniques to select features (Cardie 1993)

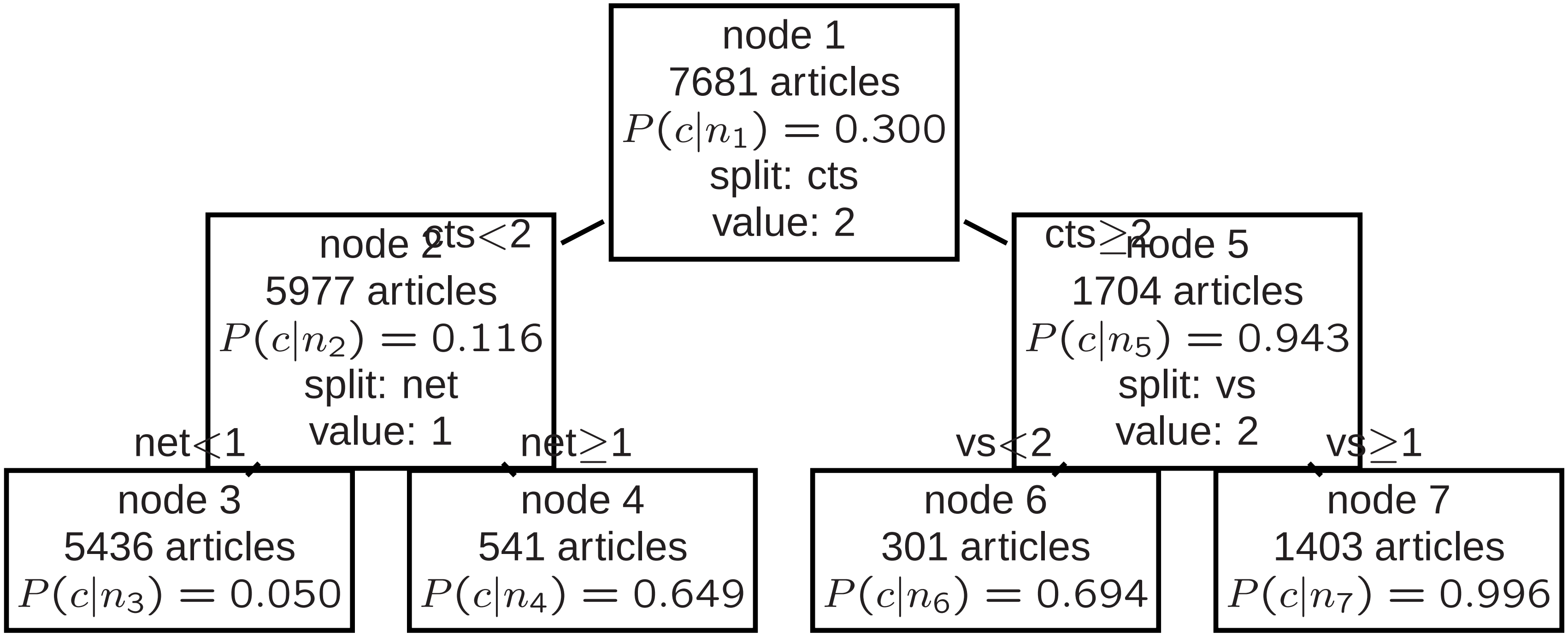
MBL

- Naive storage and search for neighbors can be expensive
- But there are efficient algorithms for indexing and hence finding neighbors (k-d trees)
- The stored data set size can be reduced by data editing so as to only store examples near decision surfaces.

Example: Part-of-Speech Tagging (Daelemans et al. 1996). Accuracy: 96.4%

- Similarity: Hamming distance ($1 -$ %matching) = L_1 norm = overlap
- Feature weighting: Information gain
- Known and unknown word models
- If known case, base features for training are:
  - disambiguated t_{i-2}, disambiguated t_{i-1}
  - ambiguity class for w_i
  - ambiguity class for w_{i+1}

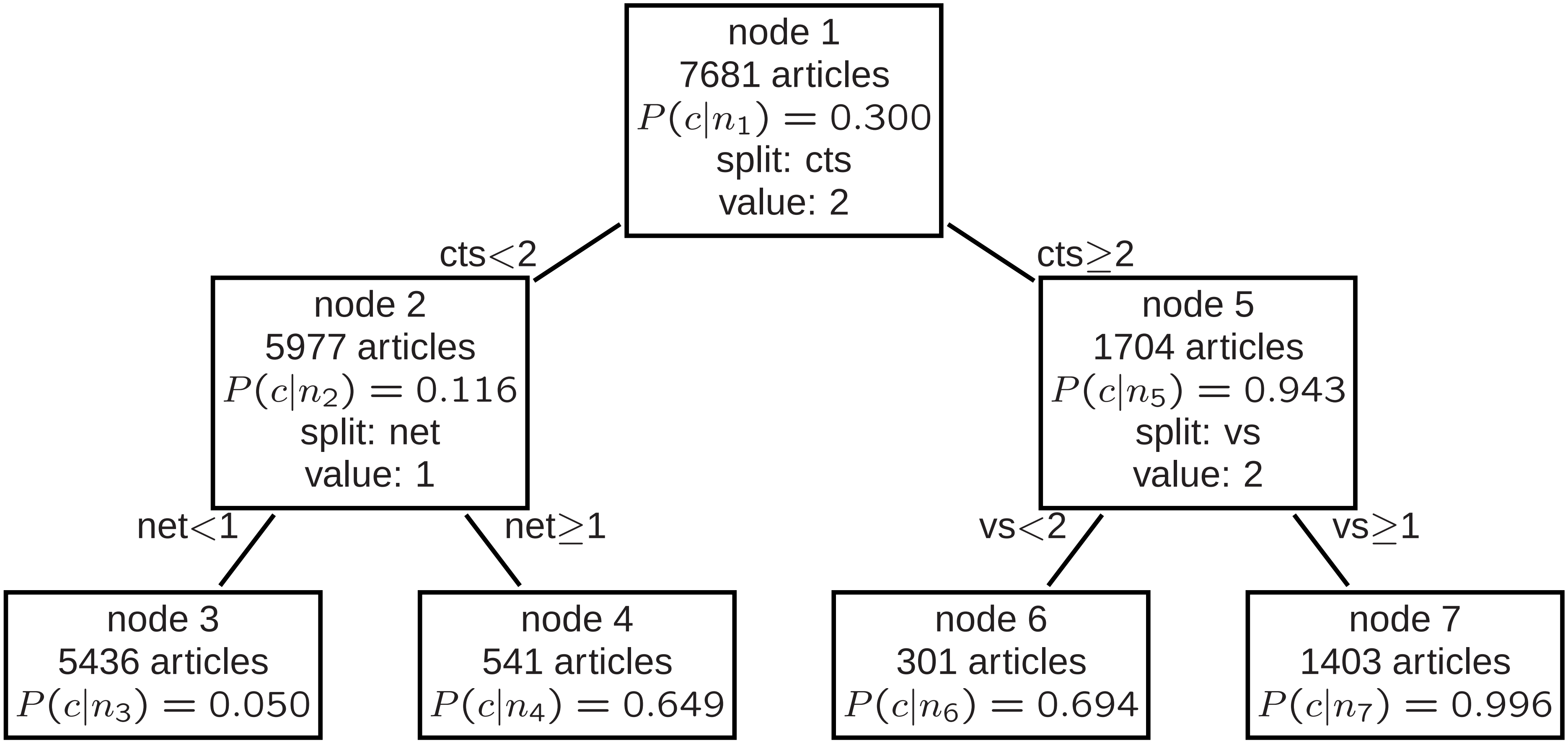
A decision tree for Reuters



(Statistical) Decision Tree Models

- Relatively unused in statistics and pattern recognition but introduced by Breiman et al. (1984)
- Widely used in AI machine learning (Quinlan 1986)
- Mainly because they are extremely easy to comprehend
- Not very efficient in combining independent information sources
- Idea is to successively test an attribute, and then branch based on its value

A text classification decision tree (for Reuters data)



Splitting criteria

- By splitting, one hopes to maximize the ‘purity’ of each daughter node
- Common criteria:
  - Information gain: an informatin theoretic measure:

    $$G(a,y) = H(\mathtt{t}) - H(\mathtt{t}|a) = H(\mathtt{t}) - (\mathtt{p}_L\, H(\mathtt{t}_L) + \mathtt{p}_R\, H(\mathtt{t}_R))$$

  - Gini index $i(p) = \sum_{i \neq j} p_i p_j$
- Either need to only allow binary splits or to penalize n-way splits, or else they are falsely favored

Statistical decision trees

- Conserve data by only asking relevant questions
- However, since each split in the tree lessens the amount of data, one needs a pruning strategy to prevent over training
- A decision tree's questions pick out boxed shaped regions of the feature hyperspace
- Practically decision trees work well, though there aren't a lot of deep theorems to justify their performance

Combining information without interaction

Intuition:

$$\begin{aligned} P(c|x_1,\ldots,x_n) &= \frac{P(x_1,\ldots,x_n|c)P(c)}{P(x_1,\ldots,x_n)} \\ &\approx P(c)\prod_{i=1}^{n}\frac{P(x_i|c)}{P(x_i)} \\ &= P(c)\prod_{i=1}^{n}\frac{P(c|x_i)}{P(c)} \end{aligned}$$

We’re updating the prior, but not necessarily in a good way

Maximum entropy/minimum divergence models

- $P(c|x_i)$ specifies a marginal constraint. We would like it to be true of the resulting model. This ain't what we get with Naive Bayes
- We will require $P(c|x_1, \ldots, x_n)$ to satisfy certain constraints
- Elsewhere $P(c|x_1, \ldots, x_n)$ should reflect our ignorance of what we have not specified

The idea of constraints: as a joint model

We want $P(x_1, x_2, c)$ chosen to be constrained by some marginals we know, $P(x_1, c)$, $P(x_2, c)$.

$$\begin{aligned}\sum_{x_1,x_2,c} P(x_1, x_2, c) f_{1ik}(x_1, x_2, c) &= P(x_i, c_k)\\ \sum_{x_1,x_2,c} P(x_1, x_2, c) f_{2jk}(x_1, x_2, c) &= P(y_j, c_k)\end{aligned}$$

where

$$\begin{aligned}f_{1ik}(x_1, x_2, c) &= 1 \text{ if } x_1 = x_i \text{ and } c = c_k, \quad 0 \text{ otherwise}\\ f_{2jk}(x_1, x_2, c) &= 1 \text{ if } x_2 = x_j \text{ and } c = c_k, \quad 0 \text{ otherwise}\end{aligned}$$

Maximum entropy/minimum divergence (Jaynes)

- Ignorance: if we know nothing but what is specified in the constraints, of all the models that satisfies the constraints, we should chose the one with maximum entropy
- It adds no information not given by the constraints
- But alternatively we can choose P to diverge as little as possible from some other probability distribution Q in terms of KL-divergence:

$$D(P\|Q) = \sum_{\vec{x}\in\mathcal{X}} P(\vec{x}) \log \frac{P(\vec{x})}{Q(\vec{x})}$$

- Maximum entropy results from making Q uniform, which minimizes divergence.

Restated as expectations

- The problem can then be restated as: determine $P(\vec{x})$ so that:
  - For given constraints f_i, it satisfies $E_P(f_i) = \sum_{\vec{x}} P(\vec{x}) f_i(\vec{x}) = d_i$
  - $D(P\|Q)$ is minimal for specified Q (usually uniform)
- We assume the constraints must be consistent.
- This is certainly the case if we estimate the d_i from training data (as ML estimates)
- Ensure probability distribution: $f_0(\vec{x}) = 1 \quad \forall \vec{x}$
- The $f_i(\cdot)$ are often all indicator functions – for computational efficiency, and because they can then all be interpreted as marginals – but they need not be

Math

- We have a constrained optimization problem and find the solution using undetermined Lagrange multipliers. For all $\vec{x}$ we set to 0 the partial derivatives wrt $P(\vec{x})$ of:

  $$D(P\|Q) - \sum_i \lambda_i [\sum_{\vec{x}'} P(\vec{x}')f_i(\vec{x}') - d(i)]$$

  The result is

  $$\log \frac{P(\vec{x})}{Q(\vec{x})} + 1 = \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \lambda_0$$

  So,

  $$P(\vec{x}) = Q(\vec{x}) e^{\lambda_0} e^{\sum_i \lambda_i f_i(x)}$$

  Where the λ_i have to be chosen to satisfy the constraints

Back to the intuitions

Intuition:

$$\begin{aligned}
P(c|x_1,\ldots,x_n) &\approx P(c)\prod_{i=1}^{n}\frac{P(c|x_i)}{P(c)} \\
&= \frac{1}{Z(\vec{x})}\prod_{i \text{ active}} e^{\lambda_i} \\
\log P(c|x_1,\ldots,x_n) &= \sum_{i \text{ active}} \lambda_i - Z(\vec{x})
\end{aligned}$$

Maximum entropy methods

- A unified framework for resolving ambiguities
- Can flexibly integrate heterogeneous sources of information
- Quite high performance tools can be built fairly quickly and automatically
- Basic algorithm is reusable; main need is annotated training data

Advantages

- If features don't overlap, exponential models can imitate other models
- If features *do* overlap, exponential models can perform better
  - They discount weights for overlap
  - But they don't automatically model complex feature interactions

Maximum entropy features

- ‘Features’ include the class assignment
- For classification, a feature f has the form $f_i(\vec{x}, c)$
- $\vec{x}$ is what we observe (e.g., words)
- c is what we are predicting (e.g., tags)
- Each feature is a constraint on the model
- E.g., we might have $f_{\text{HAVE,VBN}} = 1$ iff the word’s current tag is classified as VBN, and the previous verb’s features says it’s a perfective *have* auxiliary

How do we choose the constraints?

- The constraints can be anything, and overlap in any way. This is very good news, because it allows us to put interesting linguistics into the constraints
- However, we do need to choose consistent constraint targets
- This is easy if the targets are the empirical expectations
- But then we can easily have a problem with data sparseness: e.g., if we said unseen bigrams had an expectation of 0.
- Commonly just use constraints on “common” things
- But smoothing is a problem with MaxEnt models, and other approaches, such as using gaussian regularization terms on the lambdas have been explored.

Maximum entropy model

- We compute *expectation* of each feature based on training set (if feature is binary this is just relative frequency)
- Say that this *empirical expectation* must be met in resulting model $\longrightarrow$ the constraints
- We then compute model with *maximum* entropy subject to the constraints
- I.e., we don't go beyond the data by adding information not justified by the constraints.

Loglinear/exponential models

- Maximum entropy models are *always* exponential models. (Theorem. Proof omitted.)
- For discrete distributions – common in NLP! – we can build a contingency table model of the joint distribution of the data.
- Most common choice for modeling this is a loglinear model:

  $$\log P(X_1 = x_1, \ldots, X_p = x_p) = \sum_C \lambda_C(x_C)$$

  where $C \subset \{1, \ldots, p\}$.
- Subsets with more than one member are 'interaction terms'

Example contingency table: predicting POS JJ

		f_1		
		+hyphen	−hyphen	
f_2	+ *-al*	Y: 8 N: 2	Y: 18 N: 27	Y: 26 N: 29
	− *-al*	Y: 10 N: 20	Y: 3 N: 62	Y: 13 N: 82
		Y: 18 N: 22	Y: 21 N: 89	Y: 39 N: 111

$N = 150$

Loglinear/exponential models

- Most extreme restriction is no interaction terms:

$$P(X_1 = x_1, \ldots, X_p = x_p) = e^{\lambda_0} \prod_i e^{\lambda_i(x_i)}$$

- This is what is currently used in maxent models
- See Franz (1997) for loglinear modeling with interaction terms
  - Models with quite limited numbers of features
  - Fit by iterative proportional fitting (related to GIS)
- Really *all* the main generative probability models in StatNLP are loglinear, because they're done as a product of probabilities (HMMs, PCFGs, . . .)

Maxent Loglinear models

- $$\mathsf{p}(\vec{x}, c) = \frac{1}{Z} \prod_{i=1}^{K} \alpha_i^{f_i(\vec{x},c)}$$

  where K is the number of features, α_i is the weight for feature f_i and Z is a normalizing constant, used to ensure that a probability distribution results (partition function)

- If we take logs on both sides, then $\log \mathsf{p}$ is a linear combination of the logs of the weights:

$$\log \mathsf{p}(\vec{x}, c) = -\log Z + \sum_{i=1}^{K} f_i(\vec{x}, c) \log \alpha_i$$

Maximum entropy solution

- Want maximum entropy distribution p^* exponential model that obeys the feature constraints:

$$E_{\mathsf{p}^*} f_i = E_{\tilde{\mathsf{p}}} f_i$$

- In other words, the expected value of f_i for p^* is the same as the expected value for the empirical distribution (the training set).
- System is solved by an iterative algorithm that adjusts the α_i weights

How do we find the λ_i?

- Iteratively from an initial guess
- Hill climbing (conjugate gradient descent)
- Generalized iterative scaling
  - A generalization of the iterative proportional fitting methods used for generalized linear models in statistics

Generalized Iterative Scaling

- Algorithm requires constant sum of features for each $(\vec{x}, c)$:

$$\forall \vec{x}, c \quad \sum_i f_i(\vec{x}, c) = C$$

- Define C as the greatest possible feature sum:

$$C \stackrel{\text{def}}{=} \max_{\vec{x}, c} \sum_{i=1}^{K} f_i(\vec{x}, c)$$

- Add a feature f_{K+1} that is defined as follows:

$$f_{K+1}(\vec{x}, c) = C - \sum_{i=1}^{K} f_i(\vec{x}, c)$$

Generalized Iterative Scaling

- Expected value in model $E_{\mathsf{p}}\, f_i$ is defined as follows:

$$E_{\mathsf{p}}\, f_i = \sum_{\vec{x},c} \mathsf{p}(\vec{x}, c) f_i(\vec{x}, c)$$

  sum is over the event space – all possible vectors $\vec{x}$ and class labels c

- Empirical expectation is easy to compute:

$$E_{\tilde{\mathsf{p}}}\, f_i = \sum_{\vec{x},c} \tilde{\mathsf{p}}(\vec{x}, c) f_i(\vec{x}, c) = \frac{1}{N} \sum_{j=1}^{N} f_i(\vec{x}_j, c)$$

  where N is the number of elements in the training set and probability of unseen event is 0.

Conditional maxent framework

In general, the maximum entropy distribution $E_{\mathsf{p}}\, f_i$ cannot be computed efficiently since it would involve summing over all possible combinations of $\vec{x}$ and c, a potentially infinite set. Instead, we use the following approximation (Lau 1994: 25):

$$E_{\mathsf{p}}\, f_i \approx \sum_{\vec{x},c} \tilde{\mathsf{p}}(\vec{x})\, \mathsf{p}(c|\vec{x}) f_i(\vec{x}, c) = \frac{1}{N} \sum_{j=1}^{N} \sum_{c} \mathsf{p}(c|\vec{x}_j) f_i(\vec{x}_j, c)$$

where c ranges over all possible classes, in our case $c \in \{0, 1\}$. We just sum over seen stuff.

Generalized iterative scaling algorithm

1. Initialize $\{\alpha_i^{(1)}\}$. E.g., $\alpha_i^{(1)} = 1, \forall 1 \leq j \leq K+1$. Compute $E_{\tilde{\mathsf{p}}}\, f_i$ as shown above. Set $n = 1$.
2. Compute $\mathsf{p}^{(n)}(\vec{x}, c)$ for the distribution $\mathsf{p}^{(n)}$ given by the $\{\alpha_i^{(n)}\}$ for each element $(\vec{x}, c)$ in the training set:

$$\mathsf{p}^{(n)}(\vec{x}, c) = \frac{1}{Z} \prod_{i=1}^{K+1} (\alpha_i^{(n)})^{f_i(x,c)}$$

3. Compute $E_{\mathsf{p}^{(n)}} f_i$ for all $1 \leq i \leq K + 1$ according to equation (13).
4. Update the parameters α_i:

$$\alpha_i^{(n+1)} = \alpha_i^{(n)} \left(\frac{E_{\tilde{\mathsf{p}}} f_i}{E_{\mathsf{p}^{(n)}} f_i} \right)^{\frac{1}{C}} \qquad (8)$$

5. If the parameters of the procedure have converged, stop, otherwise increment n and go to 2.

Example system: Ratnaparkhi (1997a)

- Roughly a probabilistic shift-reduce parser
- History-based approach
- Features are parser configurations and classifications are actions
- Observed run-time is linear
- Labeled precision/recall are 87.5%/86.3%
- (Slightly behind the state of the art now, fairly competitive when it was done)

Advantages of maxent

- Diverse forms of evidence
- No data fragmentation – contrast with decision trees
- Features are allowed to overlap
- Feature weights are determined automatically

Disadvantages of maxent

- It’s difficult – but not impossible – to do smoothing and adjustment of models.
- Normally have to stick to empirical distribution, so people generally smooth that to give pseudo-empirical distribution
- Training is slow

Text classification

For text categorization, we define features as follows:

$$f_i(\vec{x}_j, c) = \begin{cases} 1 & \text{if } s_{ij} > 0 \text{ and } c = 1 \\ 0 & \text{otherwise} \end{cases}$$

Maximum Entropy text classification

- Nigam, Lafferty, and McCallum (1999 IJCAI WS)
- Features are scaled word occurrence counts in documents
- Simple model suffered from overfitting; use Gaussian prior, and do MAP estimation
- One can easily add bigrams, phrases (but they didn't)
- For WebKB data, they give an error rate of 8.1%, as against 13.7% for Naive Bayes.
- But NB was able to beat MaxEnt (by 0.7%) on a set of newsgroup data

●

Bibliography

Baker, James K. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf (eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics* 41:164–171.

Bod, Rens. 1998. *Beyond Grammar: An experience-based theory of language*. Stanford, CA: CSLI Publications.

Bod, Rens. 2000. Parsing with the shortest derivation. In *COLING 18*.

Borkar, Vinayak, Kaustubh Deshmukh, and Sunita Sarawagi. 2001. Automatic segmentation of text into structured records. In *SIGMOD 2001*, pp. 175–186.

Brants, Thorsten. 2000. TnT – a statistical part-of-speech tagger. In *ANLP 6*, pp. 224–231.

Breiman, L., and J. H. Friedman. 1988. Discussion on article by Loh and Vanichsetakul: ‘tree-structured classification via generalized discriminant analysis’. *Journal of the American Statistical Association* 83:715–727.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1991a. A statistical approach to sense disambiguation in machine translation. In *Proceedings of the DARPA Workshop on Speech and Natural Language Workshop*, pp. 146–151.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1991b. Word-sense disambiguation using statistical methods. In *ACL 29*, pp. 264–270.

Bruce, Rebecca, and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. In *ACL 32*, pp. 139–145.

Bruce, Rebecca F., and Janyce M. Wiebe. 1999. Decomposable modeling in natural language processing. *Computational Linguistics* 25:195–208.

Cardie, Claire. 1993. Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 25–32.

Charniak, Eugene. 1993. *Statistical Language Learning*. Cambridge, MA: MIT Press.

Charniak, Eugene. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, pp. 1031–1036.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI '97)*, pp. 598–603.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *NAACL 1*, pp. 132–139.

Charniak, Eugene. 2001. Immediate-head parsing for language models. In *ACL 39*, pp. 116–123.

Charniak, Eugene, and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL 43*, pp. 173–180.

Chelba, Ciprian, and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *ACL 36/COLING 17*, pp. 225–231.

Chen, Stanley F., and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University.

Church, Kenneth W., and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language* 5: 19–54.

Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL 34*, pp. 184–191.

Collins, Michael John. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 35/EACL 8*, pp. 16–23.

Collins, Michael John, and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *WVLC 3*, pp. 27–38.

Cover, Thomas M., and Joy A. Thomas. 1991. *Elements of Information Theory*. New York: John Wiley & Sons.

Daelemans, Walter, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. MBT: A memory-based part of speech tagger generator. In *WVLC 4*, pp. 14–27.

Dagan, Ido, and Alon Itai. 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics* 20: 563–596.

Dagan, Ido, Lillian Lee, and Fernando Pereira. 1999. Similarity-based models of word co-occurrence probabilities. *Machine Learning* 34:43–69.

Dempster, A.P., N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B* 39:1–38.

Domingos, Pedro, and Michael Pazzani. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29: 103–130.

Duda, Richard O., and Peter E. Hart. 1973. *Pattern classification and scene analysis*. New York: Wiley.

Ford, Marilyn, Joan Bresnan, and Ronald M. Kaplan. 1982. A competence-based theory of syntactic closure. In Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pp. 727–796. Cambridge, MA: MIT Press.

Franz, Alexander. 1997. Independence assumptions considered harmful. In *ACL 35/EACL 8*, pp. 182–189.

Freitag, Dayne, and Andrew McCallum. 1999. Information extraction with HMMs and shrinkage. In *AAAI '99 Workshop on Machine Learning for Information Extraction*, pp. 31–36.

Freitag, Dayne, and Andrew McCallum. 2000. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of AAAI*, pp. 584–589.

Friedman, Jerome H. 1997. On bias, variance, 0/1–loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1:55–77.

Friedman, Nir, and M. Goldszmidt. 1996. Building classifiers using Bayesian networks. In *AAAI '96*, pp. 1277–1284.

Gale, William A., Kenneth W. Church, and David Yarowsky. 1992. A method for disambiguating word senses in a large corpus. *Computers and the Humanities* 26:415–439.

Gale, William A., and Geoffrey Sampson. 1995. Good-Turing frequency estimation without tears. *Journal of Quantitative Linguistics* 2: 217–237.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Goodman, Joshua. 2001. A bit of progress in language modeling, extended version. Technical Report MSR-TR-2001-72, Microsoft Research.

Hindle, Donald, and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics* 19:103–120.

Horning, James Jay. 1969. *A study of grammatical inference*. PhD thesis, Stanford.

Johnson, Mark. 1998a. The effect of alternative tree representations on tree bank grammars. In *Proceedings of Joint Conference on New Methods in Language Processing and Computational Natural Language Learning (NeMLaP3/CoNLL98)*, pp. 39–48, Macquarie University.

Johnson, Mark. 1998b. PCFG models of linguistic tree representations. *Computational Linguistics* 24:613–632.

Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-35: 400–401.

Kilgarriff, Adam. 1997. "I don't believe in word senses". *Computers and the Humanities* 31:91–113.

Klein, Dan, and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 41*, pp. 423–430.

Kneser, Reinhard, and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume 1, pp. 181–184.

Koller, Daphne, and Mehran Sahami. 1996. Toward optimal feature selection. In *ICML '96*, pp. 284–292.

Kupiec, Julian. 1992. Robust part-of-speech tagging using a Hidden Markov Model. *Computer Speech and Language* 6:225–242.

Laplace, Pierre Simon marquis de. 1814. *Essai philosophique sur les probabilites*. Paris: Mme. Ve. Courcier.

Lari, K., and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.

Lau, Raymond. 1994. Adaptive statistical language modelling. Master's thesis, Massachusetts Institute of Technology.

Lesk, Michael. 1986. Automatic sense disambiguation: How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986*

SIGDOC Conference, pp. 24–26, New York. Association for Computing Machinery.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *ACL 33*, pp. 276–283.

Manning, Christopher D., and Bob Carpenter. 1997. Probabilistic parsing using left corner language models. In *Proceedings of the Fifth International Workshop on Parsing Technologies (IWPT-97)*, pp. 147–158, MIT.

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Boston, MA: MIT Press.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19:313–330.

McCallum, Andrew, and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *Working Notes of the 1998 AAAI/ICML Workshop on Learning for Text Categorization*, pp. 41–48.

McLachlan, Geoffrey J., and Thiriyambakam Krishnan. 1996. *The EM Algorithm and Extensions*. John Wiley & Sons.

Merlo, Paola, Matthew Crocker, and Cathy Berthouzoz. 1997. Attaching multiple prepositional phrases: Generalized backed-off estimation. In *EMNLP 2*, pp. 149–155.

Mitchell, Tom M. (ed.). 1997. *Machine Learning*. New York: McGraw-Hill.

Ney, Hermann, Sven Martin, and Frank Wessel. 1997. Statistical language modeling using leaving-one-out. In Steve Young and Gerrit Bloothooft (eds.), *Corpus-Based Methods in Language and Speech Processing*, pp. 174–207. Dordrecht: Kluwer Academic.

Pasca, Marius, and Sanda Harabagiu. 2001. High performance question/answering. In *SIGIR '01*, pp. 366–374.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1: 81–106. Reprinted in (Shavlik and Dietterich 1990).

Ratnaparkhi, Adwait. 1997a. A linear observed time statistical parser based on maximum entropy models. In *EMNLP 2*, pp. 1–10.

Ratnaparkhi, Adwait. 1997b. A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS Report 97–08, Institute for Research in Cognitive Science, Philadelphia, PA.

Ratnaparkhi, Adwait. 1998. Unsupervised statistical models for prepositional phrase attachment. In *ACL 36/COLING 17*, pp. 1079–1085.

Ratnaparkhi, Adwait. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning* 34:151–175.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Rosenfeld, Ronald. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language* 10: 187–228.

Rosenfeld, Ronald. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE* 88.

Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics* 24:97–124.

Shavlik, Jude W., and Thomas G. Dietterich (eds.). 1990. *Readings in Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL 3*, pp. 252–259.

van Rijsbergen, C. J. 1979. *Information Retrieval*. London: Butterworths. Second Edition.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13:260–269.

Yarowsky, David. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *COLING 14*, pp. 454–460.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL 33*, pp. 189–196.