

Introduction to NLP

What is Natural Language Processing?



Question Answering: IBM's Watson

- Won Jeopardy on February 16, 2011!

WILLIAM WILKINSON'S
“AN ACCOUNT OF THE PRINCIPALITIES OF
WALLACHIA AND MOLDOVIA”
INSPIRED THIS AUTHOR’S
MOST FAMOUS NOVEL



Bram Stoker

Dan Jurafsky



Information Extraction

Subject: **curriculum meeting**

Date: January 15, 2012

To: Dan Jura

Event: Curriculum mtg
Date: Jan-16-2012
Start: 10:00am
End: 11:30am
Where: Gates 159

Hi Dan, we've now scheduled the curriculum meeting.

It will be in Gates 159 tomorrow from 10:00-11:30.

-Chris

Create new Calendar entry



Information Extraction & Sentiment Analysis



Size and weight

- ✓ • nice and compact to carry!
- ✓ • since the camera is small and light, I around those heavy, bulky professional
- ✗ • the camera feels flimsy, is plastic and very light in weight you have to be very delicate in the handling of this camera

Attributes:
zoom
affordability
size and weight
flash
ease of use

A screenshot of a web page displaying five yellow-highlighted snippets of text from reviews. Each snippet includes a short summary and a link to the full review. The snippets are:

- Color: 100% accurate. Not a single pixel was off. It's a compact, well-made camera with a great lens.
- Zoom: Great zoom.
- Size: A really good quality plastic camera which is easy to use and looks nice. It has a decent lens, and a reasonable zoom.
- Flash: A really nice camera. It does not have a flash, but it has a good lens and a nice flash.
- Ease of use: Overall, it's a great camera. It's easy to use and has a good lens.



Machine Translation

- Fully automatic

Enter Source Text:

这 不 过 是 一 个 时 间 的 问 题 .

Translation from Stanford's *Phrasal*:

This is only a matter of time.

- Helping human translators

Enter Source Text:

تعرض الرئيس اللبناني إميل لحود لـ "#حملة عنيفة في مجلس التواب الذي انعقد أمس في جلسة تشريعية علية تحولت إلى "محاكمة" لـ "#رئيس الجمهورية على موقفه من المحكمة الدولية و "الملحاظات" التي ادلّي بها #+ها حول هذا الموضوع.

Translate Clear

Enter Translation:

lebanese |

president
suffered
exposed
president emile
before
presented
offer

Done!



Language Technology

making good progress

mostly solved

Spam detection

Let's go to Agra!



Buy V1AGRA ...



Part-of-speech (POS) tagging

ADJ ADJ NOUN VERB ADV

Colorless green ideas sleep furiously.

Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

Sentiment analysis

Best roast chicken in San Francisco!



The waiter ignored us for 20 minutes.



Coreference resolution

Carter told Mubarak he shouldn't run again.

Word sense disambiguation (WSD)

I need new batteries for my **mouse**.



Parsing



I can see Alcatraz from the window!

Machine translation (MT)

第13届上海国际电影节开幕...



The 13th Shanghai International Film Festival...

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30



Party
May 27
add

still really hard

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

Summarization

The Dow Jones is up

The S&P500 jumped



Economy is good

Dialog

Where is Citizen Kane playing in SF?



Castro Theatre at 7:30. Do you want a ticket?





Ambiguity makes NLP hard: “Crash blossoms”



Violinist Linked to JAL Crash Blossoms

Teacher Strikes Idle Kids

Red Tape Holds Up New Bridges

Hospitals Are Sued by 7 Foot Doctors

Juvenile Court to Try Shooting Defendant

Local High School Dropouts Cut in Half

Dan Jurafsky



Ambiguity is pervasive

New York Times headline (17 May 2000)

Fed raises interest rates

Fed raises interest rates

Fed raises interest rates 0.5%



In-video quizzes!

- Most lectures will include a little quiz
 - Just to check basic understanding
 - Simple, multiple-choice.
 - You can retake them if you get them wrong



Why else is natural language understanding difficult?

non-standard English

Great job @justinbieber! Were SOO PROUD of what youve accomplished! U taught us 2 #neversaynever & you yourself should never give up either♥

segmentation issues

the New York-New Haven Railroad
the New York-New Haven Railroad

idioms

dark horse
get cold feet
lose face
throw in the towel

neologisms

unfriend
Retweet
bromance

world knowledge

Mary and Sue are sisters.
Mary and Sue are mothers.

tricky entity names

Where is *A Bug's Life* playing ...
Let It Be was recorded ...
... a mutation on the *for* gene ...

But that's what makes it fun!



Making progress on this problem...

- The task is difficult! What tools do we need?
 - Knowledge about language
 - Knowledge about the world
 - A way to combine knowledge sources
- How we generally do this:
 - probabilistic models built from language data
 - $P(\text{"maison"} \rightarrow \text{"house"})$ **high**
 - $P(\text{"L'avocat général"} \rightarrow \text{"the general avocado"})$ **low**
 - Luckily, rough text features can often do half the job.



This class

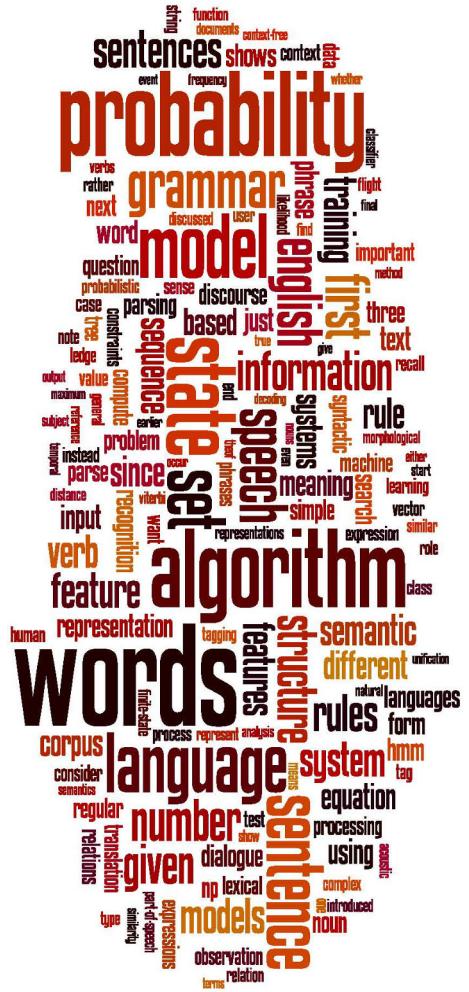
- Teaches key theory and methods for statistical NLP:
 - Viterbi
 - Naïve Bayes, Maxent classifiers
 - N-gram language modeling
 - Statistical Parsing
 - Inverted index, tf-idf, vector models of meaning
- For practical, robust real-world applications
 - Information extraction
 - Spelling correction
 - Information retrieval
 - Sentiment analysis

Dan Jurafsky



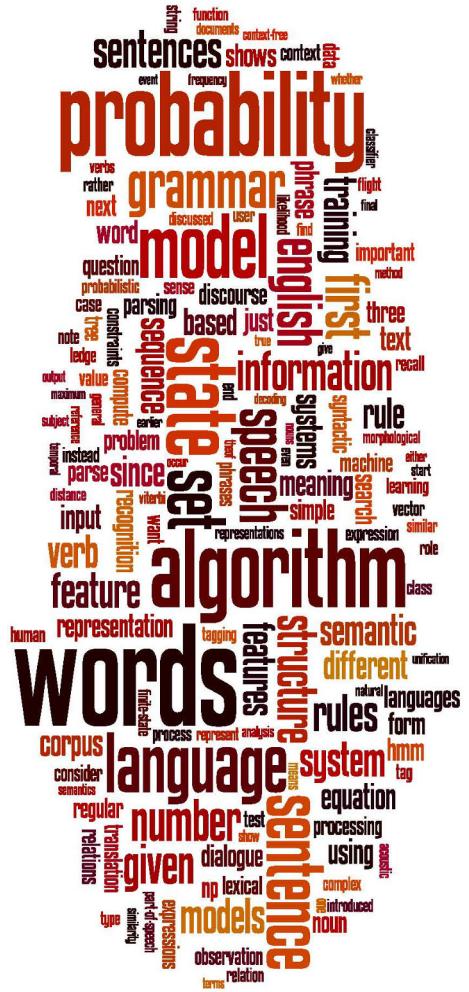
Skills you'll need

- Simple linear algebra (vectors, matrices)
- Basic probability theory
- Java or Python programming
 - Weekly programming assignments



Introduction to NLP

What is Natural Language Processing?



Basic Text Processing

Regular Expressions



Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
 - woodchuck
 - woodchucks
 - Woodchuck
 - Woodchucks





Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW] oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole



Regular Expressions: Negation in Disjunction

- Negations [^Ss]
 - Carat means negation only when first in []

Pattern	Matches	
[^A-Z]	Not an upper case letter	O <u>y</u> fn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look <u>h</u> ere
a^b	The pattern a carat b	Look up <u>a^b</u> now



Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog woodchuck	
yours mine	yours mine
a b c	= [abc]
[gG]roundhog [Ww]oodchuck	



Photo D. Fletcher

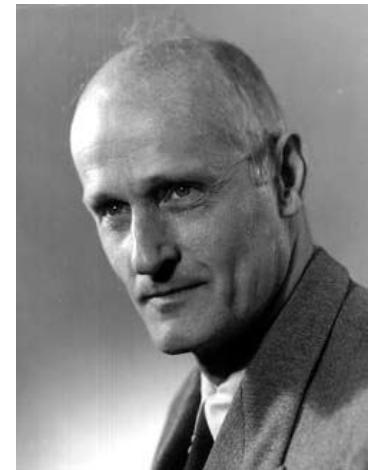


Regular Expressions: ? * + .

Pattern	Matches
colou?r	Optional previous char
oo*h!	0 or more of previous char
o+h!	1 or more of previous char
baa+	
beg.n	

The table shows regular expression patterns and their matches. The matches are underlined in blue.

- colou?r: color colour
- oo*h!: oh! ooh! oooh! ooooh!
- o+h!: oh! ooh! oooh! ooooh!
- baa+: baa baaa aaaa aaaaaa
- beg.n: begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +



Regular Expressions: Anchors \wedge $\$$

Pattern	Matches
$\wedge [A-Z]$	<u>P</u> alo Alto
$\wedge [^A-Za-z]$	<u>1</u> " <u>Hello</u> "
$\backslash . \$$	The end <u>.</u>
$. \$$	The end <u>?</u> The end <u>!</u>



Example

- Find me all instances of the word “the” in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns other or theology

[^a-zA-Z][tT]he[^a-zA-Z]



Errors

- The process we just went through was based on **fixing two kinds of errors**
 - Matching strings that we should not have matched (**there, then, other**)
 - **False positives (Type I)**
 - Not matching things that we should have matched (**The**)
 - **False negatives (Type II)**



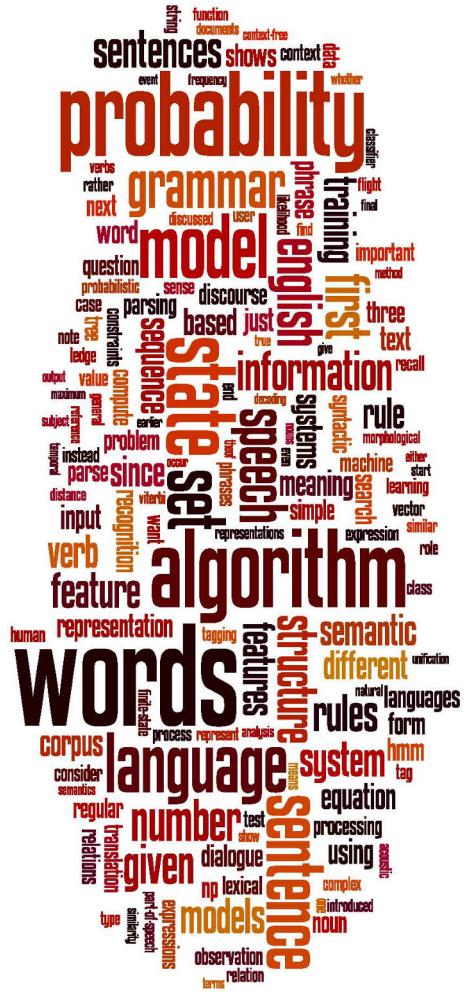
Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
 - Increasing accuracy or precision (minimizing false positives)
 - Increasing coverage or recall (minimizing false negatives).



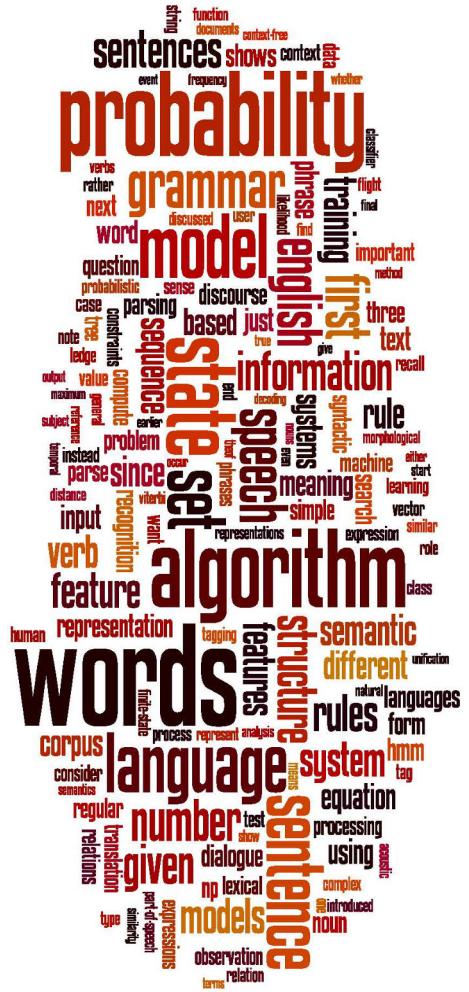
Summary

- Regular expressions play a surprisingly large role
 - Sophisticated sequences of regular expressions are often the first model for any text processing task
- For many hard tasks, we use machine learning classifiers
 - But regular expressions are used as features in the classifiers
 - Can be very useful in capturing generalizations



Basic Text Processing

Regular Expressions



Basic Text Processing

Word tokenization



Text Normalization

- Every NLP task needs to do text normalization:
 1. Segmenting/tokenizing words in running text
 2. Normalizing word formats
 3. Segmenting sentences in running text



How many words?

- I do uh main- mainly business data processing
 - Fragments, filled pauses
- Seuss's **cat** in the hat is different from other **cats!**
 - **Lemma:** same stem, part of speech, rough word sense
 - **cat** and **cats** = same lemma
 - **Wordform:** the full inflected surface form
 - **cat** and **cats** = different wordforms



How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type:** an element of the vocabulary.
- **Token:** an instance of that type in running text.
- How many?
 - 15 tokens (or 14)
 - 13 types (or 12) (or 11?)



How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{1/2})$

	Tokens = N	Types = $ V $
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million



Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt      Change all non-alpha to newlines  
| sort          Sort in alphabetical order  
| uniq -c       Merge and count each type
```

1945 A	25 Aaron
72 AARON	6 Abate
19 ABBESS	1 Abates
5 ABBOT	5 Abbess
... ...	6 Abbey
	3 Abbot

Dan Jurafsky



The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...

Dan Jurafsky



The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head
```

A

A

A

A

A

A

A

A

A

...



More counting

- Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c
```

- Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c | sort -n -r
```

23243	the
22225	i
18618	and
16339	to
15687	of
12780	a
12163	you
10839	my
10005	in
8954	d

What happened here?



Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??



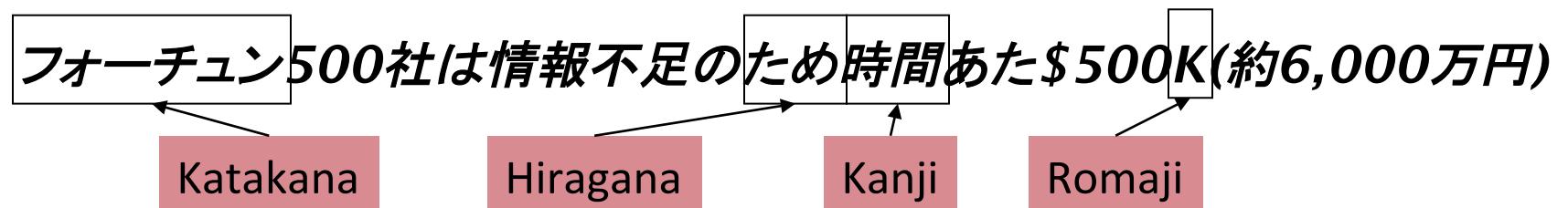
Tokenization: language issues

- French
 - *L'ensemble* → one token or two?
 - *L* ? *L'* ? *Le* ?
 - Want *l'ensemble* to match with *un ensemble*
- German noun compounds are not segmented
 - *Lebensversicherungsgesellschaftsangestellter*
 - ‘life insurance company employee’
 - German information retrieval needs **compound splitter**



Tokenization: language issues

- Chinese and Japanese no spaces between words:
 - 莎拉波娃现在居住在美国东南部的佛罗里达。
 - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
 - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
 - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!



Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
 - Characters are generally 1 syllable and 1 morpheme.
 - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
 - Maximum Matching (also called Greedy)

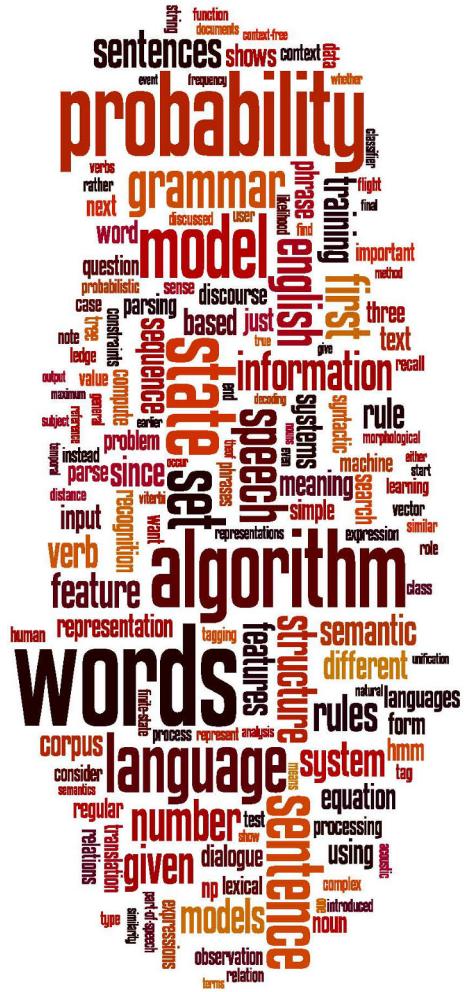


Maximum Matching Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.
 - 1) Start a pointer at the beginning of the string
 - 2) Find the longest word in dictionary that matches the string starting at pointer
 - 3) Move the pointer over the word in string
 - 4) Go to 2

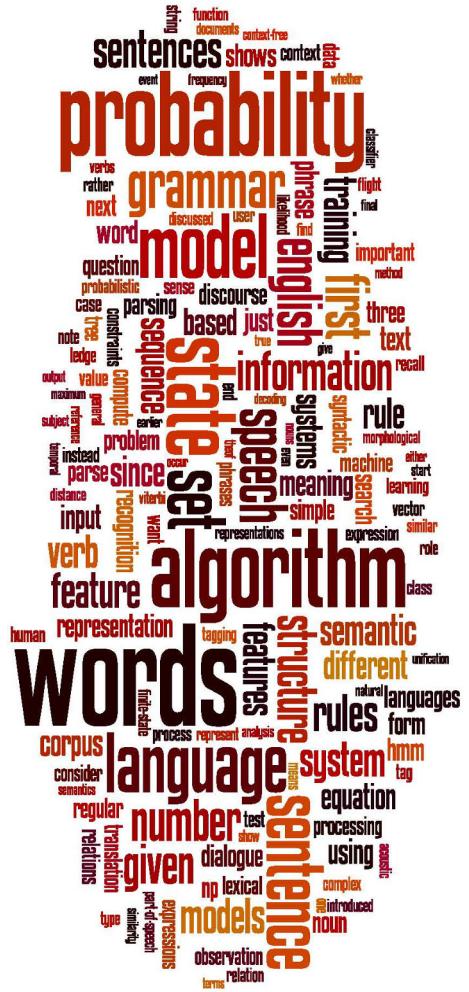


Max-match segmentation illustration



Basic Text Processing

Word tokenization



Basic Text Processing

Word Normalization and Stemming



Normalization

- Need to “normalize” terms
 - Information Retrieval: indexed text & query terms must have same form.
 - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
 - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
 - Enter: ***window*** Search: ***window, windows***
 - Enter: ***windows*** Search: ***Windows, windows, window***
 - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient



Case folding

- Applications like IR: reduce all letters to lower case
 - Since users tend to use lower case
 - Possible exception: upper case in mid-sentence?
 - e.g., *General Motors*
 - *Fed* vs. *fed*
 - *SAIL* vs. *sail*
- For sentiment analysis, MT, Information extraction
 - Case is helpful (*US* versus *us* is important)



Lemmatization

- Reduce inflections or variant forms to base form
 - *am, are, is* → *be*
 - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
 - Spanish **quiero** ('I want'), **quieres** ('you want') same lemma as **querer** 'want'



Morphology

- **Morphemes:**
 - The small meaningful units that make up words
 - **Stems:** The core meaning-bearing units
 - **Affixes:** Bits and pieces that adhere to stems
 - Often with grammatical functions



Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
 - language dependent
 - e.g., *automate(s)*, *automatic*, *automation* all reduced to *automat*.

for example compressed
and compression are both
accepted as equivalent to
compress.



for exampl compress and
compress ar both accept
as equival to compress



Porter's algorithm

The most common English stemmer

Step 1a

sses	\rightarrow	ss	caresses	\rightarrow	caress
ies	\rightarrow	i	ponies	\rightarrow	poni
ss	\rightarrow	ss	caress	\rightarrow	caress
s	\rightarrow	\emptyset	cats	\rightarrow	cat

Step 1b

(*v*)ing	\rightarrow	\emptyset	walking	\rightarrow	walk
			sing	\rightarrow	sing
(*v*)ed	\rightarrow	\emptyset	plastered	\rightarrow	plaster
...					

Step 2 (for long stems)

ational	\rightarrow	ate	relational	\rightarrow	relate
izer	\rightarrow	ize	digitizer	\rightarrow	digitize
ator	\rightarrow	ate	operator	\rightarrow	operate
...					

Step 3 (for longer stems)

al	\rightarrow	\emptyset	revival	\rightarrow	reviv
able	\rightarrow	\emptyset	adjustable	\rightarrow	adjust
ate	\rightarrow	\emptyset	activate	\rightarrow	activ
...					



Viewing morphology in a corpus

Why only strip –ing if there is a vowel?

(*v*)ing → Ø walking → walk
sing → sing



Viewing morphology in a corpus

Why only strip –ing if there is a vowel?

(**v**)ing → Ø walking → walk
sing → sing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort -nr
```

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	102 going

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort -nr
```



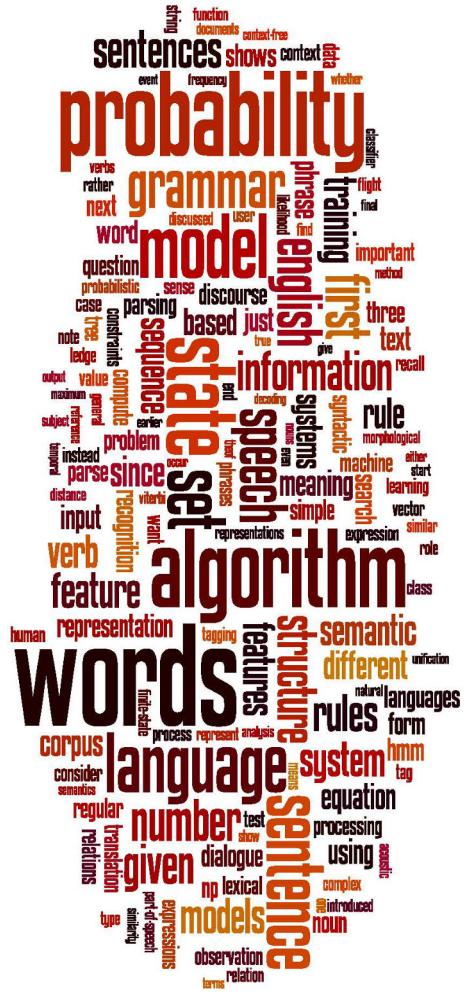
Dealing with complex morphology is sometimes necessary

- Some languages require complex morpheme segmentation
 - Turkish
 - **Uygarlastiramadiklarimizdanmissinizcasina**
 - '(behaving) as if you are among those whom we could not civilize'
 - **Uygar** 'civilized' + **las** 'become'
 - + **tır** 'cause' + **ama** 'not able'
 - + **dik** 'past' + **lar** 'plural'
 - + **imiz** 'p1pl' + **dan** 'abl'
 - + **mis** 'past' + **siniz** '2pl' + **casina** 'as if'



Basic Text Processing

Word Normalization and Stemming



Basic Text Processing

Sentence Segmentation and Decision Trees

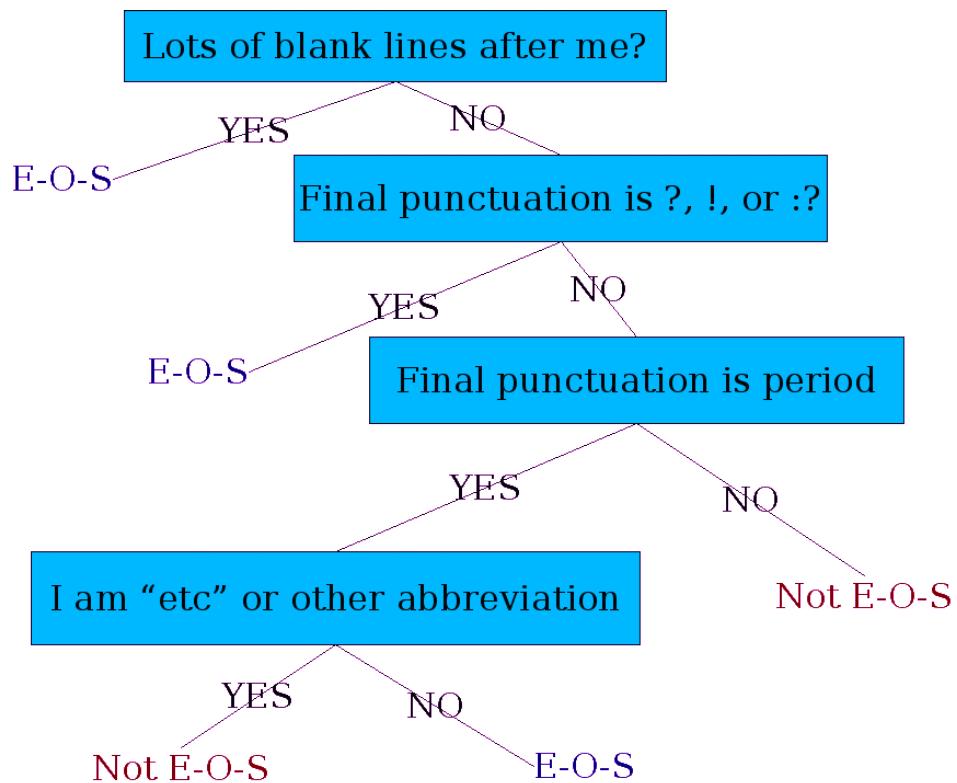


Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
 - Sentence boundary
 - Abbreviations like Inc. or Dr.
 - Numbers like .02% or 4.3
- Build a binary classifier
 - Looks at a “.”
 - Decides EndOfSentence/NotEndOfSentence
 - Classifiers: hand-written rules, regular expressions, or machine-learning



Determining if a word is end-of-sentence: a Decision Tree





More sophisticated decision tree features

- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number
- Numeric features
 - Length of word with “.”
 - Probability(word with “.” occurs at end-of-s)
 - Probability(word after “.” occurs at beginning-of-s)



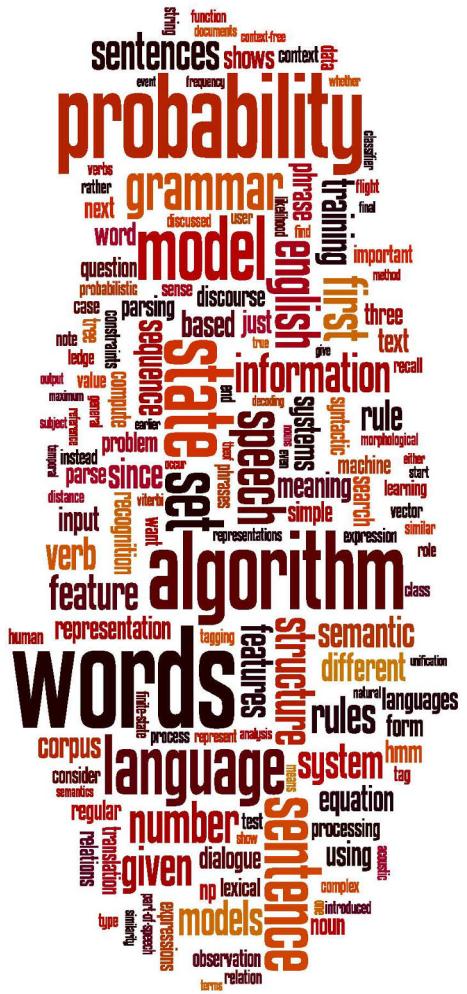
Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
 - Hand-building only possible for very simple features, domains
 - For numeric features, it's too hard to pick each threshold
 - Instead, structure usually learned by machine learning from a training corpus



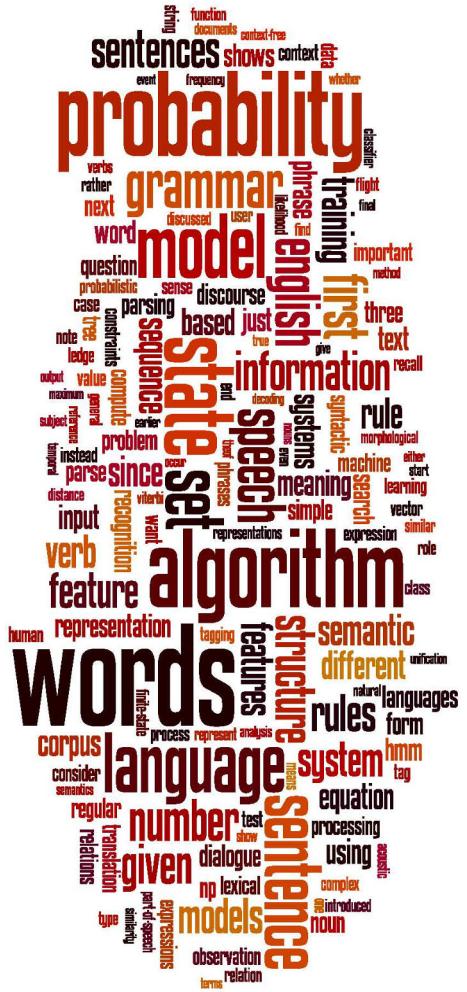
Decision Trees and other classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
 - Logistic regression
 - SVM
 - Neural Nets
 - etc.



Basic Text Processing

Sentence Segmentation
and Decision Trees



Minimum Edit Distance

Definition of Minimum Edit Distance



How similar are two strings?

- Spell correction
 - The user typed “graffe”
Which is closest?
 - graf
 - graft
 - grail
 - giraffe
- Computational Biology
 - Align two sequences of nucleotides

AGGCTATCACCTGACCTCCAGGCCGATGCC
TAGCTATCACGACC CGCGT CGATTGCCCGAC

 - Resulting alignment:

—AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACC GC--GGTCGATTGCCCGAC
- Also for Machine Translation, Information Extraction, Speech Recognition



Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
 - Insertion
 - Deletion
 - Substitution
- Needed to transform one into the other



Minimum Edit Distance

- Two strings and their **alignment**:

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N



Minimum Edit Distance

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has cost of 1
 - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
 - Distance between them is 8



Alignment in Computational Biology

- Given a sequence of bases

AGGCTATCACCTGACCTCCAGGCCGATGCC
TAGCTATCACGACCGCGGTGATTGCCCGAC

- An alignment:

-**A**G**G**CTATCAC**C**T**G**ACC**T**CC**A**GG**C**CGA--TG**C**CC---
T**A**G-CTATCAC--**G**ACC**G**C--GG**T**CGA**T**T**G**CCC**G**AC

- Given two sequences, align each letter to a letter or gap



Other uses of Edit Distance in NLP

- Evaluating Machine Translation and speech recognition

R Spokesman confirms senior government adviser was shot

H Spokesman said the senior adviser was shot dead

S I D I

- Named Entity Extraction and Entity Coreference

- IBM Inc. announced today

- IBM profits

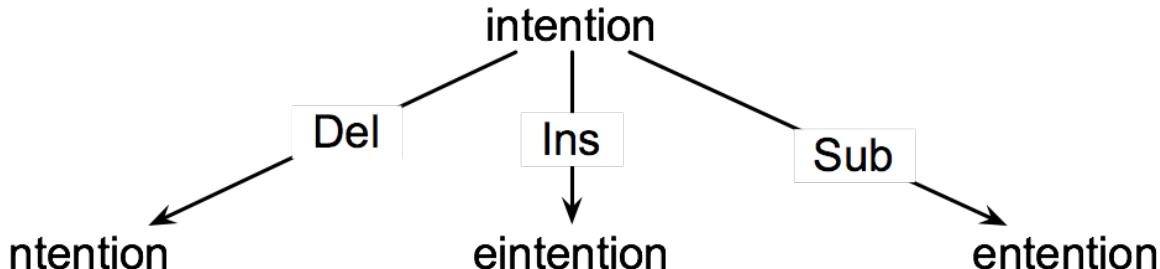
- Stanford President John Hennessy announced yesterday

- for Stanford University President John Hennessy



How to find the Min Edit Distance?

- Searching for a path (sequence of edits) from the start string to the final string:
 - **Initial state:** the word we're transforming
 - **Operators:** insert, delete, substitute
 - **Goal state:** the word we're trying to get to
 - **Path cost:** what we want to minimize: the number of edits





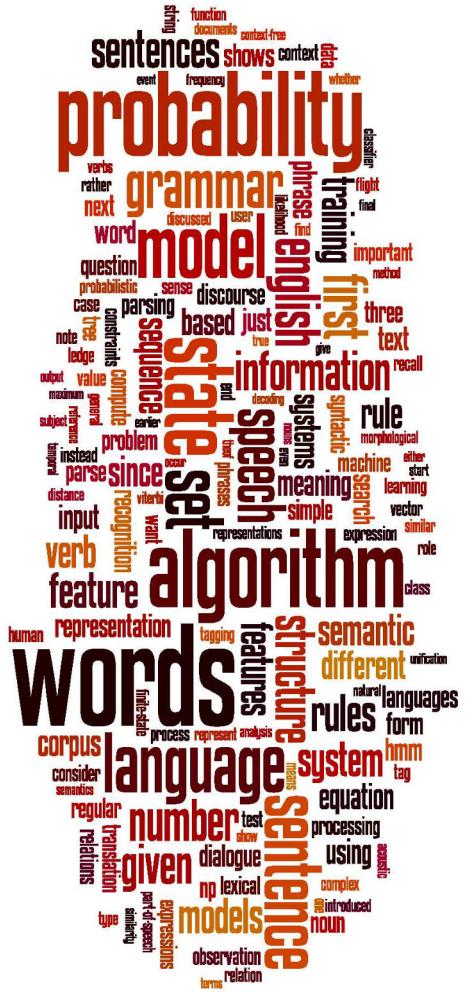
Minimum Edit as Search

- But the space of all edit sequences is huge!
 - We can't afford to navigate naïvely
 - Lots of distinct paths wind up at the same state.
 - We don't have to keep track of all of them
 - Just the shortest path to each of those revisited states.



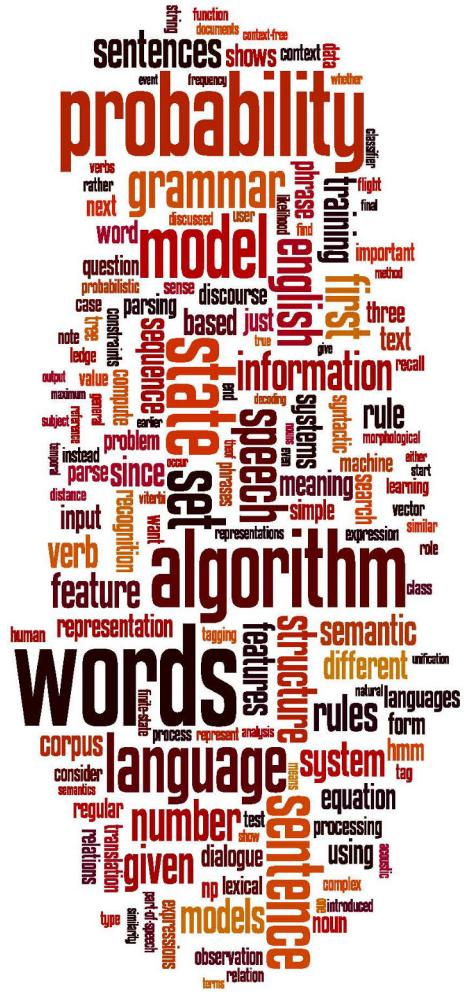
Defining Min Edit Distance

- For two strings
 - X of length n
 - Y of length m
- We define $D(i,j)$
 - the edit distance between $X[1..i]$ and $Y[1..j]$
 - i.e., the first i characters of X and the first j characters of Y
 - The edit distance between X and Y is thus $D(n,m)$



Minimum Edit Distance

Definition of Minimum Edit Distance



Minimum Edit Distance

Computing Minimum Edit Distance



Dynamic Programming for Minimum Edit Distance

- **Dynamic programming:** A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems.
- Bottom-up
 - We compute $D(i,j)$ for small i,j
 - And compute larger $D(i,j)$ based on previously computed smaller values
 - i.e., compute $D(i,j)$ for all i ($0 < i < n$) and j ($0 < j < m$)



Defining Min Edit Distance (Levenshtein)

- Initialization

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) \\ 0; & \begin{cases} \text{if } X(i) = Y(j) \end{cases} \end{cases} \end{cases}$$

- Termination:

$D(N, M)$ is distance



The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



The Edit Distance Table

N	9										
O	8										
I	7										
T	6										
N	5										
E	4										
T	3										
N	2										
I	1										
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	

$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$

A red arrow points from the value "1" in the second row of the table to the formula, indicating it is the result of the third case in the recurrence relation.



Edit Distance

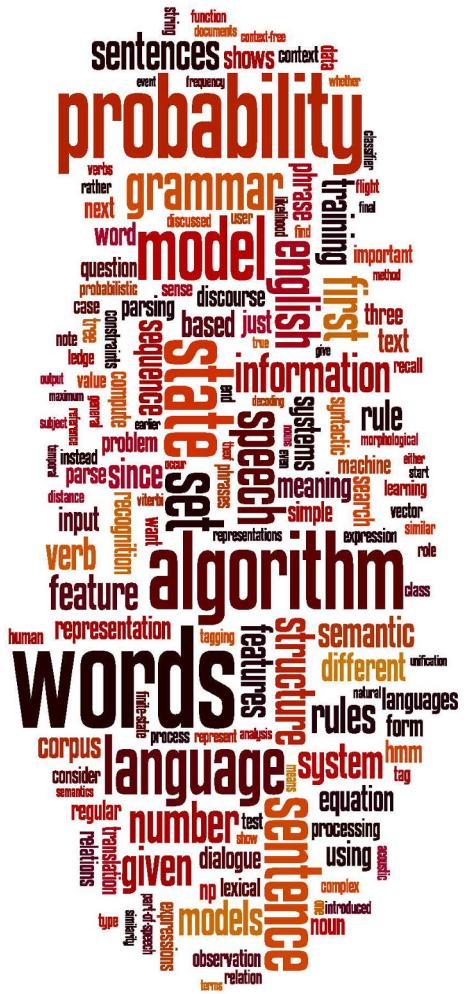
$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9										
O	8										
I	7										
T	6										
N	5										
E	4										
T	3										
N	2										
I	1										
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	



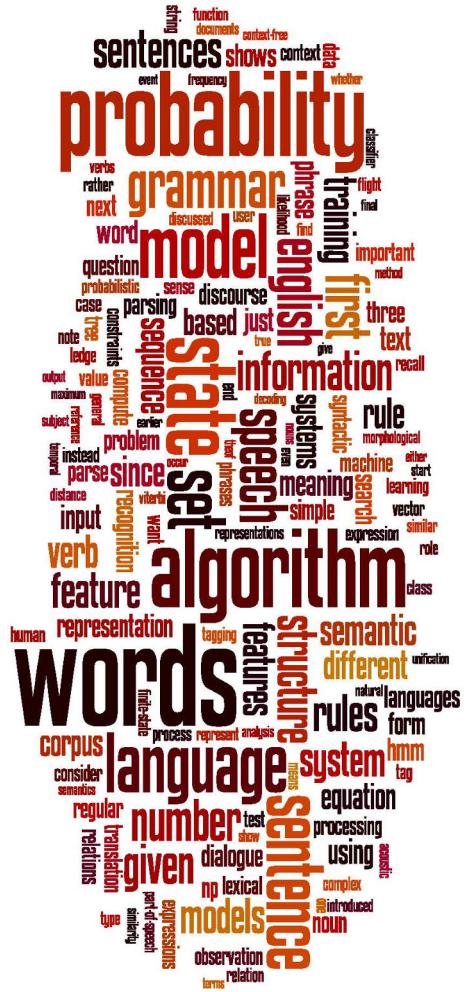
The Edit Distance Table

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



Minimum Edit Distance

Computing Minimum
Edit Distance



Minimum Edit Distance

Backtrace for Computing Alignments



Computing alignments

- Edit distance isn't sufficient
 - We often need to **align** each character of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from
- When we reach the end,
 - Trace back the path from the upper right corner to read off the alignment



Edit Distance

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9										
O	8										
I	7										
T	6										
N	5										
E	4										
T	3										
N	2										
I	1										
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	



MinEdit with Backtrace

n	9	$\downarrow 8$	$\swarrow \leftarrow \downarrow 9$	$\swarrow \leftarrow \downarrow 10$	$\swarrow \leftarrow \downarrow 11$	$\swarrow \leftarrow \downarrow 12$	$\downarrow 11$	$\downarrow 10$	$\downarrow 9$	$\swarrow 8$	
o	8	$\downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow \leftarrow \downarrow 9$	$\swarrow \leftarrow \downarrow 10$	$\swarrow \leftarrow \downarrow 11$	$\downarrow 10$	$\downarrow 9$	$\swarrow 8$	$\leftarrow 9$	
i	7	$\downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow \leftarrow \downarrow 9$	$\swarrow \leftarrow \downarrow 10$	$\downarrow 9$	$\swarrow 8$	$\leftarrow 9$	$\leftarrow 10$	
t	6	$\downarrow 5$	$\swarrow \leftarrow \downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow \leftarrow \downarrow 9$	$\swarrow 8$	$\leftarrow 9$	$\leftarrow 10$	$\leftarrow 11$	
n	5	$\downarrow 4$	$\swarrow \leftarrow \downarrow 5$	$\swarrow \leftarrow \downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow \leftarrow \downarrow 9$	$\swarrow \leftarrow \downarrow 10$	$\swarrow \leftarrow \downarrow 11$	$\swarrow \downarrow 10$	
e	4	$\swarrow 3$	$\leftarrow 4$	$\swarrow \leftarrow 5$	$\leftarrow 6$	$\leftarrow 7$	$\downarrow 8$	$\swarrow \leftarrow 9$	$\swarrow \leftarrow \downarrow 10$	$\downarrow 9$	
t	3	$\swarrow \leftarrow \downarrow 4$	$\swarrow \leftarrow \downarrow 5$	$\swarrow \leftarrow \downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow 7$	$\leftarrow 8$	$\swarrow \leftarrow 9$	$\downarrow 8$	
n	2	$\swarrow \leftarrow \downarrow 3$	$\swarrow \leftarrow \downarrow 4$	$\swarrow \leftarrow \downarrow 5$	$\swarrow \leftarrow \downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\downarrow 7$	$\swarrow \leftarrow \downarrow 8$	$\swarrow 7$	
i	1	$\swarrow \leftarrow \downarrow 2$	$\swarrow \leftarrow \downarrow 3$	$\swarrow \leftarrow \downarrow 4$	$\swarrow \leftarrow \downarrow 5$	$\swarrow \leftarrow \downarrow 6$	$\swarrow \leftarrow \downarrow 7$	$\swarrow 6$	$\leftarrow 7$	$\leftarrow 8$	
#	0	1	2	3	4	5	6	7	8	9	
	#	e	x	e	c	u	t	i	o	n	



Adding Backtrace to Minimum Edit Distance

- Base conditions:

$$D(i, 0) = i \quad D(0, j) = j$$

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) & \text{substitution} \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

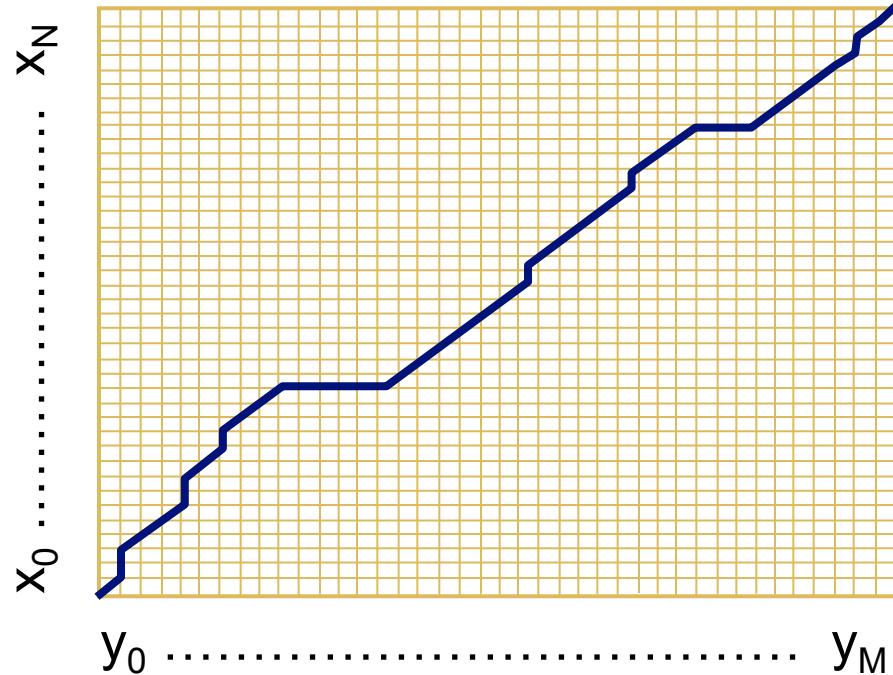
$$\text{ptr}(i, j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

Termination:

$D(N, M)$ is distance



The Distance Matrix



Every non-decreasing path

from $(0,0)$ to (M, N)

corresponds to
an alignment
of the two sequences

An optimal alignment is composed
of optimal subalignments



Result of Backtrace

- Two strings and their **alignment**:

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N



Performance

- Time:

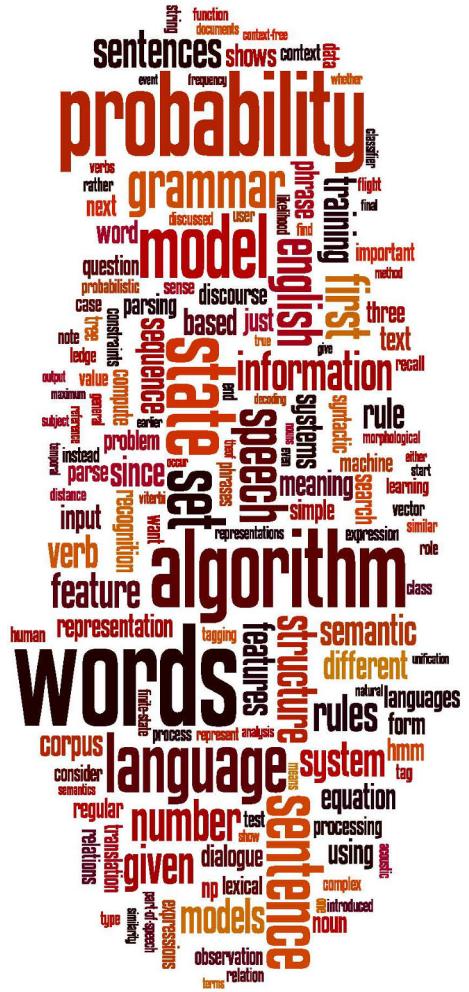
$$O(nm)$$

- Space:

$$O(nm)$$

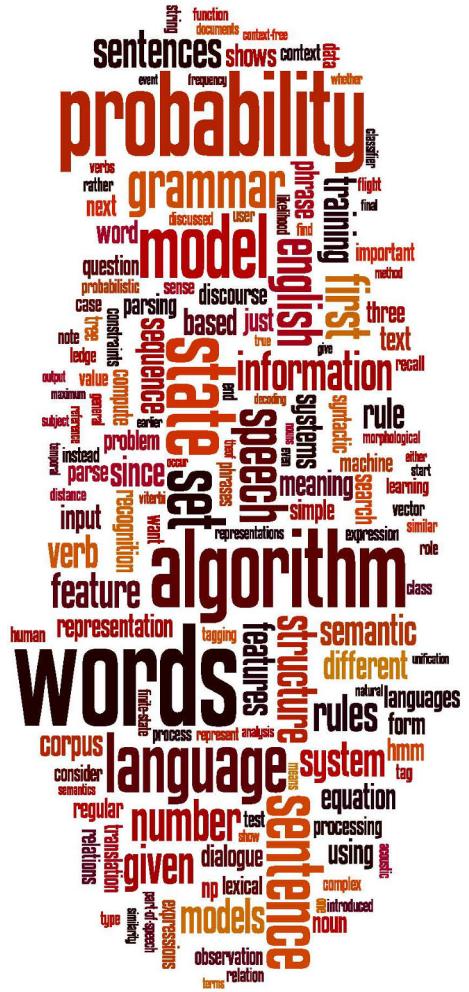
- Backtrace

$$O(n+m)$$



Minimum Edit Distance

Backtrace for Computing Alignments



Minimum Edit Distance

Weighted Minimum Edit Distance



Weighted Edit Distance

- Why would we add weights to the computation?
 - Spell Correction: some letters are more likely to be mistyped than others
 - Biology: certain kinds of deletions or insertions are more likely than others



Confusion matrix for spelling errors

X	sub[X, Y] = Substitution of X (incorrect) for Y (correct)																										
	Y (correct)																										
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0	
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3	0
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2	0
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1	0
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6	0
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0	0

Dan Jurafsky





Weighted Min Edit Distance

- Initialization:

$$D(0,0) = 0$$

$$D(i,0) = D(i-1,0) + \text{del}[x(i)]; \quad 1 < i \leq N$$

$$D(0,j) = D(0,j-1) + \text{ins}[y(j)]; \quad 1 < j \leq M$$

- Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + \text{del}[x(i)] \\ D(i,j-1) + \text{ins}[y(j)] \\ D(i-1,j-1) + \text{sub}[x(i), y(j)] \end{cases}$$

- Termination:

$D(N,M)$ is distance

Dan Jurafsky



Where did the name, dynamic programming, come from?

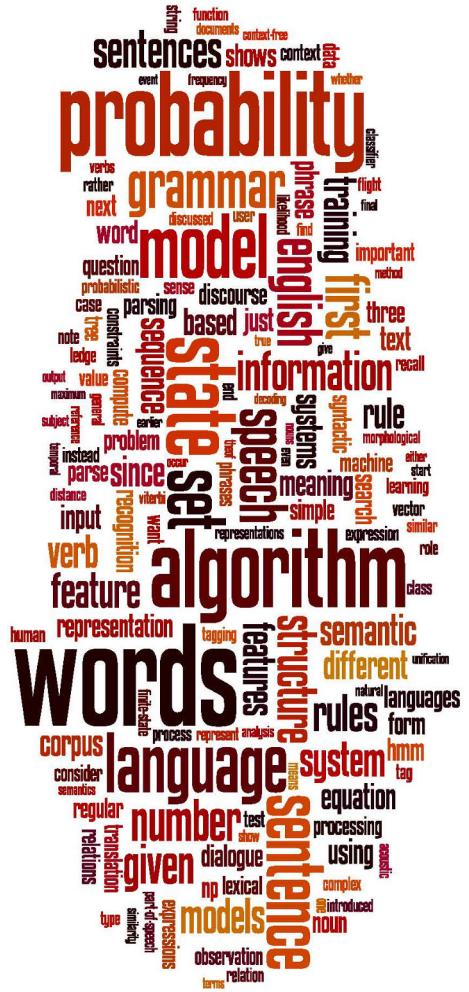
...The 1950s were not good years for mathematical research. [the] Secretary of Defense ...had a pathological fear and hatred of the word, research...

I decided therefore to use the word, “**programming**”.

I wanted to get across the idea that this was dynamic, this was multistage... I thought, let's ... take a word that has an absolutely precise meaning, namely **dynamic**... it's impossible to use the word, **dynamic**, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible.

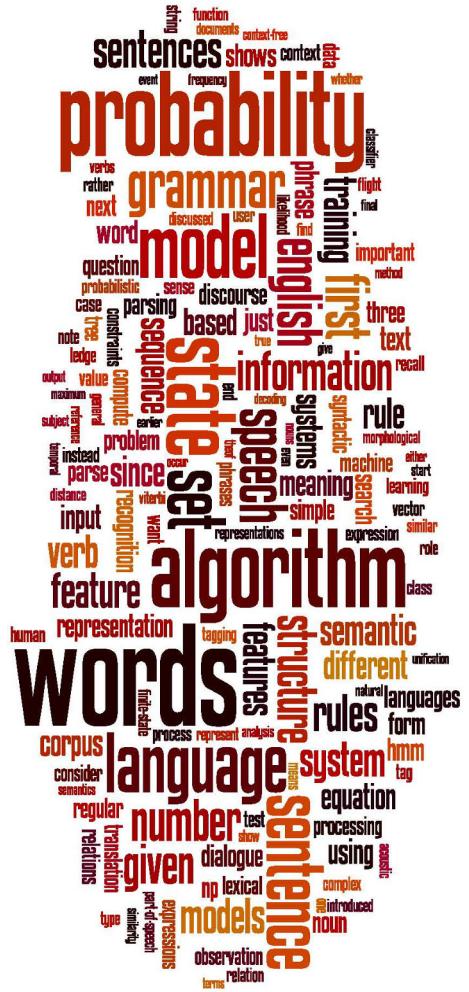
Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to.”

Richard Bellman, “Eye of the Hurricane: an autobiography” 1984.



Minimum Edit Distance

Weighted Minimum Edit Distance



Minimum Edit Distance

Minimum Edit Distance in Computational Biology



Sequence Alignment

AGGCTATCACCTGACCTCCAGGCCGATGCC
TAGCTATCACGACC CGGGT CGATTGCCCGAC

-AG**G**CTATCAC**C**T**G**AC**C**T**C**A**G**GC**C**GA--TG**C**CC---
T**A****G**-CTATCAC--**G**AC**C****G**C--**G****G****T****C****G****A****T****T****T****G****C****C****C****G****A****C**



Why sequence alignment?

- Comparing genes or regions from different species
 - to find important regions
 - determine function
 - uncover evolutionary forces
- Assembling fragments to sequence DNA
- Compare individuals to looking for mutations



Alignments in two fields

- In Natural Language Processing
 - We generally talk about **distance** (minimized)
 - And **weights**
- In Computational Biology
 - We generally talk about **similarity** (maximized)
 - And **scores**



The Needleman-Wunsch Algorithm

- Initialization:

$$D(i, 0) = -i * d$$

$$D(0, j) = -j * d$$

- Recurrence Relation:

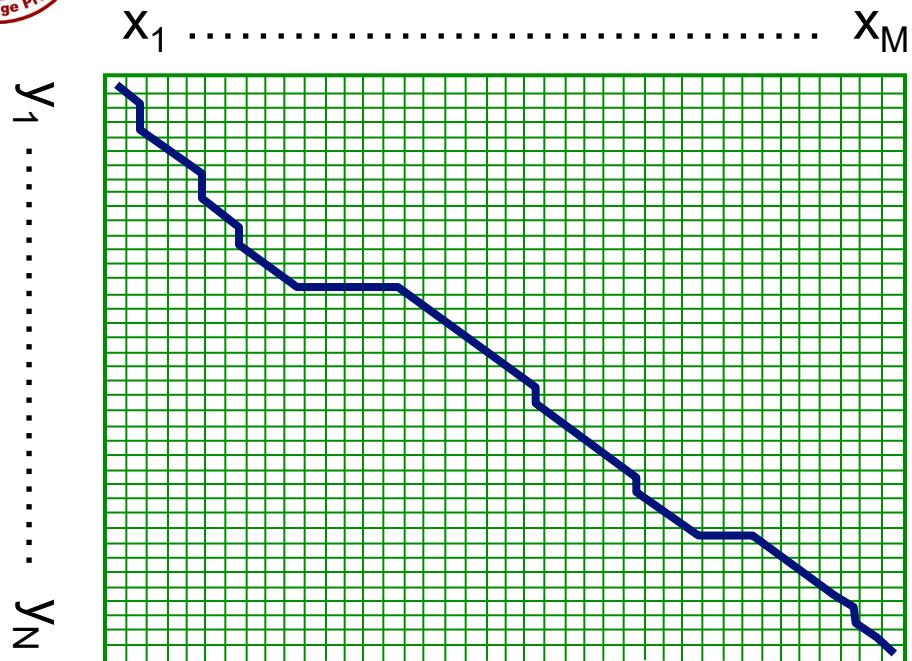
$$D(i, j) = \min \begin{cases} D(i-1, j) - d \\ D(i, j-1) - d \\ D(i-1, j-1) + s[x(i), y(j)] \end{cases}$$

- Termination:

$D(N, M)$ is distance



The Needleman-Wunsch Matrix



(Note that the origin is at the upper left.)



A variant of the basic algorithm:

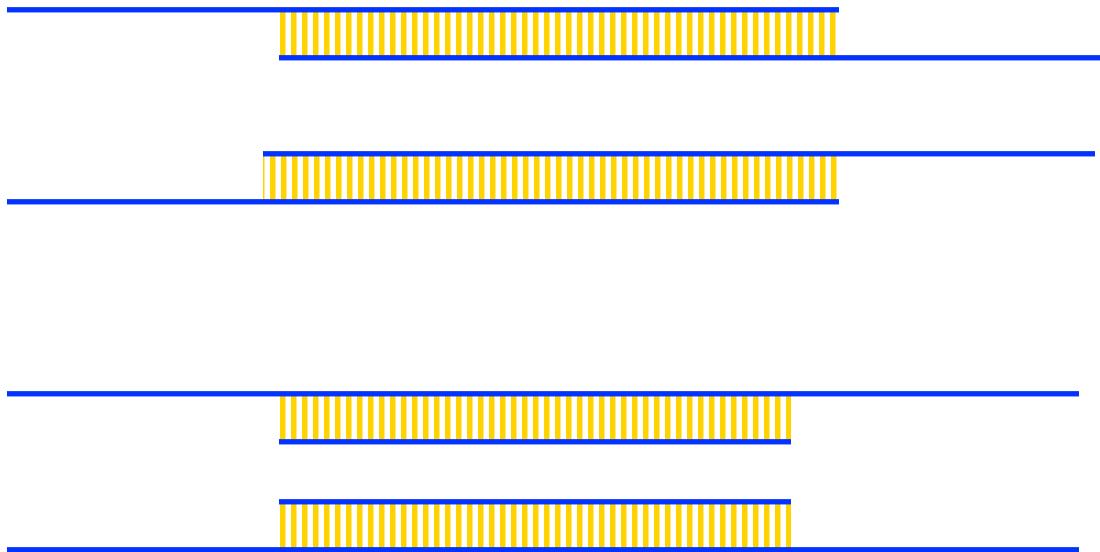
- Maybe it is OK to have an unlimited # of gaps in the beginning and end:

-----**CTATCACCTGACCTCCAGGCCGATGCCCTTCCGGC**
GCGAGTTCATCTATCAC--GACCGC--GGTCG-----

- If so, we don't want to penalize gaps at the ends



Different types of overlaps

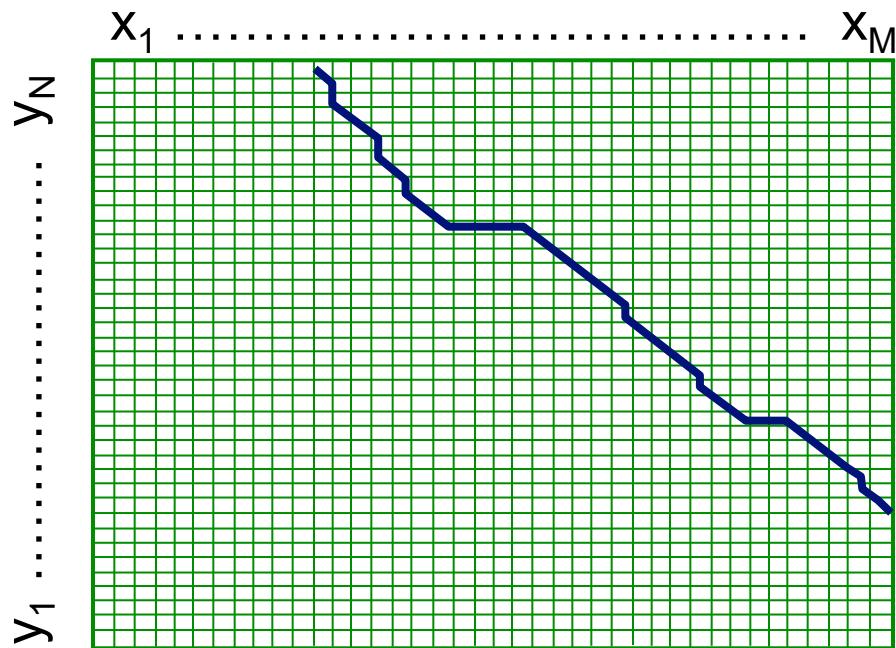


Example:
2 overlapping “reads” from a sequencing project

Example:
Search for a mouse gene within a human chromosome



The Overlap Detection variant



Changes:

1. Initialization
For all i, j ,
 $F(i, 0) = 0$
 $F(0, j) = 0$

- ## 2. Termination

$$F_{OPT} = \max \left\{ \begin{array}{l} \max_i F(i, N) \\ \max_j F(M, j) \end{array} \right.$$

Slide from Serafim Batzoglou



The Local Alignment Problem

Given two strings

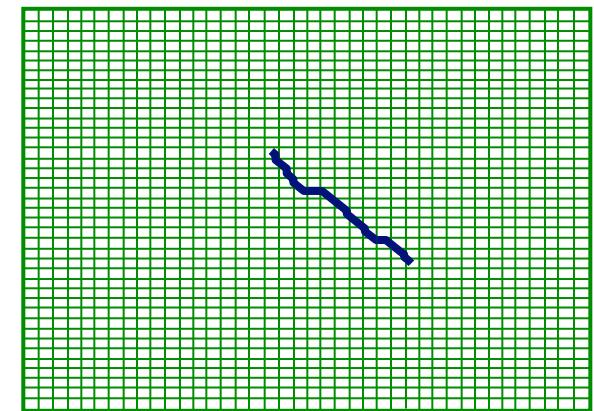
$$x = x_1 \dots x_M,$$

$$y = y_1 \dots y_N$$

Find substrings x' , y' whose similarity
(optimal global alignment value)
is maximum

$x = \text{aaaaccccccgggggtta}$

$y = \text{ttccccgggaaccaacc}$



Slide from Serafim Batzoglou



The Smith-Waterman algorithm

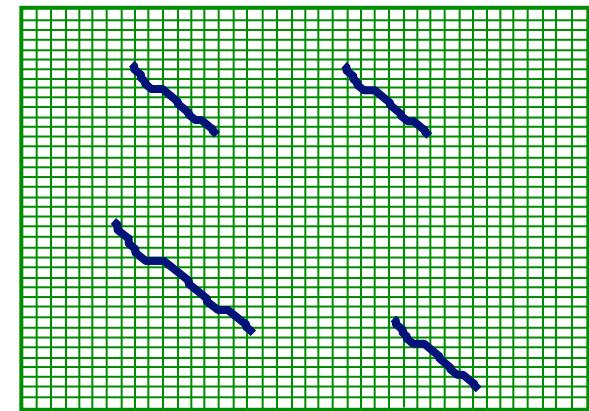
Idea: Ignore badly aligning regions

Modifications to Needleman-Wunsch:

Initialization: $F(0, j) = 0$

$F(i, 0) = 0$

Iteration: $F(i, j) = \max \begin{cases} 0 \\ F(i - 1, j) - d \\ F(i, j - 1) - d \\ F(i - 1, j - 1) + s(x_i, y_j) \end{cases}$



Slide from Serafim Batzoglou



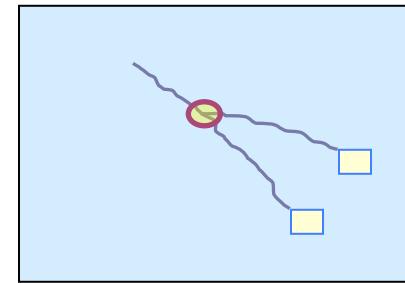
The Smith-Waterman algorithm

Termination:

1. If we want the **best** local alignment...

$$F_{OPT} = \max_{i,j} F(i, j)$$

Find F_{OPT} and trace back



2. If we want **all** local alignments **scoring > t**

??

For all i, j find $F(i, j) > t$, and trace back?

Complicated by overlapping local alignments

Slide from Serafim Batzoglou



Local alignment example

X = ATCAT

Y = ATTATC

Let:

$m = 1$ (1 point for match)

$d = 1$ (-1 point for del/ins/sub)

	A	T	T	A	T	C
	0	0	0	0	0	0
A	0					
T	0					
C	0					
A	0					
T	0					



X = ATCAT

Y = ATTATC

Local alignment example

	A	T	T	A	T	C
	0	0	0	0	0	0
A	0	1	0	0	1	0
T	0	0	2	1	0	2
C	0	0	1	1	0	1
A	0	1	0	0	2	1
T	0	0	2	0	1	3

Arrows point from the sequence Y = ATTATC to the corresponding values in the matrix. Arrows point to the first 'A' (value 0), the second 'T' (value 1), the third 'T' (value 2), the fourth 'A' (value 1), the fifth 'T' (value 2), and the sixth 'C' (value 3). The arrows originate from the sequence Y and point to the matrix cell containing the value 1, 2, 1, 0, 2, and 3 respectively.



X = **ATCAT**

Y = **ATTATC**

Local alignment example

	A	T	T	A	T	C
	0	0	0	0	0	0
A	0	1	0	0	1	0
T	0	0	2	1	0	2
C	0	0	1	1	0	1
A	0	1	0	0	2	1
T	0	0	2	0	1	③

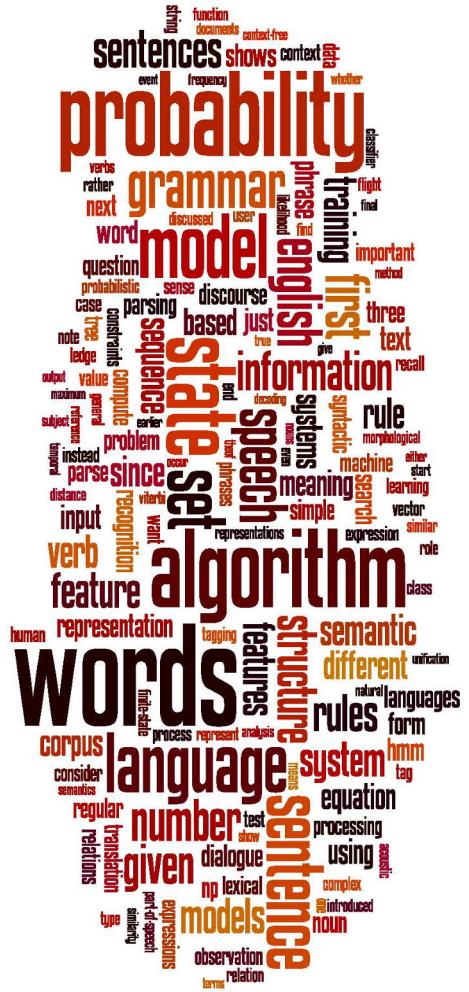


X = **ATC**AT
Y = ATT**ATC**

Local alignment example

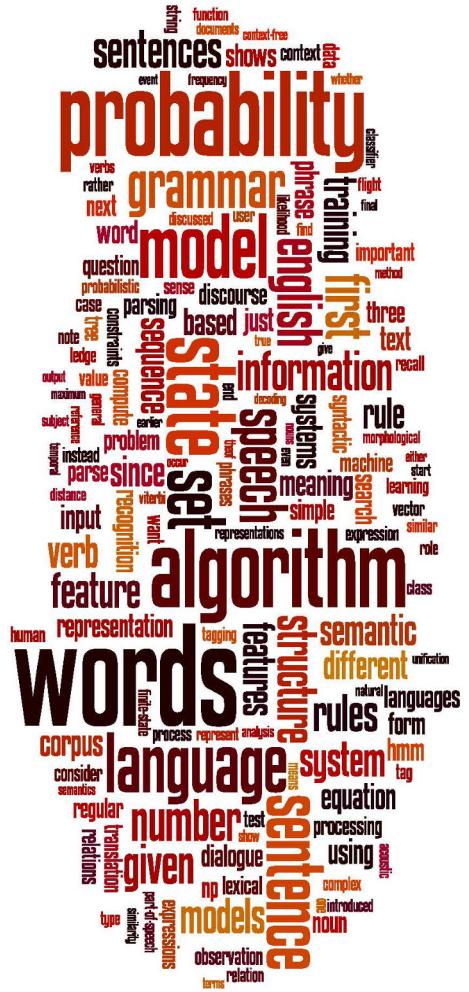
	A	T	T	A	T	C
	0	0	0	0	0	0
A	0	1	0	0	1	0
T	0	0	2	1	0	2
C	0	0	1	1	0	1
A	0	1	0	0	2	1
T	0	0	2	0	1	3

Arrows point from the sequence Y (ATTATC) to the matrix cells containing 1, 2, or 3. A red circle highlights the value 3 in the cell corresponding to the third 'T' in Y and the fourth 'A' in X.



Minimum Edit Distance

Minimum Edit Distance in Computational Biology



Language Modeling

Introduction to N-grams



Probabilistic Language Models

- Today's goal: assign a probability to a sentence
 - Machine Translation:
 - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
 - Spell Correction
 - The office is about fifteen **minuets** from my house
 - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
 - Speech Recognition
 - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
 - + Summarization, question-answering, etc., etc.!!

Why?



Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- A model that computes either of these:

$P(W)$ or $P(w_n | w_1, w_2 \dots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard



How to compute $P(W)$

- How to compute this joint probability:
 - $P(\text{its, water, is, so, transparent, that})$
- Intuition: let's rely on the Chain Rule of Probability



Reminder: The Chain Rule

- Recall the definition of conditional probabilities

Rewriting:

- More variables:

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

- The Chain Rule in General

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1, \dots, x_{n-1})$$



The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$P(\text{"its water is so transparent"}) =$

$P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$

$\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$



How to estimate these probabilities

- Could we just count and divide?

$P(\text{the} \mid \text{its water is so transparent that}) =$

$\frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$

- No! Too many possible sentences!
- We'll never see enough data for estimating these



Markov Assumption

- Simplifying assumption:



Andrei Markov

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$



Markov Assumption

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$



Simplest case: Unigram model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the



Bigram model

- Condition on the previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

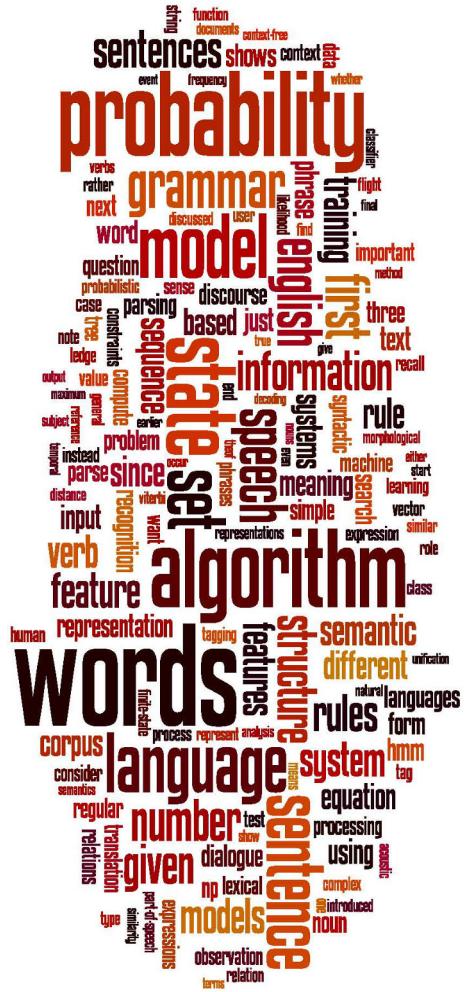
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached
this, would, be, a, record, november



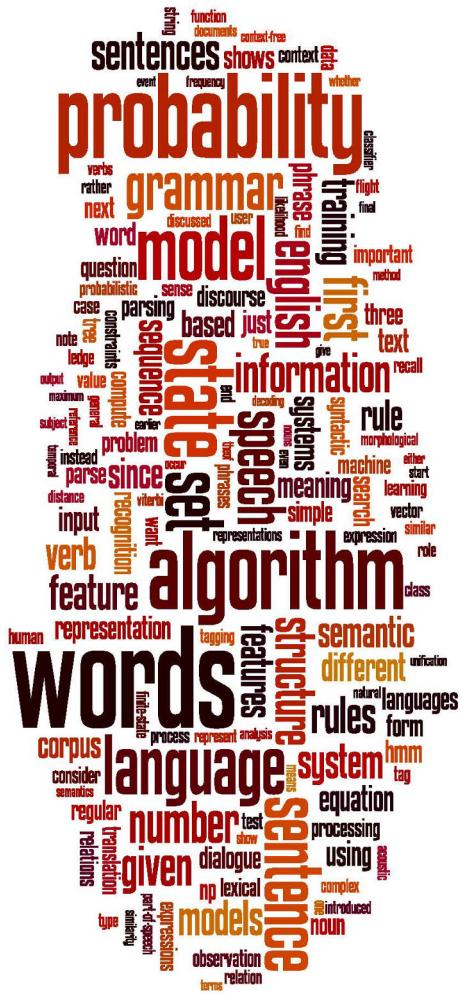
N-gram models

- We can extend to trigrams, 4-grams, 5-grams
 - In general this is an insufficient model of language
 - because language has **long-distance dependencies**:
- “The computer which I had just put into the machine room on the fifth floor crashed.”
- But we can often get away with N-gram models



Language Modeling

Introduction to N-grams



Language Modeling

Estimating N-gram
Probabilities



Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$



An example

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

< s > I am Sam < /s >

< s > Sam I am < /s >

< s > I do not like green eggs and ham < /s >

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

$$P(\text{do} | \text{I}) = \frac{1}{3} = .33$$



More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day



Raw bigram counts

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0



Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0



Bigram estimates of sentence probabilities

$P(< s > | \text{I want english food } < /s >) =$

$$P(\text{I} | < s >)$$

$$\times P(\text{want} | \text{I})$$

$$\times P(\text{english} | \text{want})$$

$$\times P(\text{food} | \text{english})$$

$$\times P(< /s > | \text{food})$$

$$= .000031$$



What kinds of knowledge?

- $P(\text{english} \mid \text{want}) = .0011$
- $P(\text{chinese} \mid \text{want}) = .0065$
- $P(\text{to} \mid \text{want}) = .66$
- $P(\text{eat} \mid \text{to}) = .28$
- $P(\text{food} \mid \text{to}) = 0$
- $P(\text{want} \mid \text{spend}) = 0$
- $P(\text{i} \mid \langle s \rangle) = .25$



Practical Issues

- We do everything in log space
 - Avoid underflow
 - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Dan Jurafsky



Language Modeling Toolkits

- SRILM
 - <http://www.speech.sri.com/projects/srilm/>

Dan Jurafsky



Google N-Gram Release, August 2006

AUG

3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word [n-gram models](#) for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.



Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

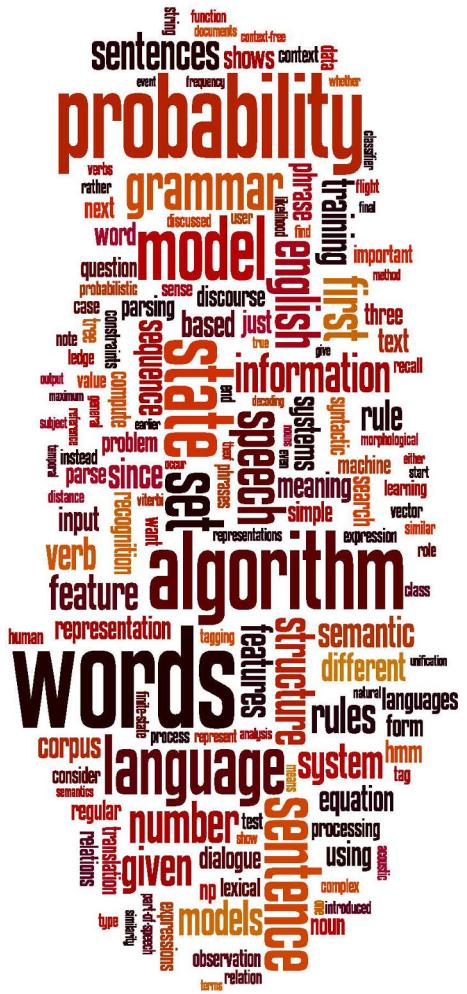
<http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html>

Dan Jurafsky



Google Book N-grams

- <http://ngrams.googlecode.com/>



Language Modeling

Estimating N-gram
Probabilities



Language Modeling

Evaluation and Perplexity



Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
 - Assign higher probability to “real” or “frequently observed” sentences
 - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a **training set**.
- We test the model’s performance on data we haven’t seen.
 - A **test set** is an unseen dataset that is different from our training set, totally unused.
 - An **evaluation metric** tells us how well our model does on the test set.



Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
 - Put each model in a task
 - spelling corrector, speech recognizer, MT system
 - Run the task, get an accuracy for A and for B
 - How many misspelled words corrected properly
 - How many words translated correctly
 - Compare accuracy for A and B



Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
 - Time-consuming; can take days or weeks
- So
 - Sometimes use **intrinsic** evaluation: **perplexity**
 - Bad approximation
 - unless the test data looks **just** like the training data
 - So **generally only useful in pilot experiments**
 - But is helpful to think about.



Intuition of Perplexity

- The Shannon Game:
 - How well can we predict the next word?

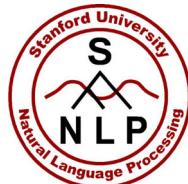
I always order pizza with cheese and _____

The 33rd President of the US was _____

I saw a _____

 - Unigrams are terrible at this game. (Why?)
- A better model of a text
 - is one which assigns a higher probability to the word that actually occurs

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
....
fried rice 0.0001
....
and 1e-100



Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest $P(\text{sentence})$

Perplexity is the inverse probability of the test set, normalized by the number of words:

Chain rule:

For bigrams:

$$\begin{aligned} PP(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \end{aligned}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability



The Shannon Game intuition for perplexity

- From Josh Goodman
- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9'
 - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
 - Perplexity = 30,000
- If a system has to recognize
 - Operator (1 in 4)
 - Sales (1 in 4)
 - Technical Support (1 in 4)
 - 30,000 names (1 in 120,000 each)
 - Perplexity is 53
- Perplexity is weighted equivalent branching factor



Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign $P=1/10$ to each digit?

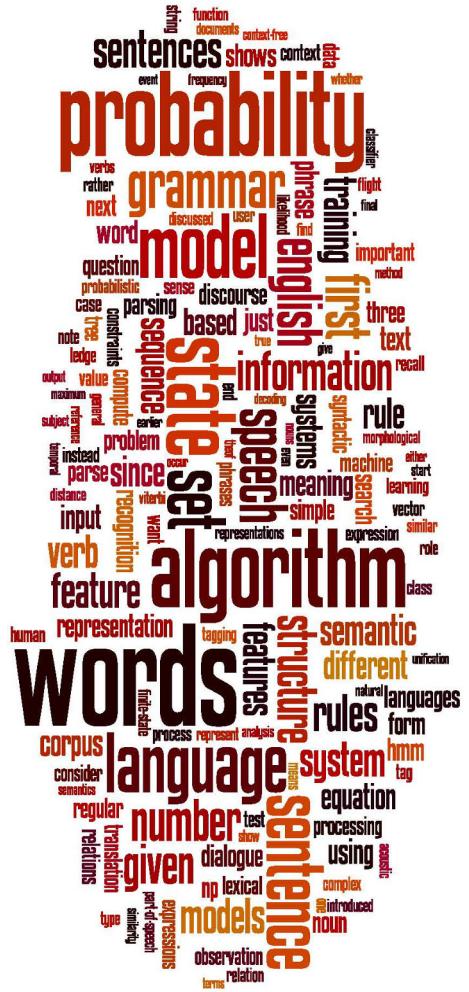
$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$



Lower perplexity = better model

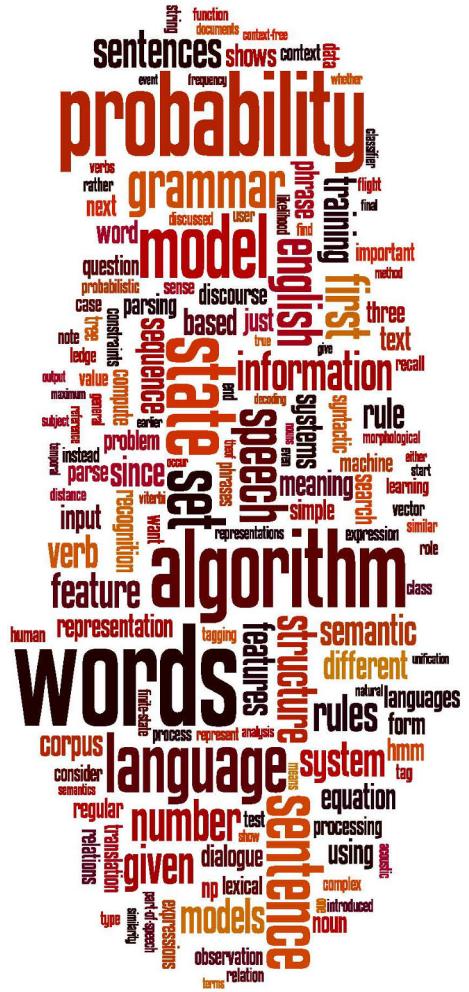
- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109



Language Modeling

Evaluation and Perplexity



Language Modeling

Generalization and zeros



The Shannon Visualization Method

- Choose a random bigram ($\langle s \rangle, w$) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose $\langle /s \rangle$
- Then string the words together

$\langle s \rangle$ I
I want
want to
to eat
eat Chinese
Chinese food
food $\langle /s \rangle$

I want to eat Chinese food



Approximating Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.



Shakespeare as corpus

- $N=884,647$ tokens, $V=29,066$
- Shakespeare produced 300,000 bigram types out of $V^2= 844$ million possible bigrams.
 - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it *is* Shakespeare



The wall street journal is not shakespeare (no offense)

Unigram

Months the my and issue of year foreign new exchange's september were recession ex-change new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions



The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
 - In real life, it often doesn't
 - We need to train robust models that generalize!
 - One kind of generalization: Zeros!
 - Things that don't ever occur in the training set
 - But occur in the test set



Zeros

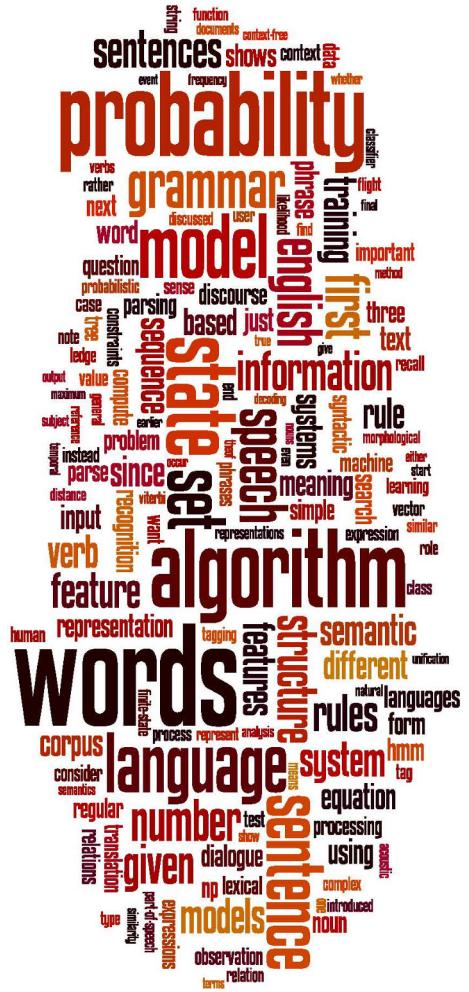
- Training set:
 - ... denied the allegations
 - ... denied the reports
 - ... denied the claims
 - ... denied the request
- Test set
 - ... denied the offer
 - ... denied the loan

$$P(\text{"offer"} \mid \text{denied the}) = 0$$



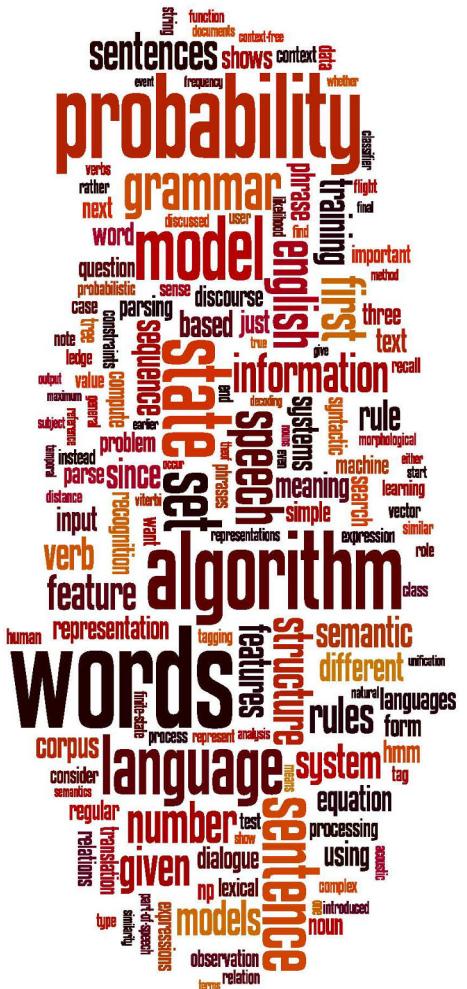
Zero probability bigrams

- Bigrams with zero probability
 - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!



Language Modeling

Generalization and zeros



Language Modeling

Smoothing: Add-one
(Laplace) smoothing



The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

$P(w | \text{denied the})$

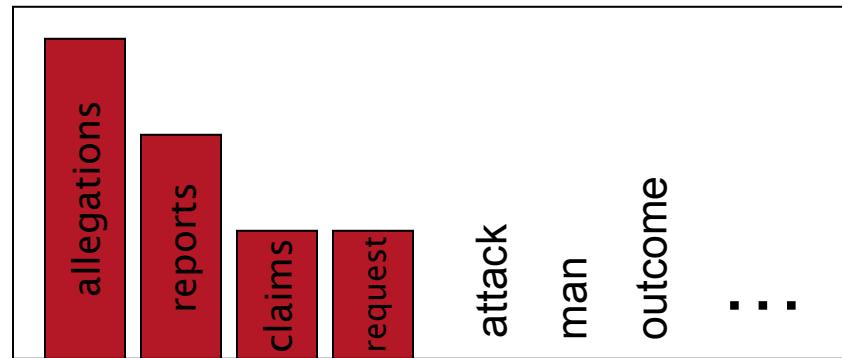
3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better

$P(w | \text{denied the})$

2.5 allegations

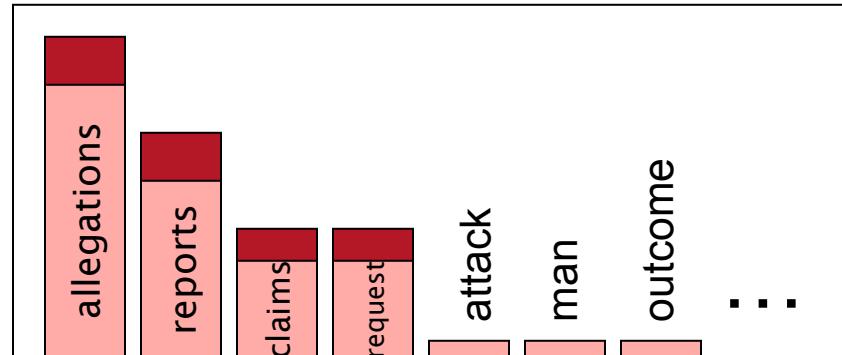
1.5 reports

0.5 claims

0.5 request

2 other

7 total





Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- MLE estimate:

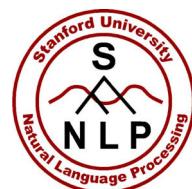
$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

- Add-1 estimate:



Maximum Likelihood Estimates

- The maximum likelihood estimate
 - of some parameter of a model M from a training set T
 - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is $400/1,000,000 = .0004$
- This may be a bad estimate for some other corpus
 - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.



Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1



Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

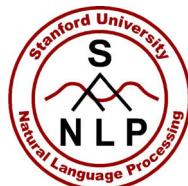
	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058



Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16



Compare with raw bigram counts

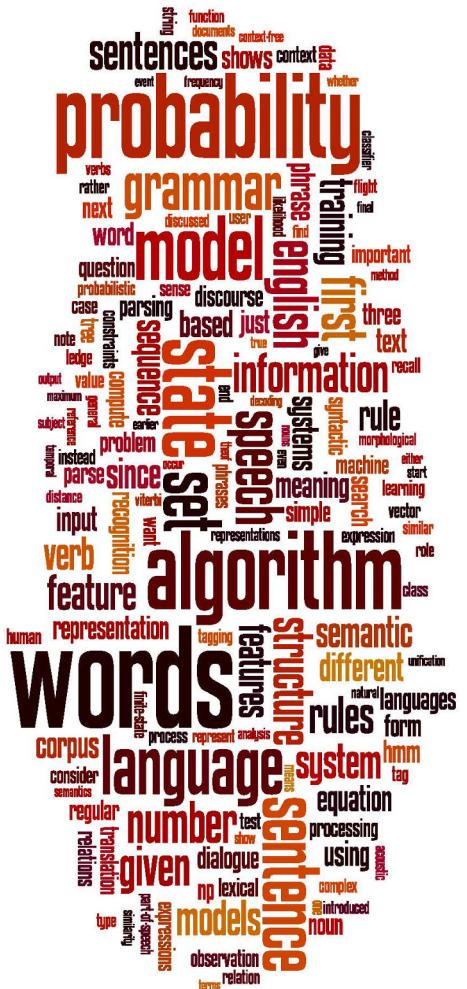
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16



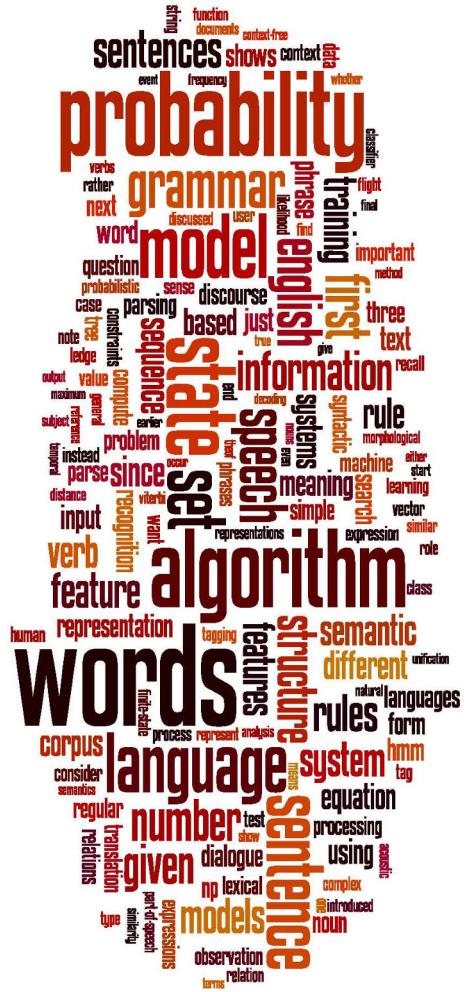
Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
 - We'll see better methods
- But add-1 is used to smooth other NLP models
 - For text classification
 - In domains where the number of zeros isn't so huge.



Language Modeling

Smoothing: Add-one
(Laplace) smoothing



Language Modeling

Interpolation, Backoff, and Web-Scale LMs



Backoff and Interpolation

- Sometimes it helps to use **less** context
 - Condition on less context for contexts you haven't learned much about
- **Backoff:**
 - use trigram if you have good evidence,
 - otherwise bigram, otherwise unigram
- **Interpolation:**
 - mix unigram, bigram, trigram
- Interpolation works better



Linear Interpolation

- Simple interpolation

$$\begin{aligned}\hat{P}(w_n | w_{n-1} w_{n-2}) = & \lambda_1 P(w_n | w_{n-1} w_{n-2}) \\ & + \lambda_2 P(w_n | w_{n-1}) \\ & + \lambda_3 P(w_n)\end{aligned}$$

$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n | w_{n-2} w_{n-1}) = & \lambda_1(w_{n-2}^{n-1}) P(w_n | w_{n-2} w_{n-1}) \\ & + \lambda_2(w_{n-2}^{n-1}) P(w_n | w_{n-1}) \\ & + \lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$



How to set the lambdas?

- Use a **held-out** corpus

Training Data

Held-Out
Data

Test
Data

- Choose λ s to maximize the probability of held-out data:
 - Fix the N-gram probabilities (on the training data)
 - Then search for λ s that give largest probability to held-out set:

$$\log P(w_1 \dots w_n | M(\lambda_1 \dots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \dots \lambda_k)}(w_i | w_{i-1})$$



Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
 - Vocabulary V is fixed
 - Closed vocabulary task
- Often we don't know this
 - **Out Of Vocabulary** = OOV words
 - Open vocabulary task
- Instead: create an unknown word token <UNK>
 - Training of <UNK> probabilities
 - Create a fixed lexicon L of size V
 - At text normalization phase, any training word not in L changed to <UNK>
 - Now we train its probabilities like a normal word
 - At decoding time
 - If text input: Use UNK probabilities for any word not in training



Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
 - Only store N-grams with count > threshold.
 - Remove singletons of higher-order n-grams
 - Entropy-based pruning
- Efficiency
 - Efficient data structures like tries
 - Bloom filters: approximate language models
 - Store words as indexes, not strings
 - Use Huffman coding to fit large numbers of words into two bytes
 - Quantize probabilities (4-8 bits instead of 8-byte float)



Smoothing for Web-scale N-grams

- “Stupid backoff” (Brants *et al.* 2007)
- No discounting, just use relative frequencies

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ 0.4S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$



N-gram Smoothing Summary

- Add-1 smoothing:
 - OK for text categorization, not for language modeling
- The most commonly used method:
 - Extended Interpolated Kneser-Ney
- For very large N-grams like the Web:
 - Stupid backoff

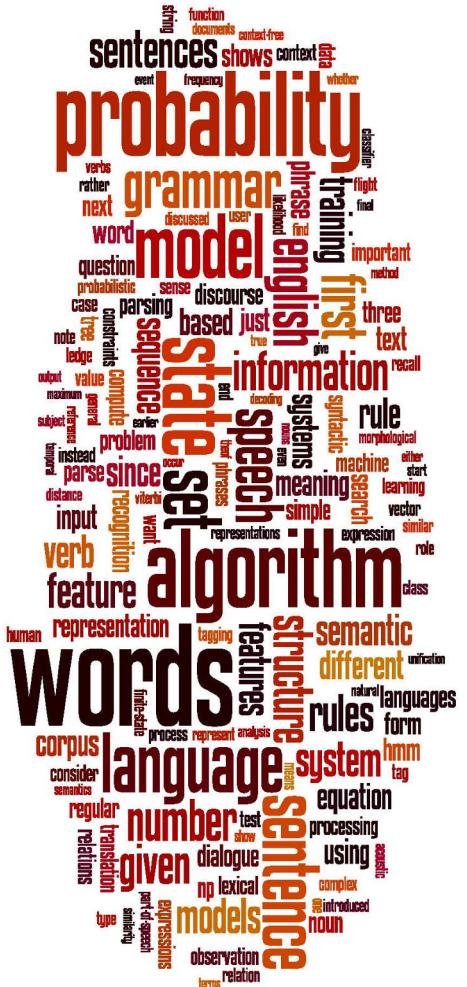


Advanced Language Modeling

- Discriminative models:
 - choose n-gram weights to improve a task, not to fit the training set
- Parsing-based models
- Caching Models
 - Recently used words are more likely to appear

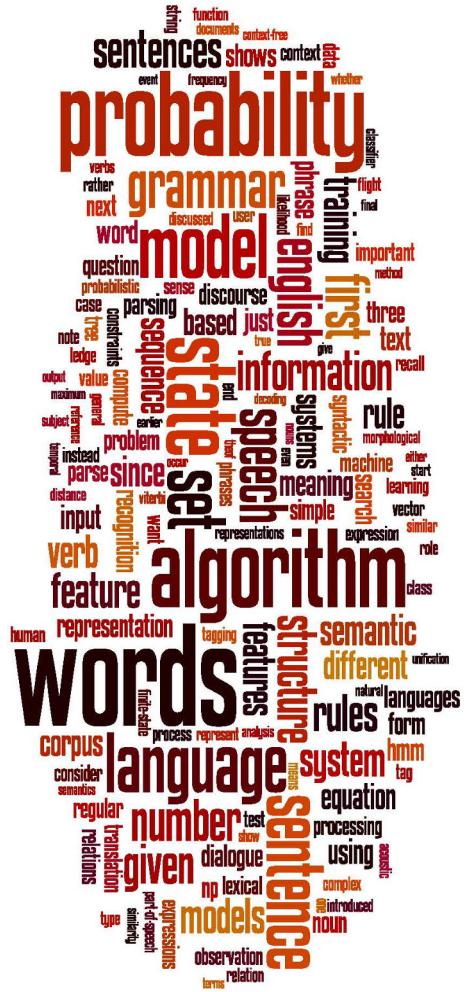
$$P_{CACHE}(w | history) = \lambda P(w_i | w_{i-2} w_{i-1}) + (1 - \lambda) \frac{c(w \in history)}{|history|}$$

- These perform very poorly for speech recognition (why?)



Language Modeling

Interpolation, Backoff,
and Web-Scale LMs



Language Modeling

Advanced: Good Turing Smoothing



Reminder: Add-1 (Laplace) Smoothing

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$



More general formulations: Add-k

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$



Unigram prior smoothing

$$P_{Add-k}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

$$P_{\text{UnigramPrior}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$



Advanced smoothing algorithms

- Intuition used by many smoothing algorithms
 - Good-Turing
 - Kneser-Ney
 - Witten-Bell
- Use the count of things we've **seen once**
 - to help estimate the count of things we've **never seen**



Notation: N_c = Frequency of frequency c

- N_c = the count of things we've seen c times
- Sam I am I am Sam I do not eat

I 3

sam 2

am 2

$$N_1 = 3$$

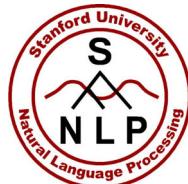
do 1

$$N_2 = 2$$

not 1

$$N_3 = 1$$

eat 1



Good-Turing smoothing intuition

- You are fishing (a scenario from Josh Goodman), and caught:
 - 10 carp, 3 perch, 2 whitefish, **1 trout, 1 salmon, 1 eel** = 18 fish
- How likely is it that next species is trout?
 - $1/18$
- How likely is it that next species is new (i.e. catfish or bass)
 - Let's use our estimate of things-we-saw-once to estimate the new things.
 - $3/18$ (because $N_1=3$)
- Assuming so, how likely is it that next species is trout?
 - Must be less than $1/18$
 - How to estimate?



Good Turing calculations

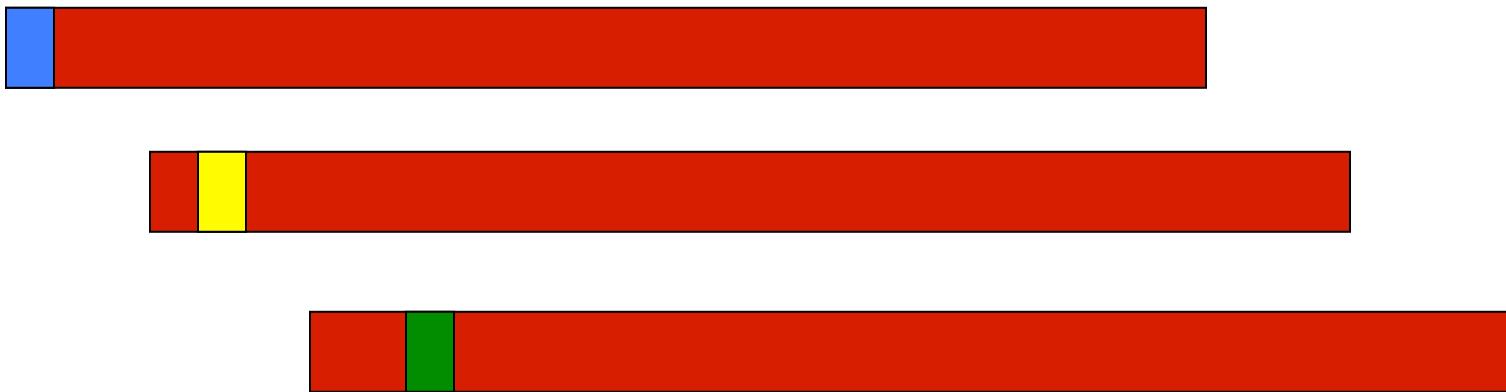
$$P_{GT}^*(\text{things with zero frequency}) = \frac{N_1}{N} \quad c^* = \frac{(c+1)N_{c+1}}{N_c}$$

- Unseen (bass or catfish)
 - $c = 0$:
 - MLE $p = 0/18 = 0$
 - $P_{GT}^*(\text{unseen}) = N_1/N = 3/18$
- Seen once (trout)
 - $c = 1$
 - MLE $p = 1/18$
 - $C^*(\text{trout}) = 2 * N_2/N_1$
 $= 2 * 1/3$
 $= 2/3$
 - $P_{GT}^*(\text{trout}) = 2/3 / 18 = 1/27$



Ney et al.'s Good Turing Intuition

H. Ney, U. Essen, and R. Kneser, 1995. On the estimation of 'small' probabilities by leaving-one-out.
IEEE Trans. PAMI. 17:12,1202-1212



Held-out words:



Ney et al. Good Turing Intuition (slide from Dan Klein)

- Intuition from leave-one-out validation
 - Take each of the c training words out in turn
 - c training sets of size $c-1$, held-out of size 1
 - What fraction of held-out words are unseen in training?
 - N_1/c
 - What fraction of held-out words are seen k times in training?
 - $(k+1)N_{k+1}/c$
 - So in the future we expect $(k+1)N_{k+1}/c$ of the words to be those with training count k
 - There are N_k words with training count k
 - Each should occur with probability:
 - $(k+1)N_{k+1}/c/N_k$
 - ...or expected count:

$$k^* = \frac{(k+1)N_{k+1}}{N_k}$$

Training

N_1
N_2
N_3
⋮
N_{3511}
N_{4417}

Held out

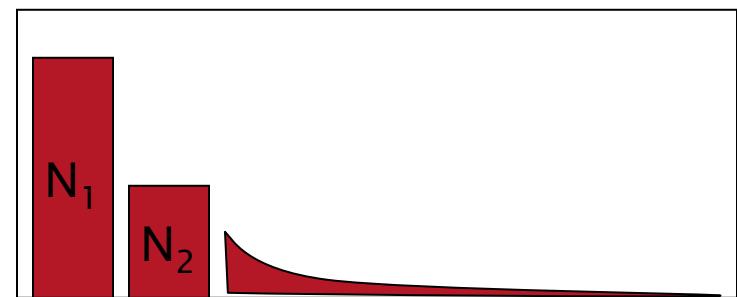
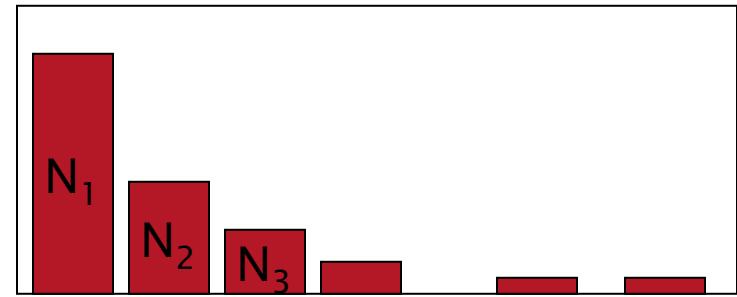
N_0
N_1
N_2
⋮
N_{3510}
N_{4416}



Good-Turing complications

(slide from Dan Klein)

- Problem: what about “the”? (say $c=4417$)
 - For small k , $N_k > N_{k+1}$
 - For large k , too jumpy, zeros wreck estimates
- Simple Good-Turing [Gale and Sampson]: replace empirical N_k with a best-fit power law once counts get unreliable



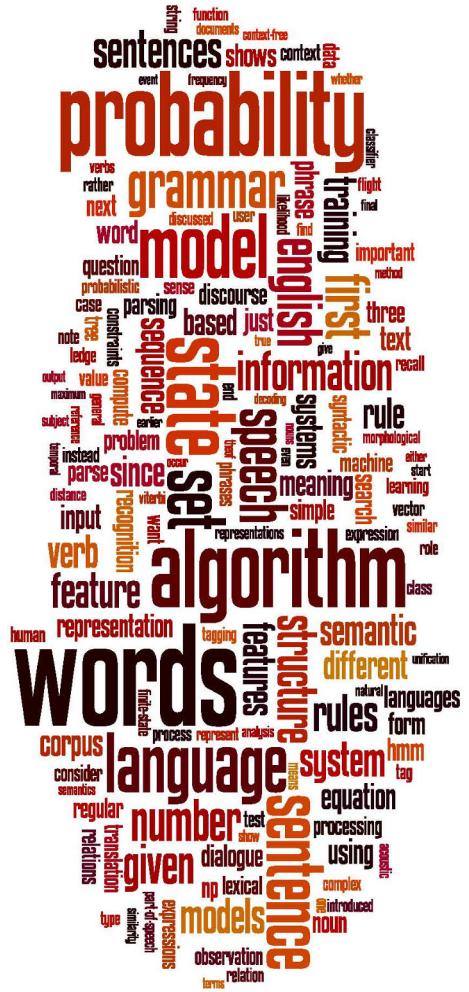


Resulting Good-Turing numbers

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

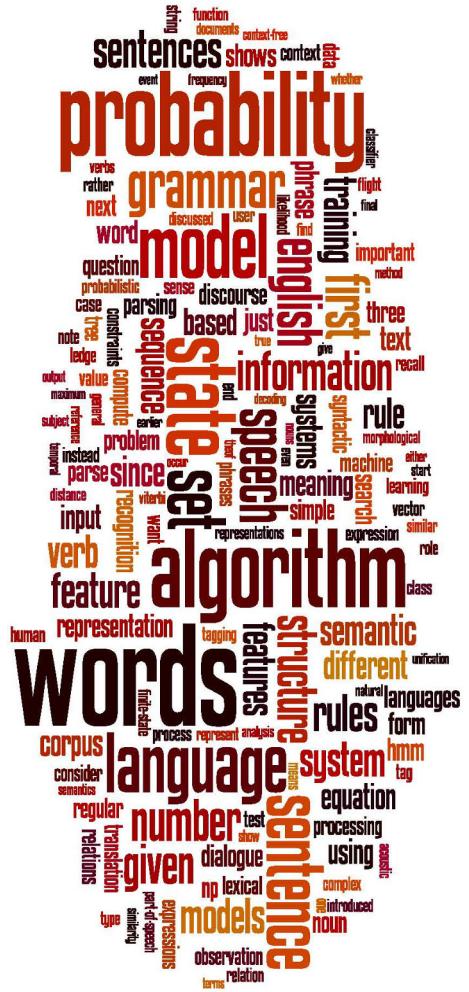
$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25



Language Modeling

Advanced: Good Turing Smoothing



Language Modeling

Advanced: Kneser-Ney Smoothing



Resulting Good-Turing numbers

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

- It sure looks like $c^* = (c - .75)$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25



Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1}) P(w)$$

discounted bigram Interpolation weight
↖ unigram

- (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram $P(w)$?



Kneser-Ney Smoothing I

- Better estimate for probabilities of lower-order unigrams!
 - Shannon game: *I can't see without my reading Francisco?*
 - “Francisco” is more common than “glasses”
 - ... but “Francisco” always follows “San”
- The unigram is useful exactly when we haven’t seen this bigram!
- Instead of $P(w)$: “How likely is w ”
- $P_{\text{continuation}}(w)$: “How likely is w to appear as a novel continuation?
 - For each word, count the number of bigram types it completes
 - Every bigram type was a novel continuation the first time it was seen

$$P_{\text{CONTINUATION}}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$



Kneser-Ney Smoothing II

- How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$



Kneser-Ney Smoothing III

- Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$$

- A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability



Kneser-Ney Smoothing IV

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

the normalized discount

The number of word types that can follow w_{i-1}
 = # of word types we discounted
 = # of times we applied normalized discount

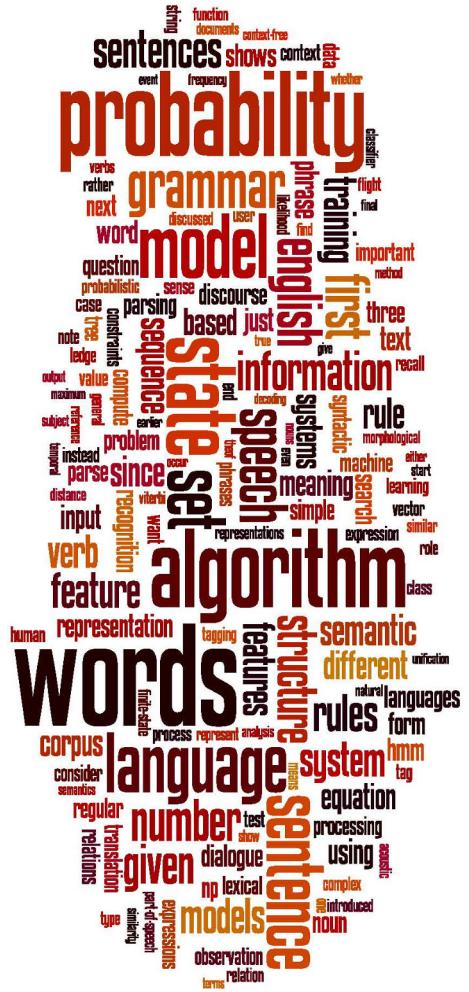


Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i | w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^i) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i | w_{i-n+2}^{i-1})$$

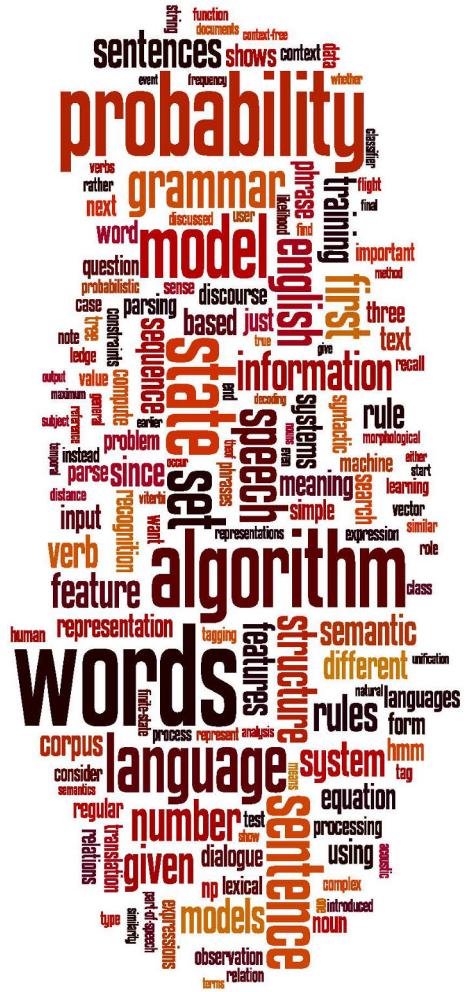
$$c_{KN}(\bullet) = \begin{cases} \text{count}(\bullet) & \text{for the highest order} \\ \text{continuation count}(\bullet) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for \bullet



Language Modeling

Advanced: Kneser-Ney Smoothing



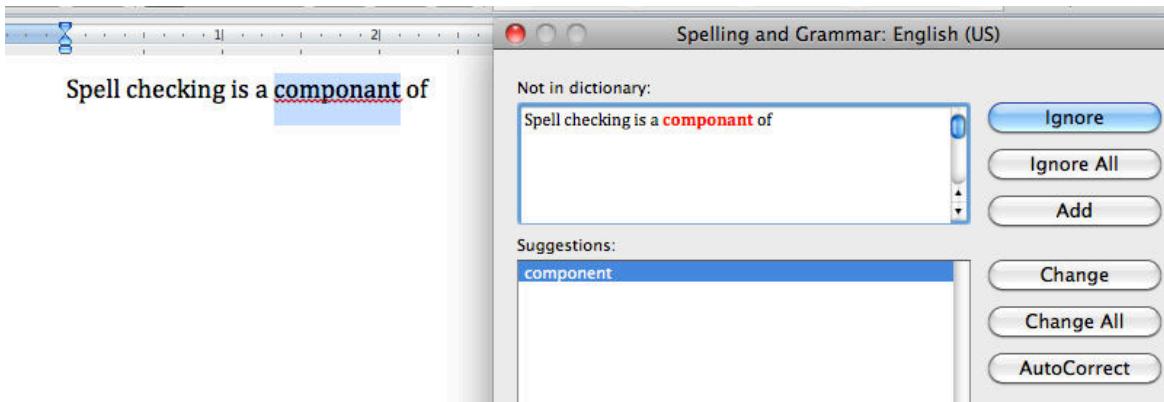
Spelling Correction and the Noisy Channel

The Spelling Correction Task



Applications for spelling correction

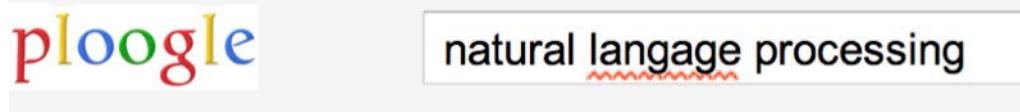
Word processing



Phones



Web search





Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
 - Autocorrect
 - hte → the
 - Suggest a correction
 - Suggestion lists



Types of spelling errors

- Non-word Errors
 - *graffe* → *giraffe*
- Real-word Errors
 - Typographical errors
 - *three* → *there*
 - Cognitive Errors (homophones)
 - *piece* → *peace*,
 - *too* → *two*



Rates of spelling errors

26%: Web queries [Wang et al. 2003](#)

13%: Retyping, no backspace: [Whitelaw et al. English&German](#)

7%: Words corrected retyping on phone-sized organizer

2%: Words uncorrected on organizer [Soukoreff & MacKenzie 2003](#)

1-2%: Retyping: [Kane and Wobbrock 2007](#), [Gruden et al. 1983](#)



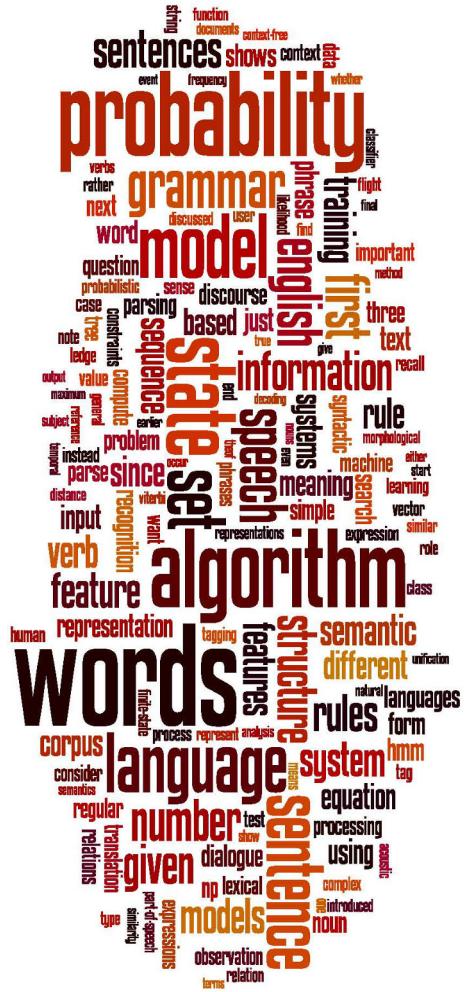
Non-word spelling errors

- Non-word spelling error detection:
 - Any word not in a **dictionary** is an error
 - The larger the dictionary the better
- Non-word spelling error correction:
 - Generate **candidates**: real words that are similar to error
 - Choose the one which is best:
 - Shortest weighted edit distance
 - Highest noisy channel probability



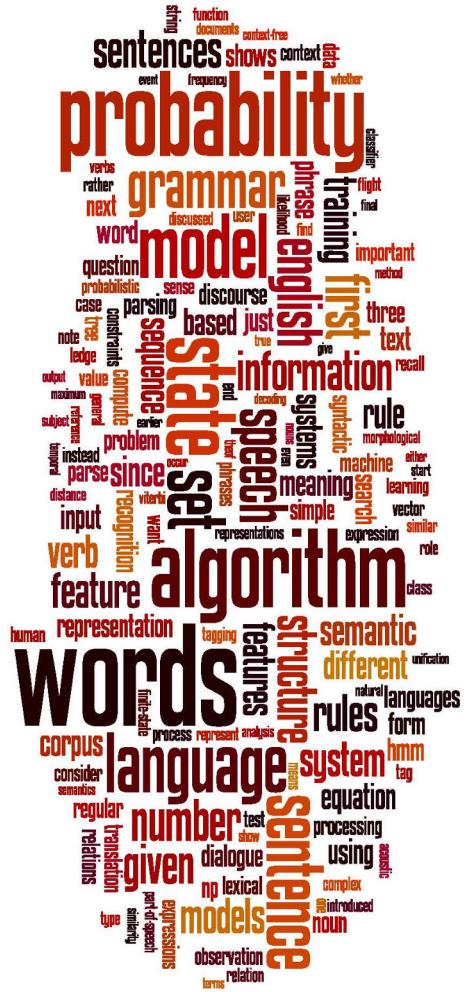
Real word spelling errors

- For each word w , generate candidate set:
 - Find candidate words with similar *pronunciations*
 - Find candidate words with similar *spelling*
 - Include w in candidate set
- Choose best candidate
 - Noisy Channel
 - Classifier



Spelling Correction and the Noisy Channel

The Spelling Correction Task

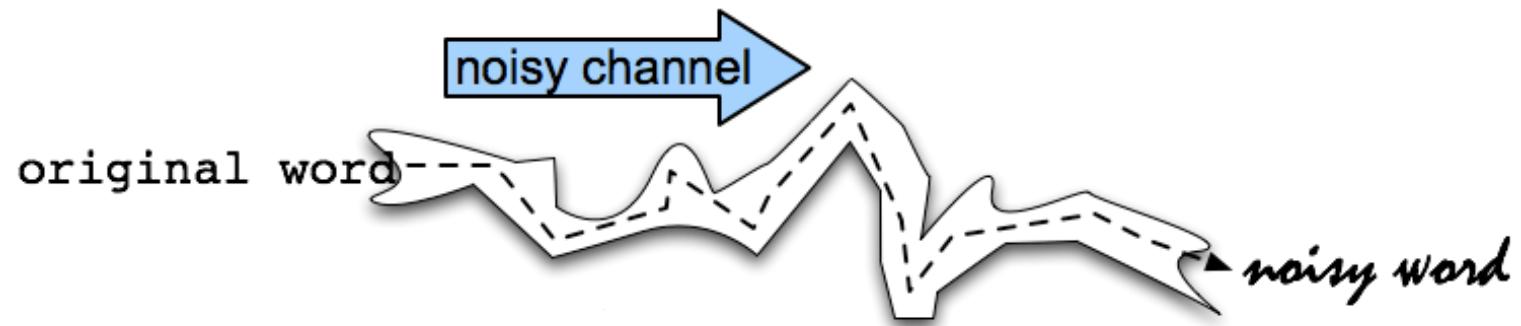


Spelling Correction and the Noisy Channel

The Noisy Channel Model of Spelling



Noisy Channel Intuition





Noisy Channel

- We see an observation x of a misspelled word
- Find the correct word w

$$\begin{aligned}\hat{w} &= \operatorname{argmax}_{w \in V} P(w | x) \\ &= \operatorname{argmax}_{w \in V} \frac{P(x | w)P(w)}{P(x)} \\ &= \operatorname{argmax}_{w \in V} P(x | w)P(w)\end{aligned}$$



History: Noisy channel for spelling proposed around 1990

- **IBM**
 - Mays, Eric, Fred J. Damerau and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5), 517–522
- **AT&T Bell Labs**
 - Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. *Proceedings of COLING 1990*, 205-210



Non-word spelling error example

acress



Candidate generation

- Words with similar spelling
 - Small edit distance to error
- Words with similar pronunciation
 - Small edit distance of pronunciation to error



Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
 - Insertion
 - Deletion
 - Substitution
 - Transposition of two adjacent letters



Words within 1 of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion
acress	acres	-	s	insertion



Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
 - `thisidea` → `this idea`
 - `inlaw` → `in-law`



Language Model

- Use any of the language modeling algorithms we've learned
- Unigram, bigram, trigram
- Web-scale spelling correction
 - Stupid backoff



Unigram Prior probability

Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463



Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*
- *Misspelled word* $x = x_1, x_2, x_3 \dots x_m$
- *Correct word* $w = w_1, w_2, w_3, \dots, w_n$
- $P(x|w)$ = probability of the edit
 - (deletion/insertion/substitution/transposition)



Computing error probability: confusion matrix

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(x typed as y)
trans[x,y]:  count(xy typed as yx)
```

Insertion and deletion conditioned on previous character



Confusion matrix for spelling errors

X	sub[X, Y] = Substitution of X (incorrect) for Y (correct)																									
	Y (correct)																									
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0



Generating the confusion matrix

- Peter Norvig's list of errors
- Peter Norvig's list of counts of single-edit errors



Channel model

Kernighan, Church, Gale 1990

$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$



Channel model for *acress*

Candidate Correction	Correct Letter	Error Letter	$x w$	$P(x word)$
actress	t	-	c ct	.000117
cress	-	a	a #	.00000144
caress	ca	ac	ac ca	.00000164
access	c	r	r c	.000000209
across	o	e	e o	.0000093
acres	-	s	es e	.0000321
acres	-	s	ss s	.0000342



Noisy channel probability for **acress**

Candidate Correction	Correct Letter	Error Letter	x w	P(x word)	P(word)	$10^9 * P(x w)P(w)$
actress	t	-	c ct	.000117	.0000231	2.7
cress	-	a	a #	.00000144	.000000544	.00078
caress	ca	ac	ac ca	.00000164	.00000170	.0028
access	c	r	r c	.000000209	.0000916	.019
across	o	e	e o	.0000093	.000299	2.8
acres	-	s	es e	.0000321	.0000318	1.0
acres	-	s	ss s	.0000342	.0000318	1.0



Noisy channel probability for **acress**

Candidate Correction	Correct Letter	Error Letter	x w	P(x word)	P(word)	$10^9 * P(x w)P(w)$
actress	t	-	c ct	.000117	.0000231	2.7
cress	-	a	a #	.00000144	.000000544	.00078
caress	ca	ac	ac ca	.00000164	.00000170	.0028
access	c	r	r c	.000000209	.0000916	.019
across	o	e	e o	.0000093	.000299	2.8
acres	-	s	es e	.0000321	.0000318	1.0
acres	-	s	ss s	.0000342	.0000318	1.0



Using a bigram language model

- “**a stellar and versatile **acress** whose combination of sass and glamour...**”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = .000021$ $P(\text{whose}|\text{actress}) = .0010$
- $P(\text{across}|\text{versatile}) = .000021$ $P(\text{whose}|\text{across}) = .000006$
- $P(\text{"versatile actress whose"}) = .000021 * .0010 = 210 \times 10^{-10}$
- $P(\text{"versatile across whose"}) = .000021 * .000006 = 1 \times 10^{-10}$



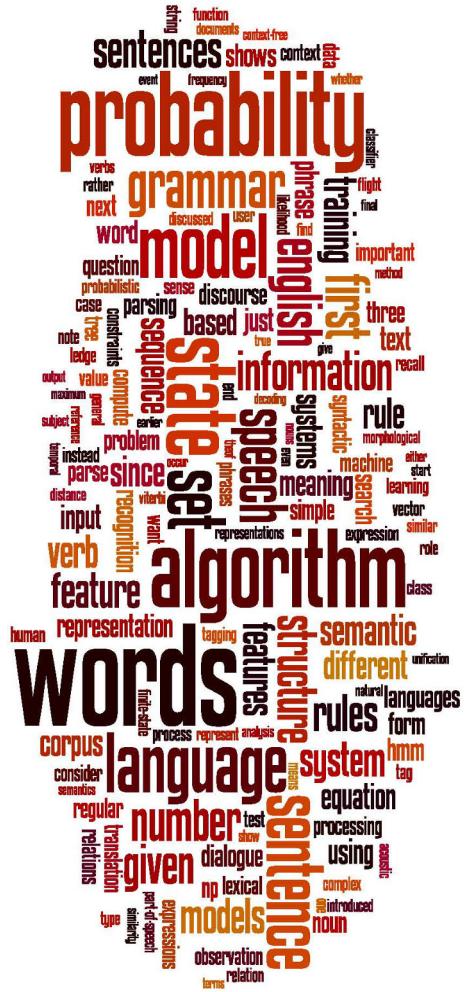
Using a bigram language model

- “**a stellar and versatile **acress** whose combination of sass and glamour...**”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = .000021$ $P(\text{whose}|\text{actress}) = .0010$
- $P(\text{across}|\text{versatile}) = .000021$ $P(\text{whose}|\text{across}) = .000006$
- **$P(\text{"versatile actress whose"}) = .000021 * .0010 = 210 \times 10^{-10}$**
- **$P(\text{"versatile across whose"}) = .000021 * .000006 = 1 \times 10^{-10}$**



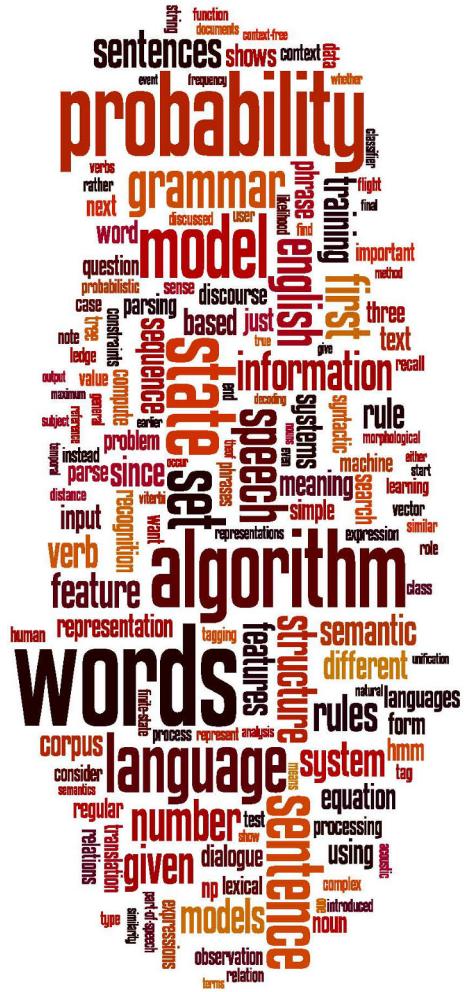
Evaluation

- Some spelling error test sets
 - [Wikipedia's list of common English misspelling](#)
 - [Aspell filtered version of that list](#)
 - [Birkbeck spelling error corpus](#)
 - [Peter Norvig's list of errors \(includes Wikipedia and Birkbeck, for training or testing\)](#)



Spelling Correction and the Noisy Channel

The Noisy Channel Model of Spelling



Spelling Correction and the Noisy Channel

Real-Word Spelling Correction



Real-word spelling errors

- ...leaving in about fifteen *minuets* to go to her house.
- The design *an* construction of the system...
- Can they *lave* him my messages?
- The study was conducted mainly *be* John Black.
- 25-40% of spelling errors are real words Kukich 1992



Solving real-world spelling errors

- For each word in sentence
 - Generate *candidate set*
 - the word itself
 - all single-letter edits that are English words
 - words that are homophones
 - Choose best candidates
 - Noisy channel model
 - Task-specific classifier

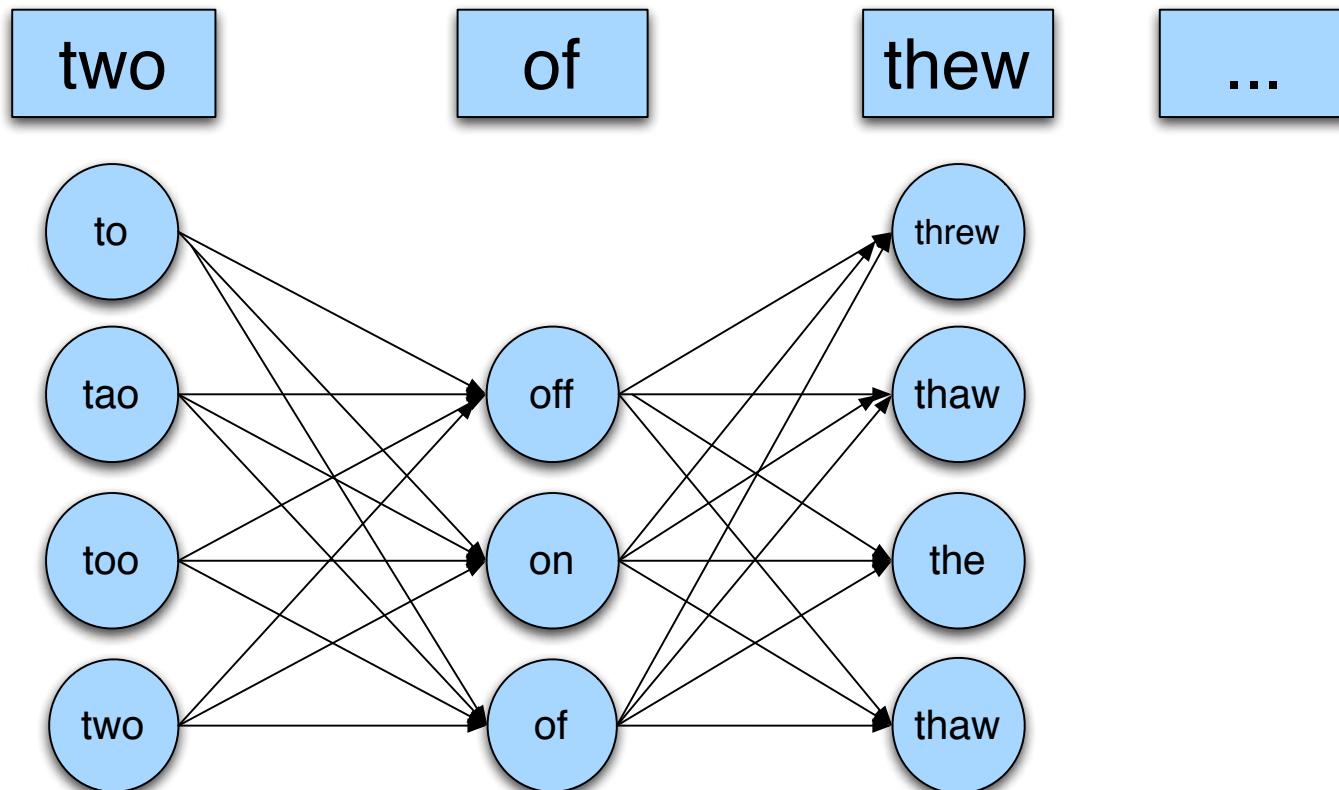


Noisy channel for real-word spell correction

- Given a sentence $w_1, w_2, w_3, \dots, w_n$
- Generate a set of candidates for each word w_i
 - Candidate(w_1) = $\{w_1, w'_1, w''_1, w'''_1, \dots\}$
 - Candidate(w_2) = $\{w_2, w'_2, w''_2, w'''_2, \dots\}$
 - Candidate(w_n) = $\{w_n, w'_n, w''_n, w'''_n, \dots\}$
- Choose the sequence W that maximizes $P(W)$

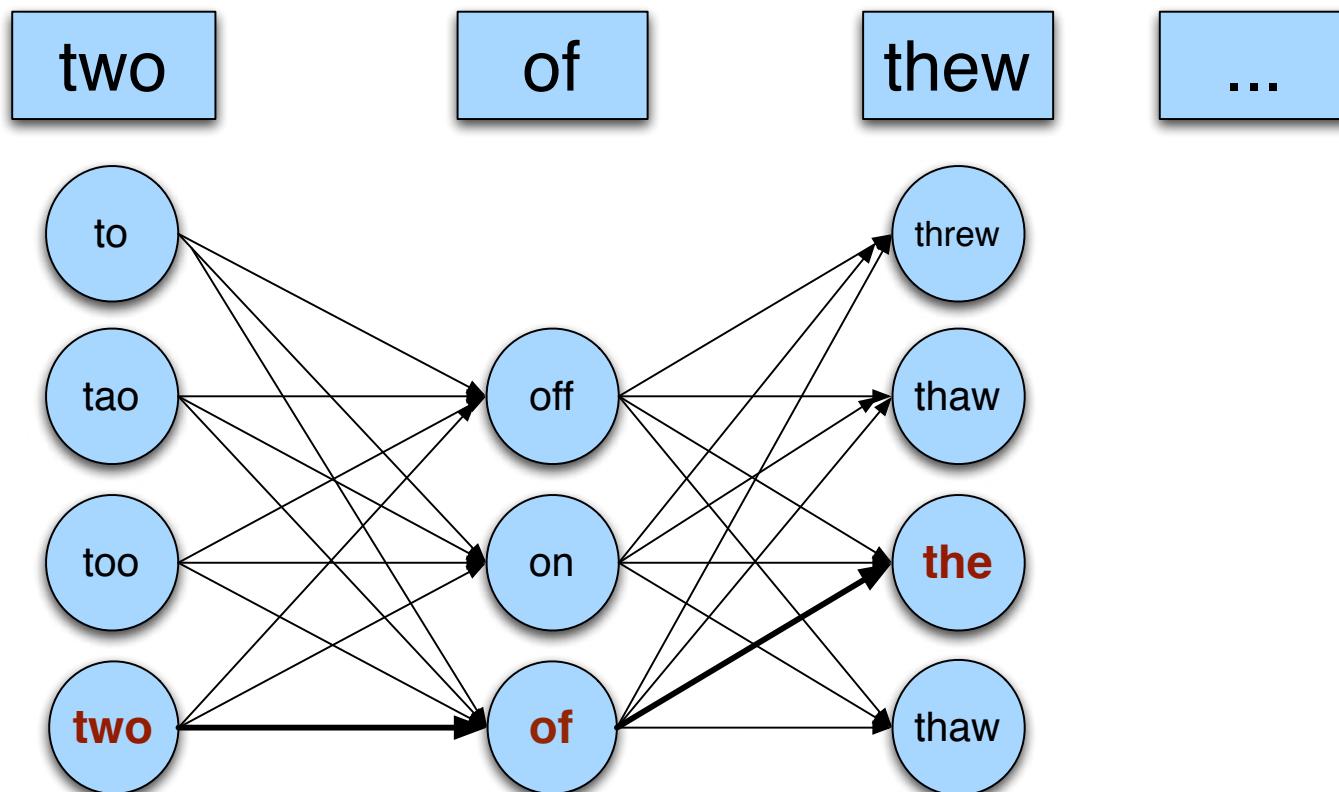


Noisy channel for real-word spell correction





Noisy channel for real-word spell correction





Simplification: One error per sentence

- Out of all possible sentences with one word replaced
 - w_1, w''_2, w_3, w_4 **two off thew**
 - w_1, w_2, w'_3, w_4 **two of the**
 - w'''_1, w_2, w_3, w_4 **too of thew**
 - ...
- Choose the sequence W that maximizes $P(W)$



Where to get the probabilities

- Language model
 - Unigram
 - Bigram
 - Etc
- Channel model
 - Same as for non-word spelling correction
 - Plus need probability for no error, $P(w|w)$



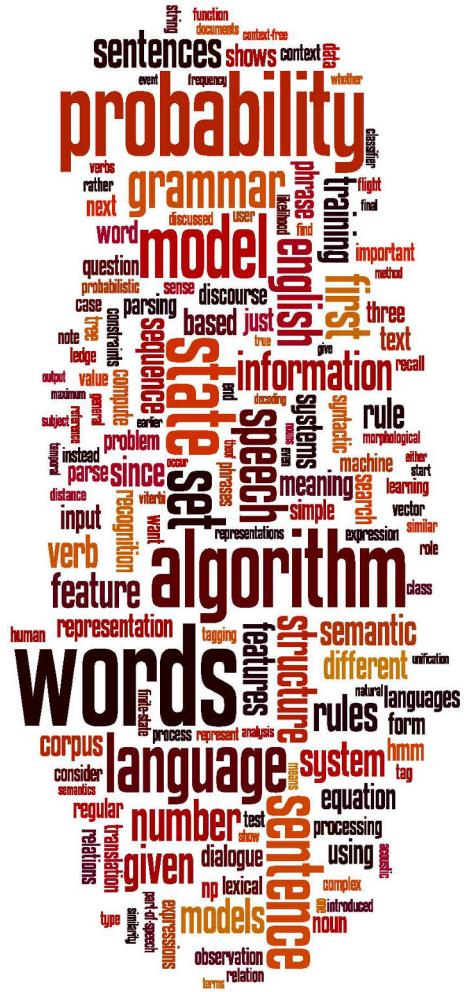
Probability of no error

- What is the channel probability for a correctly typed word?
- $P(\text{"the"} | \text{"the"})$
- Obviously this depends on the application
 - .90 (1 error in 10 words)
 - .95 (1 error in 20 words)
 - .99 (1 error in 100 words)
 - .995 (1 error in 200 words)



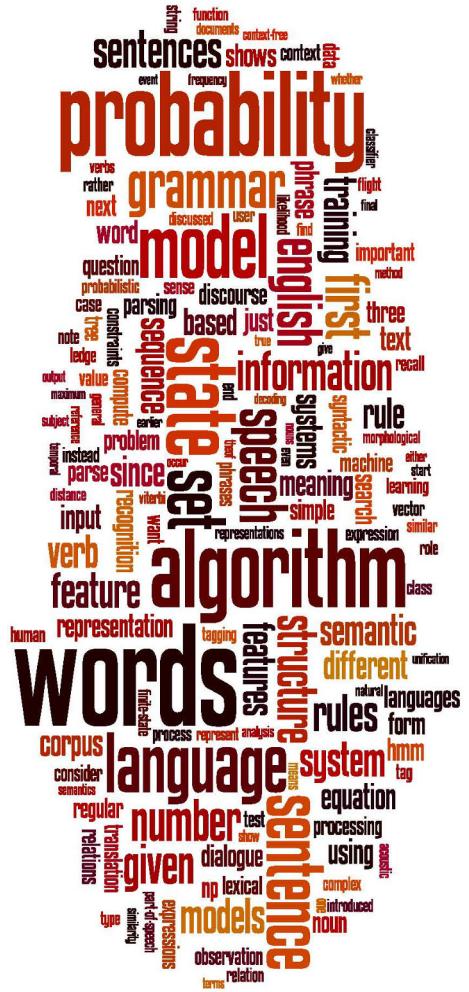
Peter Norvig's “thew” example

x	w	x w	$P(x w)$	$P(w)$	$10^9 P(x w)P(w)$
thew	the	ew e	0.000007	0.02	144
thew	thew		0.95	0.00000009	90
thew	thaw	e a	0.001	0.0000007	0.7
thew	threw	h hr	0.000008	0.000004	0.03
thew	thwe	ew we	0.000003	0.00000004	0.0001



Spelling Correction and the Noisy Channel

Real-Word Spelling Correction



Spelling Correction and the Noisy Channel

State-of-the-art Systems



HCI issues in spelling

- If very confident in correction
 - Autocorrect
- Less confident
 - Give the best correction
- Less confident
 - Give a correction list
- Unconfident
 - Just flag as an error



State of the art noisy channel

- We never just multiply the prior and the error model
- Independence assumptions → probabilities not commensurate
- Instead: Weigh them

$$\hat{w} = \operatorname{argmax}_{w \in V} P(x | w) P(w)^\lambda$$

- Learn λ from a development test set



Phonetic error model

- Metaphone, used in GNU aspell
 - Convert misspelling to metaphone pronunciation
 - “Drop duplicate adjacent letters, except for C.”
 - “If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.”
 - “Drop 'B' if after 'M' and if it is at the end of the word”
 - ...
 - Find words whose pronunciation is 1-2 edit distance from misspelling's
 - Score result list
 - Weighted edit distance of candidate to misspelling
 - Edit distance of candidate pronunciation to misspelling pronunciation



Improvements to channel model

- Allow richer edits (Brill and Moore 2000)
 - ent → ant
 - ph → f
 - le → al
- Incorporate pronunciation into channel (Toutanova and Moore 2002)



Channel model

- Factors that could influence $p(\text{misspelling} \mid \text{word})$
 - The source letter
 - The target letter
 - Surrounding letters
 - The position in the word
 - Nearby keys on the keyboard
 - Homology on the keyboard
 - Pronunciations
 - Likely morpheme transformations

Dan Jurafsky



Nearby keys



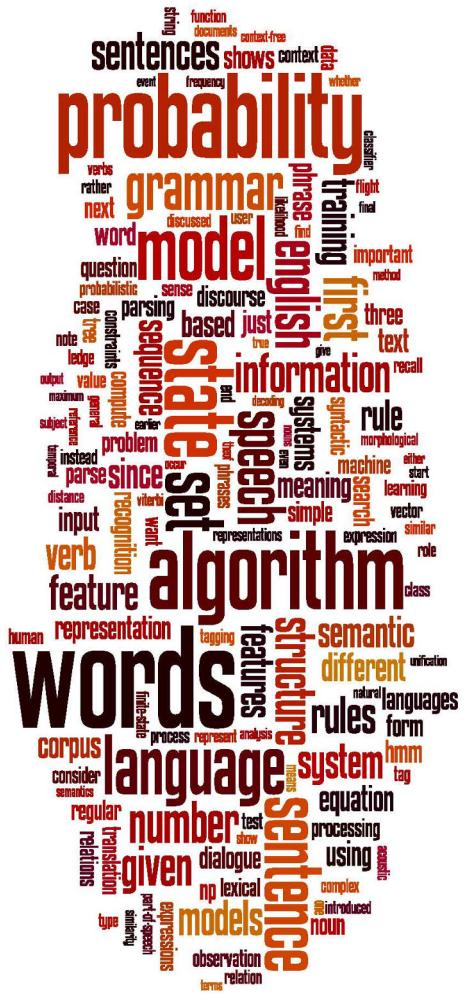


Classifier-based methods for real-word spelling correction

- Instead of just channel model and language model
- Use many features in a classifier (next lecture).
- Build a classifier for a specific pair like:

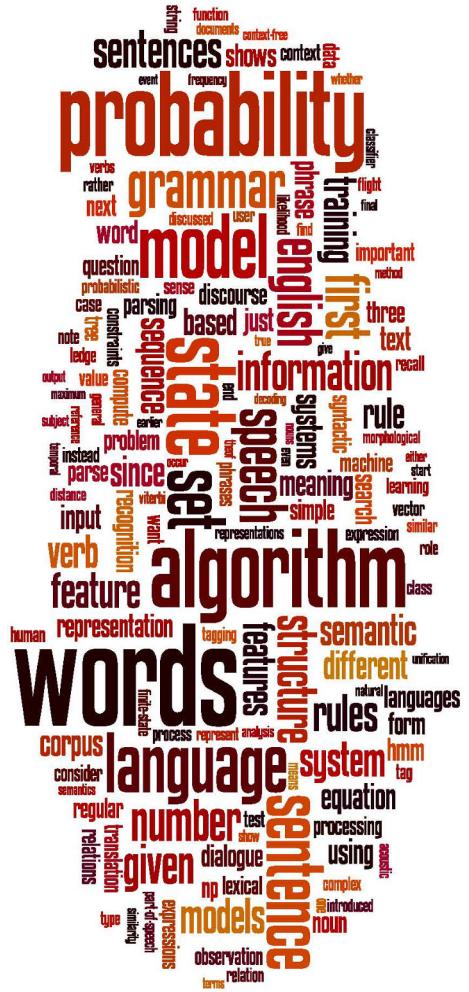
whether/weather

- “cloudy” within +- 10 words
- ____ to VERB
- ____ or not



Spelling Correction and the Noisy Channel

Real-Word Spelling
Correction



Text Classification and Naïve Bayes

The Task of Text Classification

Dan Jurafsky



Is this spam?

Subject: Important notice!

From: Stanford University <newsforum@stanford.edu>

Date: October 28, 2011 12:34:16 PM PDT

To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

<http://www.123contactform.com/contact-form-StanfordNew1-236335.html>

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

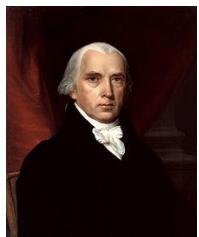
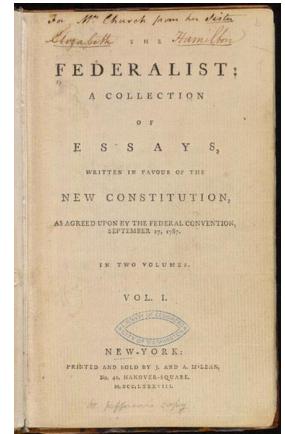
© Stanford University. All Rights Reserved.

Dan Jurafsky

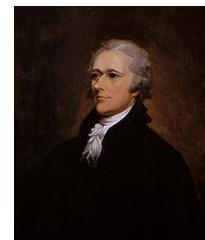


Who wrote which Federalist papers?

- 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using Bayesian methods



James Madison



Alexander Hamilton



Male or female author?

1. By 1925 present-day Vietnam was divided into three parts under French colonial rule. The southern region embracing Saigon and the Mekong delta was the colony of Cochinchina; the central area with its imperial capital at Hue was the protectorate of Annam...
2. Clara never failed to be astonished by the extraordinary felicity of her own name. She found it hard to trust herself to the mercy of fate, which had managed over the years to convert her greatest shame into one of her greatest assets...

S. Argamon, M. Koppel, J. Fine, A. R. Shimoni, 2003. "Gender, Genre, and Writing Style in Formal Written Texts," *Text*, volume 23, number 3, pp. 321–346



Positive or negative movie review?



- unbelievably disappointing



- Full of zany characters and richly applied satire, and some great plot twists



- this is the greatest screwball comedy ever filmed

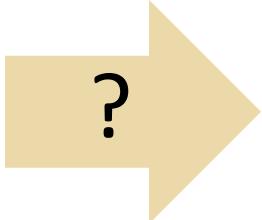


- It was pathetic. The worst part about it was the boxing scenes.



What is the subject of this article?

MEDLINE Article



MeSH Subject Category Hierarchy

- Antagonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- ...



Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- ...



Text Classification: definition

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- *Output:* a predicted class $c \in C$



Classification Methods: Hand-coded rules

- Rules based on combinations of words or other features
 - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high
 - If rules carefully refined by expert
- But building and maintaining these rules is expensive



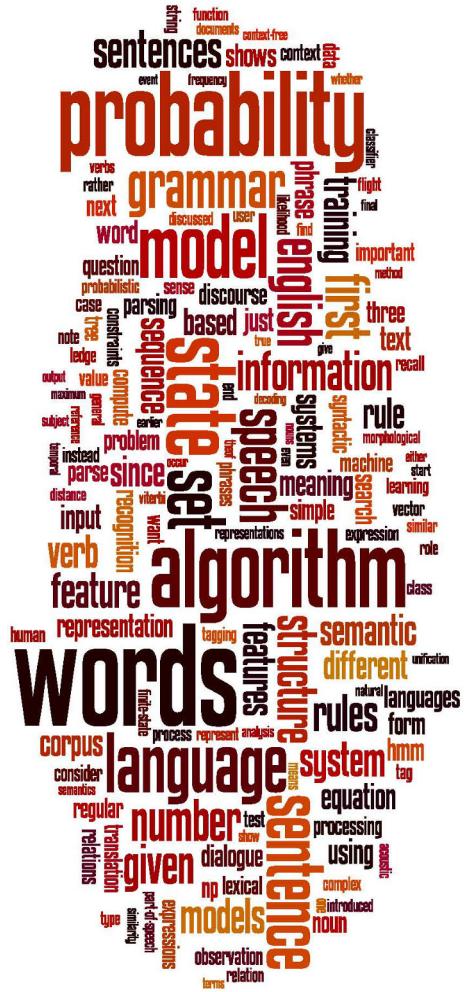
Classification Methods: Supervised Machine Learning

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
 - A training set of m hand-labeled documents $(d_1, c_1), \dots, (d_m, c_m)$
- *Output:*
 - a learned classifier $\gamma: d \rightarrow c$



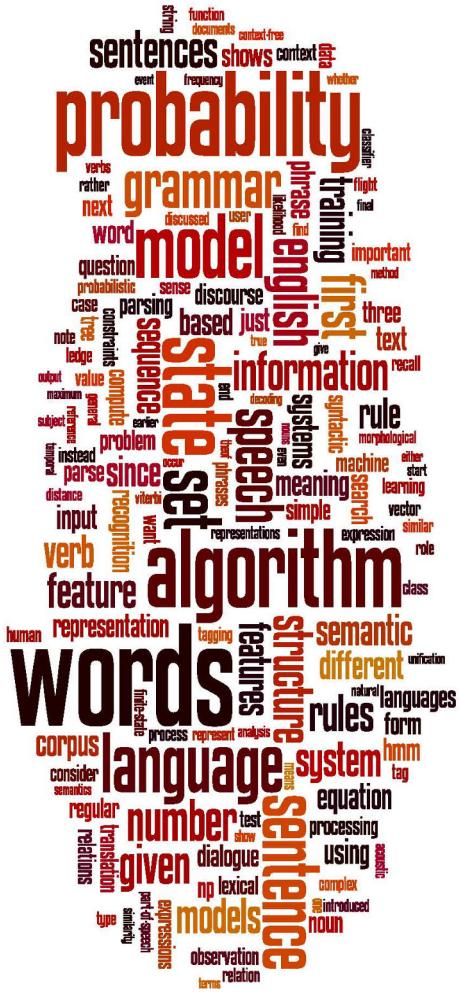
Classification Methods: Supervised Machine Learning

- Any kind of classifier
 - Naïve Bayes
 - Logistic regression
 - Support-vector machines
 - k-Nearest Neighbors
 - ...



Text Classification and Naïve Bayes

The Task of Text Classification



Text Classification and Naïve Bayes

Naïve Bayes (I)



Naïve Bayes Intuition

- Simple (“naïve”) classification method based on Bayes rule
- Relies on very simple representation of document
 - Bag of words



The bag of words representation

Y(

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

)=C





The bag of words representation

Y(

I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun**... It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.

)=C





The bag of words representation: using a subset of words

Y(

```
x love xxxxxxxxxxxxxxxx sweet
xxxxxxxx satirical xxxxxxxxxxx
xxxxxxxxxx great xxxxxxx
xxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxx recommend xxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
```

)=C





The bag of words representation

$\gamma(\quad) = C$

great	2
love	2
recommend	1
laugh	1
happy	1
...	...





Bag of words for document classification

Test
document

parser
language
label
translation
...

?

Machine
Learning

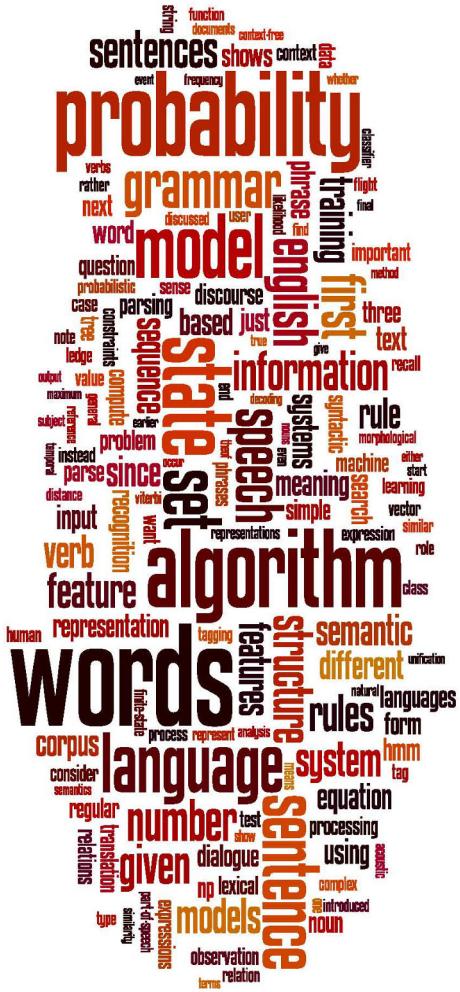
NLP

Garbage
Collection

Planning

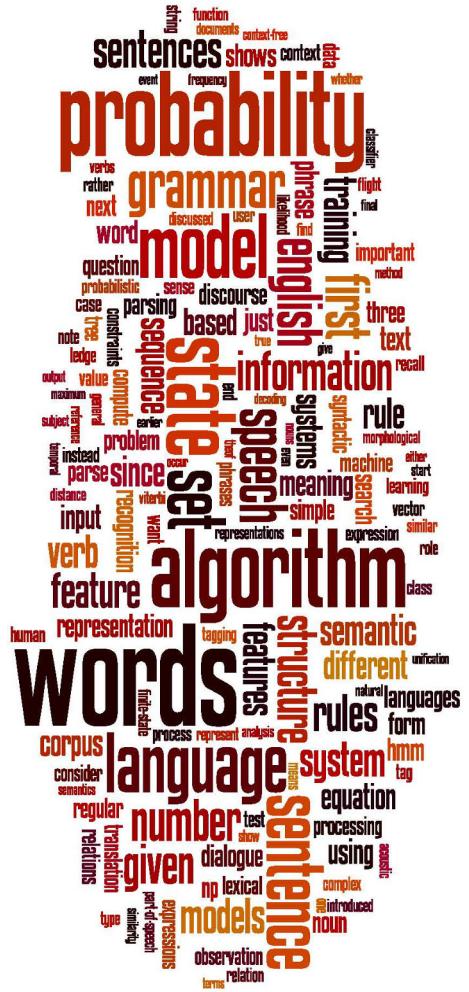
GUI

learning	<u>parser</u>	garbage	planning	...
<u>training</u>	tag	collection	temporal	
algorithm	training	memory	reasoning	
shrinkage	<u>translation</u>	optimization	plan	
network...	<u>language...</u>	region...	<u>language...</u>	



Text Classification and Naïve Bayes

Naïve Bayes (I)



Text Classification and Naïve Bayes

Formalizing the Naïve Bayes Classifier



Bayes' Rule Applied to Documents and Classes

- For a document d and a class c

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$



Naïve Bayes Classifier (I)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(c | d)$$

MAP is “maximum a posteriori” = most likely class

$$= \operatorname{argmax}_{c \in C} \frac{P(d | c)P(c)}{P(d)}$$

Bayes Rule

$$= \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

Dropping the denominator



Naïve Bayes Classifier (II)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

$$= \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n | c)P(c)$$

Document d
represented as
features
x_{1..n}



Naïve Bayes Classifier (IV)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n | c)P(c)$$

$O(|X|^n \cdot |C|)$ parameters

Could only be estimated if a very, very large number of training examples was available.

How often does this class occur?

We can just count the relative frequencies in a corpus



Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \dots, x_n | c)$$

- **Bag of Words assumption:** Assume position doesn't matter
- **Conditional Independence:** Assume the feature probabilities $P(x_i | c_j)$ are independent given the class c .

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \bullet P(x_2 | c) \bullet P(x_3 | c) \bullet \dots \bullet P(x_n | c)$$



Multinomial Naïve Bayes Classifier

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n \mid c)P(c)$$

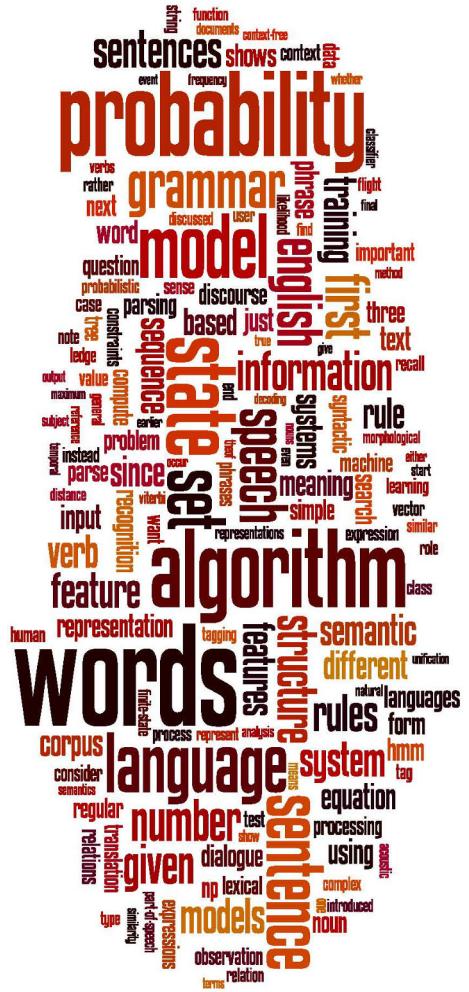
$$c_{NB} = \operatorname{argmax}_{c \in C} P(c_j) \prod_{x \in X} P(x \mid c)$$



Applying Multinomial Naive Bayes Classifiers to Text Classification

positions \leftarrow all word positions in test document

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$



Text Classification and Naïve Bayes

Formalizing the Naïve Bayes Classifier



Text Classification and Naïve Bayes

Naïve Bayes: Learning



Learning the Multinomial Naïve Bayes Model

- First attempt: maximum likelihood estimates
 - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i | c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$



Parameter estimation

$$\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

fraction of times word w_i appears
among all words in documents of topic c_j

- Create mega-document for topic j by concatenating all docs in this topic
 - Use frequency of w in mega-document



Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive (thumbs-up)**?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{\text{count}(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$



Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i | c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)}$$
$$= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V|}$$



Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate $P(c_j)$ terms
 - For each c_j in C do
 $docs_j \leftarrow$ all docs with class = c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$
 - Calculate $P(w_k | c_j)$ terms
 - $Text_j \leftarrow$ single doc containing all $docs_j$
 - For each word w_k in *Vocabulary*
 $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

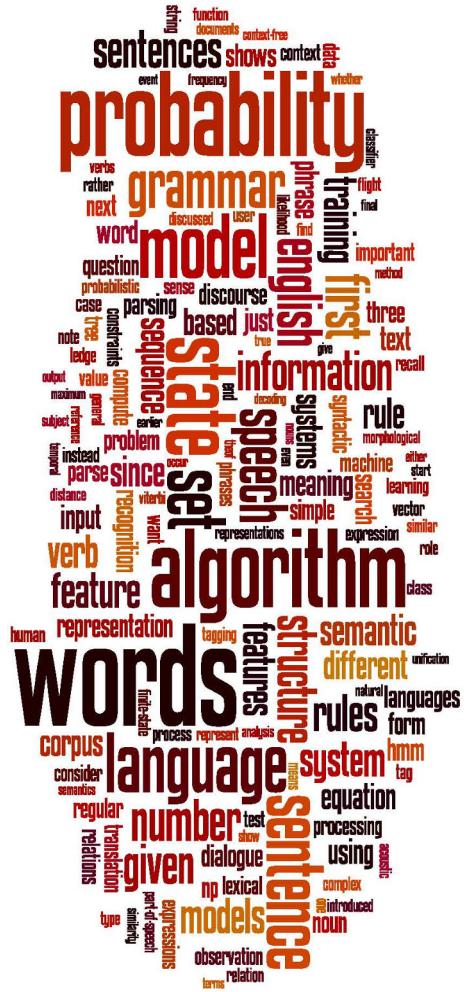
$$P(w_k | c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |\text{Vocabulary}|}$$



Laplace (add-1) smoothing: unknown words

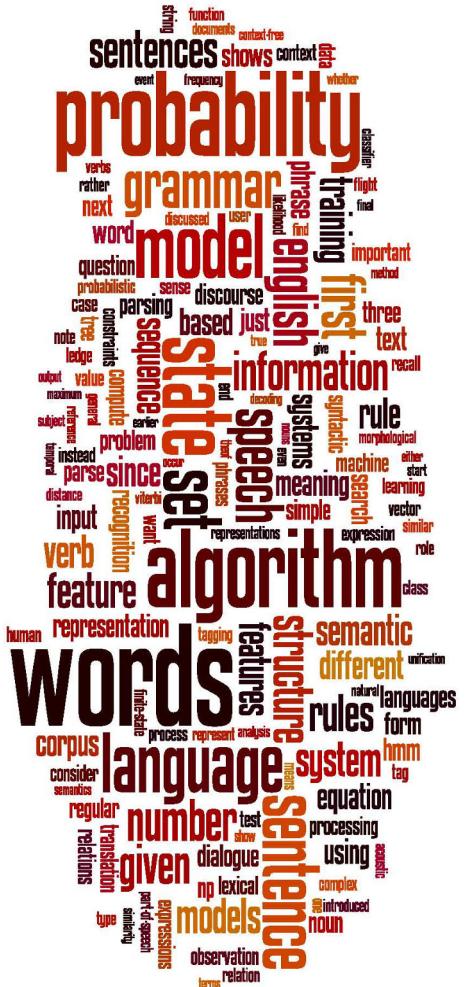
Add one extra word to the vocabulary, the “unknown word” w_u

$$\begin{aligned}\hat{P}(w_u | c) &= \frac{\text{count}(w_u, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V + 1|} \\ &= \frac{1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V + 1|}\end{aligned}$$



Text Classification and Naïve Bayes

Naïve Bayes: Learning

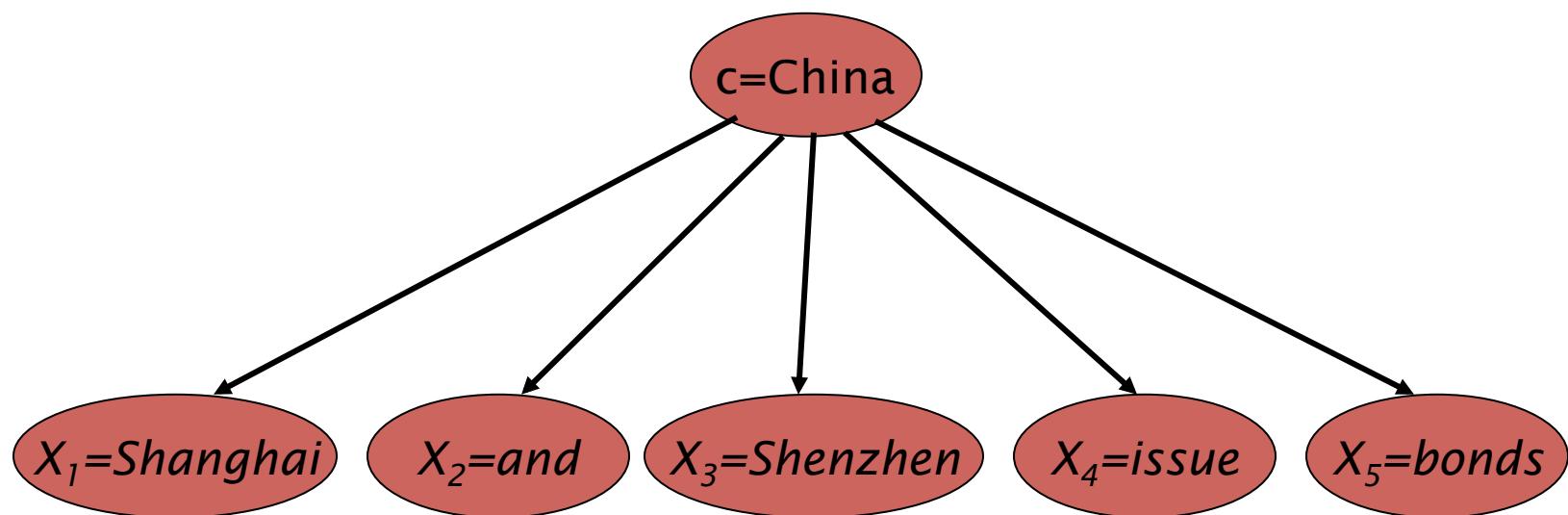


Text Classification and Naïve Bayes

Naïve Bayes:
Relationship to
Language Modeling



Generative Model for Multinomial Naïve Bayes





Naïve Bayes and Language Modeling

- Naïve bayes classifiers can use any sort of feature
 - URL, email address, dictionaries, network features
- But if, as in the previous slides
 - We use **only** word features
 - we use **all** of the words in the text (not a subset)
- Then
 - Naïve bayes has an important similarity to language modeling.



Each class = a unigram language model

- Assigning each word: $P(\text{word} \mid c)$
- Assigning each sentence: $P(s \mid c) = \prod P(\text{word} \mid c)$

Class *pos*

0.1	I		I	<u>love</u>	<u>this</u>	<u>fun</u>	<u>film</u>
0.1	love		0.1	0.1	.05	0.01	0.1
0.01	this						
0.05	fun						
0.1	film						

$$P(s \mid \text{pos}) = 0.0000005$$

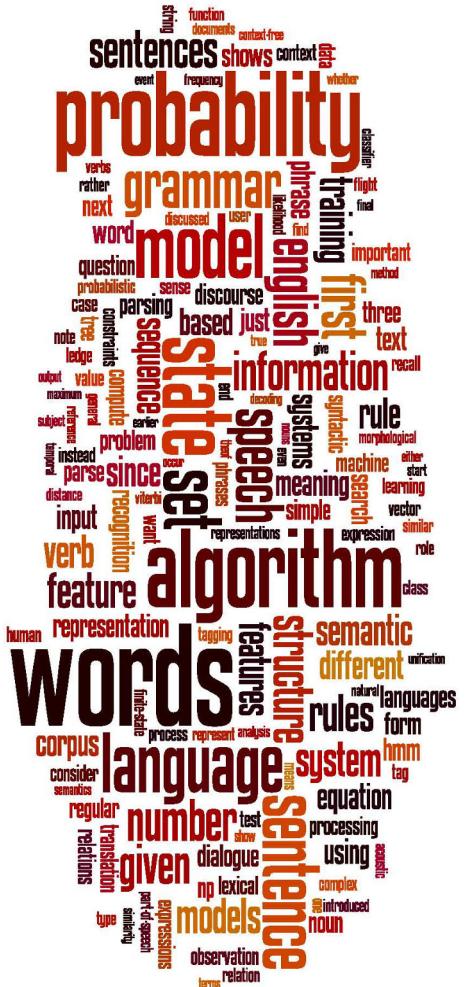


Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

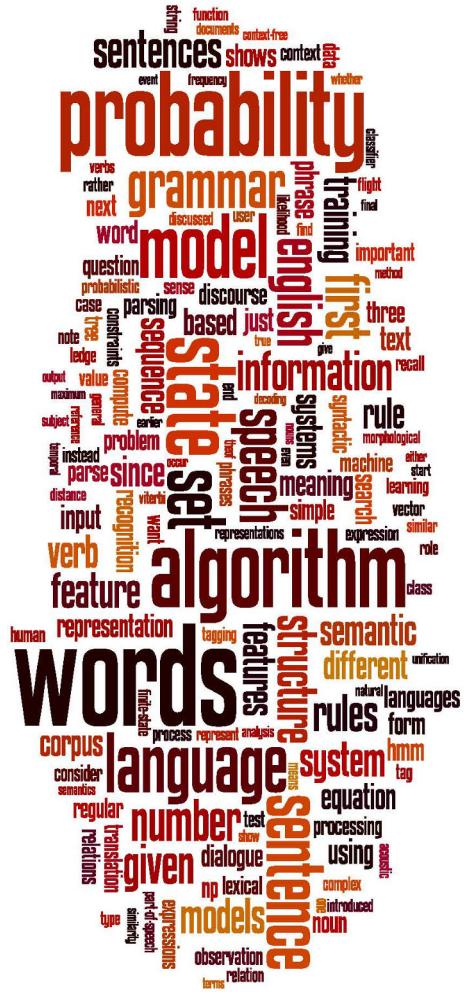
Model pos		Model neg						
0.1	I	0.2	I	I				
0.1	love	0.001	love	love				
0.01	this	0.01	this	this				
0.05	fun	0.005	fun	fun				
0.1	film	0.1	film	film				

$P(s|pos) > P(s|neg)$



Text Classification and Naïve Bayes

Naïve Bayes:
Relationship to
Language Modeling



Text Classification and Naïve Bayes

Multinomial Naïve Bayes: A Worked Example



$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w|c) = \frac{\text{count}(w,c)+1}{\text{count}(c)+|V|}$$

Priors:

$$P(c) = \frac{3}{4}$$

$$P(j) = \frac{1}{4}$$

Conditional Probabilities:

$$P(\text{Chinese}|c) = (5+1) / (8+6) = 6/14 = 3/7$$

$$P(\text{Tokyo}|c) = (0+1) / (8+6) = 1/14$$

$$P(\text{Japan}|c) = (0+1) / (8+6) = 1/14$$

$$P(\text{Chinese}|j) = (1+1) / (3+6) = 2/9$$

$$P(\text{Tokyo}|j) = (1+1) / (3+6) = 2/9$$

$$45 \quad P(\text{Japan}|j) = (1+1) / (3+6) = 2/9$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Choosing a class:

$$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \\ \approx 0.0003$$

$$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \\ \approx 0.0001$$



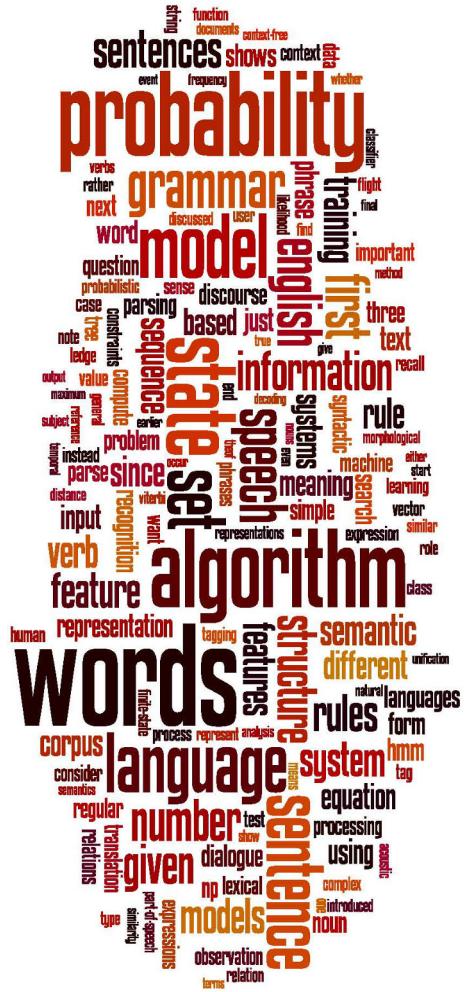
Naïve Bayes in Spam Filtering

- SpamAssassin Features:
 - Mentions Generic Viagra
 - Online Pharmacy
 - Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
 - Phrase: impress ... girl
 - From: starts with many numbers
 - Subject is all capitals
 - HTML has a low ratio of text to image area
 - One hundred percent guaranteed
 - Claims you can be removed from the list
 - 'Prestigious Non-Accredited Universities'
 - http://spamassassin.apache.org/tests_3_3_x.html



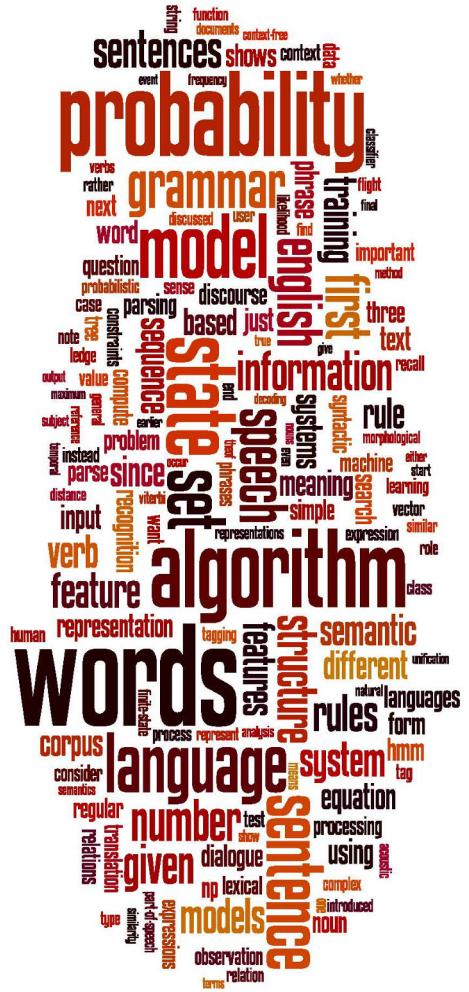
Summary: Naive Bayes is Not So Naive

- Very Fast, low storage requirements
- Robust to Irrelevant Features
 - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
 - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification
 - **But we will see other classifiers that give better accuracy**



Text Classification and Naïve Bayes

Multinomial Naïve Bayes: A Worked Example



Text Classification and Naïve Bayes

Precision, Recall, and the F measure



The 2-by-2 contingency table

	correct	not correct
selected	tp	fp
not selected	fn	tn



Precision and recall

- **Precision:** % of selected items that are correct
Recall: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn



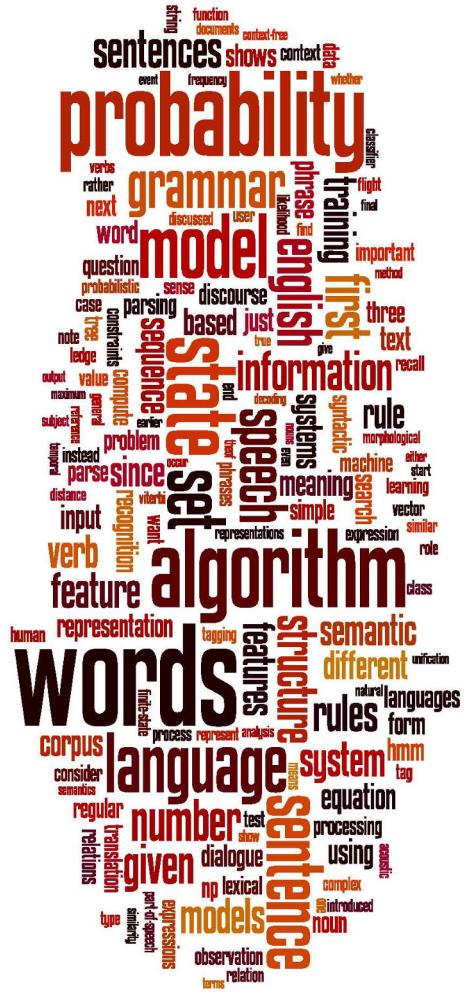
A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

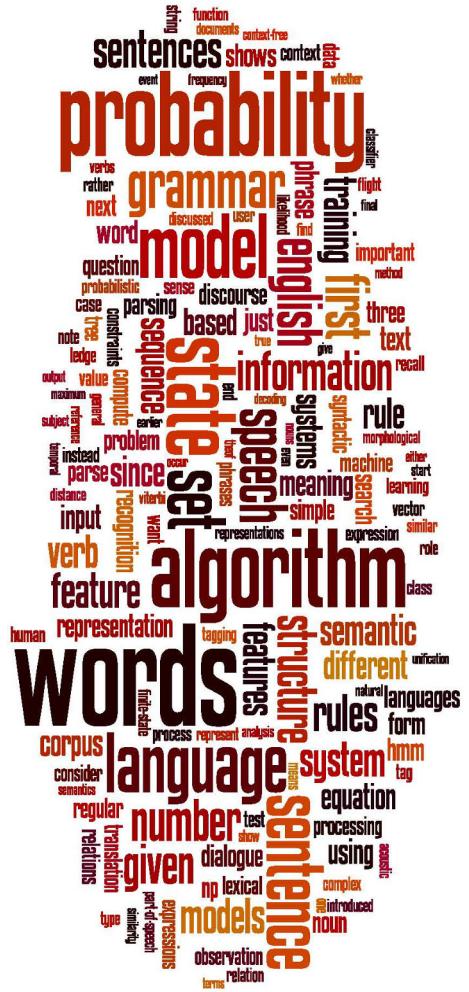
- The harmonic mean is a very conservative average; see *IIR* § 8.3
- People usually use balanced F1 measure
 - i.e., with $\beta = 1$ (that is, $\alpha = \frac{1}{2}$):

$$F = 2PR/(P+R)$$



Text Classification and Naïve Bayes

Precision, Recall, and the F measure



Text Classification and Naïve Bayes

Text Classification: Evaluation



More Than Two Classes: Sets of binary classifiers

- Dealing with **any-of** or **multivalue** classification
 - A document can belong to 0, 1, or >1 classes.
- For each class $c \in C$
 - Build a classifier γ_c to distinguish c from all other classes $c' \in C$
- Given test doc d ,
 - Evaluate it for membership in each class using each γ_c
 - d belongs to **any** class for which γ_c returns true



More Than Two Classes: Sets of binary classifiers

- One-of or multinomial classification
 - Classes are mutually exclusive: each document in exactly one class
- For each class $c \in C$
 - Build a classifier y_c to distinguish c from all other classes $c' \in C$
- Given test doc d ,
 - Evaluate it for membership in each class using each y_c
 - d belongs to the one class with maximum score



Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set, 21,578 docs (each 90 types, 200 tokens)
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
 - An article can be in more than one category
 - Learn 118 binary category distinctions
- Average document (with at least one category) has 1.24 classes
- Only about 10 out of 118 categories are large
 - Earn (2877, 1087)
 - Acquisitions (1650, 179)
 - Money-fx (538, 179)
 - Grain (433, 149)
 - Crude (389, 189)
 - Trade (369, 119)
 - Interest (347, 131)
 - Ship (197, 89)
 - Wheat (212, 71)
 - Corn (182, 56)

Common categories
(#train, #test)

57



Reuters Text Categorization data set (Reuters-21578) document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OL DID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

58
</BODY></TEXT></REUTERS>



Confusion matrix c

- For each pair of classes $\langle c_1, c_2 \rangle$ how many documents from c_1 were incorrectly assigned to c_2 ?
 - $c_{3,2}$: 90 wheat documents incorrectly assigned to poultry

Docs in test set	Assigned UK	Assigned poultry	Assigned wheat	Assigned coffee	Assigned interest	Assigned trade
True UK	95	1	13	0	1	0
True poultry	0	1	0	0	0	0
True wheat	10	90	0	1	0	0
True coffee	0	0	0	34	3	7
True interest	-	1	2	13	26	5
True trade	0	0	2	14	5	10



Per class evaluation measures

Recall:

Fraction of docs in class i classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

Precision:

Fraction of docs assigned class i that are actually about class i :

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

Accuracy:

(1 - error rate)

Fraction of docs classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$



Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- **Macroaveraging:** Compute performance for each class, then average.
- **Microaveraging:** Collect decisions for all classes, compute contingency table, evaluate.



Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: $(0.5 + 0.9)/2 = 0.7$
- Microaveraged precision: $100/120 = .83$
- Microaveraged score is dominated by score on common classes



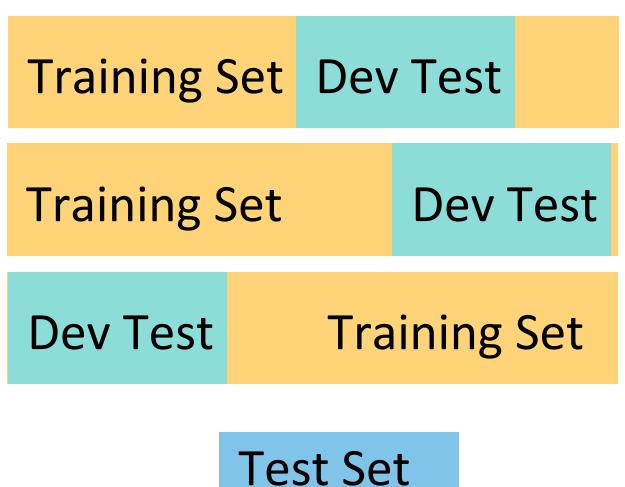
Development Test Sets and Cross-validation

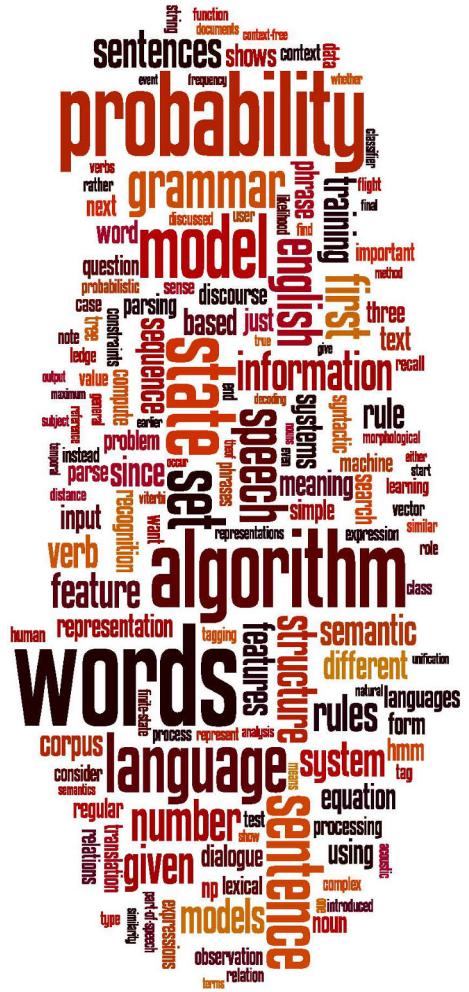
Training set

Development Test Set

Test Set

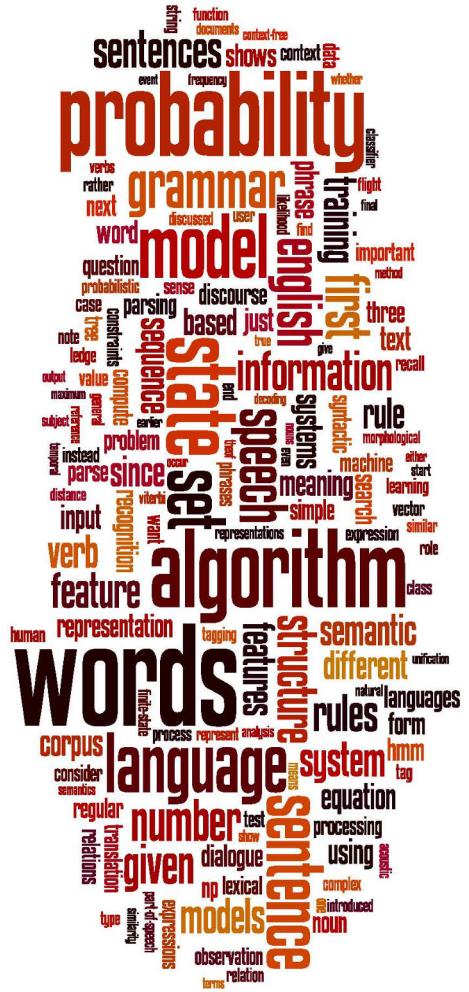
- Metric: P/R/F1 or Accuracy
- Unseen test set
 - avoid overfitting ('tuning to the test set')
 - more conservative estimate of performance
- Cross-validation over multiple splits
 - Handle sampling errors from different datasets
 - Pool results over each split
 - Compute pooled dev set performance





Text Classification and Naïve Bayes

Text Classification: Evaluation



Text Classification and Naïve Bayes

Text Classification: Practical Issues



The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?



No training data? Manually written rules

If (wheat or grain) and not (whole or bread) then
Categorize as grain

- Need careful crafting
 - Human tuning on development data
 - Time-consuming: 2 days per class



Very little data?

- Use Naïve Bayes
 - Naïve Bayes is a “high-bias” algorithm (Ng and Jordan 2002 NIPS)
- Get more labeled data
 - Find clever ways to get humans to label data for you
- Try semi-supervised training methods:
 - Bootstrapping, EM over unlabeled documents, ...



A reasonable amount of data?

- Perfect for all the clever classifiers
 - SVM
 - Regularized Logistic Regression
- You can even use user-interpretable decision trees
 - Users like to hack
 - Management likes quick fixes



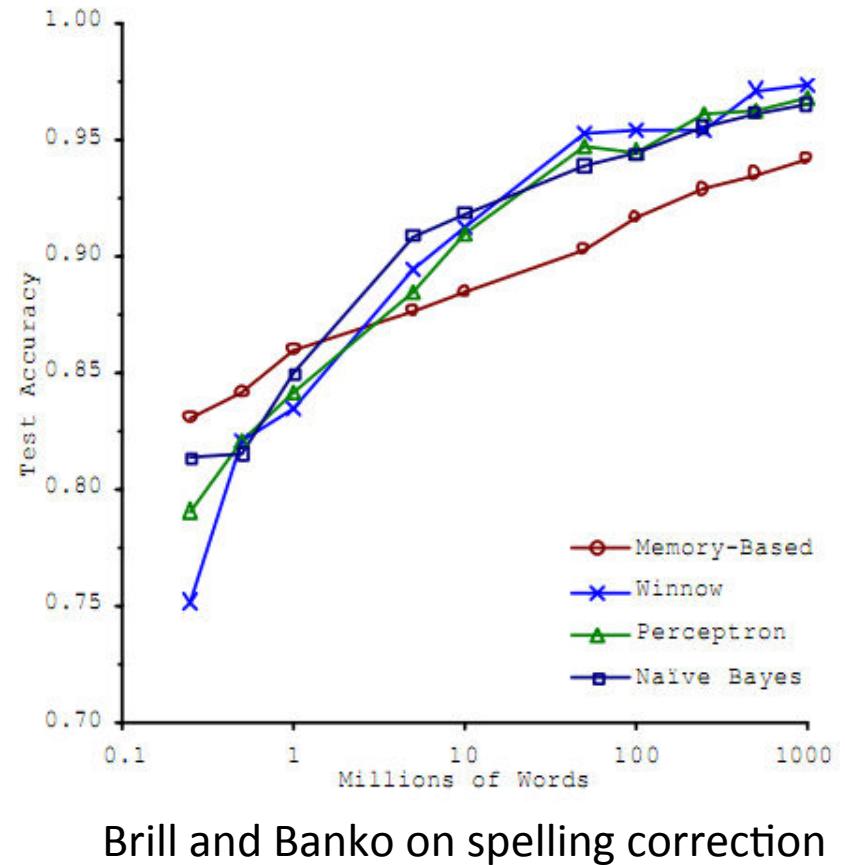
A huge amount of data?

- Can achieve high accuracy!
- At a cost:
 - SVMs (train time) or kNN (test time) can be too slow
 - Regularized logistic regression can be somewhat better
- So Naïve Bayes can come back into its own again!



Accuracy as a function of data size

- With enough data
 - Classifier may not matter





Real-world systems generally combine:

- Automatic classification
- Manual review of uncertain/difficult/"new" cases



Underflow Prevention: log space

- Multiplying lots of probabilities can result in floating-point underflow.
- Since $\log(xy) = \log(x) + \log(y)$
 - Better to sum logs of probabilities instead of multiplying probabilities.
- Class with highest un-normalized log probability score is still most probable.

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \log P(c_j) + \sum_{i \in positions} \log P(x_i | c_j)$$

- Model is now just max of sum of weights



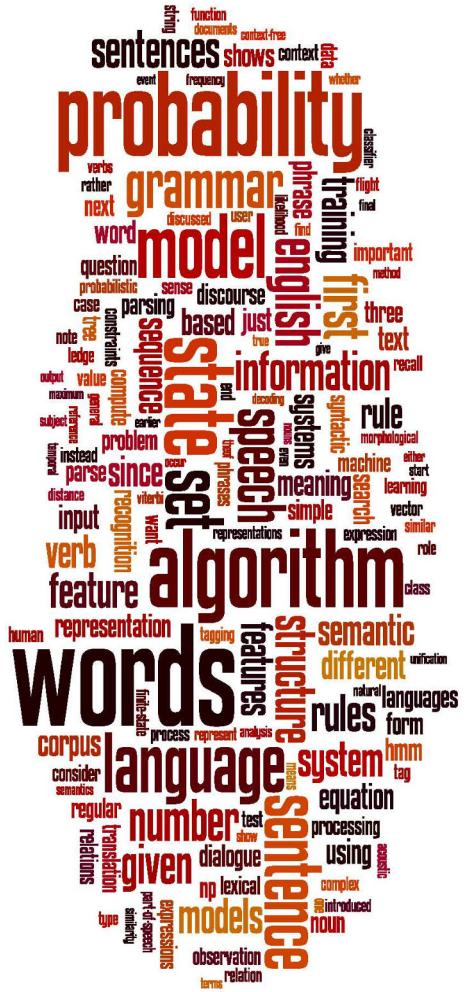
How to tweak performance

- Domain-specific features and weights: *very* important in real performance
- Sometimes need to collapse terms:
 - Part numbers, chemical formulas, ...
 - But stemming generally doesn't help
- Upweighting: Counting a word as if it occurred twice:
 - title words ([Cohen & Singer 1996](#))
 - first sentence of each paragraph ([Murata, 1999](#))
 - In sentences that contain title words ([Ko et al, 2002](#))



Text Classification and Naïve Bayes

Text Classification: Practical Issues



Sentiment Analysis

What is Sentiment Analysis?



Positive or negative movie review?



- unbelievably disappointing



- Full of zany characters and richly applied satire, and some great plot twists



- this is the greatest screwball comedy ever filmed



- It was pathetic. The worst part about it was the boxing scenes.



Google Product Search



HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner

\$89 online, \$100 nearby ★★★★☆ 377 reviews

September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

Reviews

Summary - Based on 377 reviews

1 star 2 3 4 stars 5 stars

What people are saying

ease of use	<div style="width: 80%;"></div>	"This was very easy to setup to four computers."
value	<div style="width: 80%;"></div>	"Appreciate good quality at a fair price."
setup	<div style="width: 80%;"></div>	"Overall pretty easy setup."
customer service	<div style="width: 20%;"></div>	"I DO like honest tech support people."
size	<div style="width: 40%;"></div>	"Pretty Paper weight."
mode	<div style="width: 40%;"></div>	"Photos were fair on the high quality mode."
colors	<div style="width: 80%;"></div>	"Full color prints came out with great quality."



Bing Shopping

HP Officejet 6500A E710N Multifunction Printer

[Product summary](#) [Find best price](#) **Customer reviews** [Specifications](#) [Related items](#)



\$121.53 - \$242.39 (14 stores)

Compare

Average rating (144)

(55)

(54)

(10)

(6)

(23)

(0)

Most mentioned

Performance

(57)

Ease of Use

(43)

Print Speed

(39)

Connectivity

(31)

More ▾

Show reviews by source

Best Buy (140)

CNET (5)

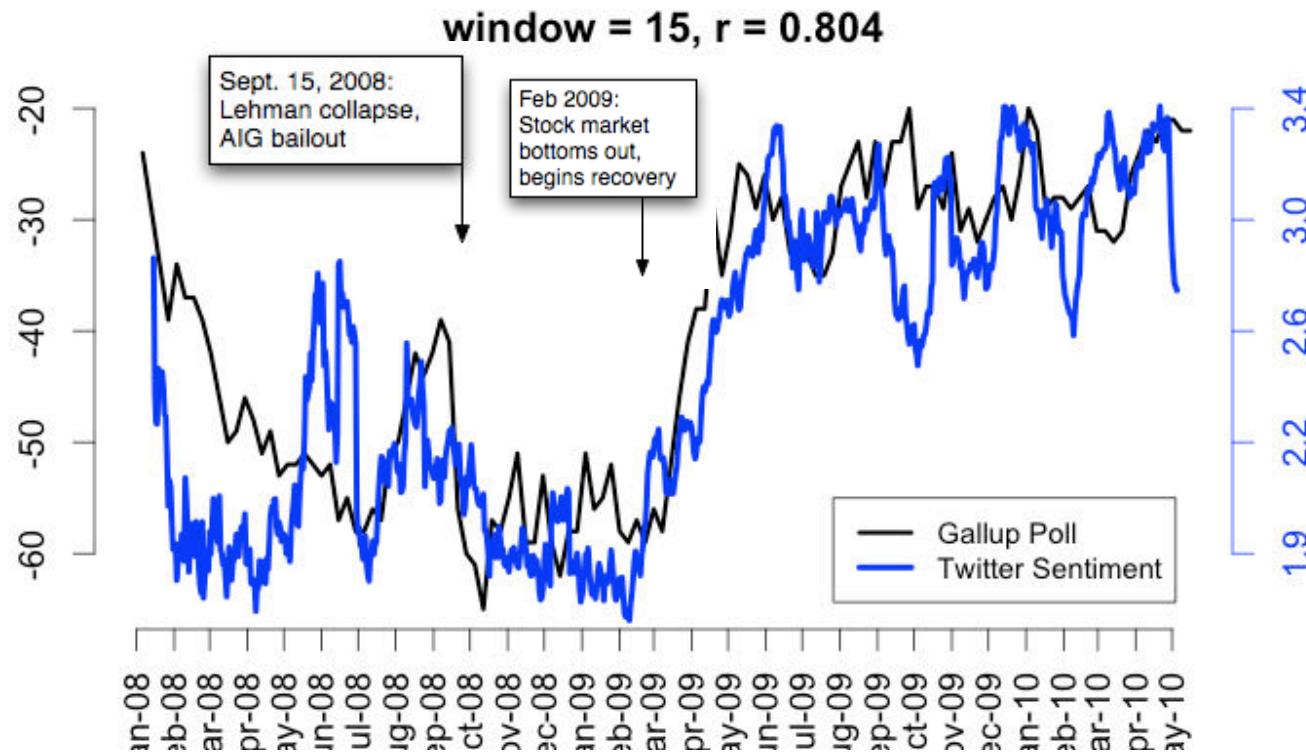
Amazon.com (3)

Dan Jurafsky



Twitter sentiment versus Gallup Poll of Consumer Confidence

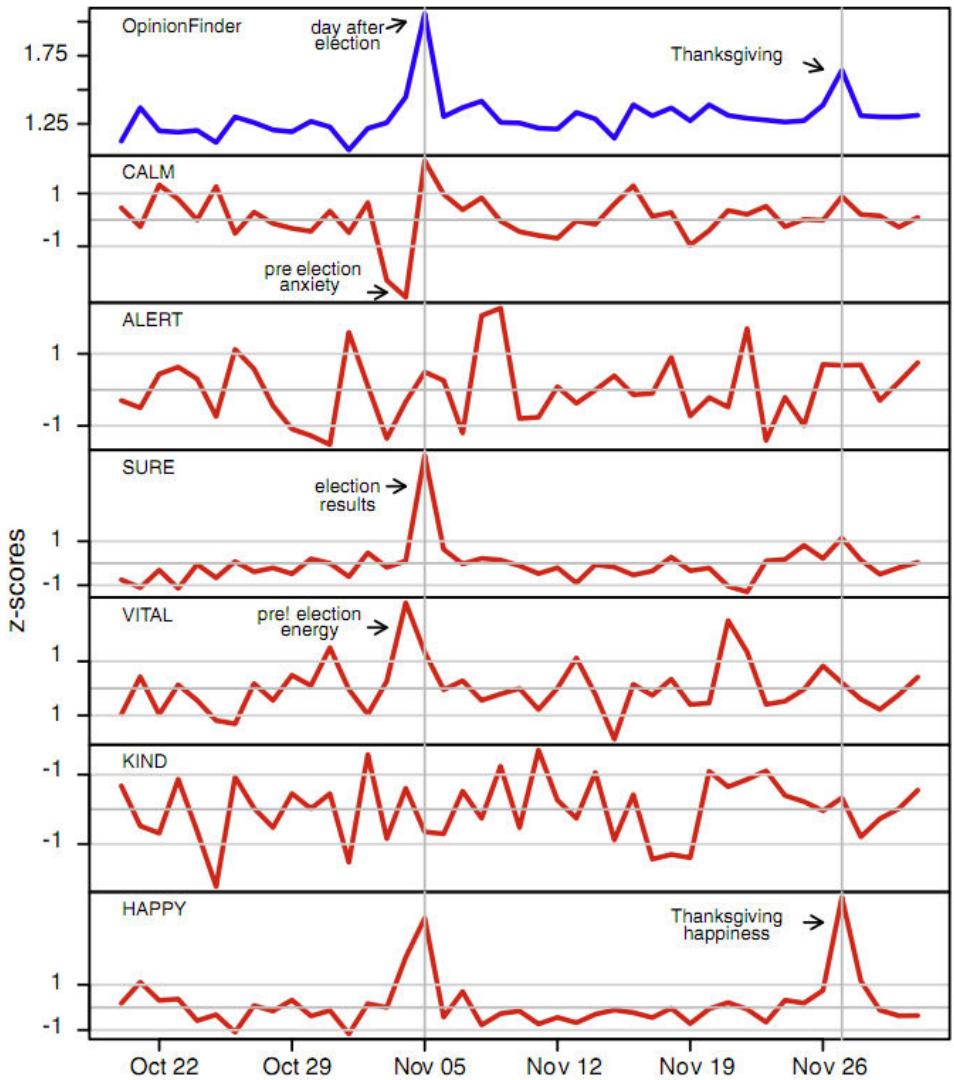
Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010.
From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In ICWSM-2010





Twitter sentiment:

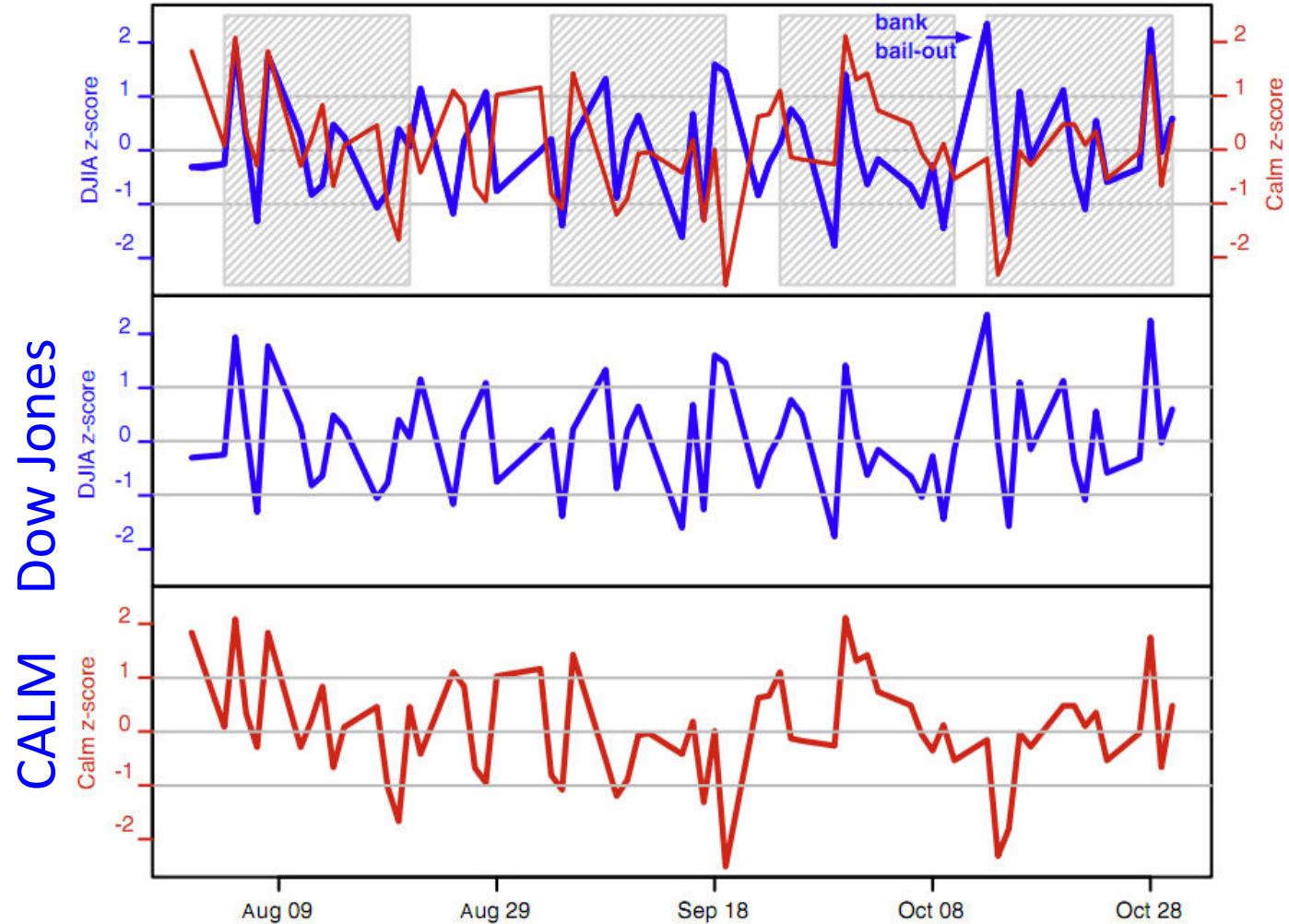
Johan Bollen, Huina Mao, Xiaojun Zeng. 2011.
Twitter mood predicts the stock market,
Journal of Computational Science 2:1, 1-8.
10.1016/j.jocs.2010.12.007.





Bollen et al. (2011)

- CALM predicts DJIA 3 days later
- At least one current hedge fund uses this algorithm





Target Sentiment on Twitter

Type in a word and we'll highlight the good and the bad

- Twitter Sentiment App

- Alec Go, Richa Bhayani, Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision

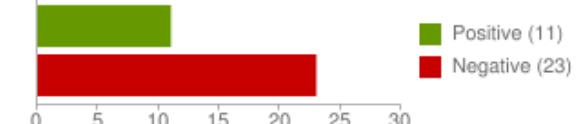
"united airlines" [Save this search](#)

Sentiment analysis for "united airlines"

Sentiment by Percent



Sentiment by Count



jljacobson: OMG... Could @United airlines have worse customer service? W8g now 15 minutes
Posted 2 hours ago

12345clumsy6789: I hate United Airlines Ceiling!!! Fukn impossible to get my conduit in this d
Posted 2 hours ago

EMLandPRGbelgiu: EML/PRG fly with Q8 united airlines and 24seven to an exotic destination
Posted 2 hours ago

CountAdam: FANTASTIC customer service from United Airlines at XNA today. Is tweet more.
Posted 4 hours ago



Sentiment analysis has many other names

- Opinion extraction
- Opinion mining
- Sentiment mining
- Subjectivity analysis



Why sentiment analysis?

- *Movie*: is this review positive or negative?
- *Products*: what do people think about the new iPhone?
- *Public sentiment*: how is consumer confidence? Is despair increasing?
- *Politics*: what do people think about this candidate or issue?
- *Prediction*: predict election outcomes or market trends from sentiment



Scherer Typology of Affective States

- **Emotion:** brief organically synchronized ... evaluation of a major event
 - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood:** diffuse non-caused low-intensity long-duration change in subjective feeling
 - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances:** affective stance toward another person in a specific interaction
 - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes:** enduring, affectively colored beliefs, dispositions towards objects or persons
 - *liking, loving, hating, valuing, desiring*
- **Personality traits:** stable personality dispositions and typical behavior tendencies
 - *nervous, anxious, reckless, morose, hostile, jealous*



Scherer Typology of Affective States

- **Emotion:** brief organically synchronized ... evaluation of a major event
 - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood:** diffuse non-caused low-intensity long-duration change in subjective feeling
 - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances:** affective stance toward another person in a specific interaction
 - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes:** enduring, affectively colored beliefs, dispositions towards objects or persons
 - *liking, loving, hating, valuing, desiring*
- **Personality traits:** stable personality dispositions and typical behavior tendencies
 - *nervous, anxious, reckless, morose, hostile, jealous*



Sentiment Analysis

- Sentiment analysis is the detection of **attitudes**
“enduring, affectively colored beliefs, dispositions towards objects or persons”
 1. **Holder (source)** of attitude
 2. **Target (aspect)** of attitude
 3. **Type** of attitude
 - From a set of types
 - *Like, love, hate, value, desire, etc.*
 - Or (more commonly) simple weighted **polarity**:
 - *positive, negative, neutral*, together with *strength*
 4. **Text** containing the attitude
 - Sentence or entire document



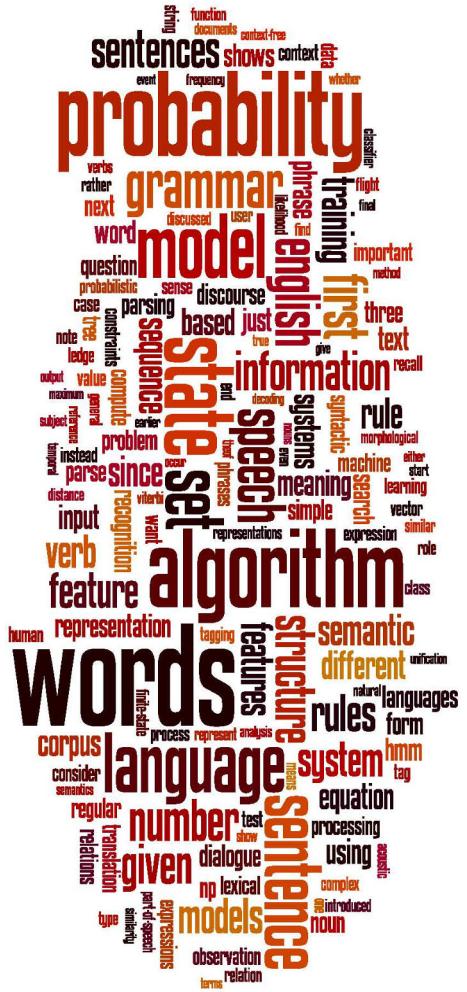
Sentiment Analysis

- Simplest task:
 - Is the attitude of this text positive or negative?
- More complex:
 - Rank the attitude of this text from 1 to 5
- Advanced:
 - Detect the target, source, or complex attitude types



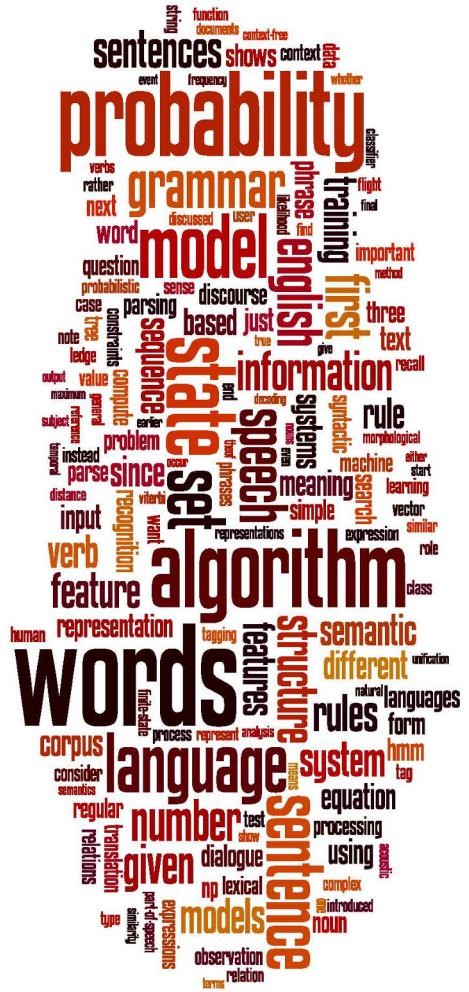
Sentiment Analysis

- Simplest task:
 - Is the attitude of this text positive or negative?
- More complex:
 - Rank the attitude of this text from 1 to 5
- Advanced:
 - Detect the target, source, or complex attitude types



Sentiment Analysis

What is Sentiment Analysis?



Sentiment Analysis

A Baseline Algorithm



Sentiment Classification in Movie Reviews

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79–86.

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. ACL, 271-278

- Polarity detection:
 - Is an IMDB movie review positive or negative?
- Data: *Polarity Data 2.0:*
 - <http://www.cs.cornell.edu/people/pabo/movie-review-data>



IMDB data in the Pang and Lee database



when _star wars_ came out some twenty years ago , the image of traveling throughout the stars has become a commonplace image . [...]

when han solo goes light speed , the stars change to bright lines , going towards the viewer in lines that converge at an invisible point .
cool .

october sky offers a much simpler image—that of a single white dot , traveling horizontally across the night sky . [...]



“ snake eyes ” is the most aggravating kind of movie : the kind that shows so much potential then becomes unbelievably disappointing .

it’s not just because this is a brian depalma film , and since he’s a great director and one who’s films are always greeted with at least some fanfare .
and it’s not even because this was a film starring nicolas cage and since he gives a brauvara performance , this film is hardly worth his talents .



Baseline Algorithm (adapted from Pang and Lee)

- Tokenization
- Feature Extraction
- Classification using different classifiers
 - Naïve Bayes
 - MaxEnt
 - SVM



Sentiment Tokenization Issues

- Deal with HTML and XML markup
- Twitter mark-up (names, hash tags)
- Capitalization (preserve for words in all caps)
- Phone numbers, dates
- Emoticons
- Useful code:

- [Christopher Potts sentiment tokenizer](#)

- [Brendan O'Connor twitter tokenizer](#)

```
[<>]?
[::;=8]
[\-o\*\' ]?
[(\\)]\\([dDpP/\:\:]\{@\|\\]
|
[(\\)]\\([dDpP/\:\:]\{@\|\\]
[\-o\*\' ]?
[::;=8]
[<>]? # optional hat/brow
# eyes
# optional nose
# mouth
#### reverse orientation
# mouth
# optional nose
# eyes
# optional hat/brow
```



Extracting Features for Sentiment Classification

- How to handle negation
 - I **didn't** like this movie
 - vs
 - I really like this movie
- Which words to use?
 - Only adjectives
 - All words
 - All words turns out to work better, at least on this data



Negation

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).
Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79–86.

Add NOT_ to every word between negation and following punctuation:

didn't like this movie , but I



didn't NOT_like NOT_this NOT_movie but I



Reminder: Naïve Bayes

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(w_i | c_j)$$

$$\hat{P}(w | c) = \frac{count(w, c) + 1}{count(c) + |V|}$$



Binarized (Boolean feature) Multinomial Naïve Bayes

- Intuition:
 - For sentiment (and probably for other text classification domains)
 - Word occurrence may matter more than word frequency
 - The occurrence of the word *fantastic* tells us a lot
 - The fact that it occurs 5 times may not tell us much more.
 - Boolean Multinomial Naïve Bayes
 - Clips all the word counts in each document at 1



Boolean Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms

 - For each c_j in C do

$docs_j \leftarrow$ all docs with class = c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{\text{total \# documents}}$$

- Calculate $P(w_k | c_j)$ terms

 - Remove single doc containing all $docs_j$

 - For each word type w_k in *Vocabulary*

$n_k \leftarrow$ # of occurrences of w_k in $text_j$

$$P(w_k | c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$



Boolean Multinomial Naïve Bayes on a test document d

- First remove all duplicate words from d
- Then compute NB using the same equation:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(w_i | c_j)$$

Dan Jurafsky



Normal vs. Boolean Multinomial NB

Normal	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Boolean	Doc	Words	Class
Training	1	Chinese Beijing	c
	2	Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Tokyo Japan	?



Binarized (Boolean feature) Multinomial Naïve Bayes

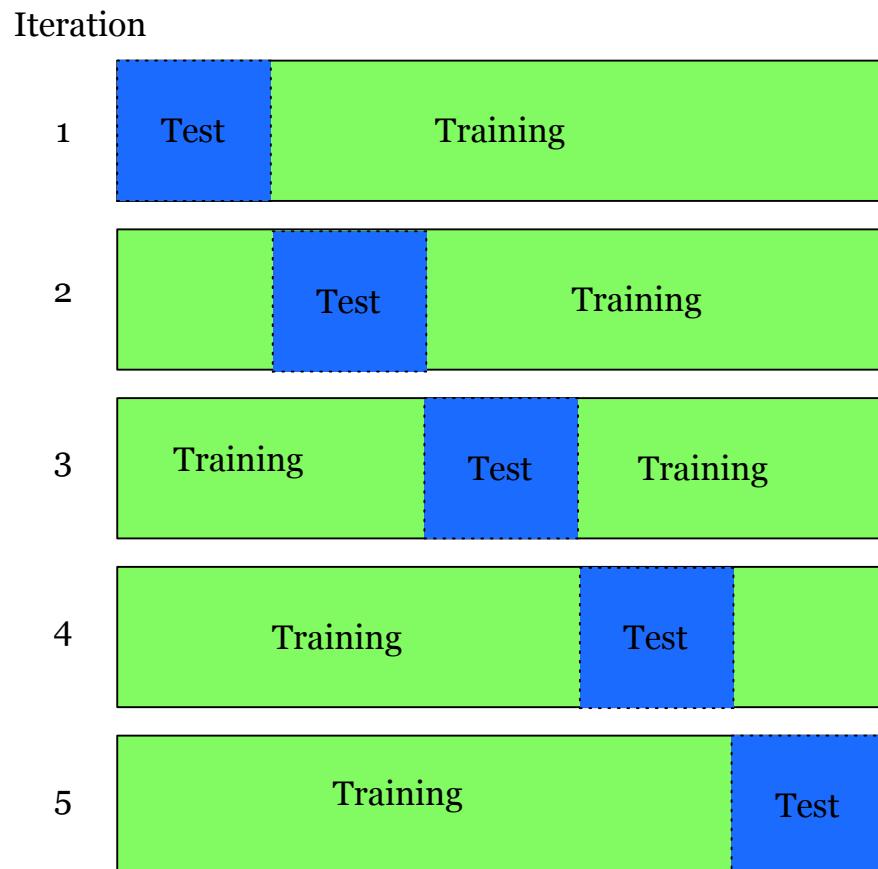
B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79–86.
V. Metsis, I. Androutsopoulos, G. Palouras. 2006. Spam Filtering with Naive Bayes – Which Naive Bayes? CEAS 2006 - Third Conference on Email and Anti-Spam.
K.-M. Schneider. 2004. On word frequency information and negative evidence in Naive Bayes text classification. ICANLP, 474-485.
JD Rennie, L Shih, J Teevan. 2003. Tackling the poor assumptions of naive bayes text classifiers. ICML 2003

- Binary seems to work better than full word counts
 - This is **not** the same as Multivariate Bernoulli Naïve Bayes
 - MBNB doesn't work well for sentiment or other text tasks
 - Other possibility: $\log(\text{freq}(w))$



Cross-Validation

- Break up data into 10 folds
 - (Equal positive and negative inside each fold?)
- For each fold
 - Choose the fold as a temporary test set
 - Train on 9 folds, compute performance on the test fold
- Report average performance of the 10 runs





Other issues in Classification

- MaxEnt and SVM tend to do better than Naïve Bayes



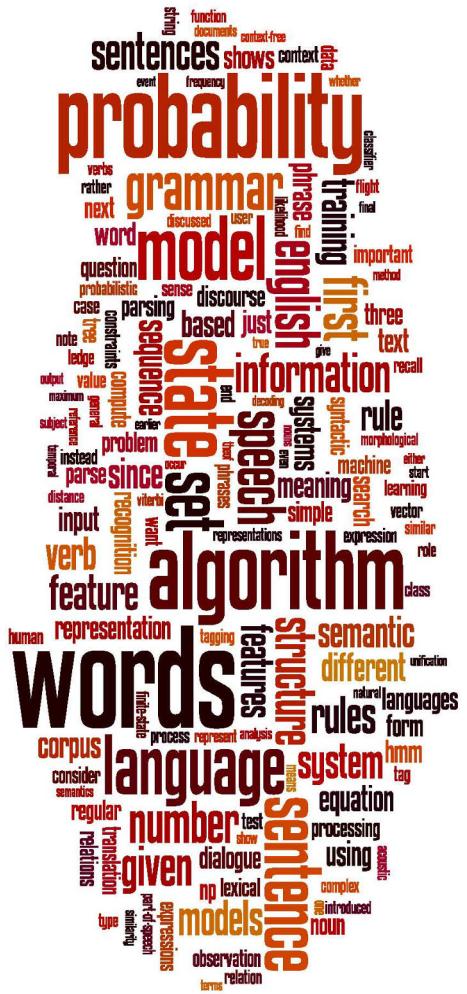
Problems: What makes reviews hard to classify?

- Subtlety:
 - Perfume review in *Perfumes: the Guide*:
 - “If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut.”
 - Dorothy Parker on Katherine Hepburn
 - “She runs the gamut of emotions from A to B”



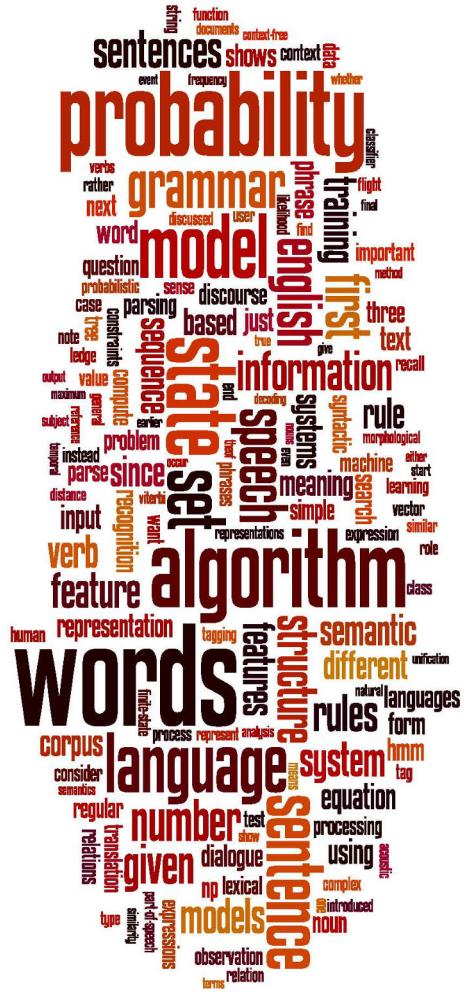
Thwarted Expectations and Ordering Effects

- “This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it **can’t hold up.**”
- Well as usual Keanu Reeves is nothing special, but surprisingly, the **very talented** Laurence Fishbourne is **not so good** either, I was surprised.



Sentiment Analysis

A Baseline
Algorithm



Sentiment Analysis

Sentiment Lexicons



The General Inquirer

Philip J. Stone, Dexter C Dunphy, Marshall S. Smith, Daniel M. Ogilvie. 1966. The General Inquirer: A Computer Approach to Content Analysis. MIT Press

- Home page: <http://www.wjh.harvard.edu/~inquirer>
- List of Categories: <http://www.wjh.harvard.edu/~inquirer/homecat.htm>
- Spreadsheet: <http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls>
- Categories:
 - Positiv (1915 words) and Negativ (2291 words)
 - Strong vs Weak, Active vs Passive, Overstated versus Understated
 - Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc
- Free for Research Use



LIWC (Linguistic Inquiry and Word Count)

Pennebaker, J.W., Booth, R.J., & Francis, M.E. (2007). Linguistic Inquiry and Word Count: LIWC 2007. Austin, TX

- Home page: <http://www.liwc.net/>
- 2300 words, >70 classes
- **Affective Processes**
 - negative emotion (*bad, weird, hate, problem, tough*)
 - positive emotion (*love, nice, sweet*)
- **Cognitive Processes**
 - Tentative (*maybe, perhaps, guess*), Inhibition (*block, constraint*)
- **Pronouns, Negation (*no, never*), Quantifiers (*few, many*)**
- \$30 or \$90 fee



MPQA Subjectivity Cues Lexicon

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proc. of HLT-EMNLP-2005.

Riloff and Wiebe (2003). Learning extraction patterns for subjective expressions. EMNLP-2003.

- Home page: http://www.cs.pitt.edu/mpqa/subj_lexicon.html
- 6885 words from 8221 lemmas
 - 2718 positive
 - 4912 negative
- Each word annotated for intensity (strong, weak)
- GNU GPL



Bing Liu Opinion Lexicon

Minqing Hu and Bing Liu. Mining and Summarizing Customer Reviews. ACM SIGKDD-2004.

- [Bing Liu's Page on Opinion Mining](#)
- <http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar>
- 6786 words
 - 2006 positive
 - 4783 negative



SentiWordNet

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010 SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. LREC-2010

- Home page: <http://sentiwordnet.isti.cnr.it/>
- All WordNet synsets automatically annotated for degrees of positivity, negativity, and neutrality/objectiveness
- [estimable(J,3)] “may be computed or estimated”

Pos 0 Neg 0 Obj 1

- [estimable(J,1)] “deserving of respect or high regard”

Pos .75 Neg 0 Obj .25



Disagreements between polarity lexicons

Christopher Potts, [Sentiment Tutorial](#), 2011

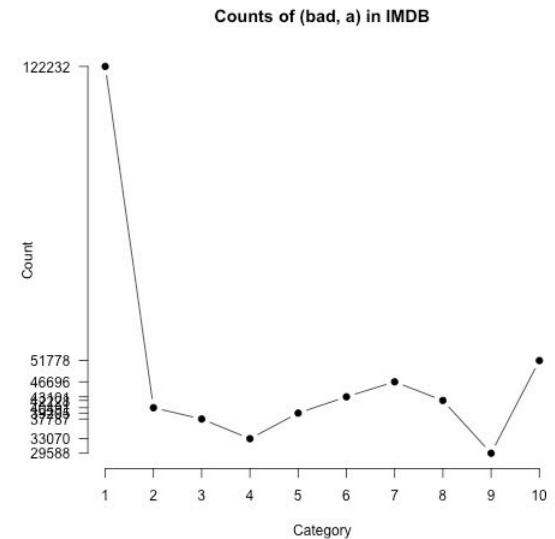
	Opinion Lexicon	General Inquirer	SentiWordNet	LIWC
MPQA	33/5402 (0.6%)	49/2867 (2%)	1127/4214 (27%)	12/363 (3%)
Opinion Lexicon		32/2411 (1%)	1004/3994 (25%)	9/403 (2%)
General Inquirer			520/2306 (23%)	1/204 (0.5%)
SentiWordNet				174/694 (25%)
LIWC				

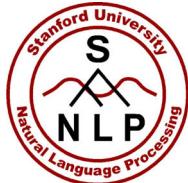


Analyzing the polarity of each word in IMDB

Potts, Christopher. 2011. On the negativity of negation. *SALT* 20, 636-659.

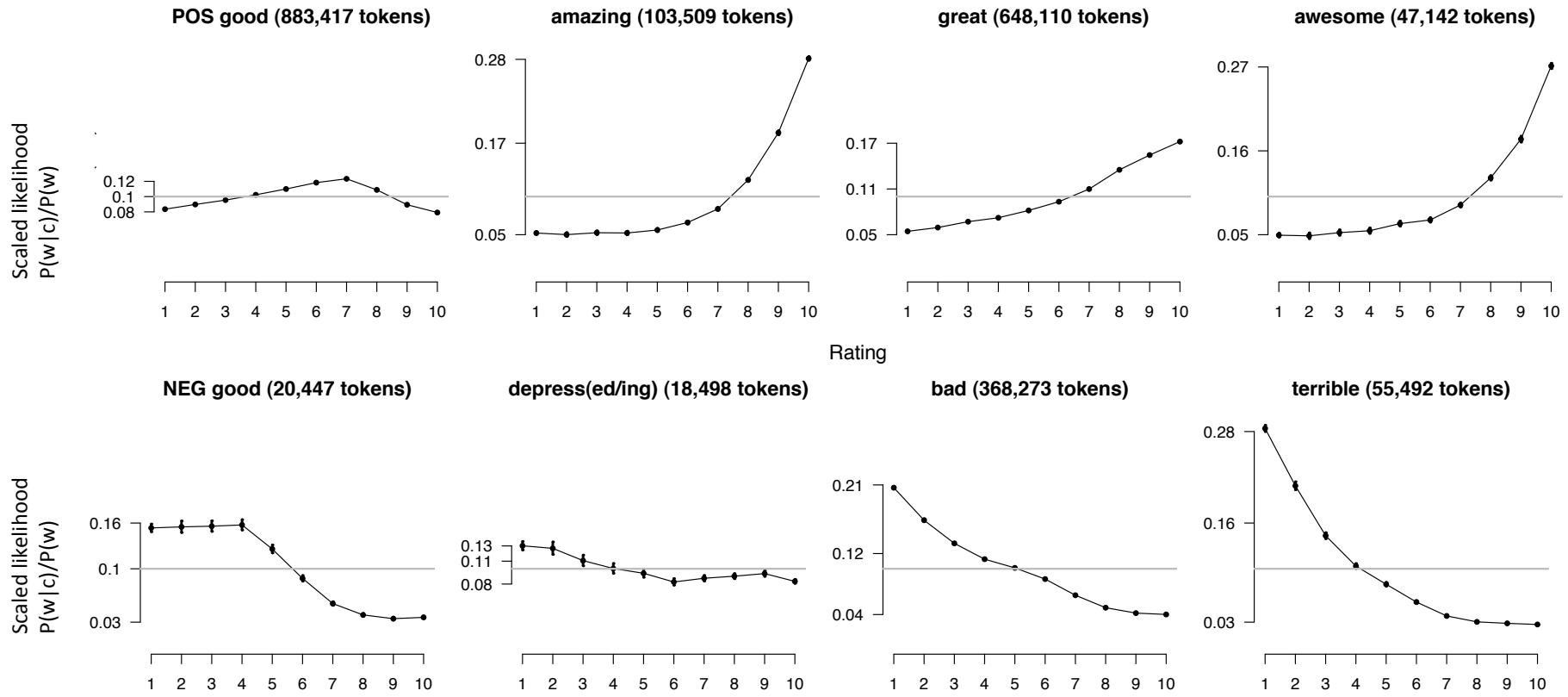
- How likely is each word to appear in each sentiment class?
- Count("bad") in 1-star, 2-star, 3-star, etc.
- But can't use raw counts:
- Instead, **likelihood**: $P(w|c) = \frac{f(w,c)}{\sum_{w \in c} f(w,c)}$
- Make them comparable between words
 - **Scaled likelihood**: $\frac{P(w|c)}{P(w)}$





Analyzing the polarity of each word in IMDB

Potts, Christopher. 2011. On the negativity of negation. SALT 20, 636-659.





Other sentiment feature: Logical negation

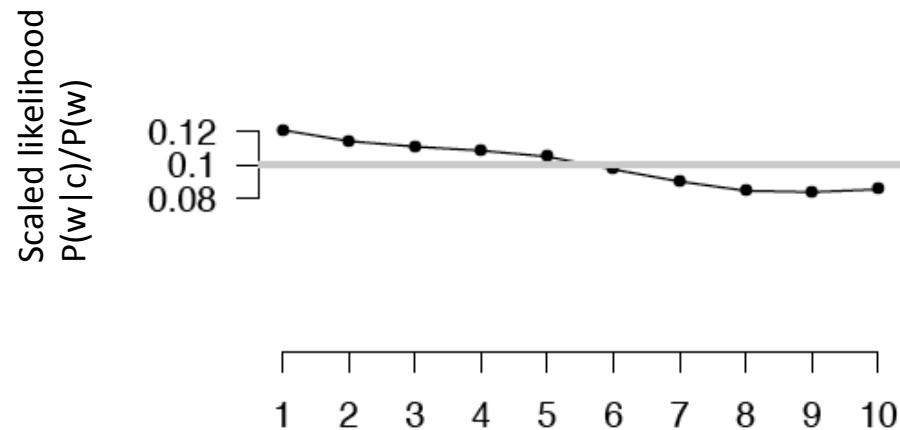
Potts, Christopher. 2011. On the negativity of negation. *SALT* 20, 636-659.

- Is logical negation (*no, not*) associated with negative sentiment?
- Potts experiment:
 - Count negation (*not, n't, no, never*) in online reviews
 - Regress against the review rating

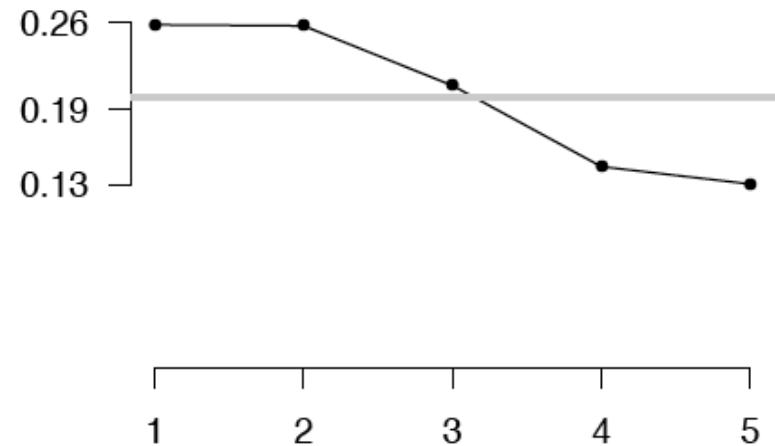


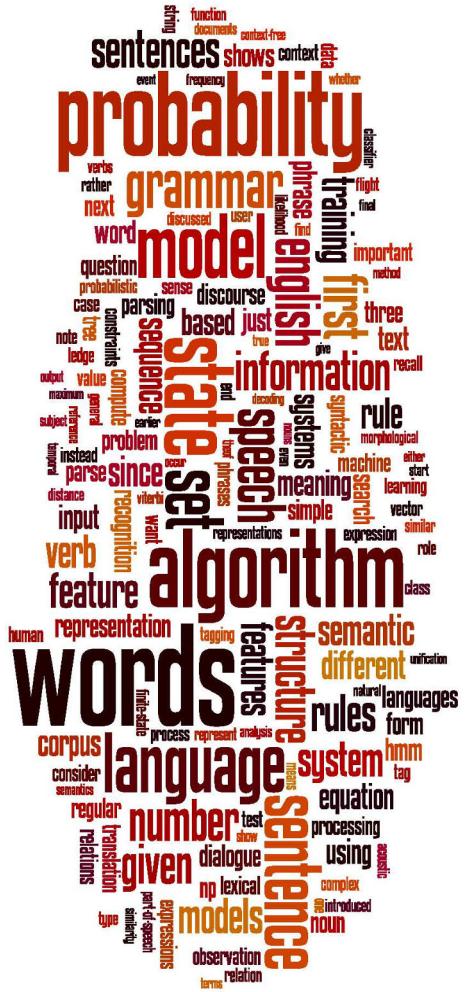
Potts 2011 Results: More negation in negative sentiment

IMDB (4,073,228 tokens)



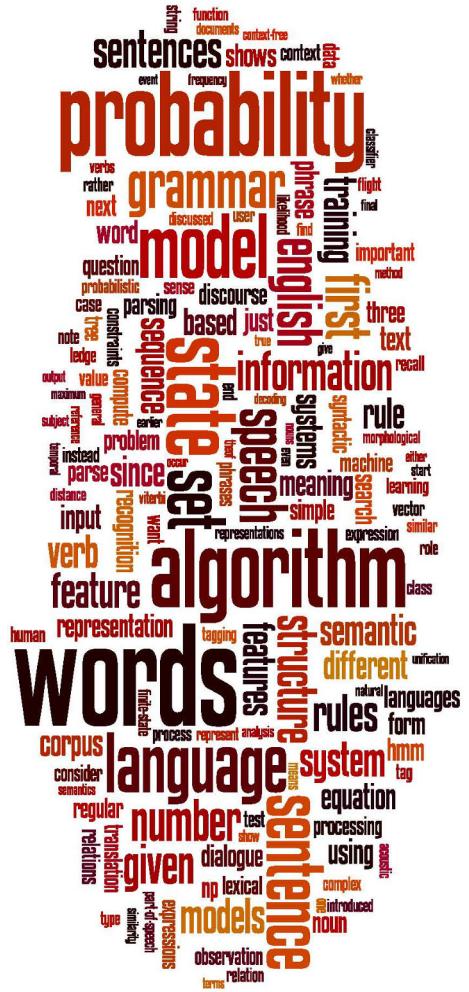
Five-star reviews (846,444 tokens)





Sentiment Analysis

Sentiment Lexicons



Sentiment Analysis

Learning Sentiment Lexicons



Semi-supervised learning of lexicons

- Use a small amount of information
 - A few labeled examples
 - A few hand-built patterns
- To bootstrap a lexicon



Hatzivassiloglou and McKeown intuition for identifying word polarity

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. ACL, 174–181

- Adjectives conjoined by “*and*” have same polarity
 - Fair **and** legitimate, corrupt **and** brutal
 - *fair **and** brutal, *corrupt **and** legitimate
- Adjectives conjoined by “*but*” do not
 - fair **but** brutal



Hatzivassiloglou & McKeown 1997

Step 1

- Label **seed set** of 1336 adjectives (all >20 in 21 million word WSJ corpus)
 - 657 positive
 - adequate central clever famous intelligent remarkable reputed sensitive slender thriving...
 - 679 negative
 - contagious drunken ignorant lanky listless primitive strident troublesome unresolved unsuspecting...



Hatzivassiloglou & McKeown 1997

Step 2

- Expand seed set to conjoined adjectives

Google

"was nice and"

Nice location in Porto and the front desk staff **was nice and helpful**...

www.tripadvisor.com>ShowUserReviews-g189180-d206904-r12068... +1

Mercure Porto Centro: Nice location in Porto and the front desk staff **was nice and helpful** - See traveler reviews, 77 candid photos, and great deals for Porto, ...

nice, helpful

If a girl **was nice and classy**, but had some vibrant purple dye in ...

answers.yahoo.com › Home › All Categories › Beauty & Style › Hair +1

4 answers - Sep 21

Question: Your personal opinion or what you think other people's opinions might ...

Top answer: I think she would be cool and confident like katy perry :)

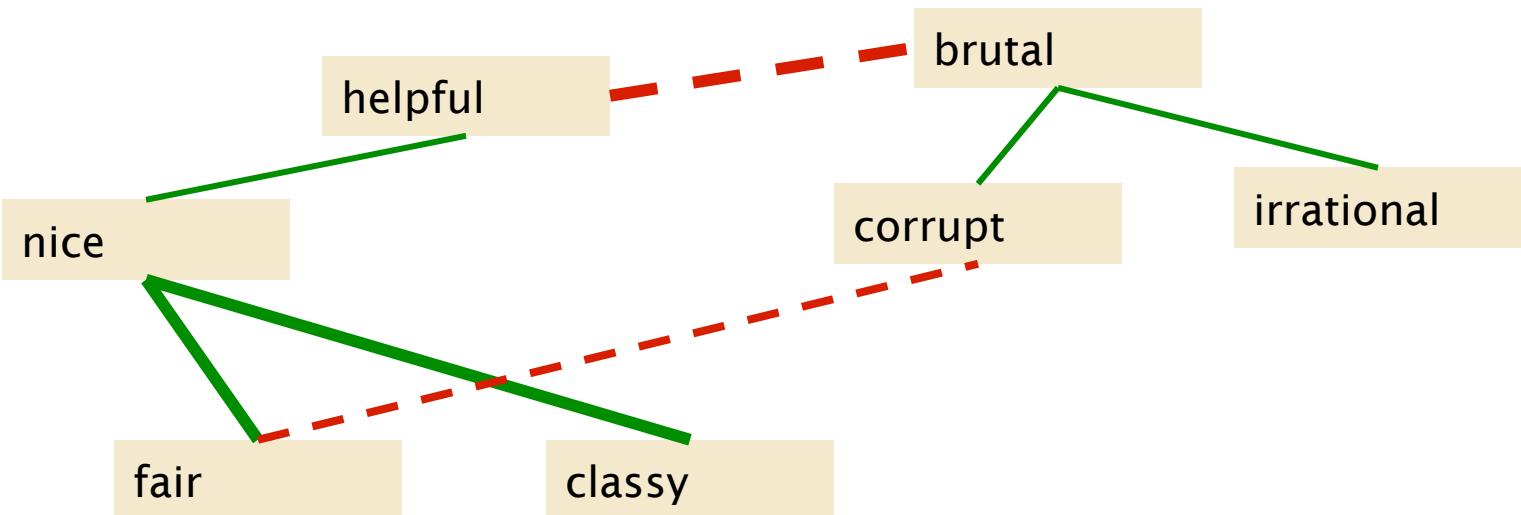
nice, classy



Hatzivassiloglou & McKeown 1997

Step 3

- Supervised classifier assigns “polarity similarity” to each word pair, resulting in graph:

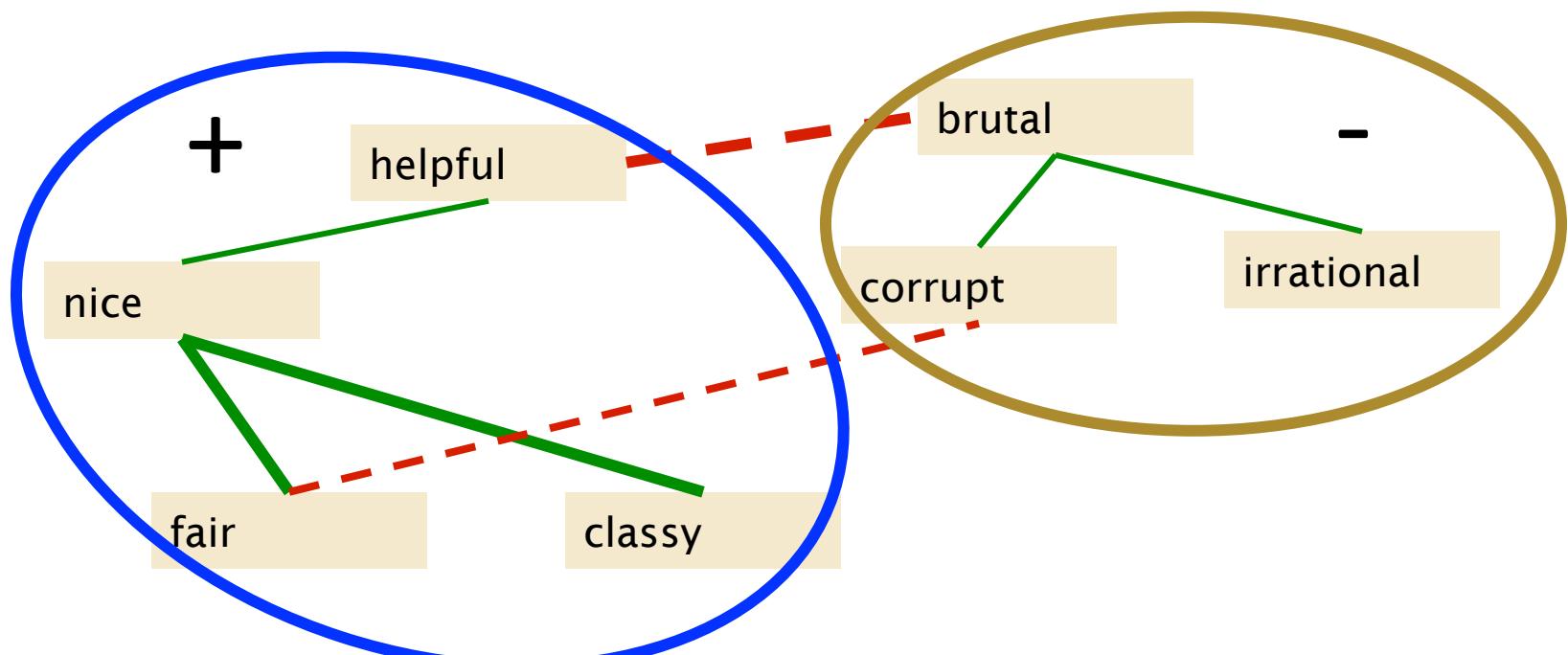




Hatzivassiloglou & McKeown 1997

Step 4

- Clustering for partitioning the graph into two





Output polarity lexicon

- Positive
 - bold decisive disturbing generous good honest important large mature patient peaceful positive proud sound stimulating straightforward strange talented vigorous witty...
- Negative
 - ambiguous cautious cynical evasive harmful hypocritical inefficient insecure irrational irresponsible minor outspoken pleasant reckless risky selfish tedious unsupported vulnerable wasteful...



Output polarity lexicon

- Positive
 - bold decisive **disturbing** generous good honest important large mature patient peaceful positive proud sound stimulating straightforward **strange** talented vigorous witty...
- Negative
 - ambiguous **cautious** cynical evasive harmful hypocritical inefficient insecure irrational irresponsible minor **outspoken pleasant** reckless risky selfish tedious unsupported vulnerable wasteful...



Turney Algorithm

Turney (2002): Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews

1. Extract a *phrasal lexicon* from reviews
2. Learn polarity of each phrase
3. Rate a review by the average polarity of its phrases



Extract two-word phrases with adjectives

First Word	Second Word	Third Word (not extracted)
JJ	NN or NNS	anything
RB, RBR, RBS	JJ	Not NN nor NNS
JJ	JJ	Not NN or NNS
NN or NNS	JJ	Not NN nor NNS
RB, RBR, or RBS	VB, VBD, VBN, VBG	anything



How to measure polarity of a phrase?

- Positive phrases co-occur more with “*excellent*”
- Negative phrases co-occur more with “*poor*”
- But how to measure co-occurrence?



Pointwise Mutual Information

- **Mutual information** between 2 random variables X and Y

$$I(X,Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- **Pointwise mutual information:**

- How much more do events x and y co-occur than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$



Pointwise Mutual Information

- **Pointwise mutual information:**
 - How much more do events x and y co-occur than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- **PMI between two words:**
 - How much more do two words co-occur than if they were independent?

$$\text{PMI}(\textit{word}_1, \textit{word}_2) = \log_2 \frac{P(\textit{word}_1, \textit{word}_2)}{P(\textit{word}_1)P(\textit{word}_2)}$$



How to Estimate Pointwise Mutual Information

- Query search engine (Altavista)
 - $P(\text{word})$ estimated by $\text{hits}(\text{word})/N$
 - $P(\text{word}_1, \text{word}_2)$ by $\text{hits}(\text{word1 NEAR word2})/N^2$

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{\text{hits}(\text{word}_1 \text{ NEAR } \text{word}_2)}{\text{hits}(\text{word}_1)\text{hits}(\text{word}_2)}$$



Does phrase appear more with “poor” or “excellent”?

$\text{Polarity}(\text{phrase}) = \text{PMI}(\text{phrase}, \text{"excellent"}) - \text{PMI}(\text{phrase}, \text{"poor"})$

$$= \log_2 \frac{\text{hits}(\text{phrase NEAR "excellent"})}{\text{hits}(\text{phrase})\text{hits}(\text{"excellent"})} - \log_2 \frac{\text{hits}(\text{phrase NEAR "poor"})}{\text{hits}(\text{phrase})\text{hits}(\text{"poor"})}$$

$$= \log_2 \frac{\text{hits}(\text{phrase NEAR "excellent"})}{\text{hits}(\text{phrase})\text{hits}(\text{"excellent"})} \frac{\text{hits}(\text{phrase})\text{hits}(\text{"poor"})}{\text{hits}(\text{phrase NEAR "poor"})}$$

$$= \log_2 \left(\frac{\text{hits}(\text{phrase NEAR "excellent"})\text{hits}(\text{"poor"})}{\text{hits}(\text{phrase NEAR "poor"})\text{hits}(\text{"excellent"})} \right)$$



Phrases from a thumbs-up review

Phrase	POS tags	Polarity
online service	JJ NN	2.8
online experience	JJ NN	2.3
direct deposit	JJ NN	1.3
local branch	JJ NN	0.42
...		
low fees	JJ NNS	0.33
true service	JJ NN	-0.73
other bank	JJ NN	-0.85
inconveniently located	JJ NN	-1.5
<i>Average</i>		0.32



Phrases from a thumbs-down review

Phrase	POS tags	Polarity
direct deposits	JJ NNS	5.8
online web	JJ NN	1.9
very handy	RB JJ	1.4
...		
virtual monopoly	JJ NN	-2.0
lesser evil	RBR JJ	-2.3
other problems	JJ NNS	-2.8
low funds	JJ NNS	-6.8
unethical practices	JJ NNS	-8.5
<i>Average</i>		-1.2



Results of Turney algorithm

- 410 reviews from Epinions
 - 170 (41%) negative
 - 240 (59%) positive
- Majority class baseline: 59%
- Turney algorithm: 74%

- Phrases rather than words
- Learns domain-specific information



Using WordNet to learn polarity

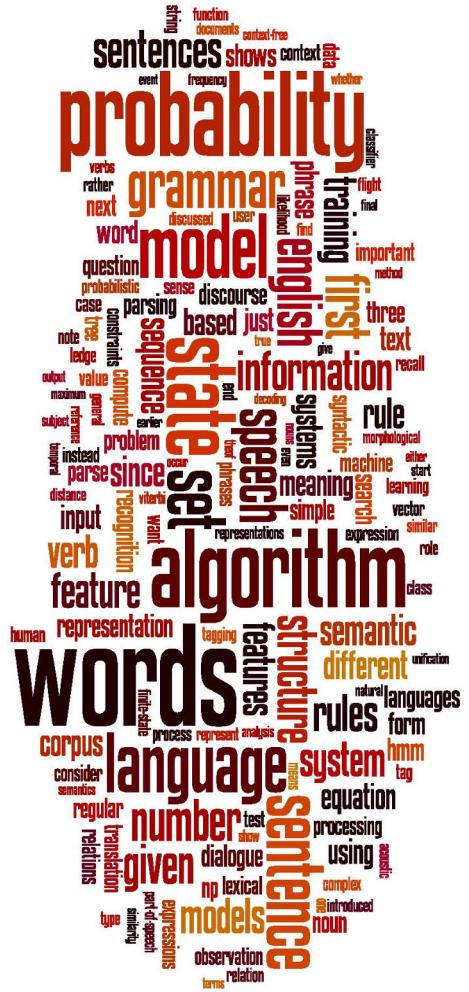
S.M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. COLING 2004
M. Hu and B. Liu. Mining and summarizing customer reviews. In Proceedings of KDD, 2004

- WordNet: online thesaurus (covered in later lecture).
- Create positive (“good”) and negative seed-words (“terrible”)
- Find Synonyms and Antonyms
 - Positive Set: Add synonyms of positive words (“well”) and antonyms of negative words
 - Negative Set: Add synonyms of negative words (“awful”) and antonyms of positive words (“evil”)
- Repeat, following chains of synonyms
- Filter



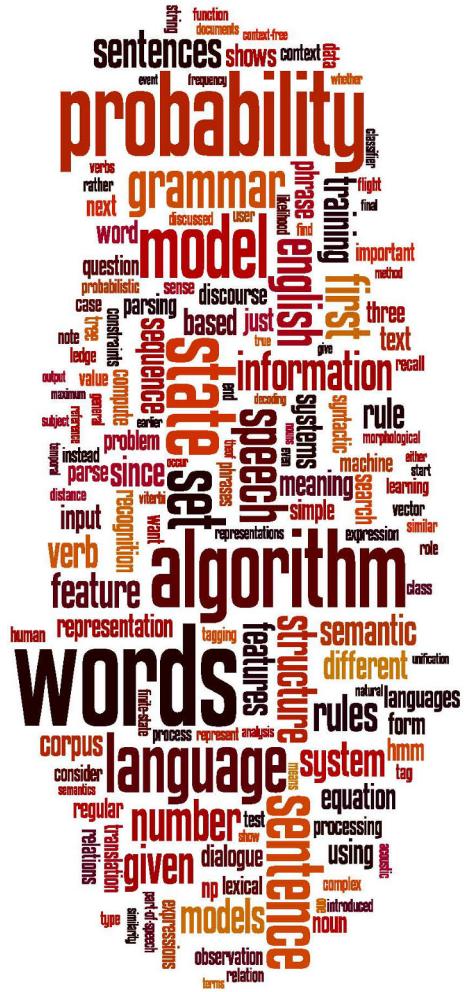
Summary on Learning Lexicons

- Advantages:
 - Can be domain-specific
 - Can be more robust (more words)
- Intuition
 - Start with a seed set of words ('good', 'poor')
 - Find other words that have similar polarity:
 - Using “and” and “but”
 - Using words that occur nearby in the same document
 - Using WordNet synonyms and antonyms



Sentiment Analysis

Learning Sentiment Lexicons



Sentiment Analysis

Other Sentiment Tasks



Finding sentiment of a sentence

- Important for finding aspects or attributes
 - Target of sentiment
- The food was great but the service was awful



Finding aspect/attribute/target of sentiment

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In Proceedings of KDD.

S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. 2008. Building a Sentiment Summarizer for Local Service Reviews. WWW Workshop.

- Frequent phrases + rules
 - Find all highly frequent phrases across reviews (“fish tacos”)
 - Filter by rules like “occurs right after sentiment word”
 - “...great fish tacos” means fish tacos a likely aspect

Casino	casino, buffet, pool, resort, beds
Children's Barber	haircut, job, experience, kids
Greek Restaurant	food, wine, service, appetizer, lamb
Department Store	selection, department, sales, shop, clothing



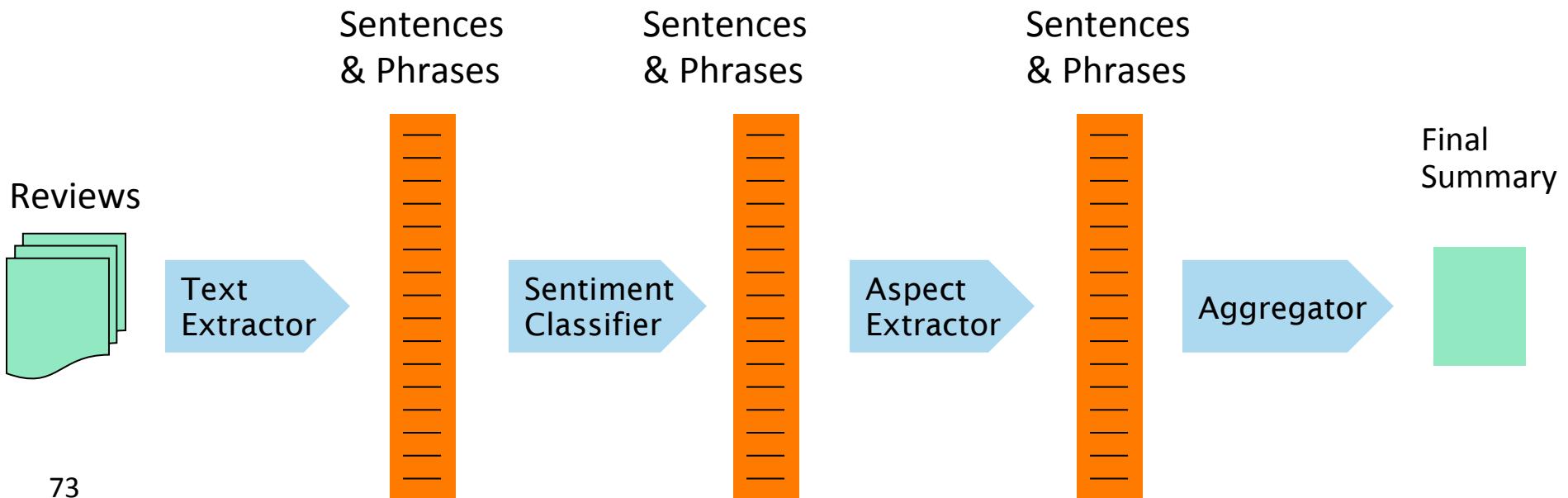
Finding aspect/attribute/target of sentiment

- The aspect name may not be in the sentence
- For restaurants/hotels, aspects are well-understood
- Supervised classification
 - Hand-label a small corpus of restaurant review sentences with aspect
 - food, décor, service, value, NONE
 - Train a classifier to assign an aspect to a sentence
 - “Given this sentence, is the aspect *food, décor, service, value, or NONE*”



Putting it all together: Finding sentiment for aspects

S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. 2008. Building a Sentiment Summarizer for Local Service Reviews. WWW Workshop





Results of Blair-Goldensohn et al. method

Rooms (3/5 stars, 41 comments)

- (+) The room was clean and everything worked fine – even the water pressure ...
- (+) We went because of the free room and was pleasantly pleased ...
- (-) ...the worst hotel I had ever stayed at ...

Service (3/5 stars, 31 comments)

- (+) Upon checking out another couple was checking early due to a problem ...
- (+) Every single hotel staff member treated us great and answered every ...
- (-) The food is cold and the service gives new meaning to SLOW.

Dining (3/5 stars, 18 comments)

- (+) our favorite place to stay in biloxi.the food is great also the service ...
- (+) Offer of free buffet for joining the Play



Baseline methods assume classes have equal frequencies!

- If not balanced (common in the real world)
 - can't use accuracies as an evaluation
 - need to use F-scores
- Severe imbalancing also can degrade classifier performance
- Two common solutions:
 1. Resampling in training
 - Random undersampling
 2. Cost-sensitive learning
 - Penalize SVM more for misclassification of the rare thing



How to deal with 7 stars?

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. ACL, 115-124

1. Map to binary
2. Use linear or ordinal regression
 - Or specialized models like metric labeling



Summary on Sentiment

- Generally modeled as classification or regression task
 - predict a binary or ordinal label
- Features:
 - Negation is important
 - Using all words (in naïve bayes) works well for some tasks
 - Finding subsets of words may help in other tasks
 - Hand-built polarity lexicons
 - Use seeds and semi-supervised learning to induce lexicons



Scherer Typology of Affective States

- **Emotion:** brief organically synchronized ... evaluation of a major event
 - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood:** diffuse non-caused low-intensity long-duration change in subjective feeling
 - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances:** affective stance toward another person in a specific interaction
 - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes:** enduring, affectively colored beliefs, dispositions towards objects or persons
 - *liking, loving, hating, valuing, desiring*
- **Personality traits:** stable personality dispositions and typical behavior tendencies
 - *nervous, anxious, reckless, morose, hostile, jealous*



Computational work on other affective states

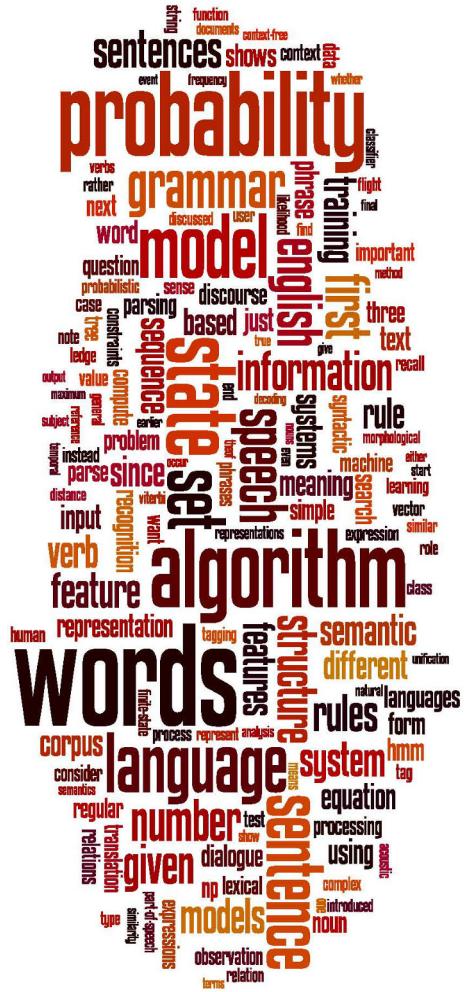
- **Emotion:**
 - Detecting annoyed callers to dialogue system
 - Detecting confused/frustrated versus confident students
- **Mood:**
 - Finding traumatized or depressed writers
- **Interpersonal stances:**
 - Detection of flirtation or friendliness in conversations
- **Personality traits:**
 - Detection of extroverts



Detection of Friendliness

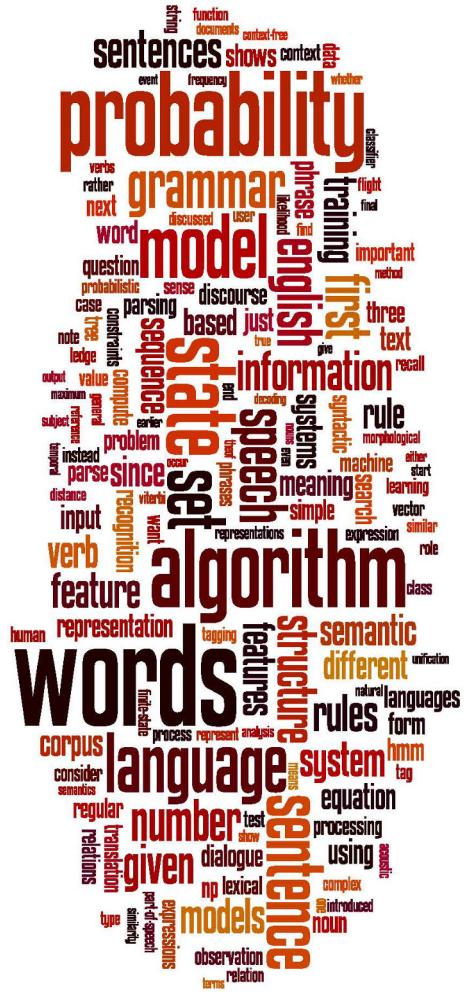
Ranganath, Jurafsky, McFarland

- Friendly speakers use collaborative conversational style
 - Laughter
 - Less use of negative emotional words
 - More sympathy
 - That's too bad I'm sorry to hear that
 - More agreement
 - I think so too
 - Less hedges
 - kind of sort of a little ...



Sentiment Analysis

Other Sentiment Tasks



Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning



Introduction

- So far we've looked at “generative models”
 - Language models, Naive Bayes
- But there is now much use of conditional or discriminative probabilistic models in NLP, Speech, IR (and ML generally)
- Because:
 - They give high accuracy performance
 - They make it easy to incorporate lots of linguistically important features
 - They allow automatic building of language independent, retargetable NLP modules



Joint vs. Conditional Models

- We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .
- **Joint (generative) models** place probabilities over both observed data and the hidden stuff (generate the observed data from hidden stuff):
 - All the classic StatNLP models:
 - n -gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models

$$P(c, d)$$



Joint vs. Conditional Models

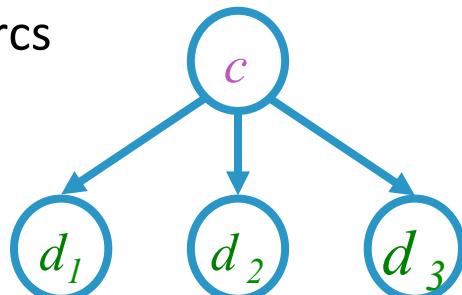
- **Discriminative (conditional) models** take the data as given, and put a probability over hidden structure given the data:
 - Logistic regression, conditional loglinear or maximum entropy models, conditional random fields
 - Also, SVMs, (averaged) perceptron, etc. are discriminative classifiers (but not directly probabilistic)

$$P(c|d)$$



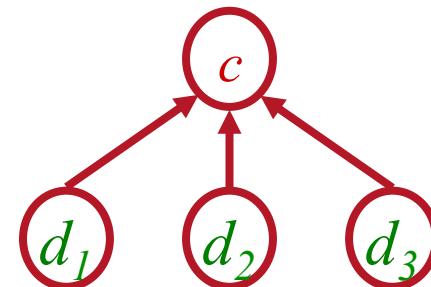
Bayes Net/Graphical Models

- Bayes net diagrams draw circles for random variables, and lines for direct dependencies
- Some variables are observed; some are hidden
- Each node is a little classifier (conditional probability table) based on incoming arcs



Naive Bayes

Generative



Logistic Regression

Discriminative



Conditional vs. Joint Likelihood

- A *joint* model gives probabilities $P(d,c)$ and tries to maximize this joint likelihood.
 - It turns out to be trivial to choose weights: just relative frequencies.
- A *conditional* model gives probabilities $P(c|d)$. It takes the data as given and models only the conditional probability of the class.
 - We seek to maximize conditional likelihood.
 - Harder to do (as we'll see...)
 - More closely related to classification error.



Conditional models work well: Word Sense Disambiguation

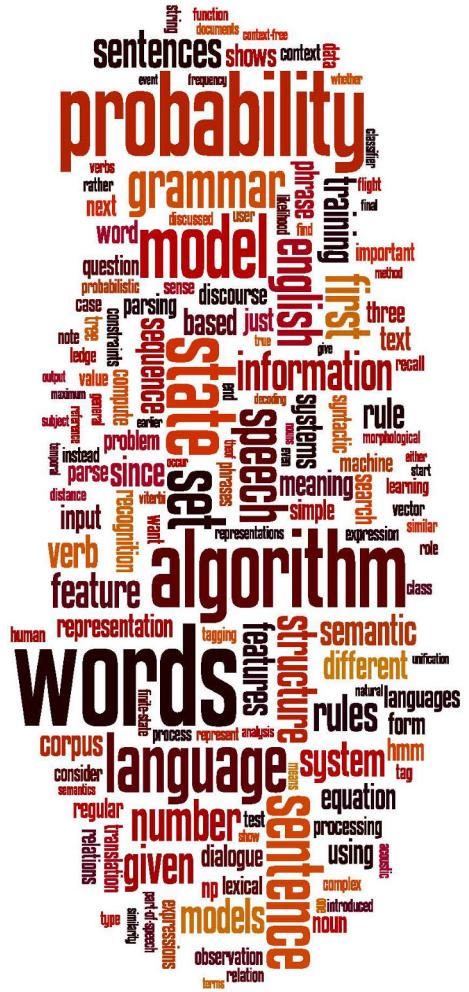
Training Set	
Objective	Accuracy
Joint Like.	86.8
Cond. Like.	98.5

- Even with exactly the same features, changing from joint to conditional estimation increases performance

Test Set	
Objective	Accuracy
Joint Like.	73.6
Cond. Like.	76.1

- That is, we use the same smoothing, and the same word-class features, we just change the numbers (parameters)

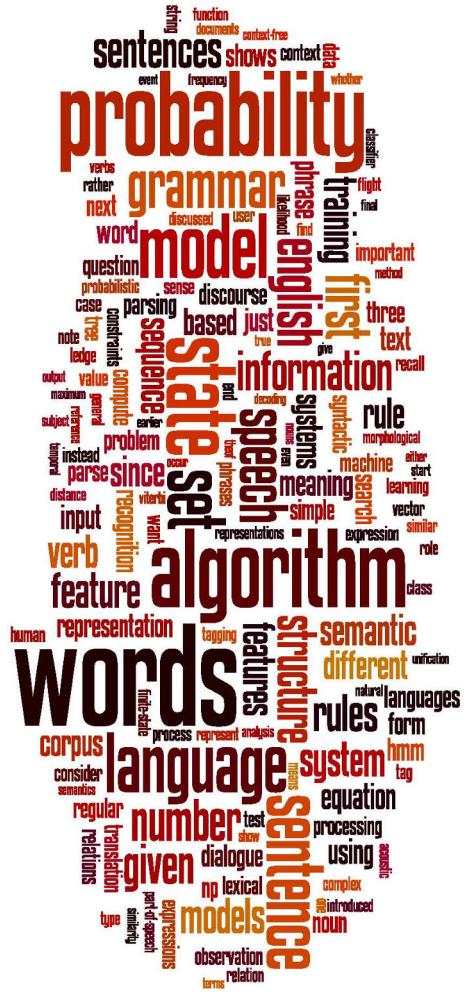
(Klein and Manning 2002, using Senseval-1 Data)



Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning



Discriminative Model Features

Making features from text for discriminative NLP models

Christopher Manning



Features

- In these slides and most maxent work: *features* f are elementary pieces of evidence that link aspects of what we observe d with a category c that we want to predict
- A feature is a function with a bounded real value: $f: C \times D \rightarrow \mathbb{R}$



Features

- In these slides and most maxent work: *features* f are elementary pieces of evidence that link aspects of what we observe d with a category c that we want to predict
- A feature is a function with a bounded real value



Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{"in"} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, "c")]$



- Models will assign to each feature a *weight*:
 - A positive weight votes that this configuration is likely correct
 - A negative weight votes that this configuration is likely incorrect



Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{"in"} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, "c")]$

LOCATION
in Arcadia

LOCATION
in Québec

DRUG
taking Zantac

PERSON
saw Sue

- Models will assign to each feature a *weight*:
 - A positive weight votes that this configuration is likely correct
 - A negative weight votes that this configuration is likely incorrect



Feature Expectations

- We will crucially make use of two *expectations*

- actual or predicted counts of a feature firing:

- Empirical count (expectation) of a feature:

$$\text{empirical } E(f_i) = \sum_{(c,d) \in \text{observed}(C,D)} f_i(c,d)$$

- Model expectation of a feature:

$$E(f_i) = \sum_{(c,d) \in (C,D)} P(c,d) f_i(c,d)$$



Features

- In NLP uses, usually a feature specifies (1) an indicator function – a yes/no boolean matching function – of properties of the input and (2) a particular class
 - $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$ [Value is 0 or 1]
 - They pick out a data subset and suggest a label for it.
- We will say that $\Phi(d)$ is a feature of the data d , when, for each c_j , the conjunction $\Phi(d) \wedge c = c_j$ is a feature of the data-class pair (c, d)



Features

- In NLP uses, usually a feature specifies
 1. an indicator function – a yes/no boolean matching function – of properties of the input and
 2. a particular class

$$f_i(c, d) \equiv [\Phi(d) \wedge c = c_j] \quad [\text{Value is 0 or 1}]$$

- Each feature picks out a data subset and suggests a label for it



Feature-Based Models

- The decision about a data point is based only on the **features** active at that point.

Data
BUSINESS: Stocks hit a yearly low ...

Label: BUSINESS
Features
{..., stocks, hit, a, yearly, low, ...}

Text
Categorization

Data
... to restructure
bank: MONEY debt.

Label: MONEY
Features
{..., w_{-1} =restructure,
 w_{+1} =debt, L=12, ...}

Word-Sense
Disambiguation

Data
DT JJ NN ...
The previous fall ...

Label: NN
Features
{ w =fall, t_{-1} =JJ
 w_{-1} =previous}

POS Tagging



Example: Text Categorization

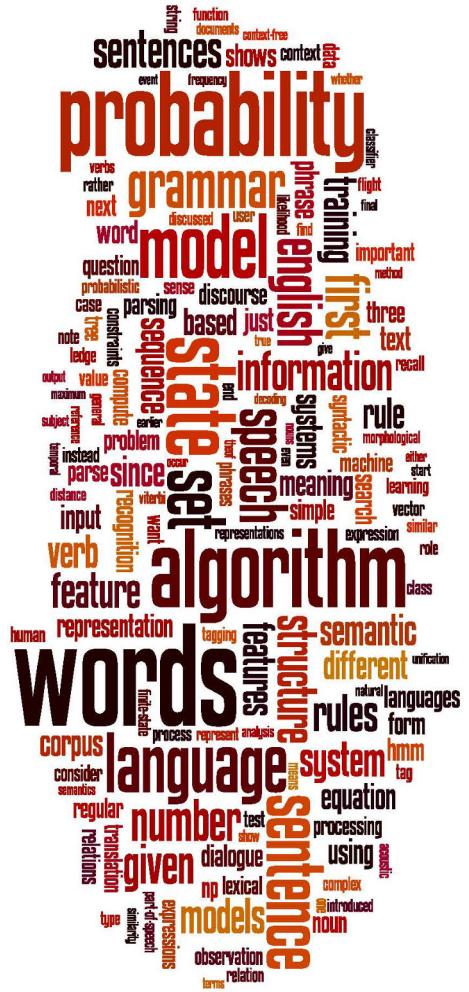
(Zhang and Oles 2001)

- Features are presence of each word in a document and the document class (they do feature selection to use reliable indicator words)
- Tests on classic Reuters data set (and others)
 - Naïve Bayes: 77.0% F_1
 - Linear regression: 86.0%
 - Logistic regression: 86.4%
 - Support vector machine: 86.5%
- Paper emphasizes the importance of *regularization* (smoothing) for successful use of discriminative methods (not used in much early NLP/IR work)



Other Maxent Classifier Examples

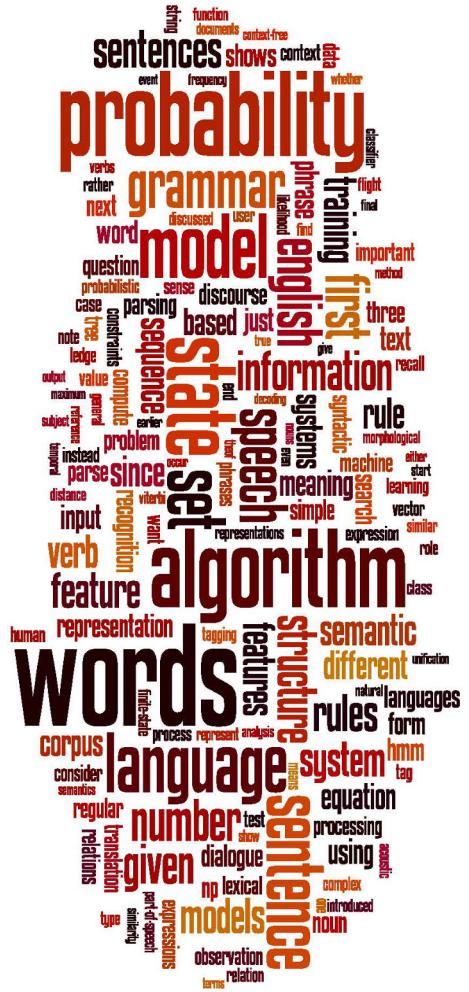
- You can use a maxent classifier whenever you want to assign data points to one of a number of classes:
 - Sentence boundary detection ([Mikheev 2000](#))
 - Is a period end of sentence or abbreviation?
 - Sentiment analysis ([Pang and Lee 2002](#))
 - Word unigrams, bigrams, POS counts, ...
 - PP attachment ([Ratnaparkhi 1998](#))
 - Attach to verb or noun? Features of head noun, preposition, etc.
 - Parsing decisions in general ([Ratnaparkhi 1997; Johnson et al. 1999, etc.](#))



Discriminative Model Features

Making features from text for discriminative NLP models

Christopher Manning



Feature-based Linear Classifiers

How to put features into a classifier



Feature-Based Linear Classifiers

- Linear classifiers at classification time:
 - Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
 - Assign a weight λ_i to each feature f_i .
 - We consider each class for an observed datum d
 - For a pair (c,d) , features vote with their weights:
 - $\text{vote}(c) = \sum \lambda_i f_i(c, d)$

PERSON
in Québec

LOCATION
in Québec

DRUG
in Québec

- Choose the class c which maximizes $\sum \lambda_i f_i(c, d)$



Feature-Based Linear Classifiers

- Linear classifiers at classification time:
 - Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
 - Assign a weight λ_i to each feature f_i .
 - We consider each class for an observed datum d
 - For a pair (c,d) , features vote with their weights:
 - $\text{vote}(c) = \sum \lambda_i f_i(c, d)$

PERSON
in Québec



0.3 DRUG
in Québec

- Choose the class c which maximizes $\sum \lambda_i f_i(c, d) = \text{LOCATION}$



Feature-Based Linear Classifiers

There are many ways to chose weights for features

- Perceptron: find a currently misclassified example, and nudge weights in the direction of its correct classification
- Margin-based methods (Support Vector Machines)

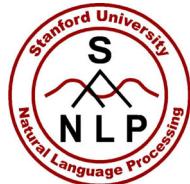


Feature-Based Linear Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
 - Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c, d)$
- $$P(c | d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$
- $\exp \sum_i \lambda_i f_i(c, d)$

← Makes votes positive
- $\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)$

← Normalizes votes
- $P(\text{LOCATION}|\text{in Québec}) = e^{1.8}e^{-0.6}/(e^{1.8}e^{-0.6} + e^{0.3} + e^0) = 0.586$
 - $P(\text{DRUG}|\text{in Québec}) = e^{0.3}/(e^{1.8}e^{-0.6} + e^{0.3} + e^0) = 0.238$
 - $P(\text{PERSON}|\text{in Québec}) = e^0/(e^{1.8}e^{-0.6} + e^{0.3} + e^0) = 0.176$
 - The weights are the parameters of the probability model, combined via a “soft max” function



Feature-Based Linear Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
 - Given this model form, we will choose parameters $\{\lambda_i\}$ that *maximize the conditional likelihood* of the data according to this model.
 - We construct not only classifications, but probability distributions over classifications.
 - There are other (good!) ways of discriminating classes – SVMs, boosting, even perceptrons – but these methods are not as trivial to interpret as distributions over classes.



Aside: logistic regression

- Maxent models in NLP are essentially the same as multiclass logistic regression models in statistics (or machine learning)
 - If you haven't seen these before, don't worry, this presentation is self-contained!
 - If you have seen these before you might think about:
 - The parameterization is slightly different in a way that is advantageous for NLP-style models with tons of sparse features (but statistically inelegant)
 - The key role of feature functions in NLP and in this presentation
 - The features are more general, with f also being a function of the class – when might this be useful?



Quiz Question

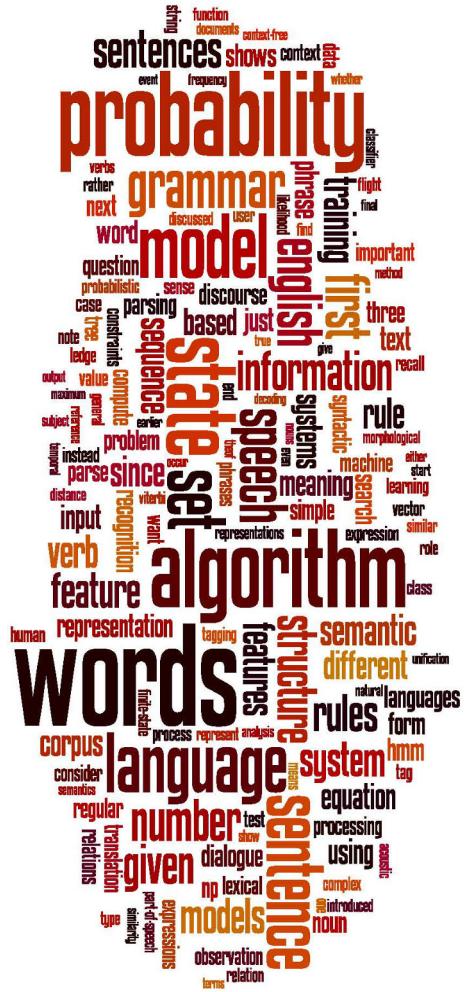
- Assuming exactly the same set up (3 class decision: LOCATION, PERSON, or DRUG; 3 features as before, maxent), what are:
 - $P(\text{PERSON} \mid \text{by Goéric}) =$
 - $P(\text{LOCATION} \mid \text{by Goéric}) =$
 - $P(\text{DRUG} \mid \text{by Goéric}) =$
 - $1.8 f_1(c, d) = [c = \text{LOCATION} \wedge w_1 = \text{"in"} \wedge \text{isCapitalized}(w)]$
 - $-0.6 f_2(c, d) = [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
 - $0.3 f_3(c, d) = [c = \text{DRUG} \wedge \text{ends}(w, "c")]$

PERSON
 by Goéric

LOCATION
 by Goéric

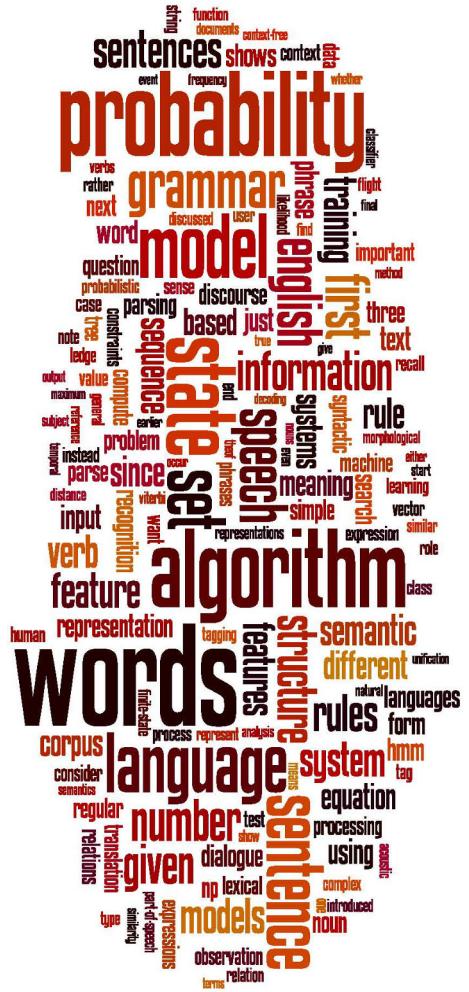
DRUG
 by Goéric

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$



Feature-based Linear Classifiers

How to put features into a classifier



Building a Maxent Model

The nuts and bolts



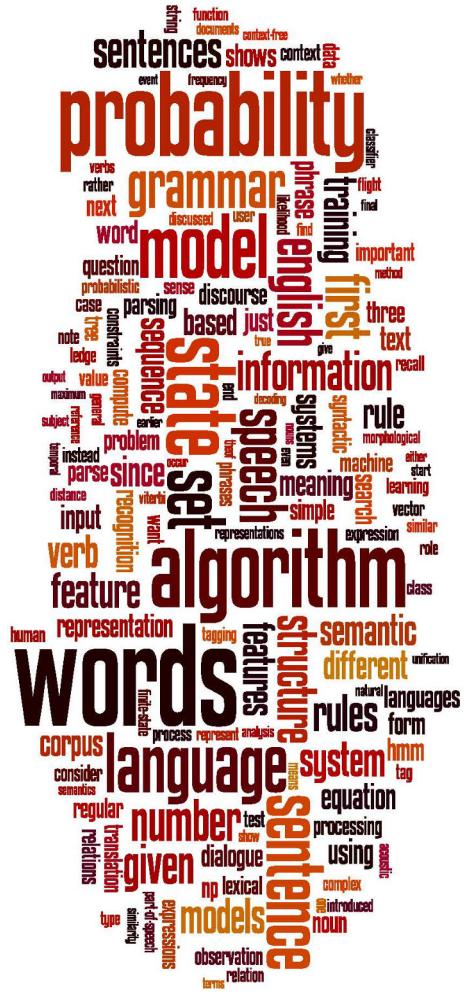
Building a Maxent Model

- We define features (indicator functions) over data points
 - Features represent sets of data points which are distinctive enough to deserve model parameters.
 - Words, but also “word contains number”, “word ends with *ing*”, etc.
- We will simply encode each Φ feature as a unique String
 - A datum will give rise to a set of Strings: the active Φ features
 - Each feature $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$ gets a real number weight
- We concentrate on Φ features but the math uses i indices of f_i



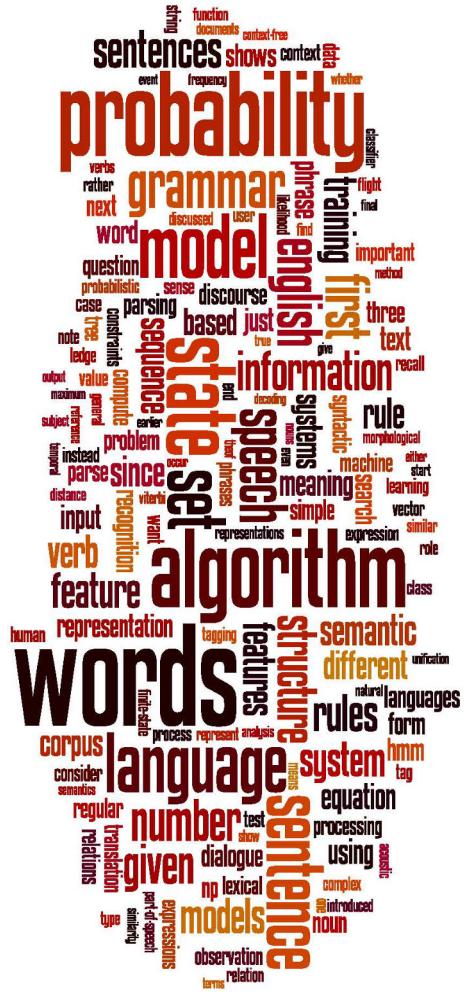
Building a Maxent Model

- Features are often added during model development to target errors
 - Often, the easiest thing to think of are features that mark bad combinations
- Then, for any given feature weights, we want to be able to calculate:
 - Data conditional likelihood
 - Derivative of the likelihood wrt each feature weight
 - Uses expectations of each feature according to the model
- We can then find the optimum feature weights (discussed later).



Building a Maxent Model

The nuts and bolts



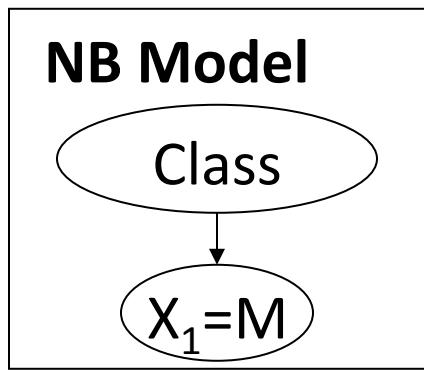
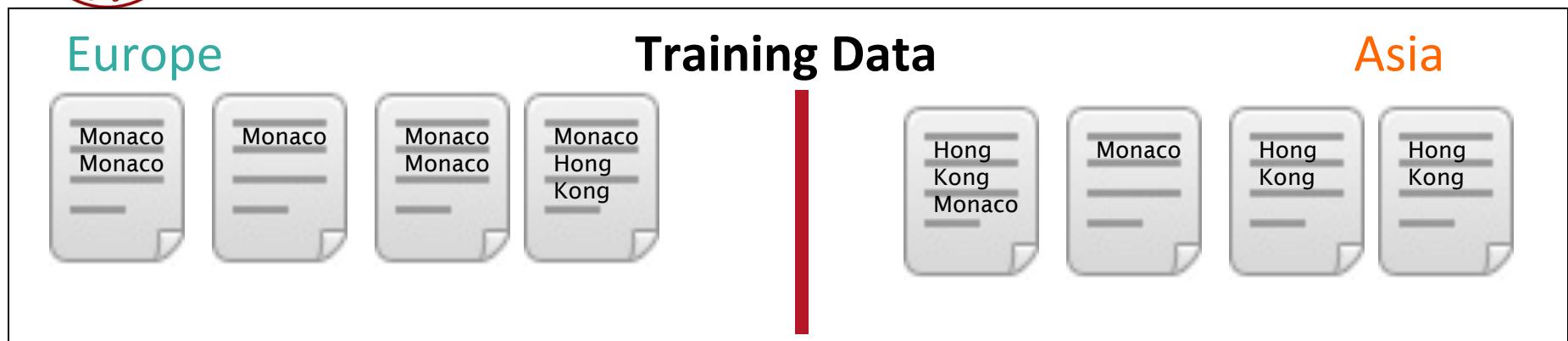
Naive Bayes vs. Maxent models

Generative vs. Discriminative models: The problem of overcounting evidence

Christopher Manning



Text classification: Asia or Europe



NB FACTORS:

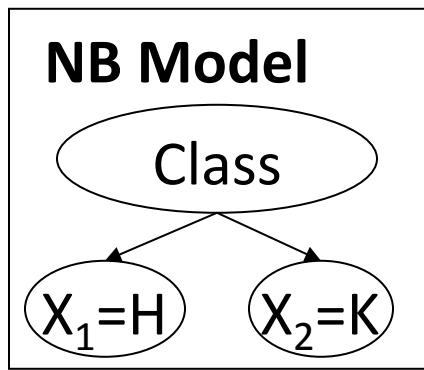
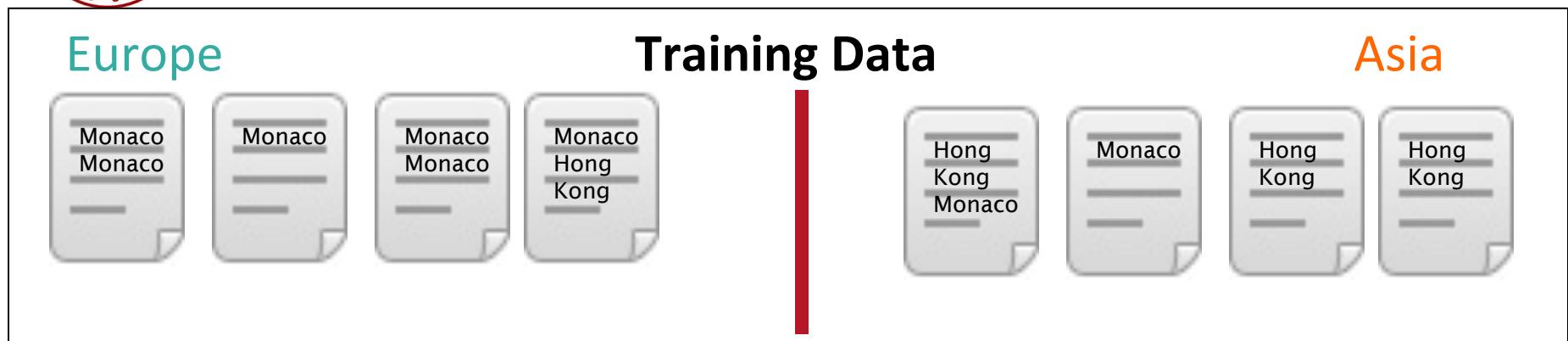
- $P(A) = P(E) =$
- $P(M|A) =$
- $P(M|E) =$

PREDICTIONS:

- $P(A,M) =$
- $P(E,M) =$
- $P(A|M) =$
- $P(E|M) =$



Text classification: Asia or Europe



NB FACTORS:

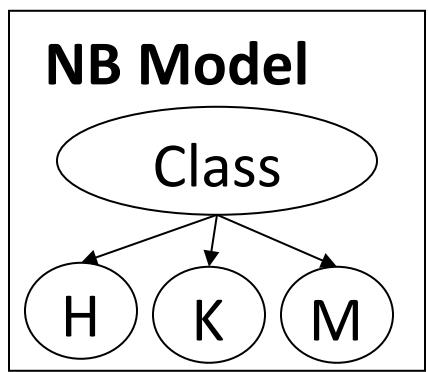
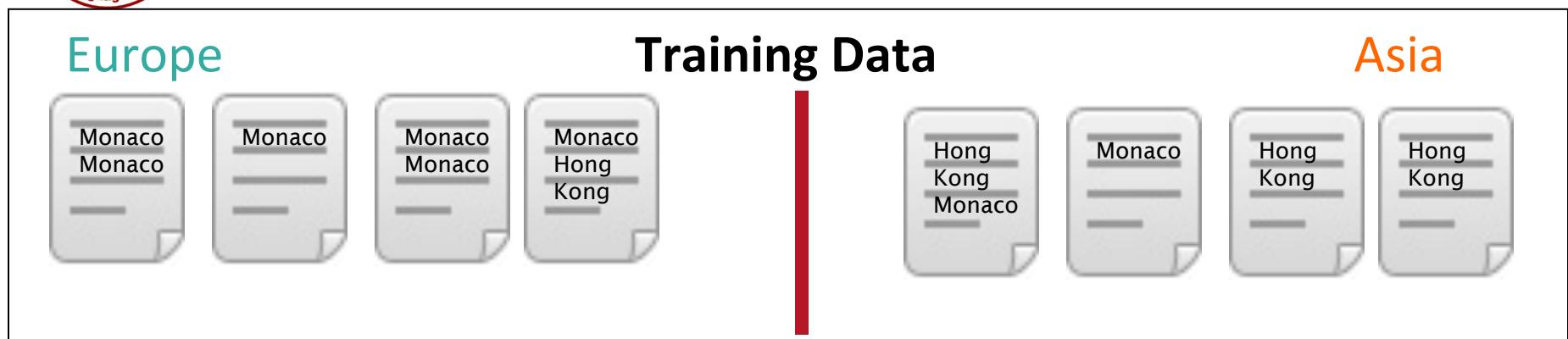
- $P(A) = P(E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$

PREDICTIONS:

- $P(A,H,K) =$
- $P(E,H,K) =$
- $P(A|H,K) =$
- $P(E|H,K) =$



Text classification: Asia or Europe



NB FACTORS:

- $P(A) = P(E) =$
- $P(M|A) =$
- $P(M|E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$

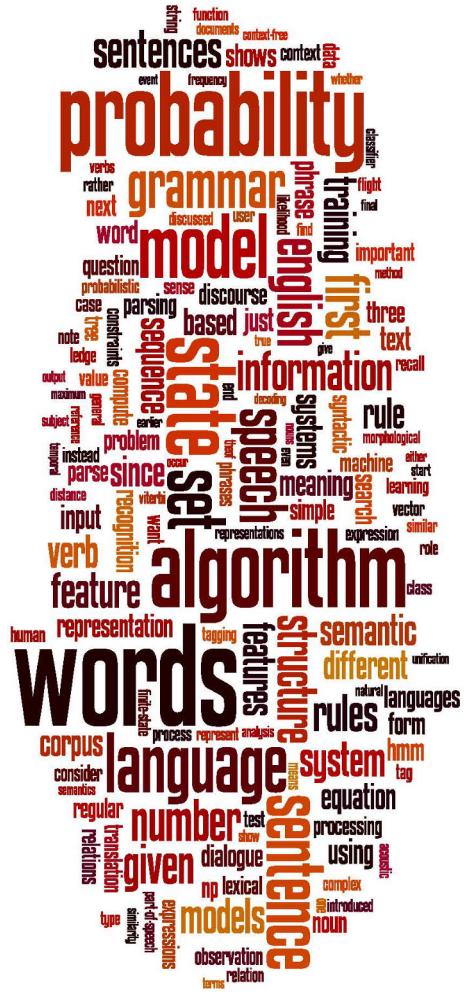
PREDICTIONS:

- $P(A,H,K,M) =$
- $P(E,H,K,M) =$
- $P(A|H,K,M) =$
- $P(E|H,K,M) =$



Naive Bayes vs. Maxent Models

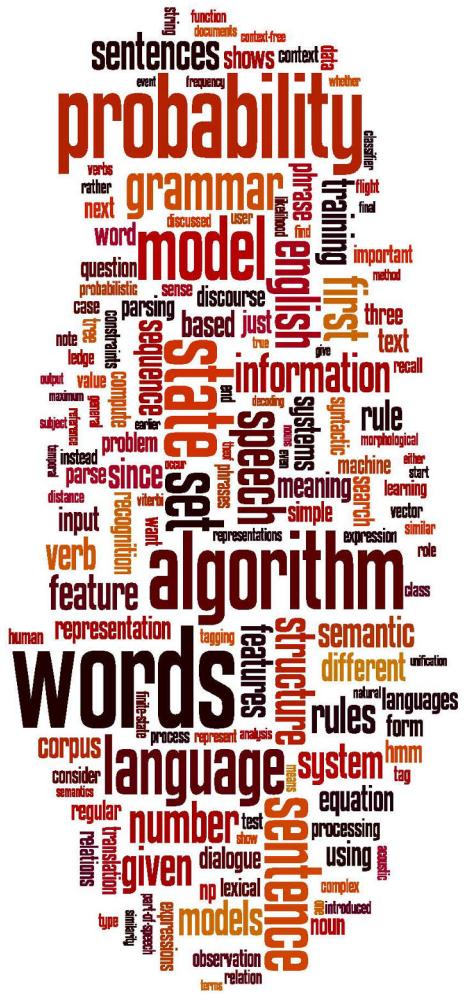
- Naive Bayes models multi-count correlated evidence
 - Each feature is multiplied in, even when you have multiple features telling you the same thing
- Maximum Entropy models (pretty much) solve this problem
 - As we will see, this is done by weighting features so that model expectations match the observed (empirical) expectations



Naive Bayes vs. Maxent models

Generative vs. Discriminative models: The problem of overcounting evidence

Christopher Manning



Maxent Models and Discriminative Estimation

Maximizing the likelihood



Exponential Model Likelihood

- Maximum (Conditional) Likelihood Models :
 - Given a model form, choose values of parameters to maximize the (conditional) likelihood of the data.

$$\log P(C | D, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c | d, \lambda) = \sum_{(c,d) \in (C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$



The Likelihood Value

- The (log) conditional likelihood of iid data (C, D) according to maxent model is a function of the data and the parameters λ :

$$\log P(C | D, \lambda) = \log \prod_{(c,d) \in (C,D)} P(c | d, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c | d, \lambda)$$

- If there aren't many values of c , it's easy to calculate:

$$\log P(C | D, \lambda) = \sum_{(c,d) \in (C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$



The Likelihood Value

- We can separate this into two components:

$$\log P(C | D, \lambda) = \sum_{(c,d) \in (C,D)} \log \exp \sum_i \lambda_i f_i(c,d) - \sum_{(c,d) \in (C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)$$

$$\log P(C | D, \lambda) = N(\lambda) - M(\lambda)$$

- The derivative is the difference between the derivatives of each component



The Derivative I: Numerator

$$\begin{aligned}
 \frac{\partial N(\lambda)}{\partial \lambda_i} &= \frac{\partial}{\partial \lambda_i} \sum_{(c,d) \in (C,D)} \log \exp \sum_i \lambda_{ci} f_i(c,d) = \frac{\partial}{\partial \lambda_i} \sum_{(c,d) \in (C,D)} \sum_i \lambda_i f_i(c,d) \\
 &= \sum_{(c,d) \in (C,D)} \frac{\partial \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} f_i(c,d)
 \end{aligned}$$

Derivative of the numerator is: the empirical count(f_i, c)



The Derivative II: Denominator

$$\begin{aligned}
 \frac{\partial M(\lambda)}{\partial \lambda_i} &= \frac{\partial}{\partial \lambda_i} \sum_{(c,d) \in (C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c', d) \\
 &= \sum_{(c,d) \in (C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \frac{\partial \sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c', d)}{1} \frac{\partial \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c', d)}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \frac{\partial \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \sum_{c'} P(c' | d, \lambda) f_i(c', d) \quad = \text{predicted count}(f_i, \lambda)
 \end{aligned}$$



The Derivative III

$$\frac{\partial \log P(C | D, \lambda)}{\partial \lambda_i} = \text{actual count}(f_i, C) - \text{predicted count}(f_i, \lambda)$$

- The optimum parameters are the ones for which each feature's **predicted expectation** equals its **empirical expectation**. The optimum distribution is:
 - Always unique (but parameters may not be unique)
 - Always exists (if feature counts are from actual data).
- These models are also called maximum entropy models because we find the model having maximum entropy and satisfying the constraints: $E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$



Finding the optimal parameters

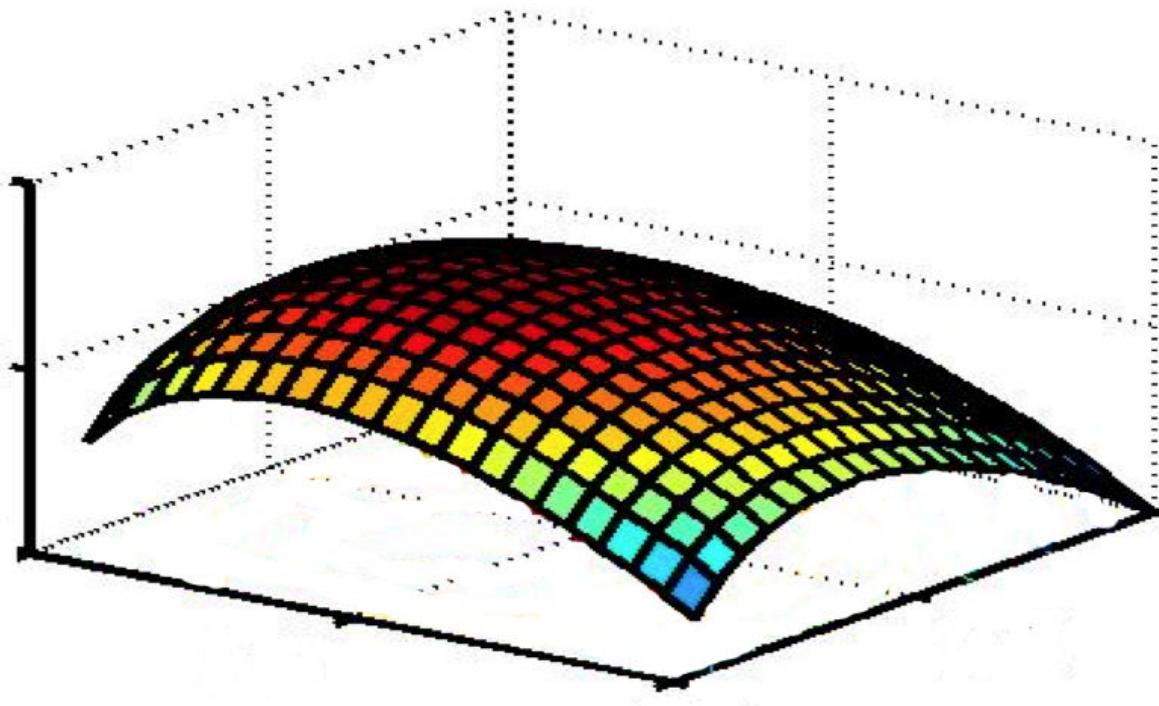
- We want to choose parameters $\lambda_1, \lambda_2, \lambda_3, \dots$ that maximize the conditional log-likelihood of the training data

$$CLogLik(D) = \sum_{i=1}^n \log P(c_i | d_i)$$

- To be able to do that, we've worked out how to calculate the function value and its partial derivatives (its gradient)



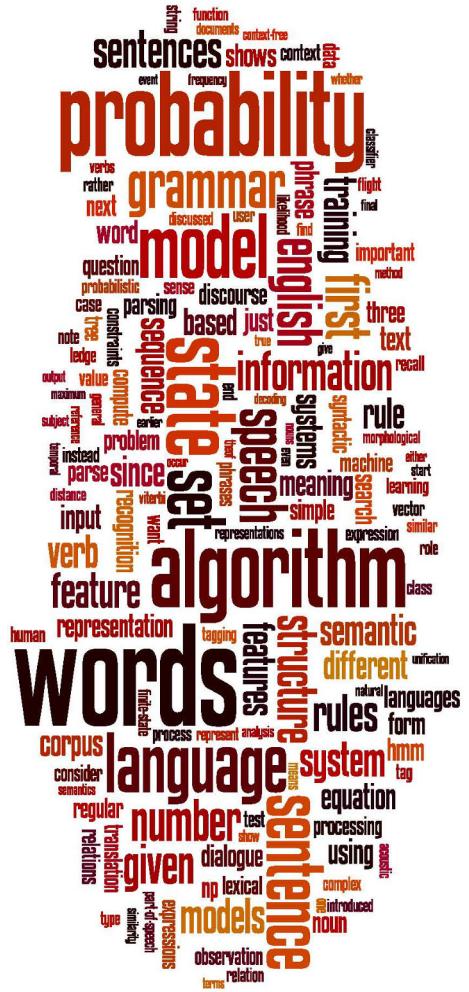
A likelihood surface





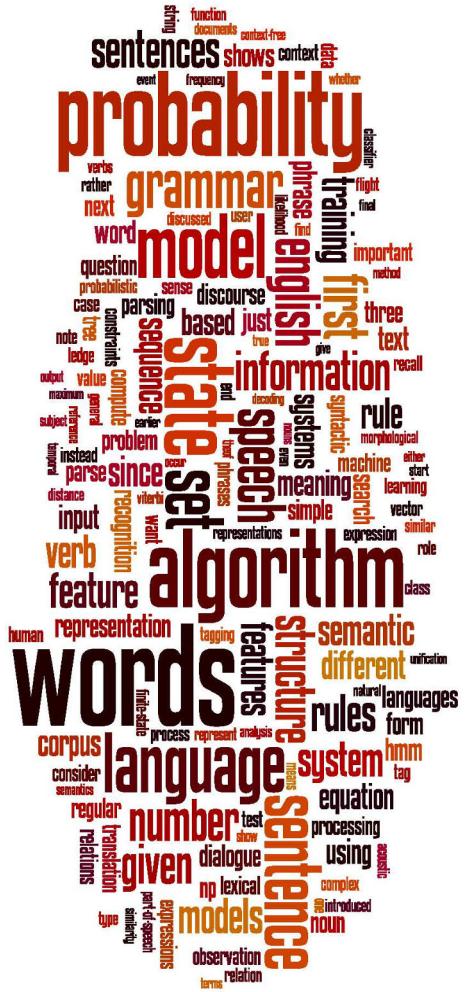
Finding the optimal parameters

- Use your favorite numerical optimization package....
 - Commonly (and in our code), you **minimize** the negative of $CLogLik$
 1. Gradient descent (GD); Stochastic gradient descent (SGD)
 2. Iterative proportional fitting methods: Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS)
 3. Conjugate gradient (CG), perhaps with preconditioning
 4. Quasi-Newton methods – limited memory variable metric (LMVM) methods, in particular, L-BFGS



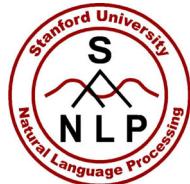
Maxent Models and Discriminative Estimation

Maximizing the likelihood



Information Extraction and Named Entity Recognition

Introducing the tasks:
Getting simple structured
information out of text



Information Extraction

- Information extraction (IE) systems
 - Find and understand limited relevant parts of texts
 - Gather information from many pieces of text
 - Produce a structured representation of relevant information:
 - *relations* (in the database sense), a.k.a.,
 - a *knowledge base*
 - Goals:
 1. Organize information so that it is useful to people
 2. Put information in a semantically precise form that allows further inferences to be made by computer algorithms



Information Extraction (IE)

- IE systems extract clear, factual information
 - Roughly: *Who did what to whom when?*
- E.g.,
 - Gathering earnings, profits, board members, headquarters, etc. from company reports
 - The headquarters of BHP Billiton Limited, and the global headquarters of the combined BHP Billiton Group, are located in Melbourne, Australia.
 - **headquarters("BHP Biliton Limited", "Melbourne, Australia")**
 - Learn drug-gene product interactions from medical research literature



Low-level information extraction

- Is now available – and I think popular – in applications like Apple or Google mail, and web indexing



- Often seems to be based on regular expressions and name lists



Low-level information extraction



Search

About 123,000 results (0.23 seconds)

Everything

Best guess for BHP Billiton Ltd. Headquarters is **Melbourne, London**

Images

Mentioned on at least 9 websites including [wikipedia.org](#), [bhpbilliton.com](#) and [bhpbilliton.com](#) - Feedback

Maps

[**BHP Billiton - Wikipedia, the free encyclopedia**](#)

en.wikipedia.org/wiki/BHP_Billiton

Videos

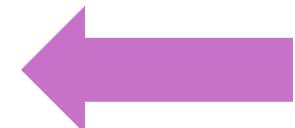
Merger of BHP & Billiton 2001 (creation of a DLC). **Headquarters, Melbourne,**

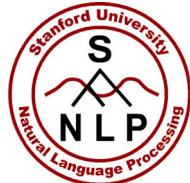
Australia (BHP Billiton Limited and BHP Billiton Group) London, United Kingdom ...

News

[History - Corporate affairs - Operations - Accidents](#)

Shopping





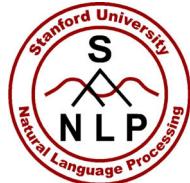
Named Entity Recognition (NER)

- A very important sub-task: **find** and **classify** names in text, for example:
 - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.



Named Entity Recognition (NER)

- A very important sub-task: **find** and **classify** names in text, for example:
 - The decision by the independent MP **Andrew Wilkie** to withdraw his support for the minority **Labor** government sounded dramatic but it should not further threaten its stability. When, after the **2010** election, **Wilkie, Rob Oakeshott, Tony Windsor** and the **Greens** agreed to support **Labor**, they gave just two guarantees: confidence and supply.



Named Entity Recognition (NER)

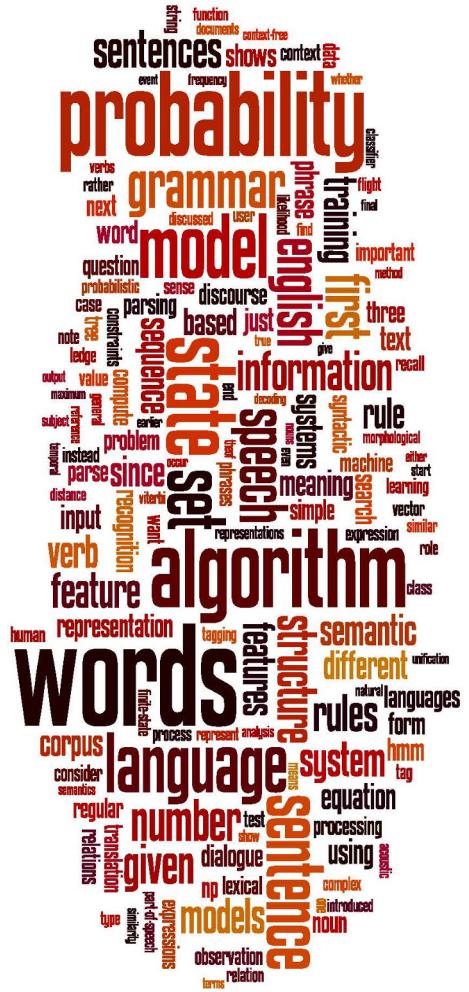
- A very important sub-task: **find** and **classify** names in text, for example:
 - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organization



Named Entity Recognition (NER)

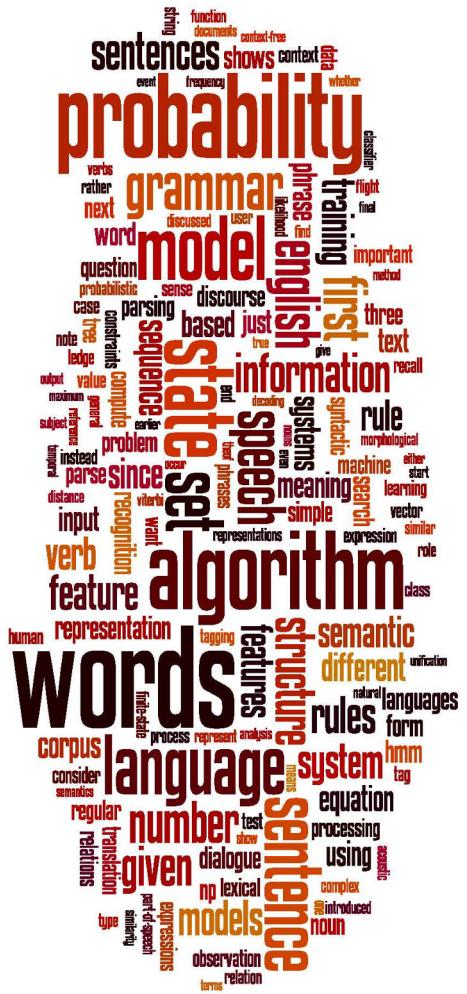
- The uses:
 - Named entities can be indexed, linked off, etc.
 - Sentiment can be attributed to companies or products
 - A lot of IE relations are associations between named entities
 - For question answering, answers are often named entities.
- Concretely:
 - Many web pages tag various entities, with links to bio or topic pages, etc.
 - Reuters' OpenCalais, Evri, AlchemyAPI, Yahoo's Term Extraction, ...
 - Apple/Google/Microsoft/... smart recognizers for document content



Information Extraction and Named Entity Recognition

Introducing the tasks:

Getting simple structured information out of text



Evaluation of Named Entity Recognition

The extension of Precision, Recall, and the F measure to sequences



The Named Entity Recognition Task

Task: Predict entities in a text

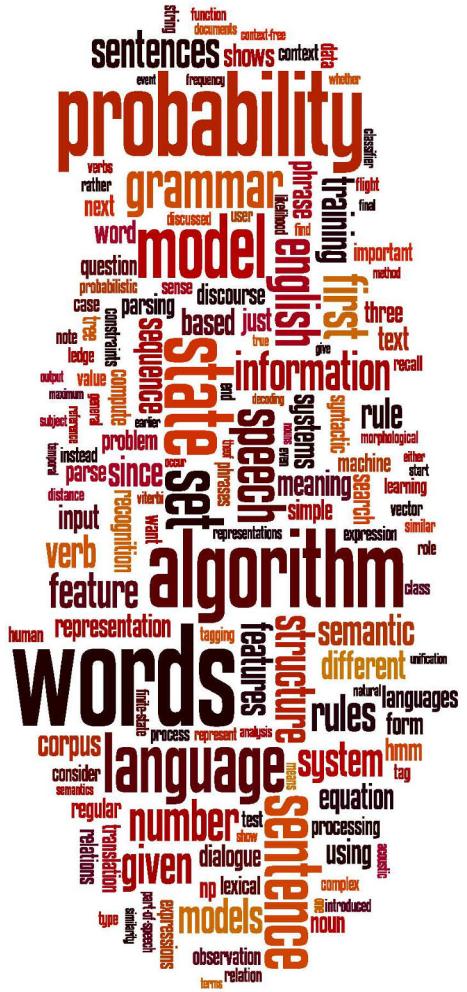
Foreign	ORG
Ministry	ORG
spokesman	O
Shen	PER
Guofang	PER
told	O
Reuters	ORG
:	:

} Standard
evaluation
is per entity,
not per token



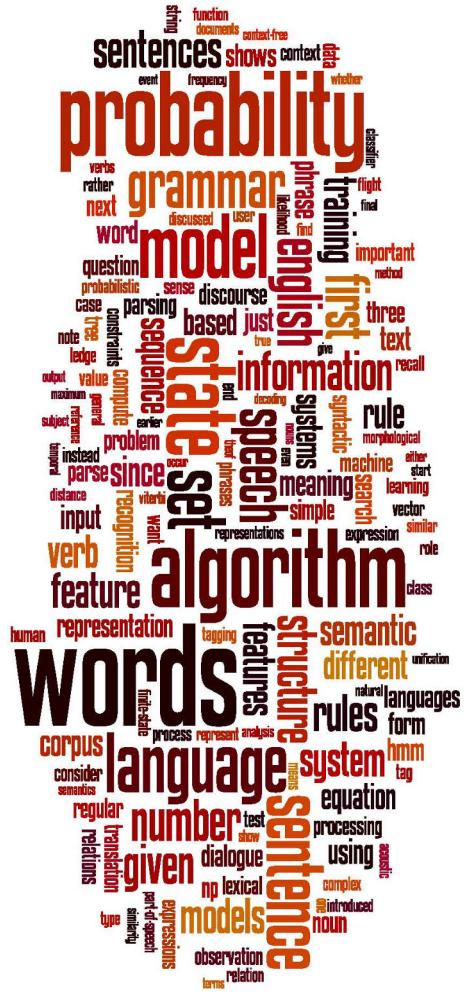
Precision/Recall/F1 for IE/NER

- Recall and precision are straightforward for tasks like IR and text categorization, where there is only one grain size (documents)
- The measure behaves a bit funny for IE/NER when there are *boundary errors* (which are *common*):
 - First Bank of Chicago announced earnings ...
- This counts as both a fp and a fn
- Selecting *nothing* would have been better
- Some other metrics (e.g., MUC scorer) give partial credit (according to complex rules)



Evaluation of Named Entity Recognition

The extension of Precision, Recall, and the F measure to sequences



Sequence Models for Named Entity Recognition



The ML sequence model approach to NER

Training

1. Collect a set of representative training documents
2. Label each token for its entity class or other (O)
3. Design feature extractors appropriate to the text and classes
4. Train a sequence classifier to predict the labels from the data

Testing

1. Receive a set of testing documents
2. Run sequence model inference to label each token
3. Appropriately output the recognized entities



Encoding classes for sequence labeling

	IO encoding	IOB encoding
Fred	PER	B-PER
showed	O	O
Sue	PER	B-PER
Mengqiu	PER	B-PER
Huang	PER	I-PER
's	O	O
new	O	O
painting	O	O



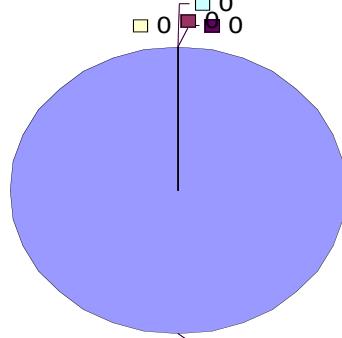
Features for sequence labeling

- Words
 - Current word (essentially like a learned dictionary)
 - Previous/next word (context)
- Other kinds of inferred linguistic classification
 - Part-of-speech tags
- Label context
 - Previous (and perhaps next) label



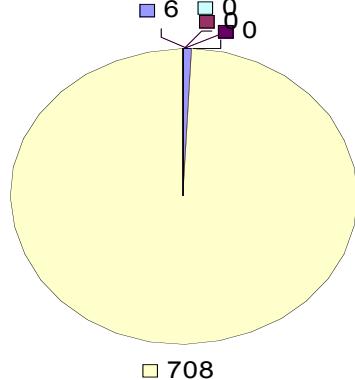
Features: Word substrings

oxa



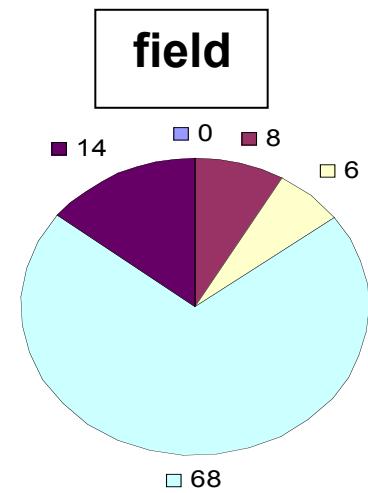
- drug
- company
- movie
- place
- person

:



708

field



Cotrimoxazole

Wethersfield

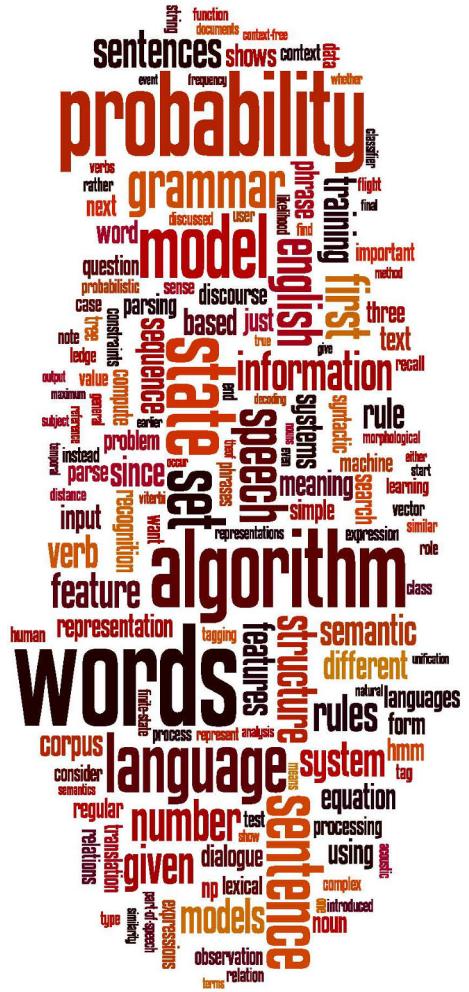
Alien Fury: Countdown to Invasion



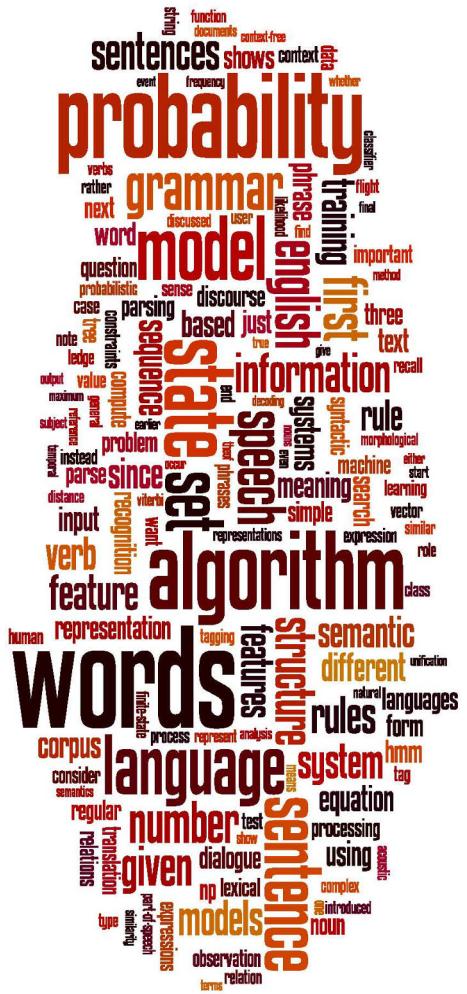
Features: Word shapes

- Word Shapes
 - Map words to simplified representation that encodes attributes such as length, capitalization, numerals, Greek letters, internal punctuation, etc.

Varicella-zoster	Xx-xxx
mRNA	xXXX
CPA1	XXXd



Sequence Models for Named Entity Recognition



Maximum entropy sequence models

Maximum entropy Markov
models (MEMMs) or
Conditional Markov models



Sequence problems

- Many problems in NLP have data which is a sequence of characters, words, phrases, lines, or sentences ...
 - We can think of our task as one of labeling each item

VBG	NN	IN	DT	NN	IN	NN
Chasing	opportunity	in	an	age	of	upheaval

POS tagging

PERS	O	O	O	ORG	ORG
Murdoch	discusses	future	of	News	Corp.

Named entity recognition

而相对于这些品牌的价

Word segmentation

Q
A
Q
A
A
A
A
Q
A

Text segmen- tation



MEMM inference in systems

- For a Conditional Markov Model (CMM) a.k.a. a Maximum Entropy Markov Model (MEMM), the classifier makes a single decision at a time, conditioned on evidence from observations *and previous decisions*
- A larger space of sequences is usually explored via search

Local Context					Decision Point
-3	-2	-1	0	+1	
DT	NNP	VBD	???	???	
The	Dow	fell	22.6	%	

(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)

Features

w_0	22.6
w_{+1}	%
w_{-1}	fell
T_{-1}	VBD
$T_{-1}-T_{-2}$	NNP-VBD
hasDigit?	true
...	...



Example: POS Tagging

- Scoring individual labeling decisions is no more complex than standard classification decisions
 - We have some assumed labels to use for prior positions
 - We use features of those and the observed data (which can include current, previous, and next words) to predict the current label

Local Context					Decision Point
-3	-2	-1	0	+1	
DT	NNP	VBD	???	???	
The	Dow	fell	22.6	%	

Features

w_0	22.6
w_{+1}	%
w_{-1}	fell
T_{-1}	VBD
$T_{-1}-T_{-2}$	NNP-VBD
hasDigit?	true
...	...

(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)



Example: POS Tagging

- POS tagging Features can include:
 - Current, previous, next words in isolation or together.
 - Previous one, two, three tags.
 - Word-internal features: word types, suffixes, dashes, etc.

Local Context					Decision Point
-3	-2	-1	0	+1	
DT	NNP	VBD	???	???	
The	Dow	fell	22.6	%	

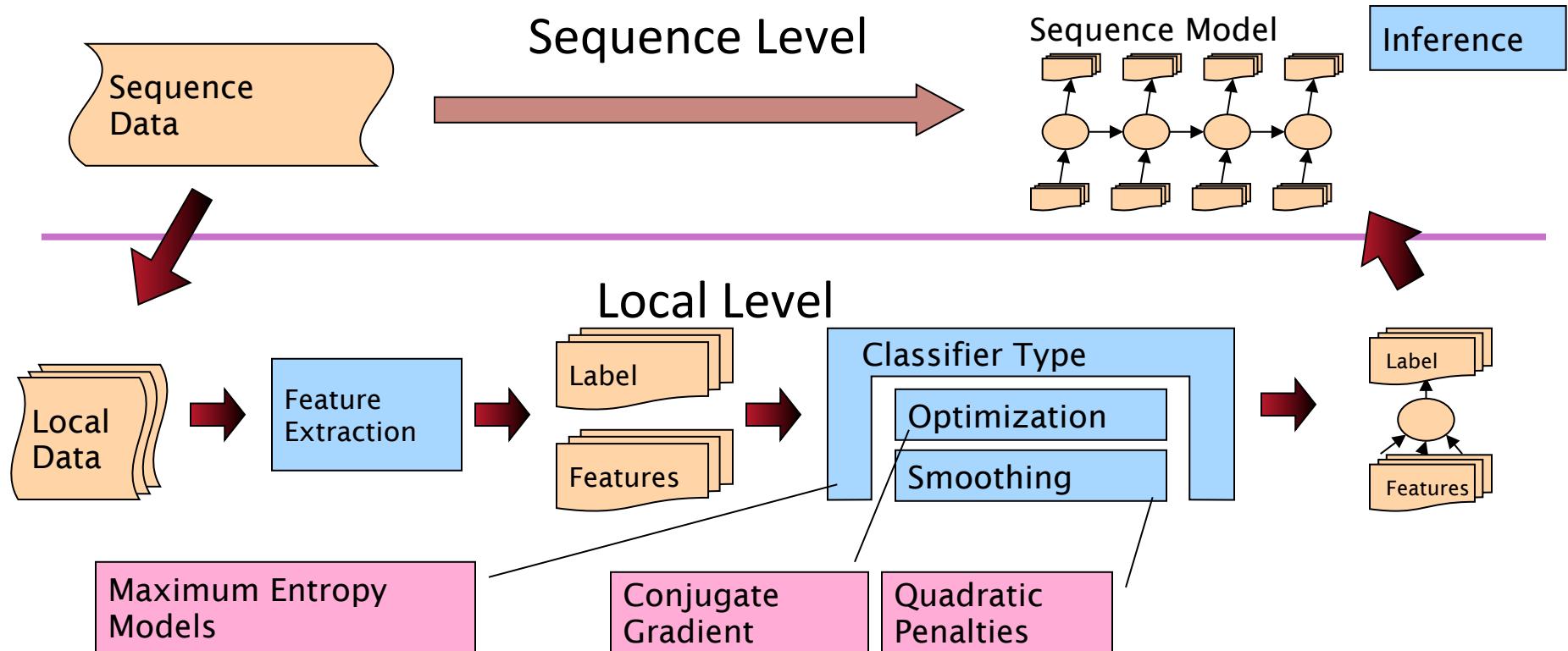
(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)

Features

w_0	22.6
w_{+1}	%
w_{-1}	fell
T_{-1}	VBD
$T_{-1}-T_{-2}$	NNP-VBD
hasDigit?	true
...	...

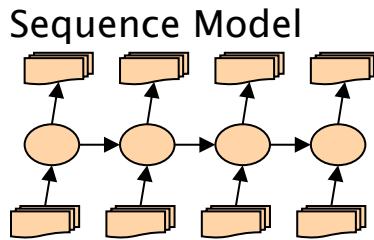


Inference in Systems

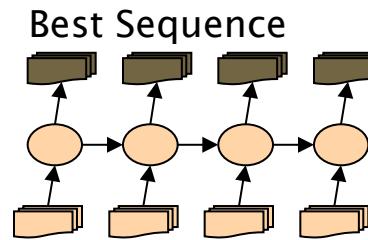




Greedy Inference



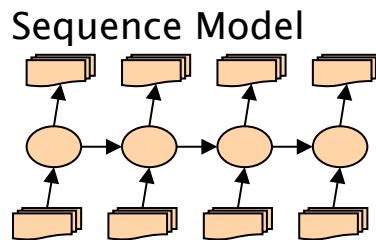
Inference



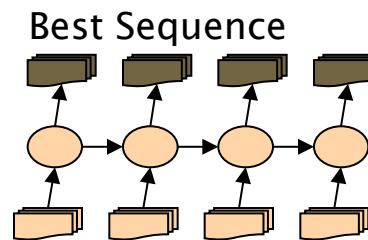
- Greedy inference:
 - We just start at the left, and use our classifier at each position to assign a label
 - The classifier can depend on previous labeling decisions as well as observed data
- Advantages:
 - Fast, no extra memory requirements
 - Very easy to implement
 - With rich features including observations to the right, it may perform quite well
- Disadvantage:
 - Greedy. We make commit errors we cannot recover from



Beam Inference



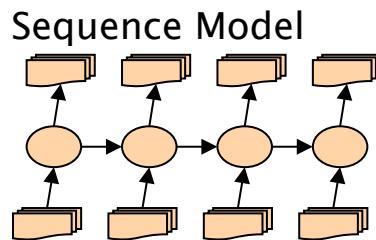
Inference



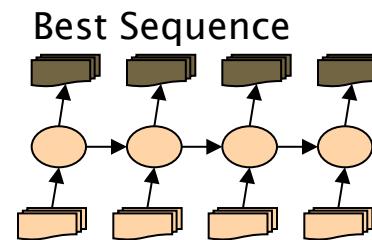
- Beam inference:
 - At each position keep the top k complete sequences.
 - Extend each sequence in each local way.
 - The extensions compete for the k slots at the next position.
- Advantages:
 - Fast; beam sizes of 3-5 are almost as good as exact inference in many cases.
 - Easy to implement (no dynamic programming required).
- Disadvantage:
 - Inexact: the globally best sequence can fall off the beam.



Viterbi Inference



Inference →



- Viterbi inference:
 - Dynamic programming or memoization.
 - Requires small window of state influence (e.g., past two states are relevant).
- Advantage:
 - Exact: the global best sequence is returned.
- Disadvantage:
 - Harder to implement long-distance state-state interactions (but beam inference tends not to allow long-distance resurrection of sequences anyway).

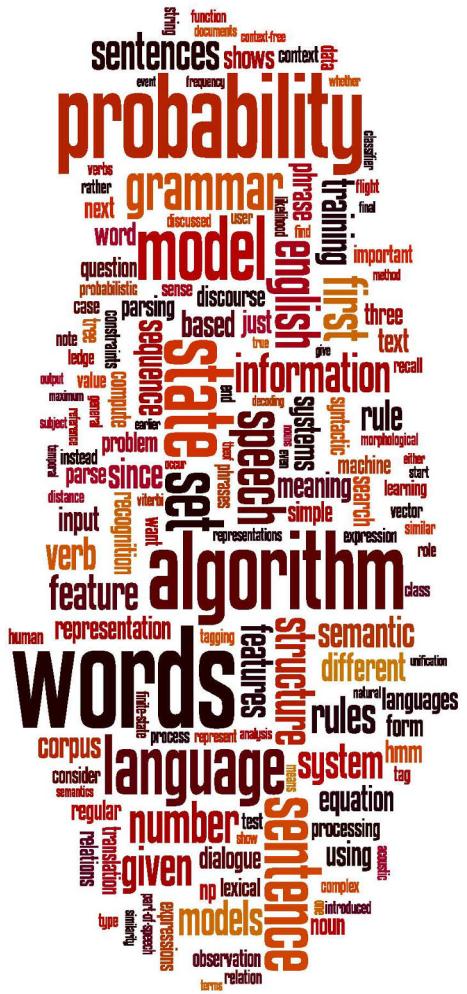


CRFs [Lafferty, Pereira, and McCallum 2001]

- Another sequence model: Conditional Random Fields (CRFs)
- A whole-sequence conditional model rather than a chaining of local models.

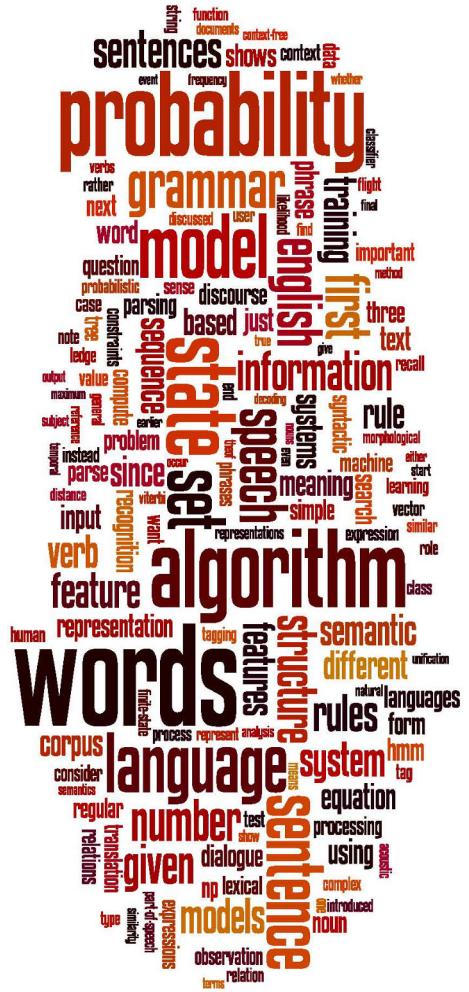
$$P(c | d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$

- The space of c 's is now the space of sequences
 - But if the features f_i remain local, the conditional sequence likelihood can be calculated exactly using dynamic programming
- Training is slower, but CRFs avoid causal-competition biases
- These (or a variant using a max margin criterion) are seen as the state-of-the-art these days ... but in practice usually work much the same as MEMMs.



Maximum entropy sequence models

Maximum entropy Markov
models (MEMMs) or
Conditional Markov models



Relation Extraction

What is relation extraction?



Extracting relations from text

- **Company report:** “International Business Machines Corporation (IBM or the company) was incorporated in the State of New York on June 16, 1911, as the Computing-Tabulating-Recording Co. (C-T-R)...”
- Extracted Complex Relation:

Company-Founding

Company	IBM
Location	New York
Date	June 16, 1911
Original-Name	Computing-Tabulating-Recording Co.

- But we will focus on the simpler task of extracting relation **triples**

Founding-year(IBM,1911)

Founding-location(IBM,New York)

Dan Jurafsky



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

Languages
العربية
Azerbaijani
Беларуская
Беларуская (тарашкевіца)

Extracting Relation Triples from Text

Stanford University

From Wikipedia, the free encyclopedia

"Stanford" redirects here. For other uses, see [Stanford \(disambiguation\)](#).

Not to be confused with [Stanford University \(disambiguation\)](#).

The Leland Stanford Junior University, commonly referred to as **Stanford University** or **Stanford**, is an American private research university located in Stanford, California on an 8,180-acre (3,310 ha) campus near Palo Alto, California, United States. It is situated in the northwestern Santa Clara Valley on the San Francisco Peninsula, approximately 20 miles (32 km) northwest of San Jose and 37 miles (60 km) southeast of San Francisco.^[6]

Leland Stanford, a Californian railroad tycoon and politician, founded the university in 1891 in honor of his son, Leland Stanford, Jr., who died of typhoid two months before his 16th birthday. The university was established as a coeducational and nondenominational institution, but struggled financially after the senior Stanford's 1893 death and after much of the campus was damaged by the 1906 San Francisco earthquake. Following World War II, Provost Frederick Terman supported faculty and graduates' entrepreneurialism to build a self-sufficient local industry in what would become known as Silicon Valley. By 1970, Stanford was home to a linear accelerator, was one of the original four ARPANET nodes, and had transformed itself into a major research university in computer science, mathematics, natural sciences, and social sciences. More than 50 Stanford faculty, staff, and alumni have won the Nobel Prize and Stanford has the largest number of Turing award winners for a single institution. Stanford faculty and alumni have founded many prominent technology companies including Cisco Systems, Google, Hewlett-Packard, LinkedIn, Rambus, Silicon Graphics, Sun Microsystems, Varian Associates, and Yahoo!^[7]

The university is organized into seven schools including academic schools of Humanities,



Stanford EQ Leland Stanford Junior University
Stanford LOC IN California
Stanford IS-A research university
Stanford LOC NEAR Palo Alto
Stanford FOUNDED IN 1891
Stanford FOUNDER Leland Stanford

Junior University,
also known as Stanford
University, is an American
university located in
... near Palo Alto,
Stanford...founded
91





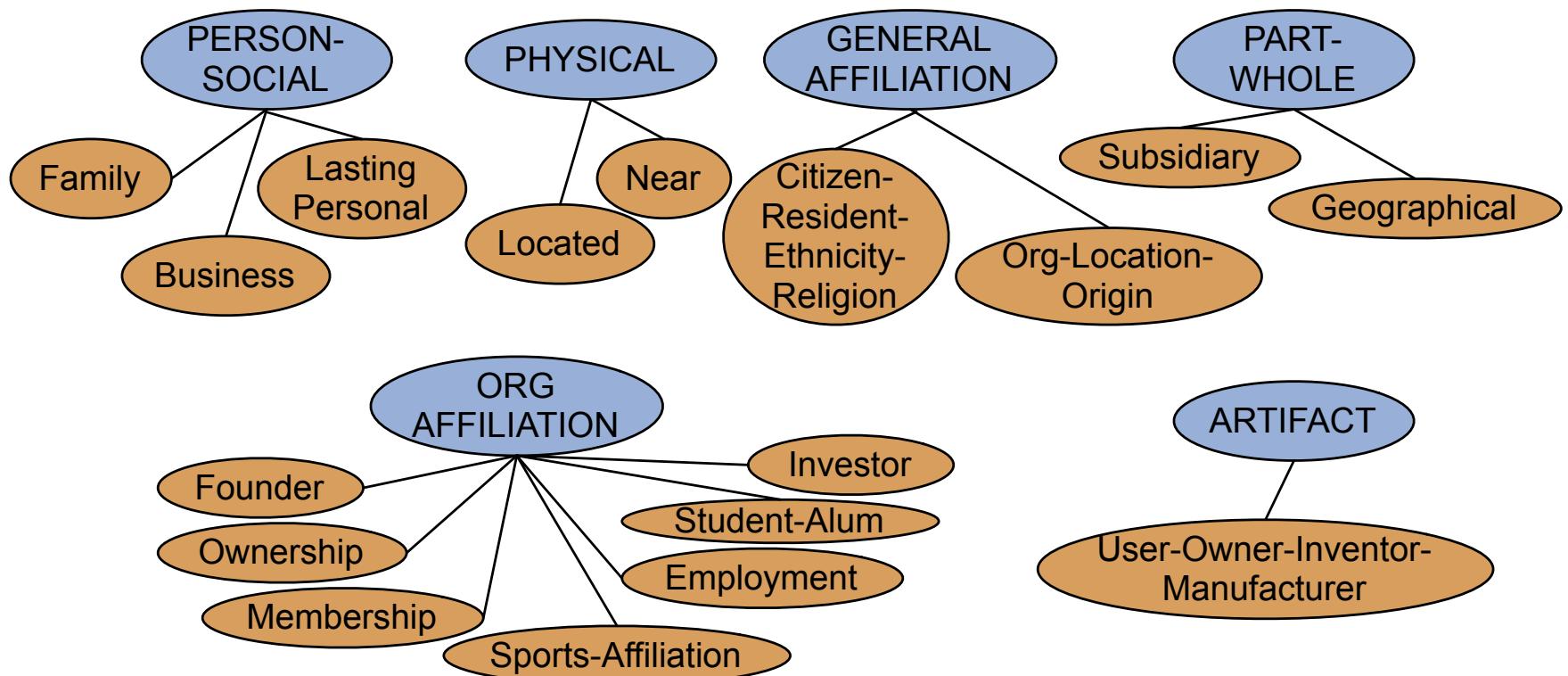
Why Relation Extraction?

- Create new structured knowledge bases, useful for any app
- Augment current knowledge bases
 - Adding words to WordNet thesaurus, facts to FreeBase or DBpedia
- Support question answering
 - [The granddaughter of which actor starred in the movie “E.T.”?](#)
(acted-in ?x “E.T.”)(is-a ?y actor)(granddaughter-of ?x ?y)
- But which relations should we extract?



Automated Content Extraction (ACE)

17 relations from 2008 “Relation Extraction Task”





Automated Content Extraction (ACE)

- Physical-Located PER-GPE
He was in Tennessee
- Part-Whole-Subsidiary ORG-ORG
XYZ, the parent company of ABC
- Person-Social-Family PER-PER
John's wife Yoko
- Org-AFF-Founder PER-ORG
Steve Jobs, co-founder of Apple...



UMLS: Unified Medical Language System

- 134 entity types, 54 relations

Injury	disrupts	Physiological Function
Bodily Location	location-of	Biologic Function
Anatomical Structure	part-of	Organism
Pharmacologic Substance	causes	Pathological Function
Pharmacologic Substance	treats	Pathologic Function



Extracting UMLS relations from a sentence

Doppler echocardiography can be used to diagnose left anterior descending artery stenosis in patients with type 2 diabetes



Echocardiography, Doppler **DIAGNOSES** Acquired stenosis



```

{{Infobox university
|image_name= Stanford University seal.svg
|image_size= 210px
|caption = Seal of Stanford University
|name =Stanford University
|native_name =Leland Stanford Junior Uni
|motto = {{lang|de|"Die Luft der Freiheit weht
name="casper">{{cite speech|title=Die Lu
Casper|first=Gerhard|last=Casper|author
05|url=http://www.stanford.edu/dept/pr
|mottoeng = The wind of freedom blows<
|established = 1891<ref>{{cite web |
url=http://www.stanford.edu/home/stan
publisher = Stanford University | accessdate=
|type = [[private university|Private]]
|calendar= Quarter
|president = [[John L. Hennessy]]
|provost = [[John Etchemendy]]
|city = [[Stanford, California|Stanford]]
|state = California
|country = U.S.

```

Databases of Wikipedia Relations

Wikipedia Infobox

Relations extracted from Infobox		
Type	Private	Stanford state California
Endowment	US\$ 16.5 billion (2011) ^[3]	Stanford motto "Die Luft der Freiheit weht"
President	John L. Hennessy	
Provost	John Etchemendy	
Academic staff	1,910 ^[4]	
Students	15,319	
Undergraduates	6,878 ^[5]	
Postgraduates	8,441 ^[5]	
Location	Stanford, California, U.S.	
Campus	Suburban, 8,180 acres (3,310 ha) ^[6]	
Colors	Cardinal red and white	



Relation databases that draw from Wikipedia

- Resource Description Framework (RDF) triples
subject predicate object

Golden Gate Park `location` San Francisco

dbpedia:Golden_Gate_Park dbpedia-owl:location dbpedia:San_Francisco

- DBpedia: 1 billion RDF triples, 385 from English Wikipedia
- Frequent Freebase relations:

people/person/nationality,
people/person/profession,
biology/organism_higher_classification

location/location/contains
people/person/place-of-birth
film/film/genre



Ontological relations

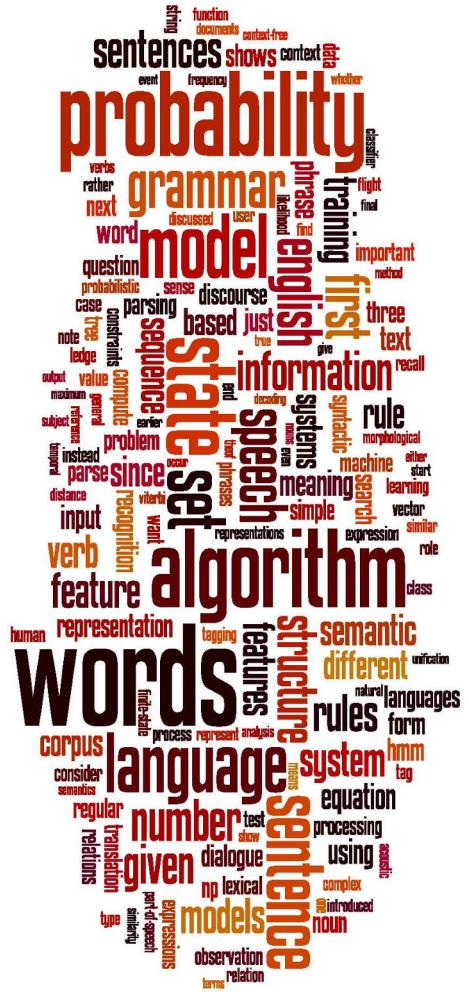
Examples from the WordNet Thesaurus

- IS-A (hypernym): subsumption between classes
 - Giraffe IS-A ruminant IS-A ungulate IS-A mammal IS-A vertebrate IS-A animal...
- Instance-of: relation between individual and class
 - San Francisco instance-of city



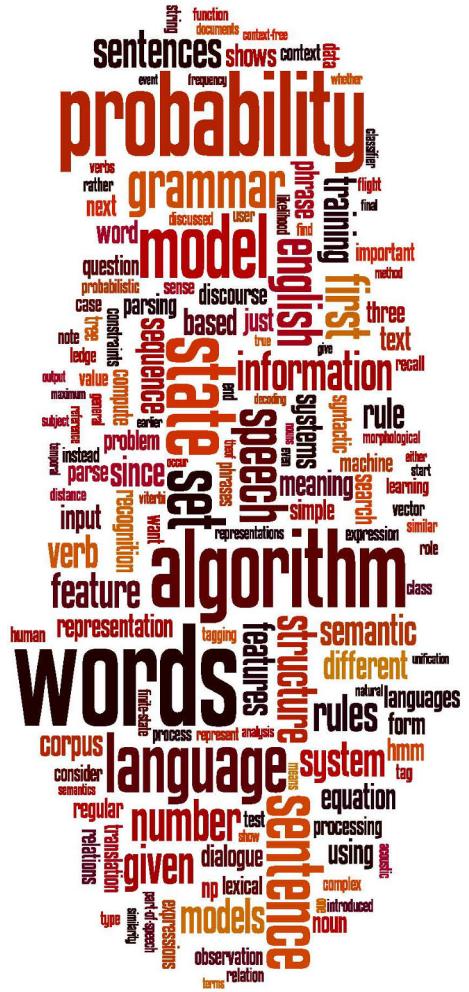
How to build relation extractors

1. Hand-written patterns
2. Supervised machine learning
3. Semi-supervised and unsupervised
 - Bootstrapping (using seeds)
 - Distant supervision
 - Unsupervised learning from the web



Relation Extraction

What is relation extraction?



Relation Extraction

Using patterns to extract relations



Rules for extracting IS-A relation

Early intuition from Hearst (1992)

- “Agar is a substance prepared from a mixture of red algae, such as *Gelidium*, for laboratory or industrial use”
- What does *Gelidium* mean?
- How do you know?`



Rules for extracting IS-A relation

Early intuition from Hearst (1992)

- “Agar is a substance prepared from a mixture of red algae, such as **Gelidium**, for laboratory or industrial use”
- What does *Gelidium* mean?
- How do you know?`



Hearst's Patterns for extracting IS-A relations

(Hearst, 1992): Automatic Acquisition of Hyponyms

"Y such as X ((, X)* (, and|or) X)"

"such Y as X"

"X or other Y"

"X and other Y"

"Y including X"

"Y, especially X"



Hearst's Patterns for extracting IS-A relations

Hearst pattern	Example occurrences
X and other Y	...temples, treasuries, and other important civic buildings.
X or other Y	Bruises, wounds, broken bones or other injuries...
Y such as X	The bow lute, such as the Bambara ndang...
Such Y as X	... such authors as Herrick, Goldsmith, and Shakespeare.
Y including X	...common-law countries, including Canada and England...
Y , especially X	European countries, especially France, England, and Spain...



Extracting Richer Relations Using Rules

- Intuition: relations often hold between specific entities
 - **located-in** (ORGANIZATION, LOCATION)
 - **founded** (PERSON, ORGANIZATION)
 - **cures** (DRUG, DISEASE)
- Start with Named Entity tags to help extract relation!



Named Entities aren't quite enough. Which relations hold between 2 entities?



Drug

Cure?

Prevent?

Cause?



Disease



What relations hold between 2 entities?



PERSON

Founder?

Investor?

Member?

Employee?

President?



ORGANIZATION



Extracting Richer Relations Using Rules and Named Entities

Who holds what office in what organization?

PERSON, POSITION of ORG

- George Marshall, Secretary of State of the United States

PERSON (named | appointed | chose | etc.) PERSON Prep? POSITION

- Truman appointed Marshall Secretary of State

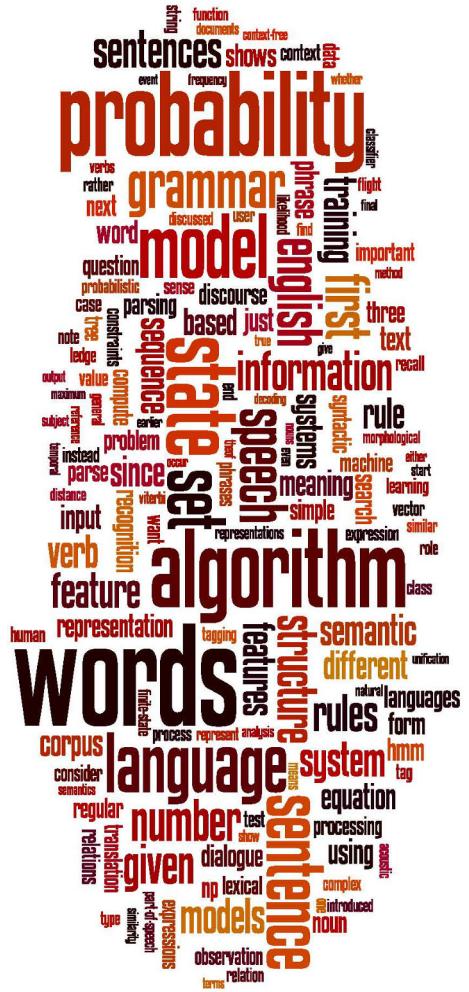
PERSON [be]? (named | appointed | etc.) Prep? ORG POSITION

- George Marshall was named US Secretary of State



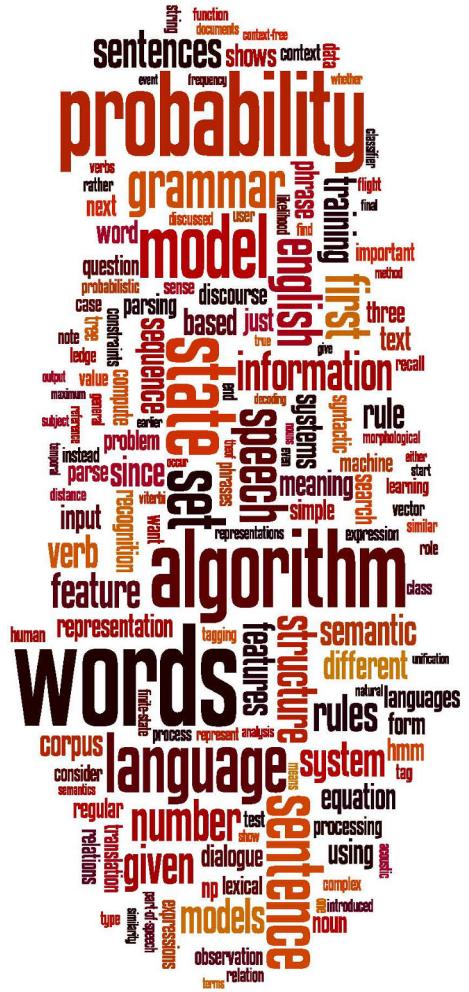
Hand-built patterns for relations

- Plus:
 - Human patterns tend to be high-precision
 - Can be tailored to specific domains
- Minus
 - Human patterns are often low-recall
 - A lot of work to think of all possible patterns!
 - Don't want to have to do this for every relation!
 - We'd like better accuracy



Relation Extraction

Using patterns to extract relations



Relation Extraction

Supervised relation extraction



Supervised machine learning for relations

- Choose a set of relations we'd like to extract
- Choose a set of relevant named entities
- Find and label data
 - Choose a representative corpus
 - Label the named entities in the corpus
 - Hand-label the relations between these entities
 - Break into training, development, and test
- Train a classifier on the training set



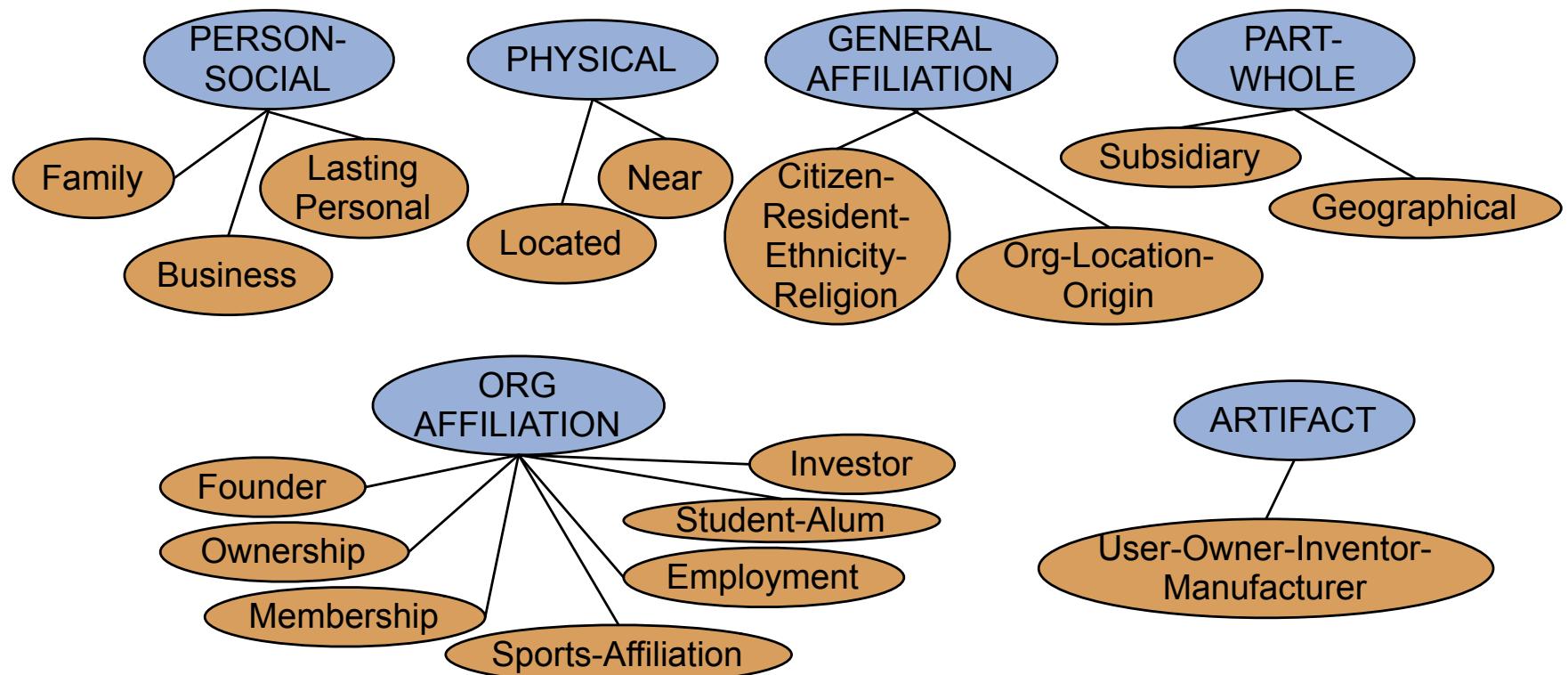
How to do classification in supervised relation extraction

1. Find all pairs of named entities (usually in same sentence)
2. Decide if 2 entities are related
3. If yes, classify the relation
 - Why the extra step?
 - Faster classification training by eliminating most pairs
 - Can use distinct feature-sets appropriate for each task.



Automated Content Extraction (ACE)

17 sub-relations of 6 relations from 2008 “Relation Extraction Task”



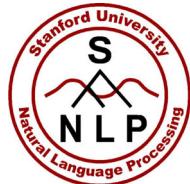


Relation Extraction

Classify the relation between two entities in a sentence

American Airlines, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said.





Word Features for Relation Extraction

American Airlines, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said
Mention 1
Mention 2

- Headwords of M1 and M2, and combination
Airlines Wagner Airlines-Wagner
 - Bag of words and bigrams in M1 and M2
{American, Airlines, Tim, Wagner, American Airlines, Tim Wagner}
 - Words or bigrams in particular positions left and right of M1/M2
M2: -1 *spokesman*
M2: +1 *said*
 - Bag of words or bigrams between the two entities
{a, AMR, of, immediately, matched, move, spokesman, the, unit}



Named Entity Type and Mention Level Features for Relation Extraction

American Airlines, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said

- Named-entity types
 - M1: ORG
 - M2: PERSON
 - Concatenation of the two named-entity types
 - ORG-PERSON
 - Entity Level of M1 and M2 (NAME, NOMINAL, PRONOUN)
 - M1: NAME [it or he would be PRONOUN]
 - M2: NAME [the company would be NOMINAL]



Parse Features for Relation Extraction

American Airlines, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said

- Base syntactic chunk sequence from one to the other
NP NP PP VP NP NP
 - Constituent path through the tree from one to the other
NP ↑ NP ↑ S ↑ S ↓ NP
 - Dependency path

Airlines matched Wagner said



Gazetteer and trigger word features for relation extraction

- Trigger list for family: kinship terms
 - [parent](#), [wife](#), [husband](#), [grandparent](#), etc. [from WordNet]
- Gazetteer:
 - Lists of useful geo or geopolitical words
 - Country name list
 - Other sub-entities



American Airlines, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said.

Entity-based features

Entity ₁ type	ORG
Entity ₁ head	<i>airlines</i>
Entity ₂ type	PERS
Entity ₂ head	<i>Wagner</i>
Concatenated types	ORGPERS

Word-based features

Between-entity bag of words	{ <i>a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman</i> }
Word(s) before Entity ₁	NONE
Word(s) after Entity ₂	<i>said</i>

Syntactic features

Constituent path	$NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
Base syntactic chunk path	$NP \rightarrow NP \rightarrow PP \rightarrow NP \rightarrow VP \rightarrow NP \rightarrow NP$
Typed-dependency path	<i>Airlines</i> \leftarrow_{subj} <i>matched</i> \leftarrow_{comp} <i>said</i> \rightarrow_{subj} <i>Wagner</i>



Classifiers for supervised methods

- Now you can use any classifier you like
 - MaxEnt
 - Naïve Bayes
 - SVM
 - ...
- Train it on the training set, tune on the dev set, test on the test set



Evaluation of Supervised Relation Extraction

- Compute P/R/ F_1 for each relation

$$P = \frac{\text{\# of correctly extracted relations}}{\text{Total \# of extracted relations}}$$

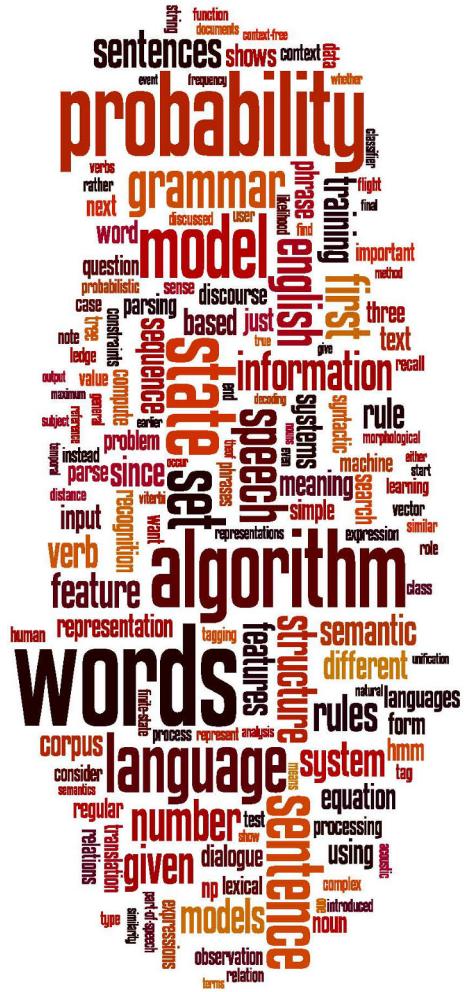
$$F_1 = \frac{2PR}{P + R}$$

$$R = \frac{\text{\# of correctly extracted relations}}{\text{Total \# of gold relations}}$$



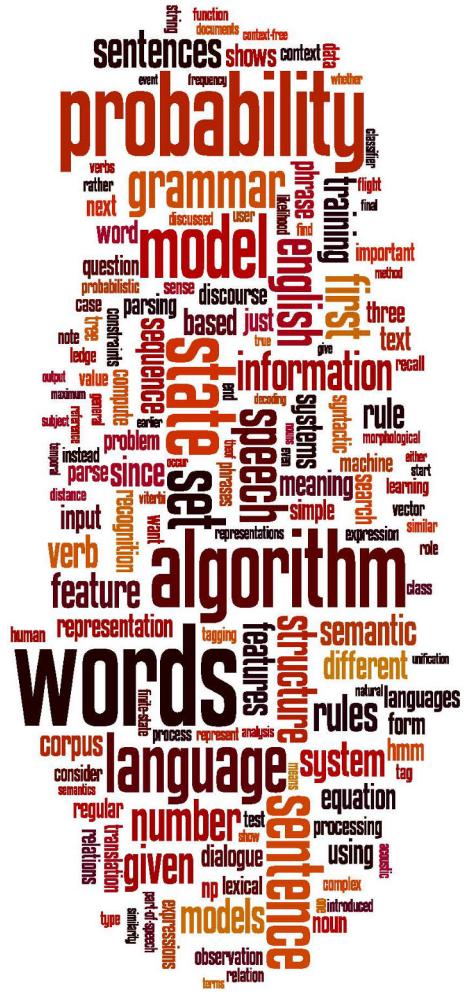
Summary: Supervised Relation Extraction

- + Can get high accuracies with enough hand-labeled training data, if test similar enough to training
- Labeling a large training set is expensive
- Supervised models are brittle, don't generalize well to different genres



Relation Extraction

Supervised relation extraction



Relation Extraction

Semi-supervised and unsupervised relation extraction



Seed-based or bootstrapping approaches to relation extraction

- No training set? Maybe you have:
 - A few seed tuples or
 - A few high-precision patterns
- Can you use those seeds to do something useful?
 - Bootstrapping: use the seeds to directly learn to populate a relation



Relation Bootstrapping (Hearst 1992)

- Gather a set of seed pairs that have relation R
- Iterate:
 1. Find sentences with these pairs
 2. Look at the context between or around the pair and generalize the context to create patterns
 3. Use the patterns for grep for more pairs



Bootstrapping

- <Mark Twain, Elmira> **Seed tuple**
 - Grep (google) for the environments of the seed tuple
 - “Mark Twain is buried in Elmira, NY.”
 - X is buried in Y
 - “The grave of Mark Twain is in Elmira”
 - The grave of X is in Y
 - “Elmira is Mark Twain’s final resting place”
 - Y is X’s final resting place.
 - Use those patterns to grep for new tuples
 - Iterate



Dipre: Extract <author,book> pairs

Brin, Sergei. 1998. Extracting Patterns and Relations from the World Wide Web.

- Start with 5 seeds:

Author	Book
Isaac Asimov	The Robots of Dawn
David Brin	Startide Rising
James Gleick	Chaos: Making a New Science
Charles Dickens	Great Expectations
William Shakespeare	The Comedy of Errors

- Find Instances:

The Comedy of Errors, by William Shakespeare, was

The Comedy of Errors, by William Shakespeare, is

The Comedy of Errors, one of William Shakespeare's earliest attempts

The Comedy of Errors, one of William Shakespeare's most

- Extract patterns (group by middle, take longest common prefix/suffix)

?x , by ?y ,

?x , one of ?y 's

- Now iterate, finding new seeds that match the pattern



Snowball

E. Agichtein and L. Gravano 2000. Snowball: Extracting Relations from Large Plain-Text Collections. ICDL

- Similar iterative algorithm
- Group instances w/similar prefix, middle, suffix, extract patterns
 - But require that X and Y be named entities
 - And compute a confidence for each pattern

Organization	Location of Headquarters
Microsoft	Redmond
Exxon	Irving
IBM	Armonk

.69 ORGANIZATION { 's, in, headquarters } LOCATION

.75 LOCATION { in, based } ORGANIZATION



Distant Supervision

Snow, Jurafsky, Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. NIPS 17

Fei Wu and Daniel S. Weld. 2007. Autonomously Semantifying Wikipedia. CIKM 2007

Mintz, Bills, Snow, Jurafsky. 2009. Distant supervision for relation extraction without labeled data. ACL09

- Combine bootstrapping with supervised learning
 - Instead of 5 seeds,
 - Use a large database to get huge # of seed examples
 - Create lots of features from all these examples
 - Combine in a supervised classifier



Distant supervision paradigm

- Like supervised classification:
 - Uses a classifier with lots of features
 - Supervised by detailed hand-created knowledge
 - Doesn't require iteratively expanding patterns
- Like unsupervised classification:
 - Uses very large amounts of unlabeled data
 - Not sensitive to genre issues in training corpus



Distantly supervised learning of relation extraction patterns

- 1 For each relation Born-In
- 2 For each tuple in big database <Edwin Hubble, Marshfield>
<Albert Einstein, Ulm>
- 3 Find sentences in large corpus with both entities Hubble was born in Marshfield
Einstein, born (1879), Ulm
Hubble's birthplace in Marshfield
- 4 Extract frequent features (parse, words, etc) PER was born in LOC
PER, born (XXXX), LOC
PER's birthplace in LOC
- 5 Train supervised classifier using thousands of patterns P(born-in | f₁, f₂, f₃, ..., f₇₀₀₀₀)



Unsupervised relation extraction

M. Banko, M. Cararella, S. Soderland, M. Broadhead, and O. Etzioni.
2007. Open information extraction from the web. IJCAI

- Open Information Extraction:
 - extract relations from the web with no training data, no list of relations
1. Use parsed data to train a “trustworthy tuple” classifier
 2. Single-pass extract all relations between NPs, keep if trustworthy
 3. Assessor ranks relations based on text redundancy
 - (FCI, specializes in, software development)
 - (Tesla, invented, coil transformer)



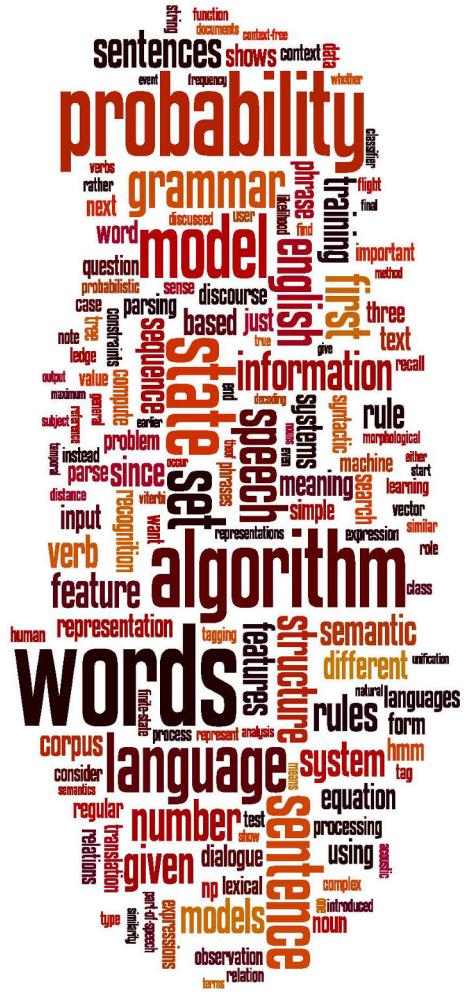
Evaluation of Semi-supervised and Unsupervised Relation Extraction

- Since it extracts totally new relations from the web
 - There is no gold set of correct instances of relations!
 - Can't compute precision (don't know which ones are correct)
 - Can't compute recall (don't know which ones were missed)
 - Instead, we can approximate precision (only)
 - Draw a random sample of relations from output, check precision manually
$$\hat{P} = \frac{\text{\# of correctly extracted relations in the sample}}{\text{Total \# of extracted relations in the sample}}$$
 - Can also compute precision at different levels of recall.
 - Precision for top 1000 new relations, top 10,000 new relations, top 100,000
 - In each case taking a random sample of that set
- 49 But no way to evaluate recall



Relation Extraction

Semi-supervised and unsupervised relation extraction



Maxent Models and Discriminative Estimation

The maximum entropy model presentation



Maximum Entropy Models

- An equivalent approach:
 - Lots of distributions out there, most of them very spiked, specific, overfit.
 - We want a distribution which is uniform except in specific ways we require.
 - Uniformity means **high entropy** – we can search for distributions which have properties we desire, but also have high entropy.

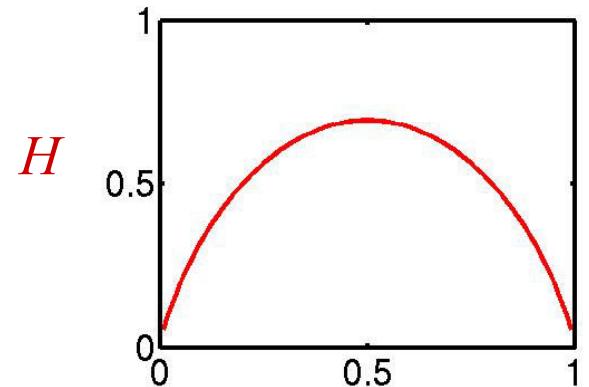
Ignorance is preferable to error and he is less remote from the truth who believes nothing than he who believes what is wrong – Thomas Jefferson (1781)



(Maximum) Entropy

- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty (“surprise”):
 - Event x
 - Probability p_x
 - “Surprise” $\log(1/p_x)$
- Entropy: expected surprise (over p):

$$H(p) = E_p \left[\log_2 \frac{1}{p_x} \right] = - \sum_x p_x \log_2 p_x$$



A coin-flip is most uncertain for a fair coin.

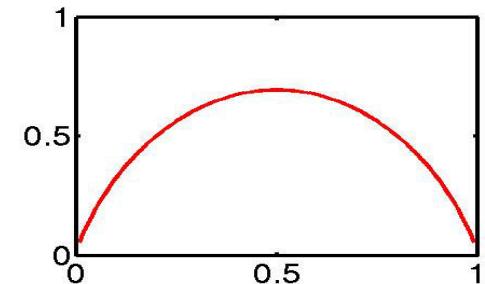


Maxent Examples I

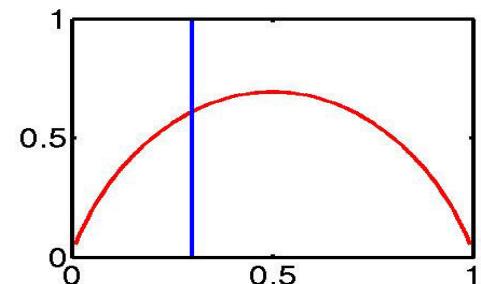
- What do we want from a distribution?
 - Minimize commitment = maximize entropy.
 - Resemble some reference distribution (data).
- Solution: maximize entropy H , subject to feature-based constraints:

$$E_p[f_i] = E_{\hat{p}}[f_i] \quad \leftrightarrow \quad \sum_{x \in f_i} p_x = C_i$$

- Adding constraints (features):
 - Lowers maximum entropy
 - Raises maximum likelihood of data
 - Brings the distribution further from uniform
 - Brings the distribution closer to data



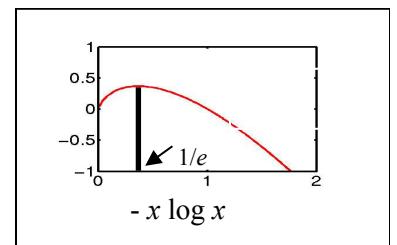
Unconstrained,
max at 0.5



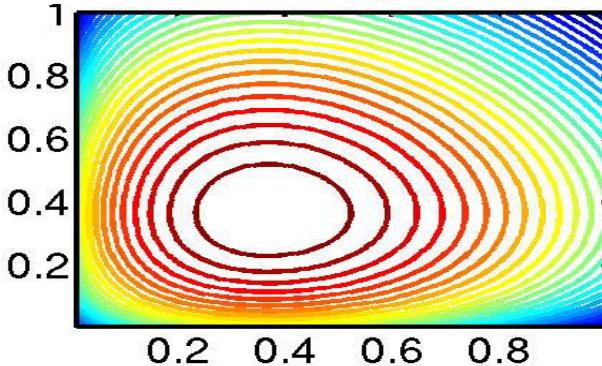
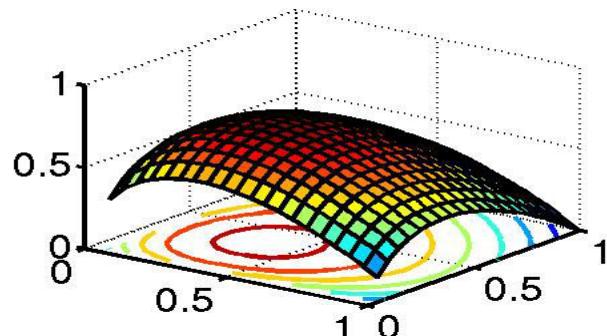
Constraint that
 $p_{\text{HEADS}} = 0.3$



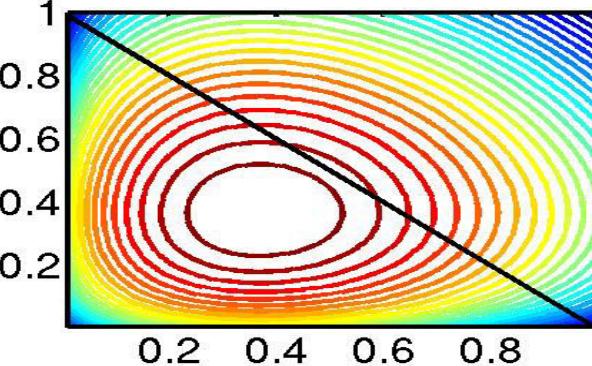
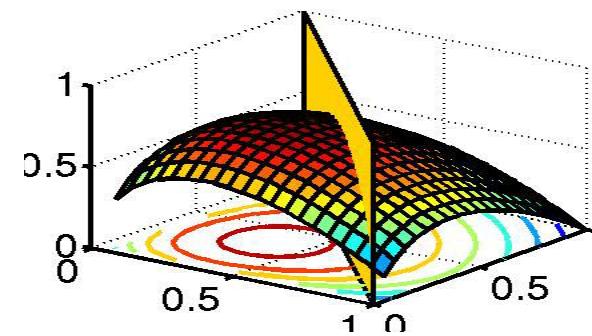
Maxent Examples II



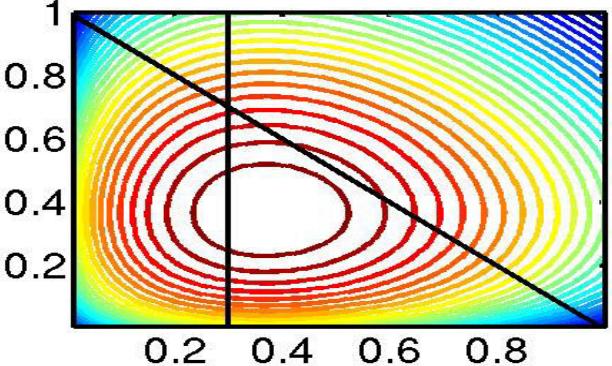
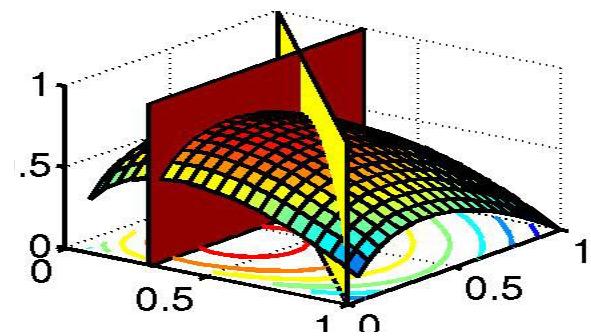
$H(p_H p_T,)$



$p_H + p_T = 1$



$p_H = 0.3$





Maxent Examples III

- Let's say we have the following event space:

NN	NNS	NNP	NNPS	VBZ	VBD
----	-----	-----	------	-----	-----

- ... and the following empirical data:

3	5	11	13	3	1
---	---	----	----	---	---

- Maximize H:

$1/e$	$1/e$	$1/e$	$1/e$	$1/e$	$1/e$
-------	-------	-------	-------	-------	-------

- ... want probabilities: $E[NN, NNS, NNP, NNPS, VBZ, VBD] = 1$

1/6	1/6	1/6	1/6	1/6	1/6
-----	-----	-----	-----	-----	-----



Maxent Examples IV

- Too uniform!
- N^* are more common than V^* , so we add the feature $f_N = \{NN, NNS, NNP, NNPS\}$, with $E[f_N] = 32/36$

NN	NNS	NNP	NNPS	VBZ	VBD
8/36	8/36	8/36	8/36	2/36	2/36

- ... and proper nouns are more frequent than common nouns, so we add $f_P = \{NNP, NNPS\}$, with $E[f_P] = 24/36$

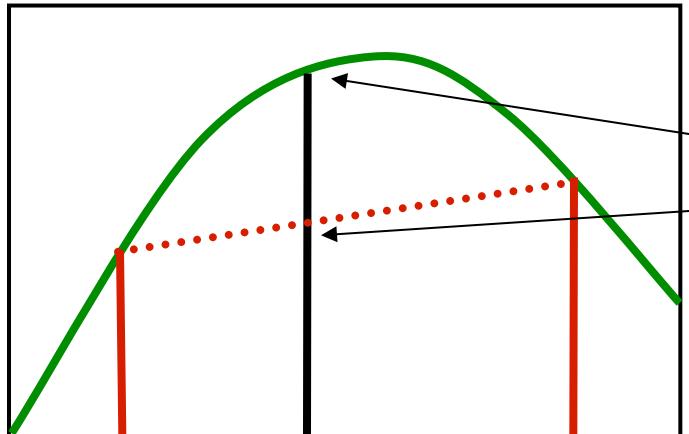
4/36	4/36	12/36	12/36	2/36	2/36
------	------	-------	-------	------	------

- ... we could keep refining the models, e.g., by adding a feature to distinguish singular vs. plural nouns, or verb types.

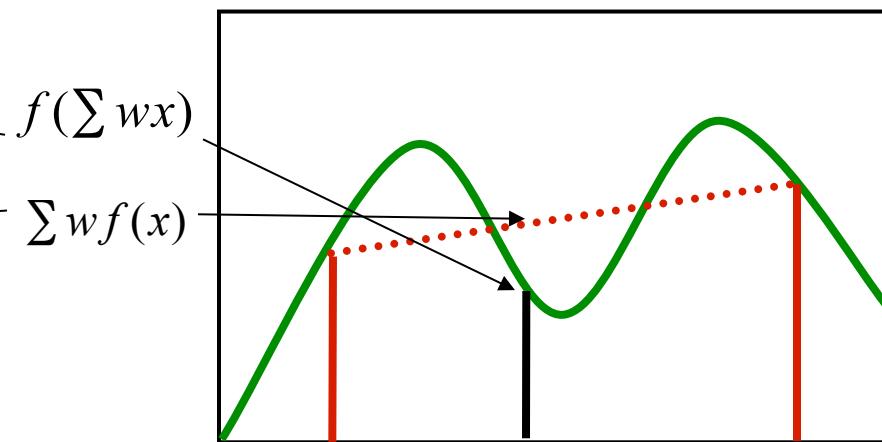


Convexity

$$f\left(\sum_i w_i x_i\right) \geq \sum_i w_i f(x_i) \quad \sum_i w_i = 1$$



Convex



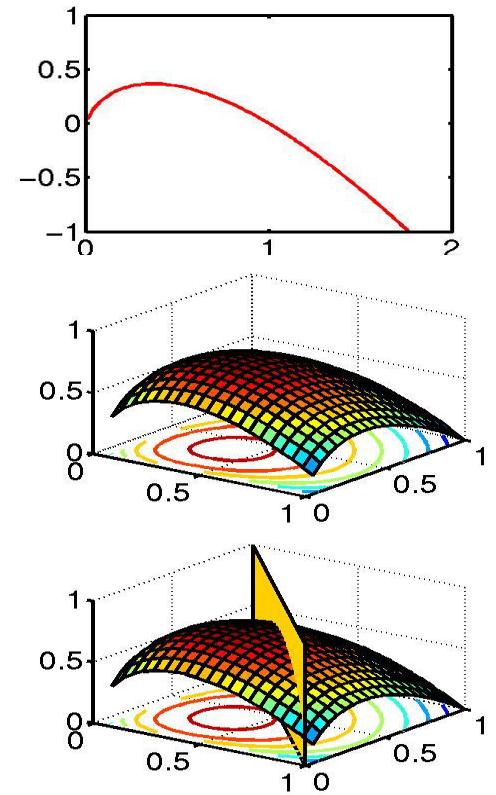
Non-Convex

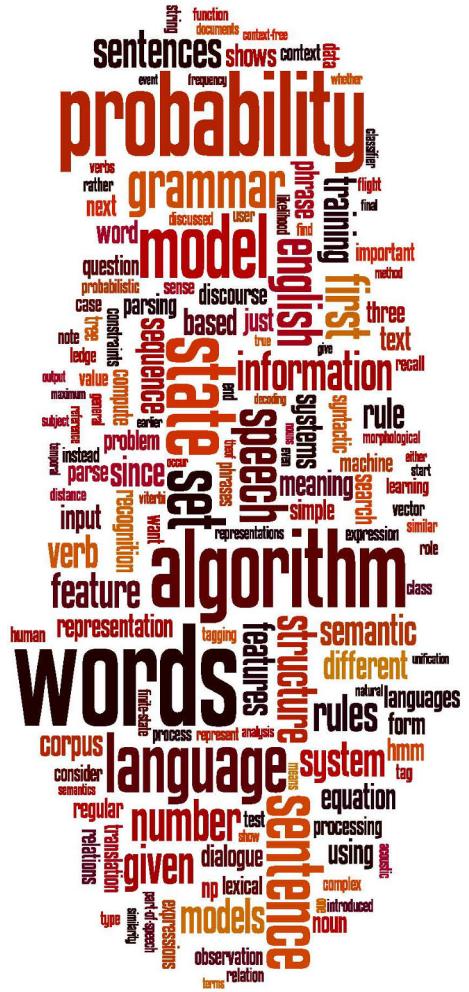
Convexity guarantees a single, global maximum because any higher points are greedily reachable.



Convexity II

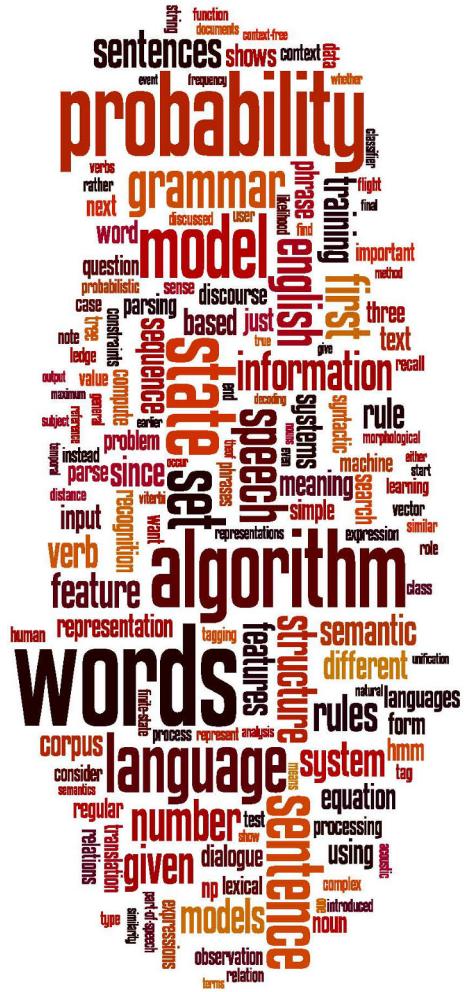
- Constrained $H(p) = - \sum x \log x$ is convex:
 - $-x \log x$ is convex
 - $-\sum x \log x$ is convex (sum of convex functions is convex).
 - The feasible region of constrained H is a linear subspace (which is convex)
 - The constrained entropy surface is therefore convex.
- The maximum likelihood exponential model (dual) formulation is also convex.





Maxent Models and Discriminative Estimation

The maximum entropy model presentation



Feature Overlap/ Feature Interaction

How overlapping features work in maxent models



Feature Overlap

- Maxent models handle overlapping features well.
- Unlike a NB model, there is no double counting!

		A	a
		B	2
		b	2
			1
			1

Empirical

	A	a
B		
b		

All = 1

	A	a
B	1/4	1/4
b	1/4	1/4

 $A = 2/3$

	A	a
B	1/3	1/6
b	1/3	1/6

 $A = 2/3$

	A	a
B	1/3	1/6
b	1/3	1/6

	A	a
B		
b		

	A	a
B	λ_A	
b	λ_A	

	A	a
B	$\lambda'_A + \lambda''_A$	
b	$\lambda'_A + \lambda''_A$	



Example: Named Entity Feature Overlap

Grace is correlated with PERSON, but does not add much evidence **on top of** already knowing prefix features.

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Feature Weights

Feature Type	Feature	PERS	LOC
Previous word	at	-0.73	0.94
Current word	Grace	0.03	0.00
Beginning bigram	<C	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
...			
Total:		-0.58	2.68



Feature Interaction

- Maxent models handle overlapping features well, but do not automatically model feature interactions.

		A	a
B	1	1	
b	1	0	

Empirical

	A	a
B		
b		

All = 1

	A	a
B	1/4	1/4
b	1/4	1/4

A = 2/3

	A	a
B	1/3	1/6
b	1/3	1/6

B = 2/3

	A	a
B	4/9	2/9
b	2/9	1/9

	A	a
B	0	0
b	0	0

	A	a
B	λ_A	
b	λ_A	

	A	a
B	$\lambda_A + \lambda_B$	λ_B
b	λ_A	



Feature Interaction

- If you want interaction terms, you have to add them:

	A	a
B	1	1
b	1	0

Empirical

	A	a
B		
b		

$$A = 2/3$$

	A	a
B	1/3	1/6
b	1/3	1/6

	A	a
B		
b		

$$B = 2/3$$

	A	a
B	4/9	2/9
b	2/9	1/9

	A	a
B		
b		

$$AB = 1/3$$

	A	a
B	1/3	1/3
b	1/3	0

- A disjunctive feature would also have done it (alone):

	A	a
B		
b		

	A	a
B	1/3	1/3
b	1/3	0



Quiz Question

- Suppose we have a 1 feature maxent model built over observed data as shown.
- What is the constructed model's probability distribution over the four possible outcomes?

Empirical		
	A	a
B	2	1
b	2	1

Features

	A	a
B		
b		

Expectations

Probabilities

	A	a
B		
b		



Feature Interaction

- For loglinear/logistic regression models in statistics, it is standard to do a greedy stepwise search over the space of all possible interaction terms.
- This combinatorial space is exponential in size, but that's okay as most statistics models only have 4–8 features.
- In NLP, our models commonly use hundreds of thousands of features, so that's not okay.
- Commonly, interaction terms are added by hand based on linguistic intuitions.



Example: NER Interaction

Previous-state and current-signature have interactions, e.g. $P=PERS-C=Xx$ indicates $C=PERS$ much more strongly than $C=Xx$ and $P=PERS$ independently.

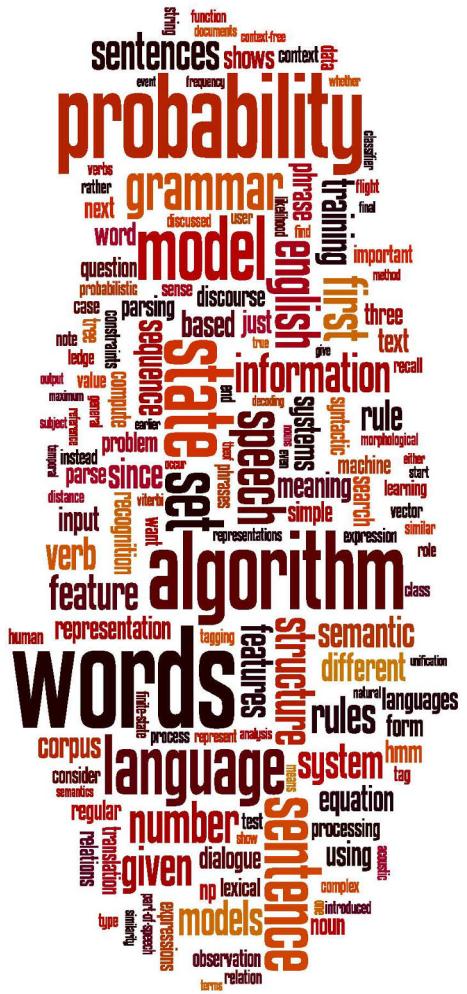
This feature type allows the model to capture this interaction.

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

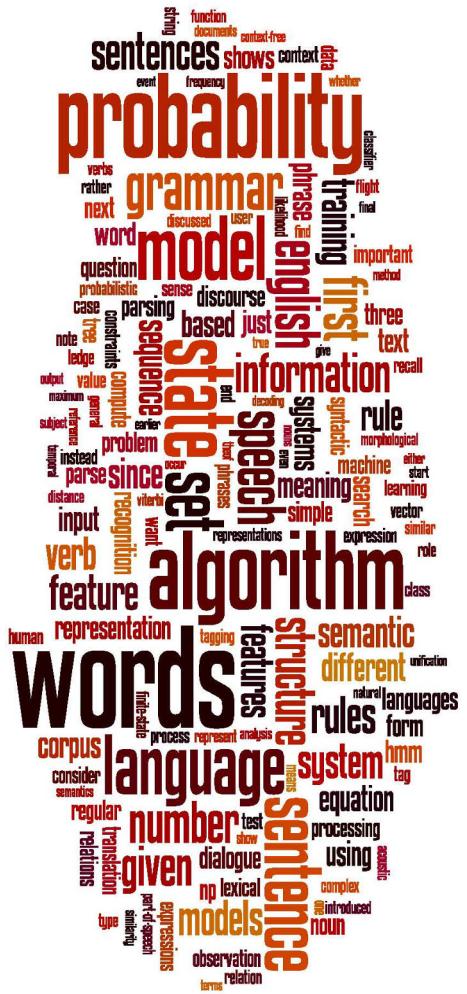
Feature Weights

Feature Type	Feature	PERS	LOC
Previous word	at	-0.73	0.94
Current word	Grace	0.03	0.00
Beginning bigram	<G	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
...			
Total:		-0.58	2.68



Feature Overlap/ Feature Interaction

How overlapping features
work in maxent models



Conditional Maxent Models for Classification

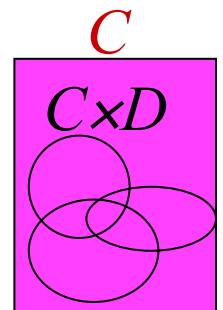
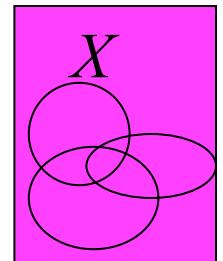
The relationship between
conditional and joint maxent/
exponential models

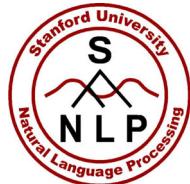


Classification

- What do these joint models of $P(\textcolor{violet}{X})$ have to do with conditional models $P(\textcolor{red}{C}|\textcolor{teal}{D})$?
- Think of the space $\textcolor{red}{C} \times \textcolor{teal}{D}$ as a complex $\textcolor{violet}{X}$.
 - $\textcolor{red}{C}$ is generally small (e.g., 2-100 topic classes)
 - $\textcolor{teal}{D}$ is generally huge (e.g., space of documents)
- We can, in principle, build models over $P(\textcolor{red}{C}, \textcolor{teal}{D})$.
- This will involve calculating expectations of features (over $\textcolor{red}{C} \times \textcolor{teal}{D}$):

$$E(f_i) = \sum_{(c,d) \in (C,D)} P(c,d) f_i(c,d)$$
- Generally impractical: can't enumerate $\textcolor{violet}{X}$ efficiently.





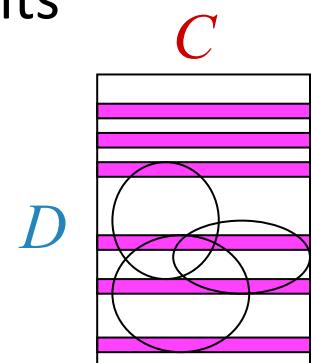
Classification II

- D may be huge or infinite, but only a few d occur in our data.
- What if we add one feature for each d and constrain its expectation to match our empirical data?

$$\forall(d) \in D \quad P(d) = \hat{P}(d)$$

- Now, most entries of $P(c,d)$ will be zero.
- We can therefore use the much easier sum:

$$\begin{aligned} E(f_i) &= \sum_{(c,d) \in (C,D)} P(c,d) f_i(c,d) \\ &= \sum_{(c,d) \in (C,D) \wedge \hat{P}(d) > 0} P(c,d) f_i(c,d) \end{aligned}$$





Classification III

- But if we've constrained the D marginals
$$\forall(d) \in D \quad P(d) = \hat{P}(d)$$
- then the only thing that can vary is the conditional distributions:

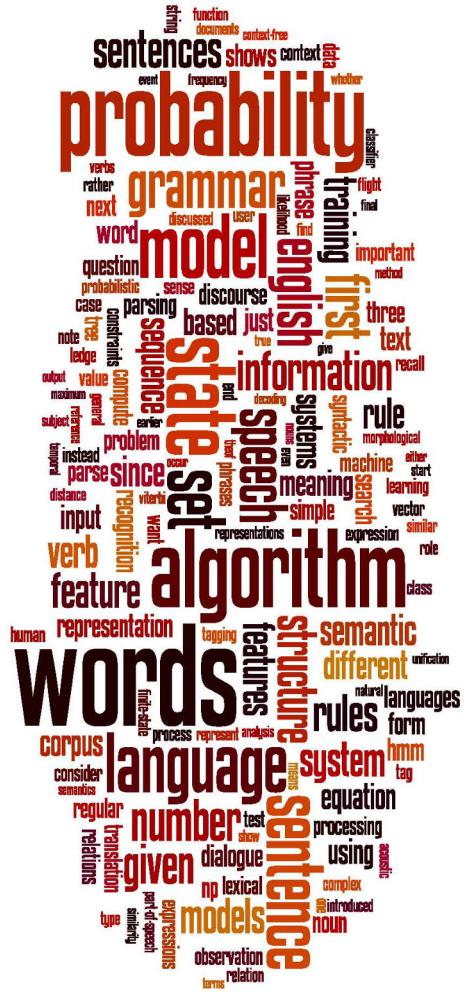
$$P(c, d) = P(c | d)P(d)$$

$$= P(c | d)\hat{P}(d)$$



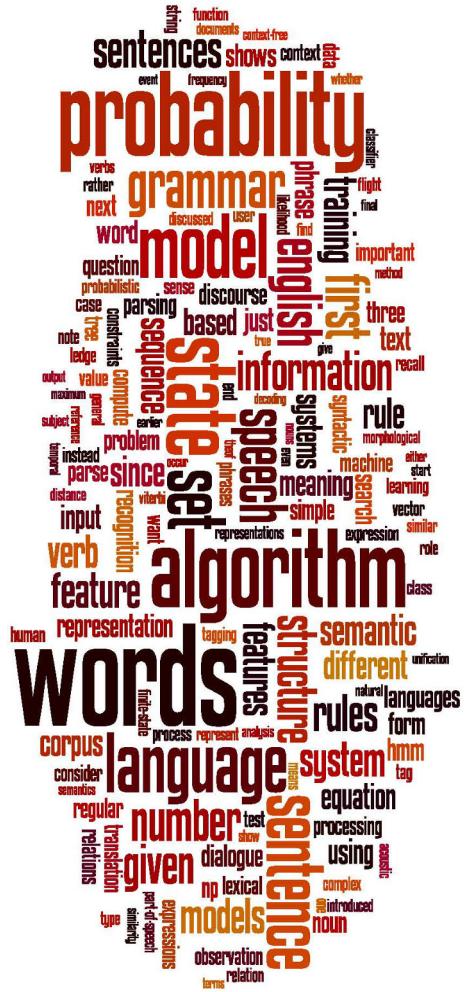
Classification IV

- This is the connection between joint and conditional maxent / exponential models:
 - Conditional models can be thought of as joint models with marginal constraints.
- Maximizing joint likelihood and conditional likelihood of the data in this model are equivalent!



Conditional Maxent Models for Classification

The relationship between conditional and joint maxent/exponential models



Smoothing/Priors/ Regularization for Maxent Models



Smoothing: Issues of Scale

- Lots of features:
 - NLP maxent models can have well over a million features.
 - Even storing a single array of parameter values can have a substantial memory cost.
- Lots of sparsity:
 - Overfitting very easy – we need smoothing!
 - Many features seen in training will never occur again at test time.
- Optimization problems:
 - Feature weights can be infinite, and iterative solvers can take a long time to get to those infinities.



Smoothing: Issues

- Assume the following empirical distribution:

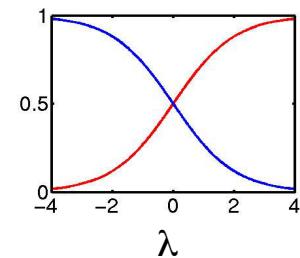
Heads	Tails
h	t

- Features: {Heads}, {Tails}
- We'll have the following model distribution:

$$p_{\text{HEADS}} = \frac{e^{\lambda_H}}{e^{\lambda_H} + e^{\lambda_T}} \quad p_{\text{TAILS}} = \frac{e^{\lambda_T}}{e^{\lambda_H} + e^{\lambda_T}}$$

- Really, only one degree of freedom ($\lambda = \lambda_H - \lambda_T$)

$$p_{\text{HEADS}} = \frac{e^{\lambda_H} e^{-\lambda_T}}{e^{\lambda_H} e^{-\lambda_T} + e^{\lambda_T} e^{-\lambda_T}} = \frac{e^\lambda}{e^\lambda + e^0} \quad p_{\text{TAILS}} = \frac{e^0}{e^\lambda + e^0}$$



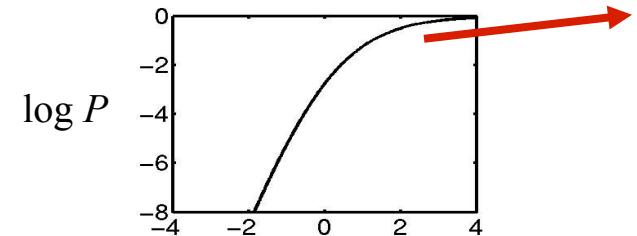
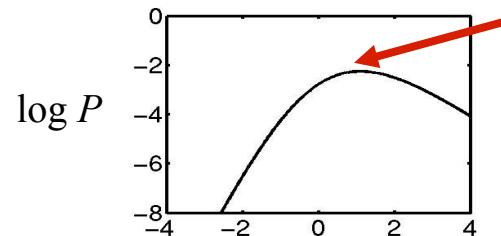
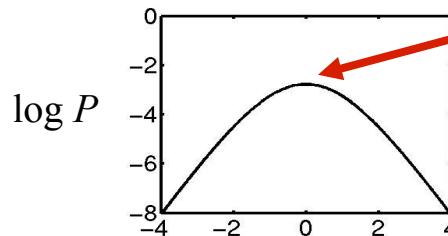


Smoothing: Issues

- The data likelihood in this model is:

$$\log P(h, t | \lambda) = h \log p_{\text{HEADS}} + t \log p_{\text{TAILS}}$$

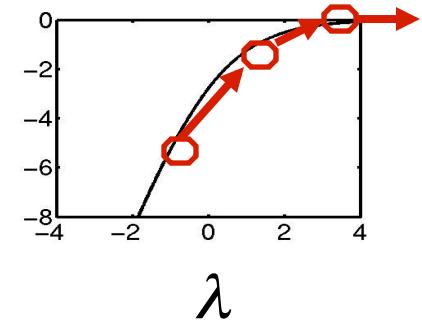
$$\log P(h, t | \lambda) = h\lambda - (t + h)\log(1 + e^\lambda)$$





Smoothing: Early Stopping

- In the 4/0 case, there were two problems:
 - The optimal value of λ was ∞ , which is a long trip for an optimization procedure.
 - The learned distribution is just as spiked as the empirical one – no smoothing.
- One way to solve both issues is to just stop the optimization early, after a few iterations.
 - The value of λ will be finite (but presumably big).
 - The optimization won't take forever (clearly).
 - Commonly used in early maxent work.



Heads	Tails
4	0

Input

Heads	Tails
1	0

Output



Smoothing: Priors (MAP)

- What if we had a prior expectation that parameter values wouldn't be very large?
- We could then balance evidence suggesting large parameters (or infinite) against our prior.
- The evidence would never totally defeat the prior, and parameters would be smoothed (and kept finite!).
- We can do this explicitly by changing the optimization objective to maximum posterior likelihood:

$$\log P(C, \lambda | D) = \log P(\lambda) + \log P(C | D, \lambda)$$

Posterior

Prior

Evidence

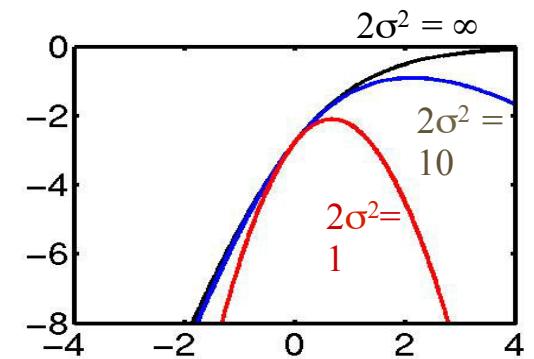


Smoothing: Priors

- Gaussian, or quadratic, or L_2 priors:
 - Intuition: parameters shouldn't be large.
 - Formalization: prior expectation that each parameter will be distributed according to a gaussian with mean μ and variance σ^2 .

$$P(\lambda_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(\lambda_i - \mu_i)^2}{2\sigma_i^2}\right)$$

- Penalizes parameters for drifting to far from their mean prior value (usually $\mu=0$).
- $2\sigma^2=1$ works surprisingly well.



They don't even
capitalize my
name anymore!





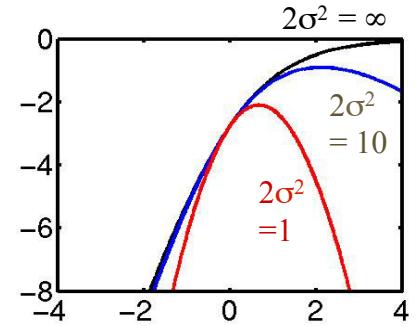
Smoothing: Priors

- If we use gaussian priors:
 - Trade off some expectation-matching for smaller parameters.
 - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
 - Accuracy generally goes up!
- Change the objective:

$$\log P(C, \lambda | D) = \log P(C | D, \lambda) - \log P(\lambda)$$

$$\log P(C, \lambda | D) = \sum_{(c,d) \in C,D} P(c | d, \lambda) - \sum_i \frac{(\lambda_i - \mu_i)^2}{2\sigma_i^2} + k$$
- Change the derivative:

$$\partial \log P(C, \lambda | D) / \partial \lambda_i = \text{actual}(f_i, C) - \text{predicted}(f_i, \lambda) - (\lambda_i - \mu_i) / \sigma^2$$





Smoothing: Priors

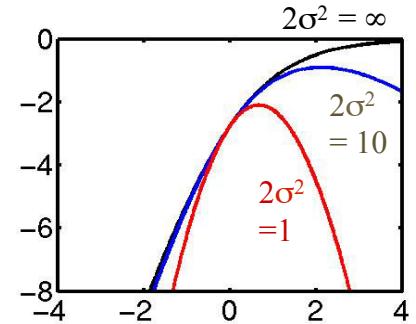
- If we use gaussian priors:
 - Trade off some expectation-matching for smaller parameters.
 - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
 - Accuracy generally goes up!
- Change the objective:

$$\log P(C, \lambda | D) = \log P(C | D, \lambda) - \log P(\lambda)$$

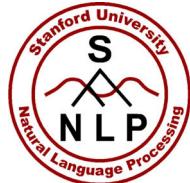
$$\log P(C, \lambda | D) = \sum_{(c,d) \in C,D} P(c | d, \lambda) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2} + k$$

- Change the derivative:

$$\partial \log P(C, \lambda | D) / \partial \lambda_i = \text{actual}(f_i, C) - \text{predicted}(f_i, \lambda) - \lambda_i / \sigma^2$$



Taking prior
mean as 0



Example: NER Smoothing

Because of smoothing, the more common prefix and single-tag features have larger weights even though entire-word and tag-pair features are more specific.

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Feature Weights

Feature Type	Feature	PERS	LOC
Previous word	at	-0.73	0.94
Current word	Grace	0.03	0.00
Beginning bigram	<G	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
...			
Total:		-0.58	2.68

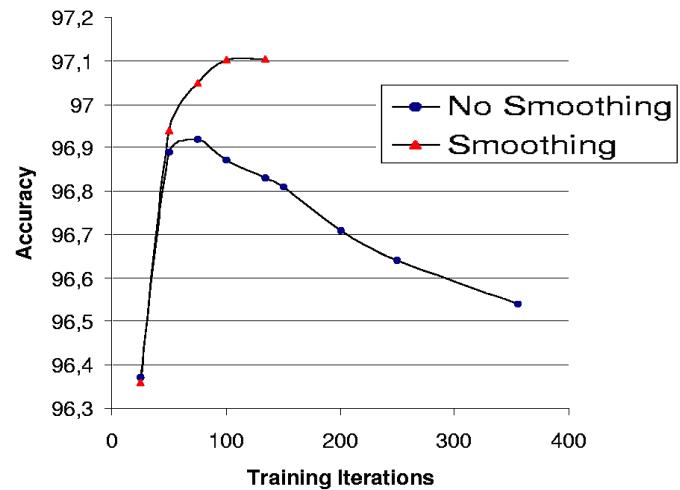


Example: POS Tagging

- From (Toutanova et al., 2003):

	Overall Accuracy	Unknown Word Acc
Without Smoothing	96.54	85.20
With Smoothing	97.10	88.20

- Smoothing helps:
 - Softens distributions.
 - Pushes weight onto more explanatory features.
 - Allows many features to be dumped safely into the mix.
 - Speeds up convergence (if both are allowed to converge)!





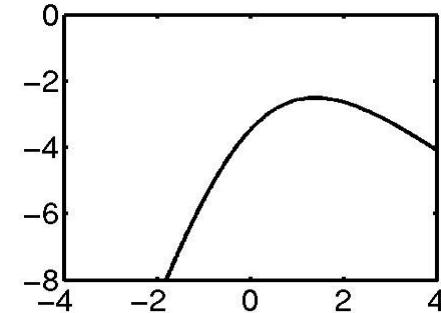
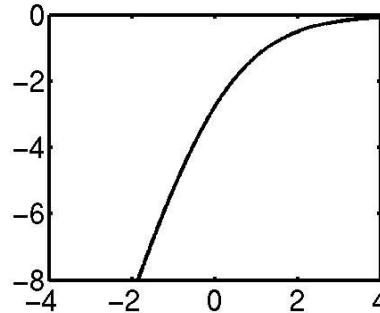
Smoothing: Regularization

- Talking of “priors” and “MAP estimation” is Bayesian language
- In frequentist statistics, people will instead talk about using “regularization”, and in particular, a gaussian prior is “ L_2 regularization”
- The choice of names makes no difference to the math



Smoothing: Virtual Data

- Another option: smooth the data, not the parameters.
- Example:



Heads	Tails
4	0



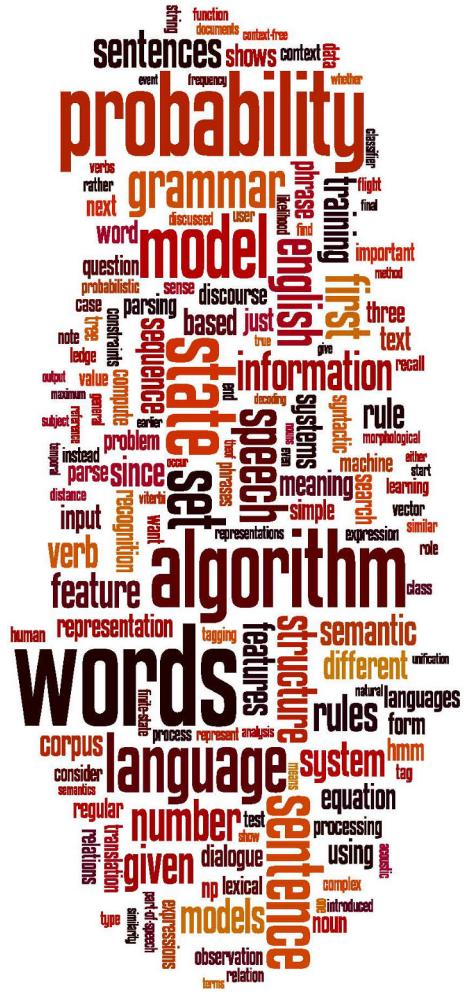
Heads	Tails
5	1

- Equivalent to adding two extra data points.
- Similar to add-one smoothing for generative models.
- Hard to know what artificial data to create!

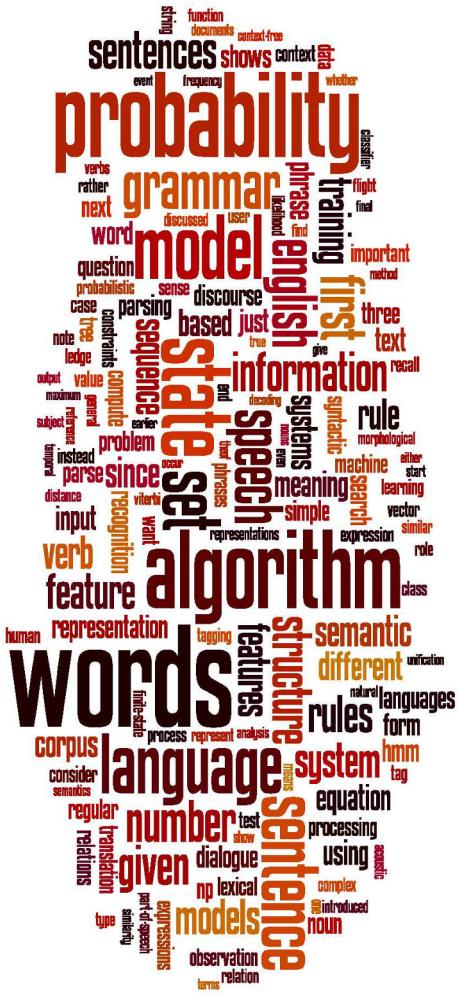


Smoothing: Count Cutoffs

- In NLP, features with low empirical counts are often dropped.
 - Very weak and indirect smoothing method.
 - Equivalent to locking their weight to be zero.
 - Equivalent to assigning them gaussian priors with mean zero and variance zero.
 - Dropping low counts does remove the features which were most in need of smoothing...
 - ... and speeds up the estimation by reducing model size ...
 - ... but count cutoffs generally hurt accuracy in the presence of proper smoothing.
- We recommend: don't use count cutoffs unless absolutely necessary for memory usage reasons.



Smoothing/Priors/ Regularization for Maxent Models



Part-of-speech tagging

A simple but useful form of linguistic analysis

Christopher Manning



Parts of Speech

- Perhaps starting with Aristotle in the West (384–322 BCE), there was the idea of having parts of speech
 - a.k.a lexical categories, word classes, “tags”, POS
- It comes from Dionysius Thrax of Alexandria (c. 100 BCE) the idea that is still with us that there are 8 parts of speech
 - But actually his 8 aren't exactly the ones we are taught today
 - Thrax: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
 - School grammar: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection

Open class (lexical) words

Nouns

Proper

IBM

Italy

Common

cat / cats

snow

Verbs

Main

see

registered

Adjectives

old older oldest

Adverbs

slowly

Numbers

122,312

one

... more

Closed class (functional)

Determiners

the some

Conjunctions

and or

Pronouns

he its

Modals

can

had

Prepositions

to with

Particles

off up

... more

Interjections

Ow Eh



Open vs. Closed classes

- Open vs. Closed classes
 - Closed:
 - determiners: *a, an, the*
 - pronouns: *she, he, I*
 - prepositions: *on, under, over, near, by, ...*
 - Why “closed”?
 - Open:
 - Nouns, Verbs, Adjectives, Adverbs.



POS Tagging

- 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

- Input: Plays well with others
- Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
- Output: Plays/VBZ well/RB with/IN others/NNS
- Uses:
 - Text-to-speech (how do we pronounce “lead”?)
 - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
 - As input to or to speed up a full parser
 - If you know the tag, you can back off to it in other tasks

Penn
Treebank
POS tags



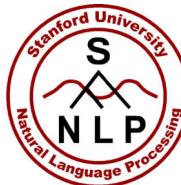
POS tagging performance

- How many tags are correct? (Tag accuracy)
 - About 97% currently
 - But baseline is already 90%
 - Baseline is performance of stupidest possible method
 - Tag every word with its most frequent tag
 - Tag unknown words as nouns
 - Partly easy because
 - Many words are unambiguous
 - You get points for them (*the*, *a*, etc.) and for punctuation marks!



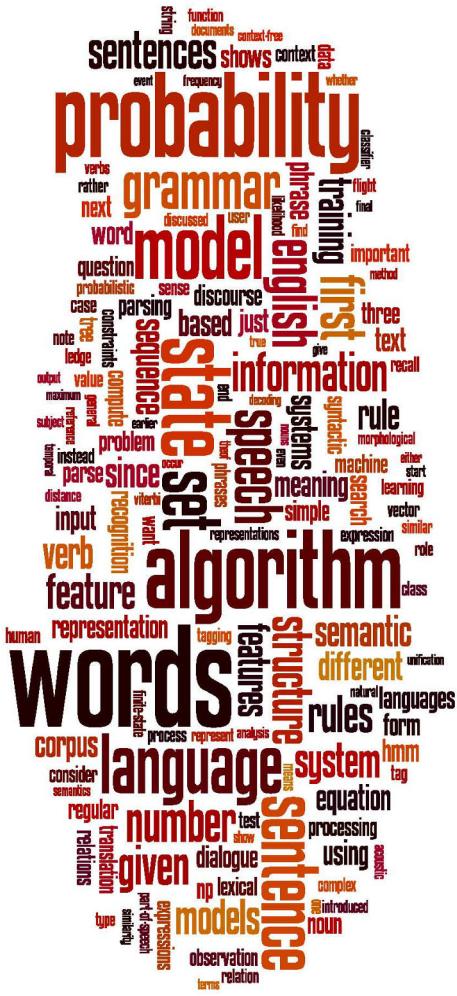
Deciding on the correct part of speech can be difficult even for people

- Mrs/NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD



How difficult is POS tagging?

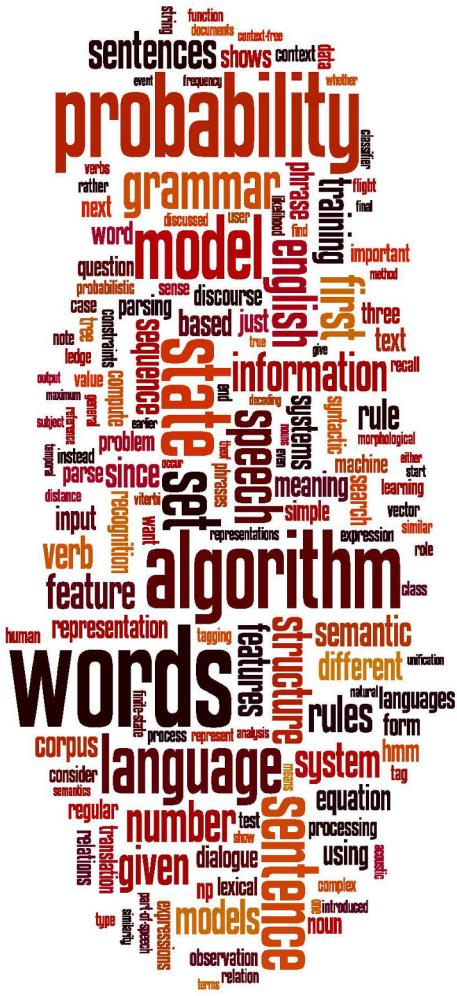
- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
 - I know *that* he is honest = IN
 - Yes, *that* play was nice = DT
 - You can't go *that* far = RB
- 40% of the word tokens are ambiguous



Part-of-speech tagging

A simple but useful form of linguistic analysis

Christopher Manning



Part-of-speech tagging revisited

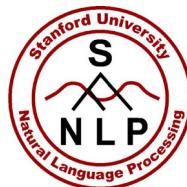
A simple but useful form of linguistic analysis

Christopher Manning



Sources of information

- What are the main sources of information for POS tagging?
 - Knowledge of neighboring words
 - Bill saw that man yesterday
 - NNP NN DT NN NN
 - VB VB(D) IN VB NN
 - Knowledge of word probabilities
 - *man* is rarely used as a verb....
- The latter proves the most useful, but the former also helps



More and Better Features → Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
 - Word the: the → DT
 - Lowercased word Importantly: importantly → RB
 - Prefixes unfathomable: un- → JJ
 - Suffixes Importantly: -ly → RB
 - Capitalization Meridian: CAP → NNP
 - Word shapes 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
 - Maxent $P(t|w)$: 93.7% overall / 82.6% unknown



Overview: POS Tagging Accuracies

- Rough accuracies:
 - Most freq tag:
 - Trigram HMM:
 - Maxent $P(t|w)$:
 - TnT (HMM++):
 - MEMM tagger:
 - Bidirectional dependencies:
 - Upper bound:

~90% / ~50%

~95% / ~55%

93.7% / 82.6%

96.2% / 86.0%

96.9% / 86.9%

97.2% / 90.0%

~98% (human agreement)

Most errors
on unknown
words



How to improve supervised results?

- Build better features!

PRP RB
VBD IN RB IN PRP VBD .
They left as soon as he arrived .

- We could fix this with a feature that looked at the next word

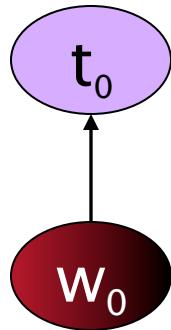
JJ
NNP NNS VBD VBN .
Intrinsic flaws remained undetected .

- We could fix this by linking capitalized words to their lowercase versions

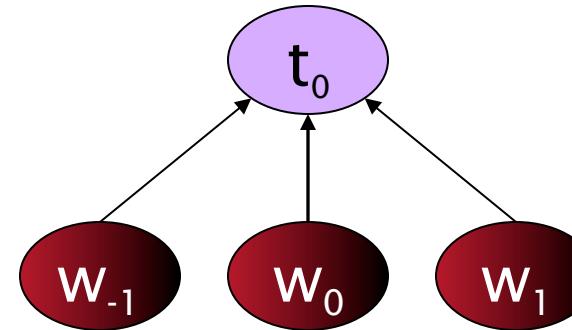


Tagging Without Sequence Information

Baseline



Three Words



Model	Features	Token	Unknown	Sentence
Baseline	56,805	93.69%	82.61%	26.74%
3Words	239,767	96.57%	86.78%	48.27%

Using words only in a straight classifier works as well as a basic (HMM or discriminative) sequence model!!



Summary of POS Tagging

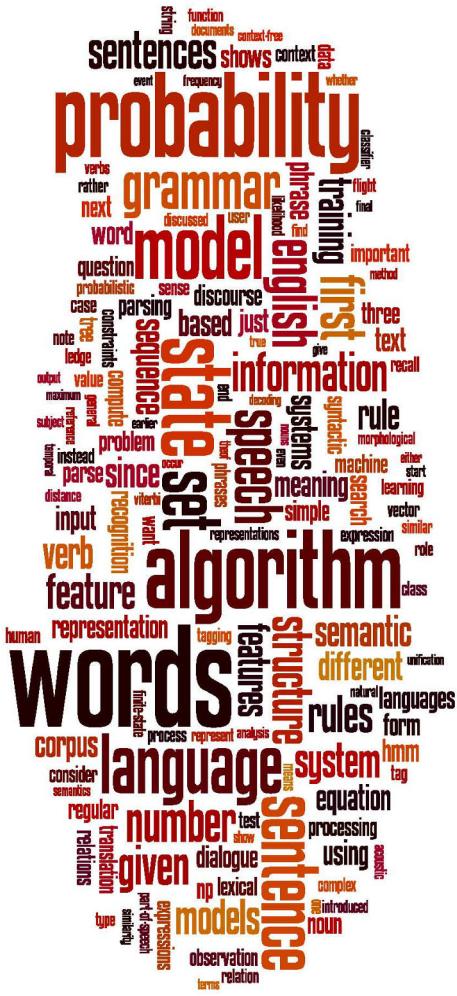
For tagging, the change from generative to discriminative model **does not by itself** result in great improvement

One profits from models for specifying dependence on **overlapping features of the observation** such as spelling, suffix analysis, etc.

An MEMM allows integration of rich features of the observations, but can suffer strongly from assuming independence from following observations; this effect can be relieved by adding dependence on following words

This additional power (of the MEMM ,CRF, Perceptron models) has been shown to result in improvements in accuracy

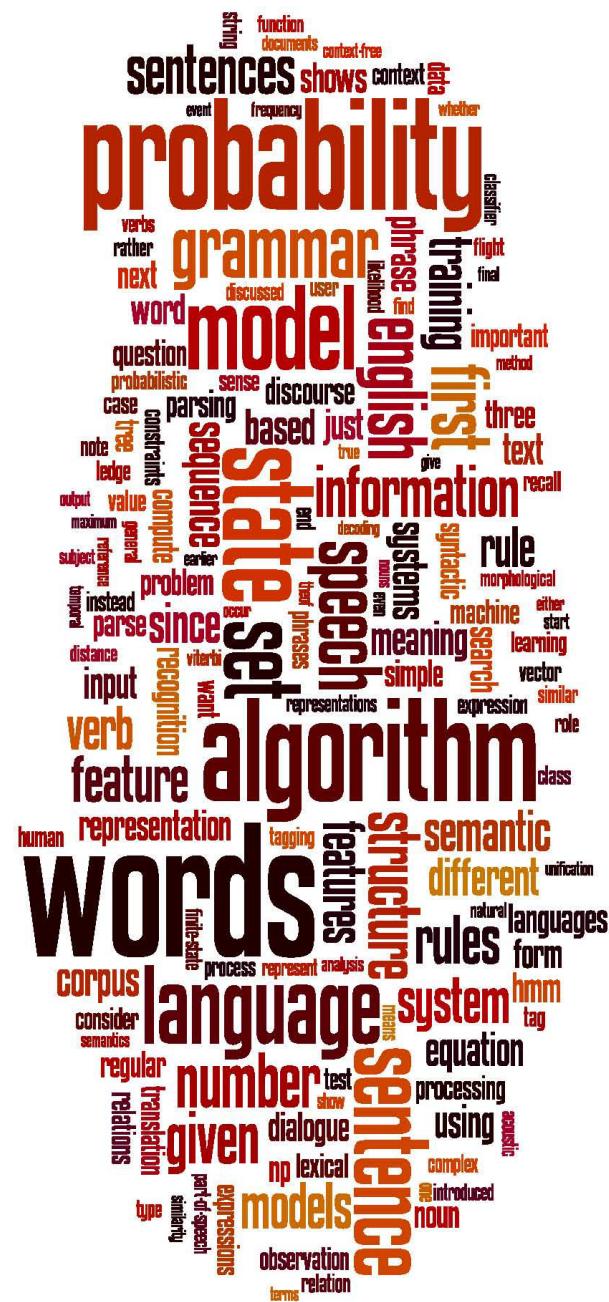
The **higher accuracy** of discriminative models comes at the price of **much slower training**



Part-of-speech tagging revisited

A simple but useful form of linguistic analysis

Christopher Manning



Statistical Natural Language Parsing

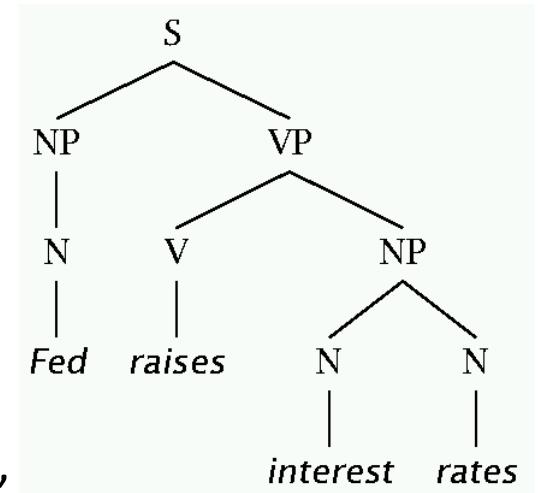
Two views of syntactic structure



Two views of linguistic structure:

1. Constituency (phrase structure)

- 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: a constituent behaves as a unit that can appear 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, regular internal structure, no intrusion, fragments, semantics, ...

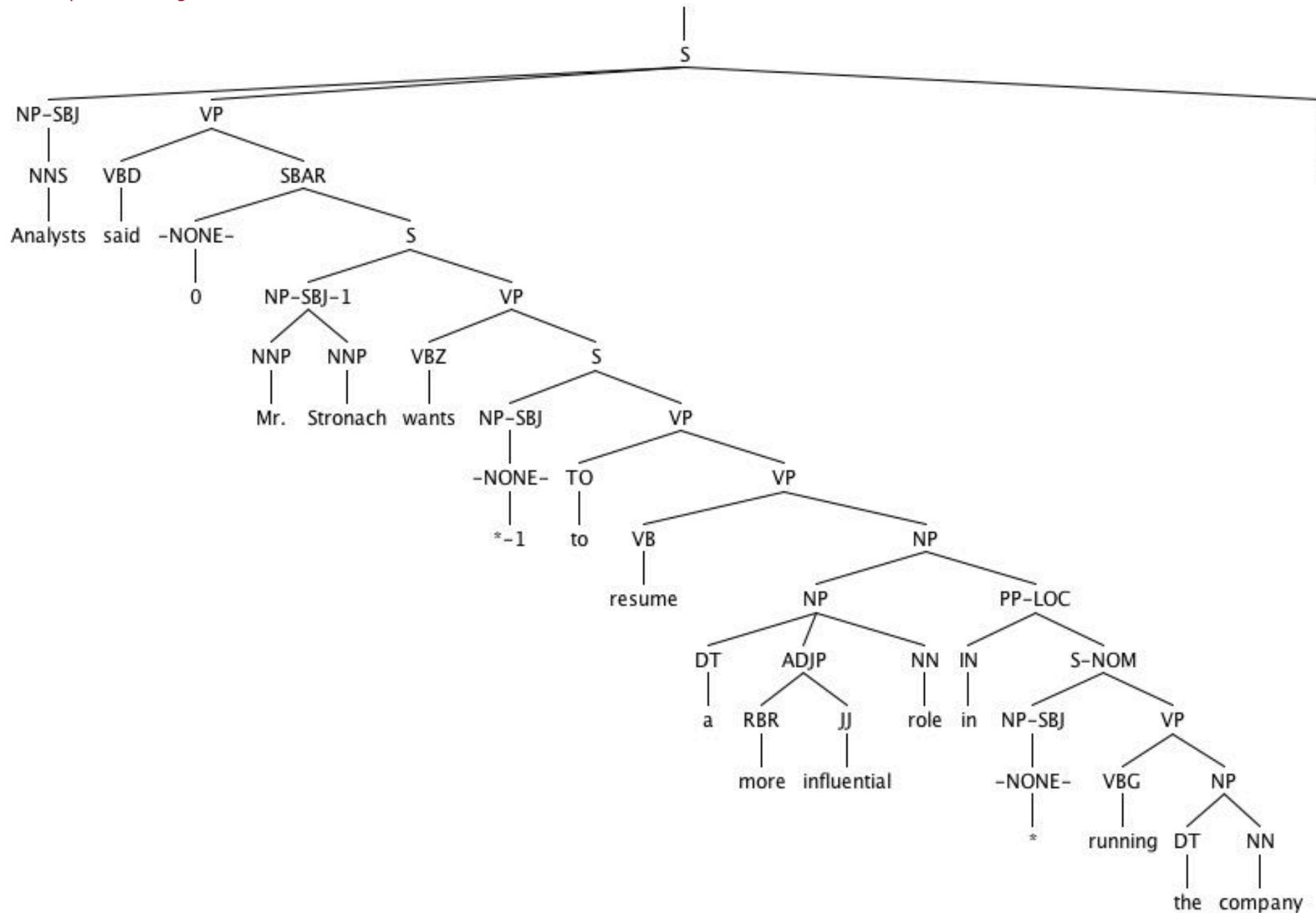




Two views of linguistic structure:

1. Constituency (phrase structure)

- 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: a constituent behaves as a unit that can appear 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, regular internal structure, no intrusion, fragments, semantics, ...





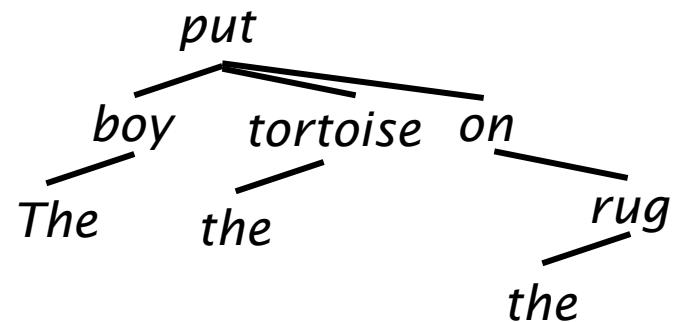
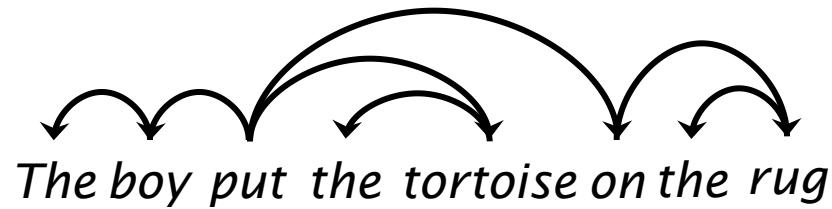
Headed phrase structure

- VP → ... VB* ...
- NP → ... NN* ...
- ADJP → ... JJ* ...
- ADVP → ... RB* ...
- SBAR(Q) → S | SINV | SQ → ... NP VP ...
- Plus minor phrase types:
 - QP (quantifier phrase in NP), CONJP (multi word constructions: *as well as*), INTJ (interjections), etc.



Two views of linguistic structure: 2. Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.

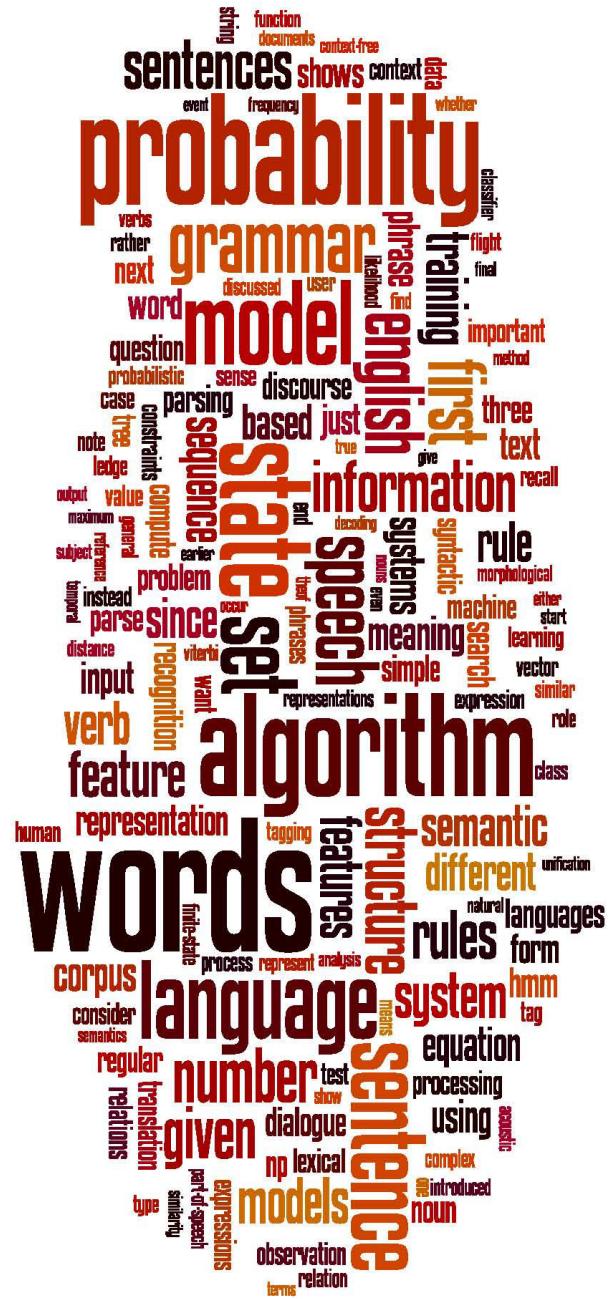




Two views of linguistic structure: 2. Dependency structure

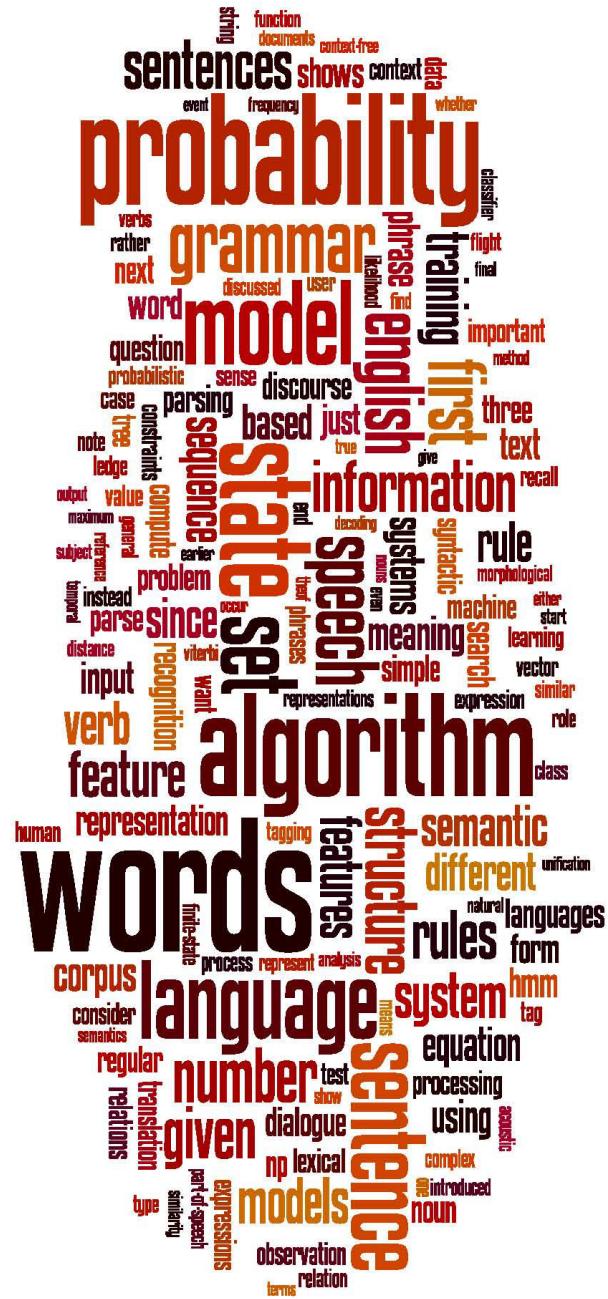
- Dependency structure shows which words depend on (modify or are arguments of) which other words.

The boy put the tortoise on the rug



Statistical Natural Language Parsing

Two views of
syntactic structure



Statistical Natural Language Parsing

Parsing: The rise of
data and statistics



Pre 1990 (“Classical”) NLP Parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon
 - $S \rightarrow NP\ VP$
 - $NP \rightarrow (DT)\ NN$
 - $NP \rightarrow NN\ NNS$
 - $NP \rightarrow NNP$
 - $VP \rightarrow V\ NP$
 - $NN \rightarrow interest$
 - $NNS \rightarrow rates$
 - $NNS \rightarrow raises$
 - $VBP \rightarrow interest$
 - $VBZ \rightarrow rates$
- Used grammar/proof systems to prove parses from words
- This scaled very badly and didn’t give coverage. For sentence:

Fed raises interest rates 0.5% in effort to control inflation

- Minimal grammar: 36 parses
- Simple 10 rule grammar: 592 parses
- Real-size broad-coverage grammar: millions of parses



Classical NLP Parsing: The problem and its solution

- Categorical constraints can be added to grammars to limit unlikely/weird parses for sentences
 - But the attempt make the grammars not robust
 - In traditional systems, commonly 30% of sentences in even an edited text would have *no* parse.
- A less constrained grammar can parse more sentences
 - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
 - Statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but still quickly find the best parse(s)



The rise of annotated data: The Penn Treebank

[Marcus et al. 1993, *Computational Linguistics*]

```
( (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)))))))
  (. .)))
```



The rise of annotated data

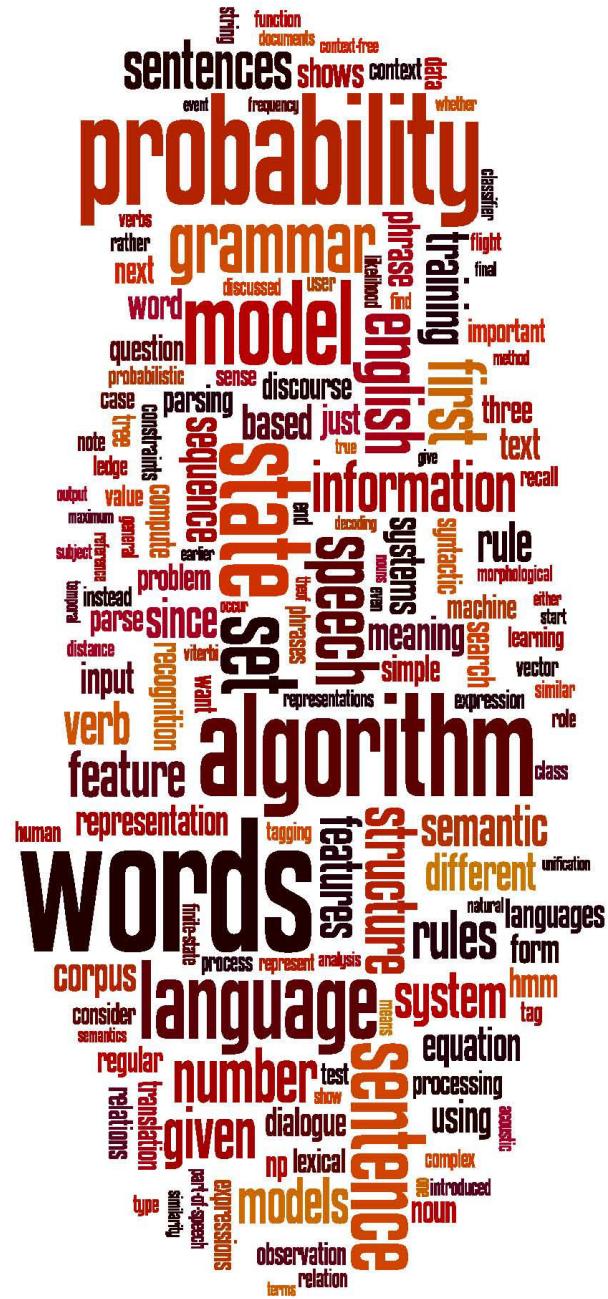
- Starting off, building a treebank seems a lot slower and less useful than building a grammar
- But a treebank gives us many things
 - Reusability of the labor
 - Many parsers, POS taggers, etc.
 - Valuable resource for linguistics
 - Broad coverage
 - Frequencies and distributional information
 - A way to evaluate systems



Statistical parsing applications

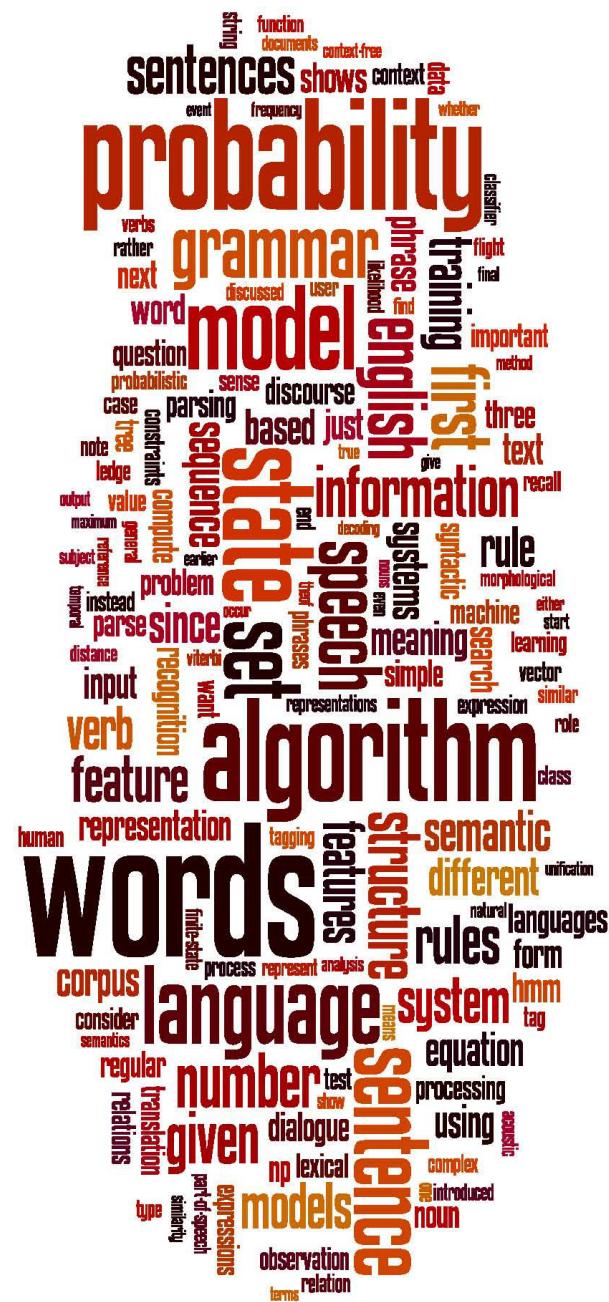
Statistical parsers are now robust and widely used in larger NLP applications:

- High precision question answering [Pasca and Harabagiu SIGIR 2001]
- Improving biological named entity finding [Finkel et al. JNLPBA 2004]
- Syntactically based sentence compression [Lin and Wilbur 2007]
- Extracting opinions about products [Bloom et al. NAACL 2007]
- Improved interaction in computer games [Gorniak and Roy 2005]
- Helping linguists find data [Resnik et al. BLS 2005]
- Source sentence analysis for machine translation [Xu et al. 2009]
- Relation extraction systems [Fundel et al. *Bioinformatics* 2006]



Statistical Natural Language Parsing

Parsing: The rise of
data and statistics



Statistical Natural Language Parsing

An exponential number of attachments



Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
 - PPs, adverbial or participial phrases, infinitives, coordinations, etc.

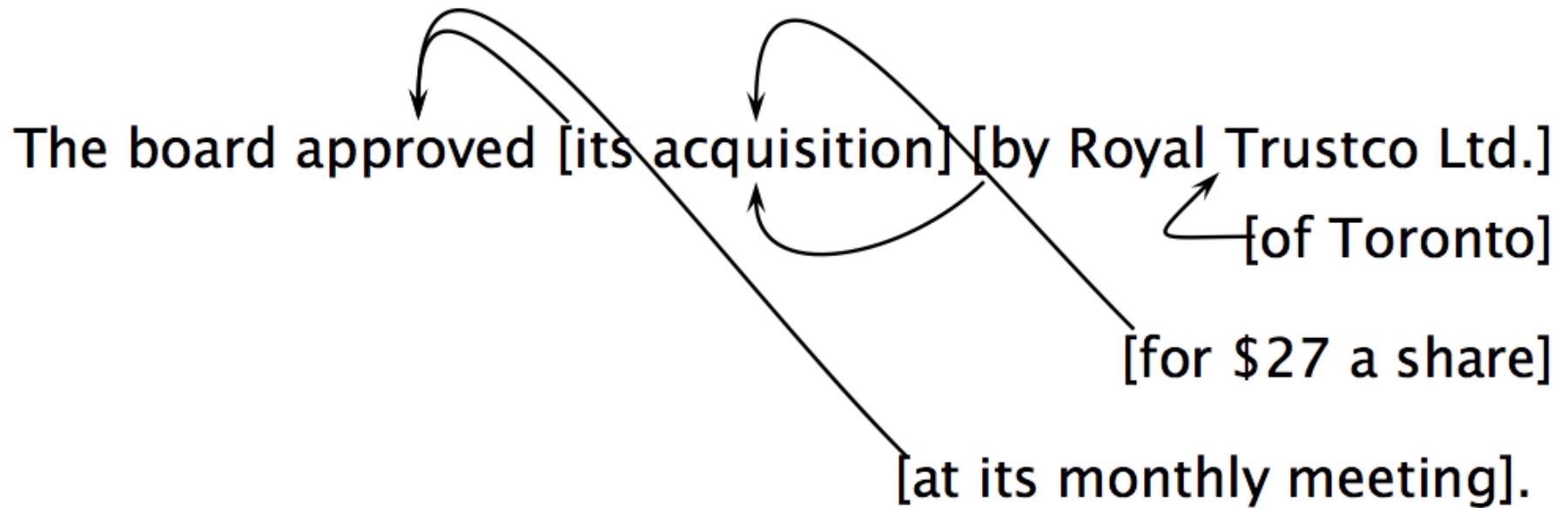
The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for \$27 a share]
[at its monthly meeting].

- Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts:
 - E.g., the number of possible triangulations of a polygon with $n+2$ sides
 - Turns up in triangulation of probabilistic graphical models....



Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
 - PPs, adverbial or participial phrases, infinitives, coordinations, etc.



- Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts:
 - E.g., the number of possible triangulations of a polygon with $n+2$ sides
 - Turns up in triangulation of probabilistic graphical models....



Quiz Question!

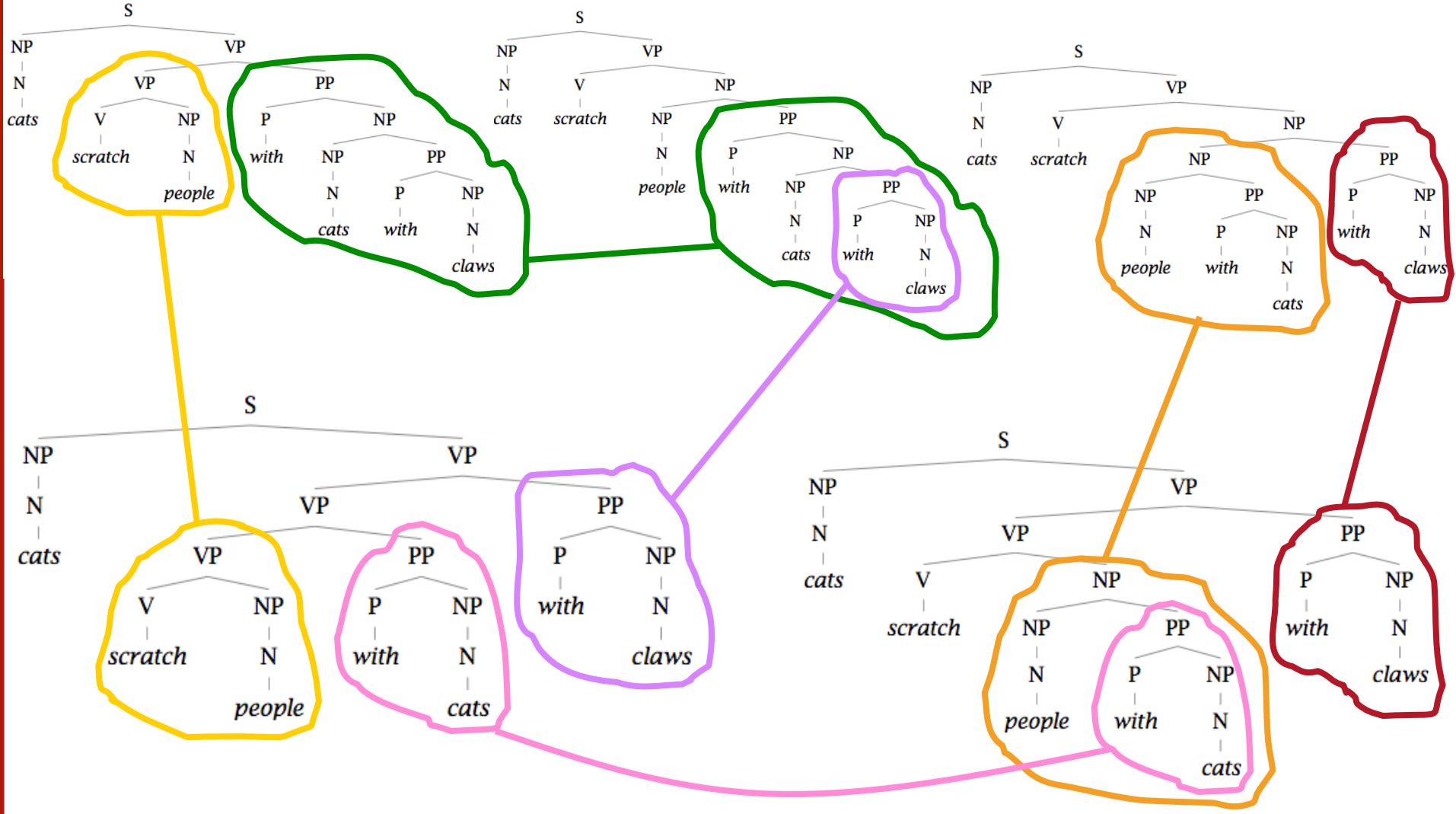
- How many distinct parses does the following sentence have due to PP attachment ambiguities?
 - A PP can attach to any preceding V or N within the verb phrase, subject only to the parse still being a tree.
 - (This is equivalent to there being no crossing dependencies, where if d_2 is a dependent of d_1 and d_3 is a dependent of d_2 , then the line d_2-d_3 begins at d_2 under the line from d_1 to d_2 .)

John wrote the book with a pen in the room.



Two problems to solve:

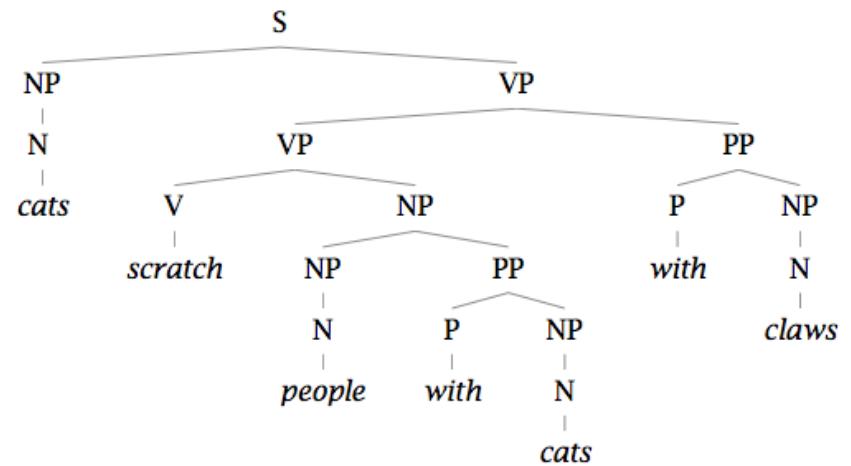
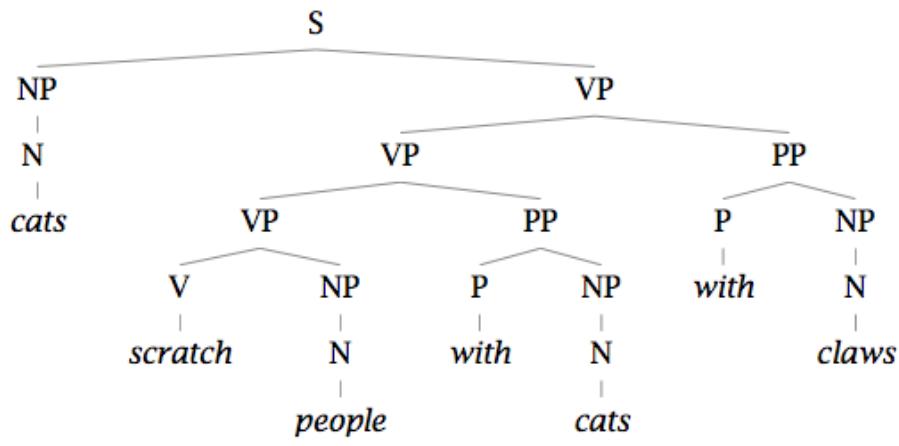
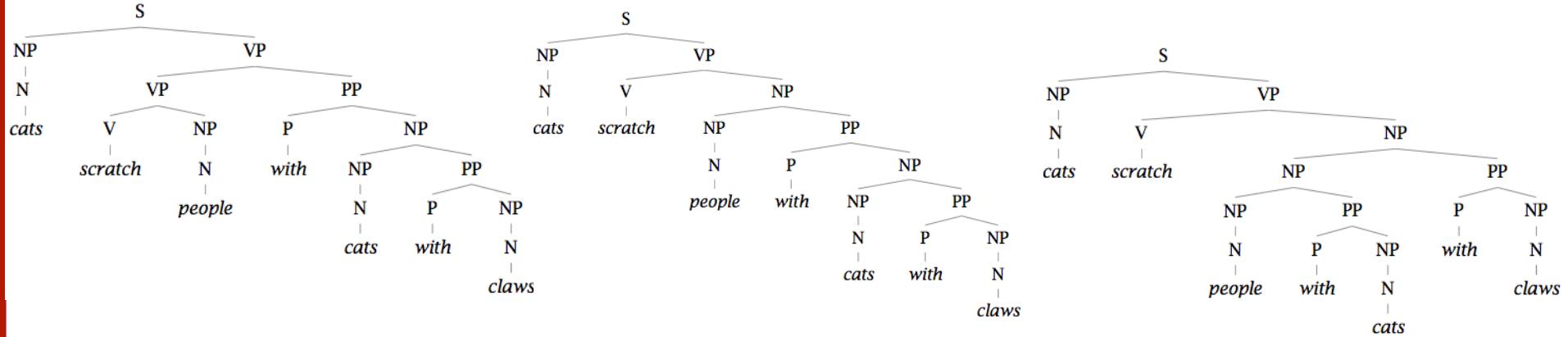
1. Repeated work...





Two problems to solve:

1. Repeated work...

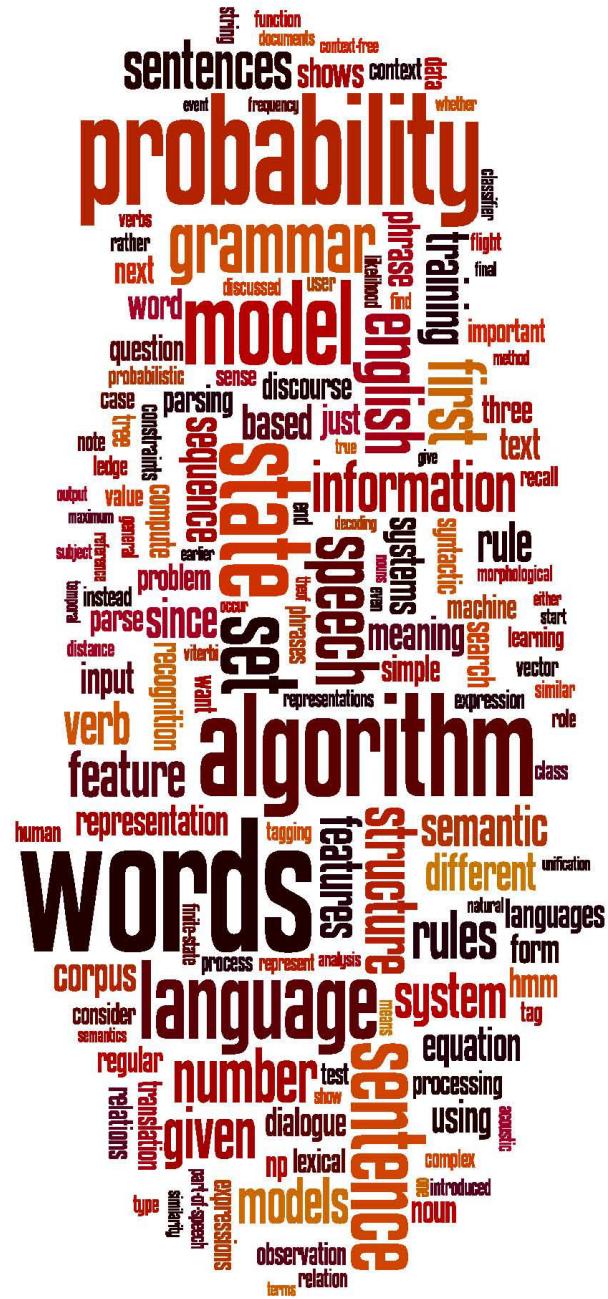




Two problems to solve:

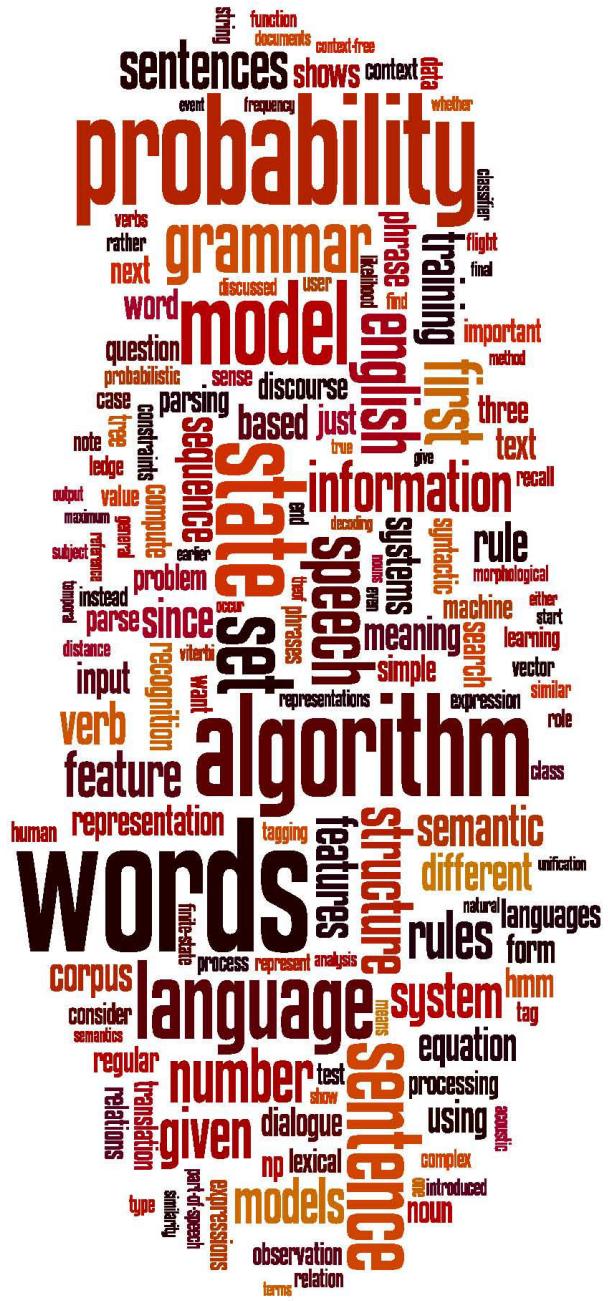
2. Choosing the correct parse

- How do we work out the correct attachment:
 - She saw the man with a telescope
 - Is the problem ‘AI complete’? Yes, but ...
 - Words are good predictors of attachment
 - Even absent full understanding
 - Moscow sent more than 100,000 soldiers into Afghanistan ...
 - Sydney Water breached an agreement with NSW Health ...
- Our statistical parsers will try to exploit such statistics.



Statistical Natural Language Parsing

An exponential
number of
attachments



CFGs and PCFGs

(Probabilistic)
Context-Free
Grammars



A phrase structure grammar

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow V NP PP$

$NP \rightarrow NP NP$

$NP \rightarrow NP PP$

$NP \rightarrow N$

$NP \rightarrow e$

$PP \rightarrow P NP$

$N \rightarrow people$

$N \rightarrow fish$

$N \rightarrow tanks$

$N \rightarrow rods$

$V \rightarrow people$

$V \rightarrow fish$

$V \rightarrow tanks$

$P \rightarrow with$

people fish tanks

people fish with rods



Phrase structure grammars = context-free grammars (CFGs)

- $G = (T, N, S, R)$
 - T is a set of terminal symbols
 - N is a set of nonterminal symbols
 - S is the start symbol ($S \in N$)
 - R is a set of rules/productions of the form $X \rightarrow \gamma$
 - $X \in N$ and $\gamma \in (N \cup T)^*$
- A grammar G generates a language L .



Phrase structure grammars in NLP

- $G = (T, C, N, S, L, R)$
 - T is a set of terminal symbols
 - C is a set of preterminal symbols
 - N is a set of nonterminal symbols
 - S is the start symbol ($S \in N$)
 - L is the lexicon, a set of items of the form $X \rightarrow x$
 - $X \in P$ and $x \in T$
 - R is the grammar, a set of items of the form $X \rightarrow \gamma$
 - $X \in N$ and $\gamma \in (N \cup C)^*$
- By usual convention, S is the start symbol, but in statistical NLP, we usually have an extra node at the top (ROOT, TOP)
- We usually write e for an empty sequence, rather than nothing



A phrase structure grammar

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ NP\ PP$

$NP \rightarrow NP\ NP$

$NP \rightarrow NP\ PP$

$NP \rightarrow N$

$NP \rightarrow e$

$PP \rightarrow P\ NP$

$N \rightarrow people$

$N \rightarrow fish$

$N \rightarrow tanks$

$N \rightarrow rods$

$V \rightarrow people$

$V \rightarrow fish$

$V \rightarrow tanks$

$P \rightarrow with$

people fish tanks

people fish with rods



Probabilistic – or stochastic – context-free grammars (PCFGs)

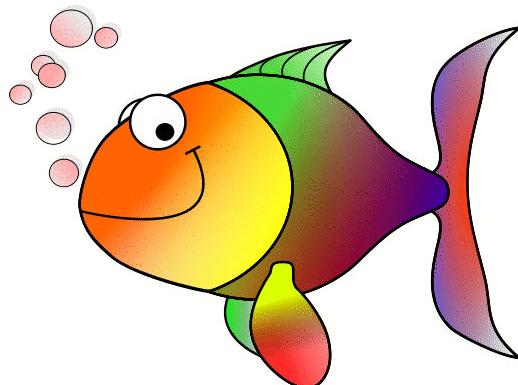
- $G = (T, N, S, R, P)$
 - T is a set of terminal symbols
 - N is a set of nonterminal symbols
 - S is the start symbol ($S \in N$)
 - R is a set of rules/productions of the form $X \rightarrow \gamma$
 - P is a probability function
 - $P: R \rightarrow [0,1]$
 - $\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$
- A grammar G generates a language model L .

$$\sum_{\gamma \in T^*} P(\gamma) = 1$$



A PCFG

$S \rightarrow NP\ VP$	1.0	$N \rightarrow people$	0.5
$VP \rightarrow V\ NP$	0.6	$N \rightarrow fish$	0.2
$VP \rightarrow V\ NP\ PP$	0.4	$N \rightarrow tanks$	0.2
$NP \rightarrow NP\ NP$	0.1	$N \rightarrow rods$	0.1
$NP \rightarrow NP\ PP$	0.2	$V \rightarrow people$	0.1
$NP \rightarrow N$	0.7	$V \rightarrow fish$	0.6
$PP \rightarrow P\ NP$	1.0	$V \rightarrow tanks$	0.3
		$P \rightarrow with$	1.0



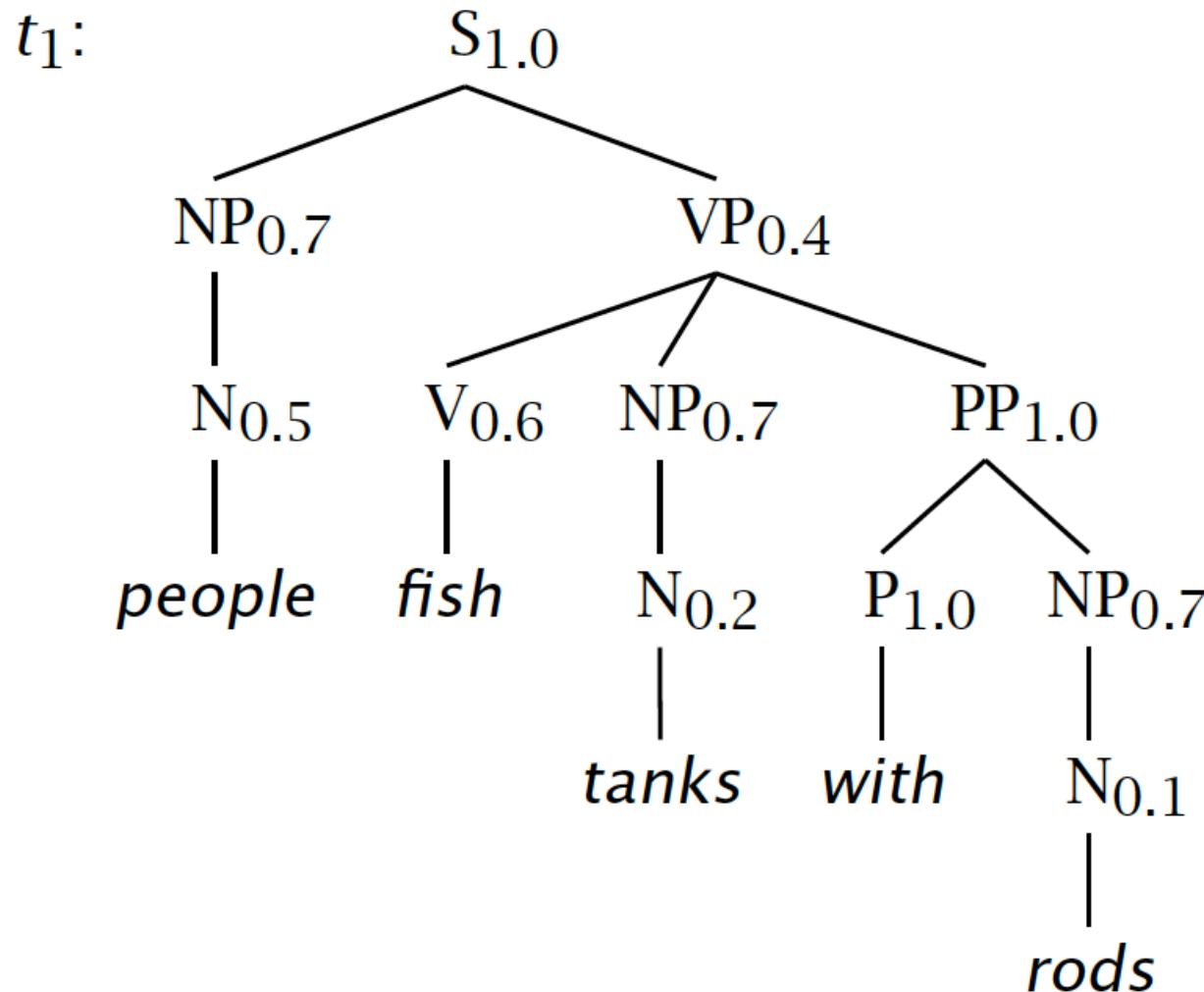
[With empty NP removed
so less ambiguous]

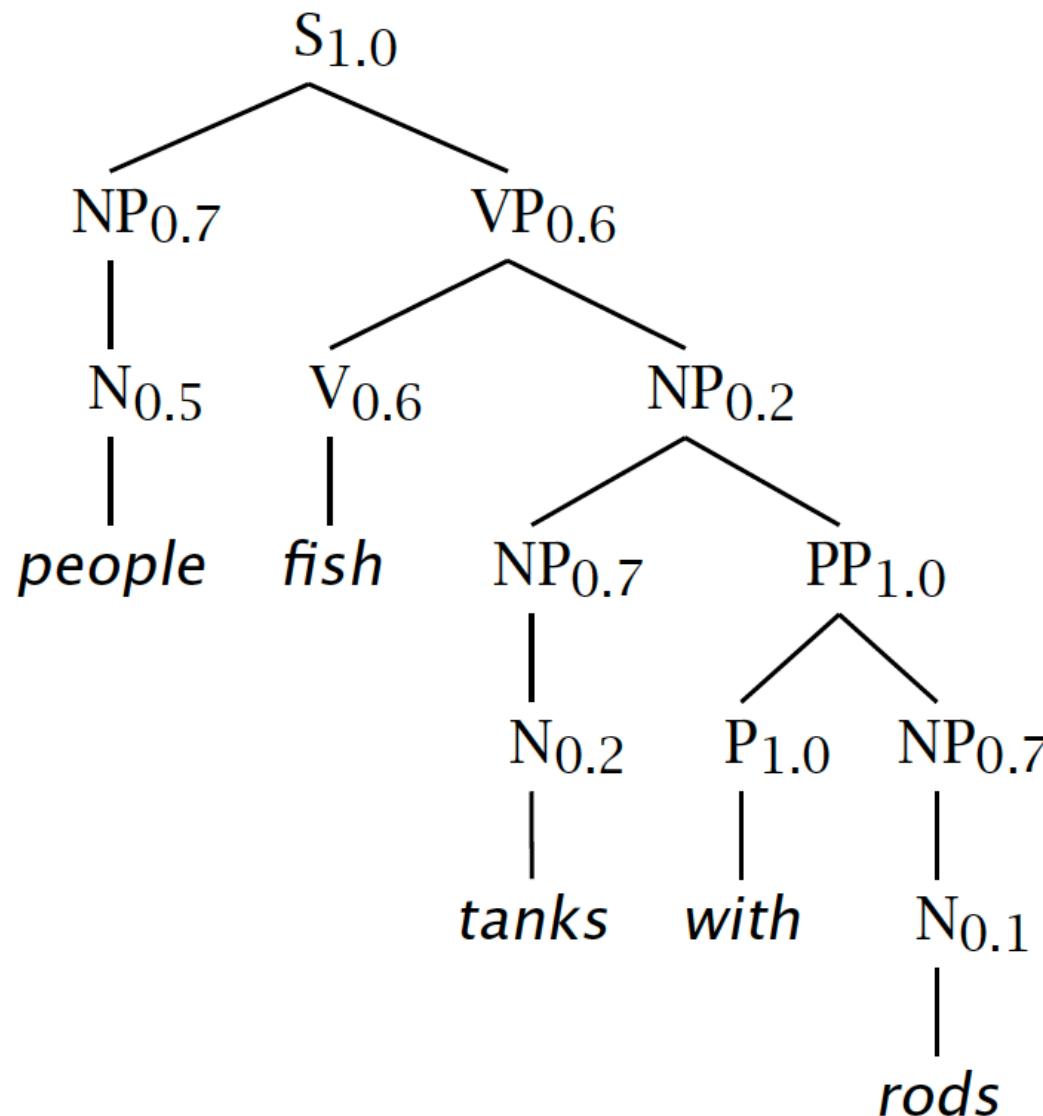


The probability of trees and strings

- $P(t)$ – The probability of a tree t is the product of the probabilities of the rules used to generate it.
- $P(s)$ – The probability of the string s is the sum of the probabilities of the trees which have that string as their yield

$$\begin{aligned} P(s) &= \sum_j P(s, t) \text{ where } t \text{ is a parse of } s \\ &= \sum_j P(t) \end{aligned}$$



 $t_2:$ 

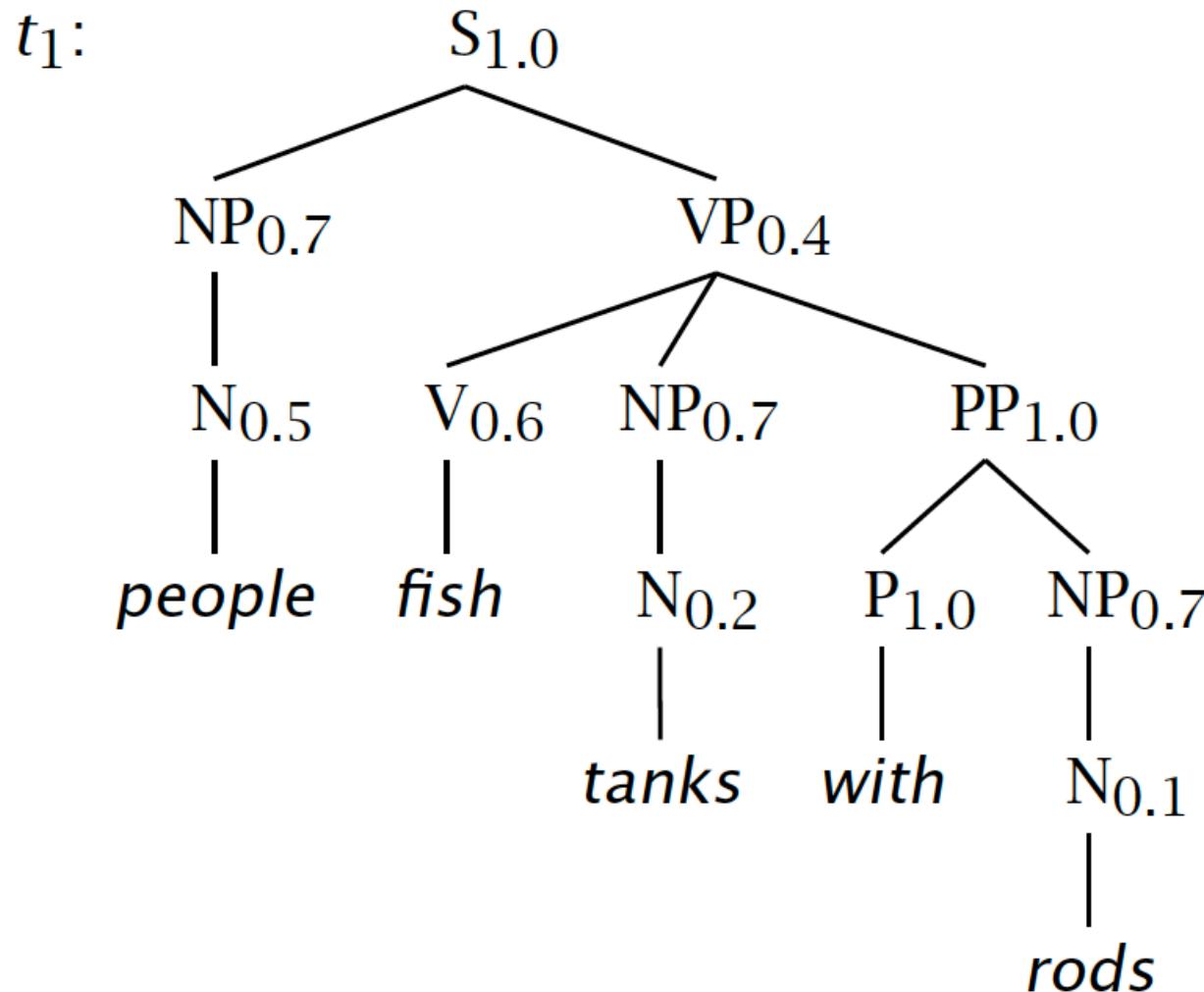


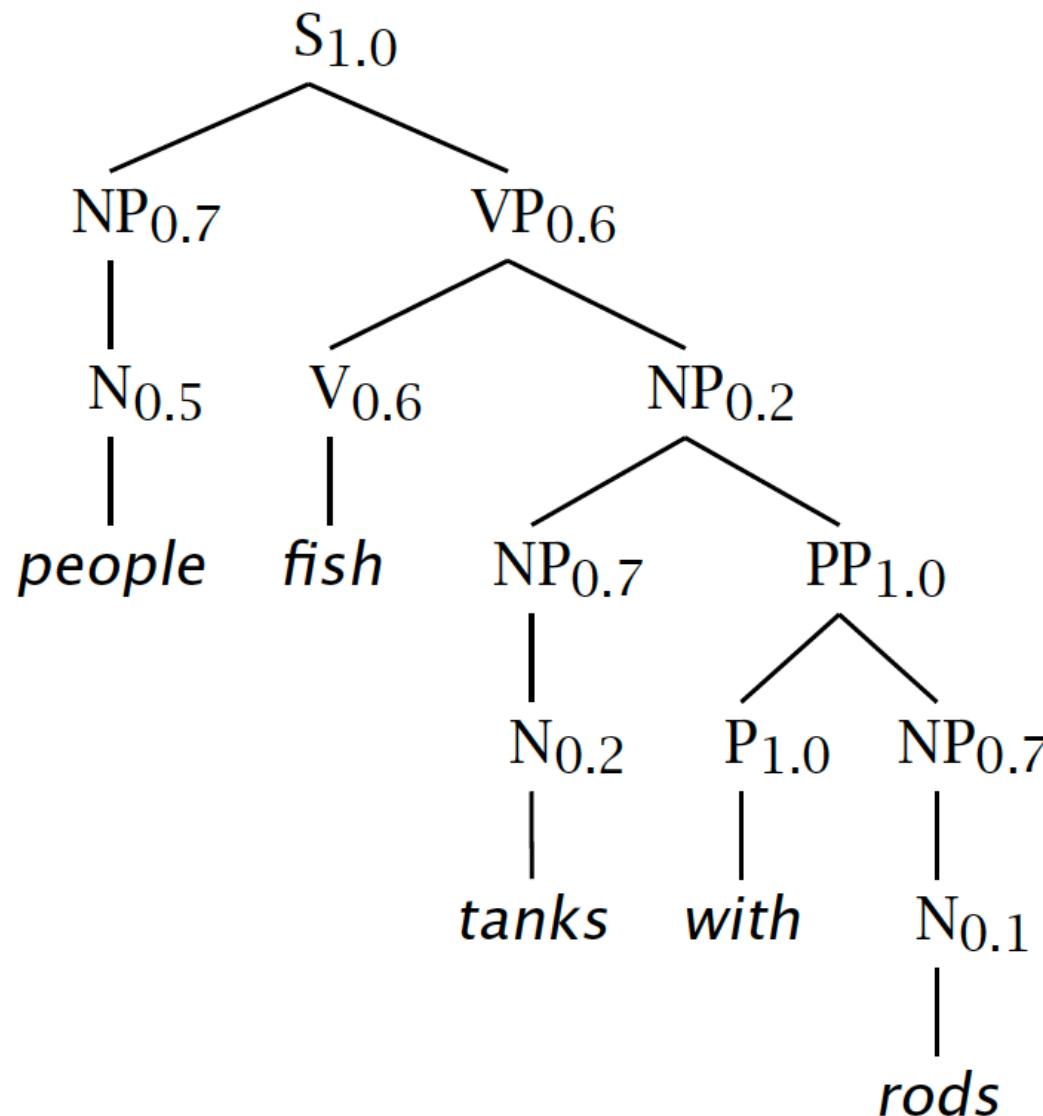
Tree and String Probabilities

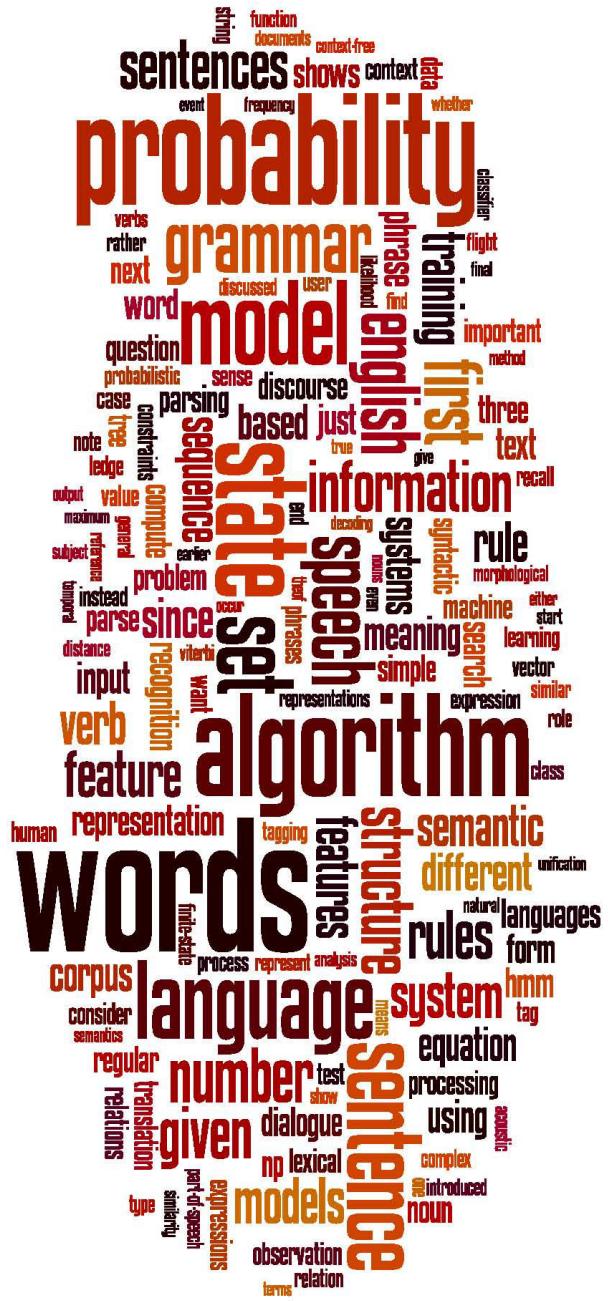
- $s = \text{people fish tanks with rods}$
- $P(t_1) = 1.0 \times 0.7 \times 0.4 \times 0.5 \times 0.6 \times 0.7 \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1$
$$= 0.0008232$$
- $P(t_2) = 1.0 \times 0.7 \times 0.6 \times 0.5 \times 0.6 \times 0.2 \times 0.7 \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1$
$$= 0.00024696$$
- $P(s) = P(t_1) + P(t_2)$
$$= 0.0008232 + 0.00024696$$
$$= 0.00107016$$

Verb attach

Noun attach

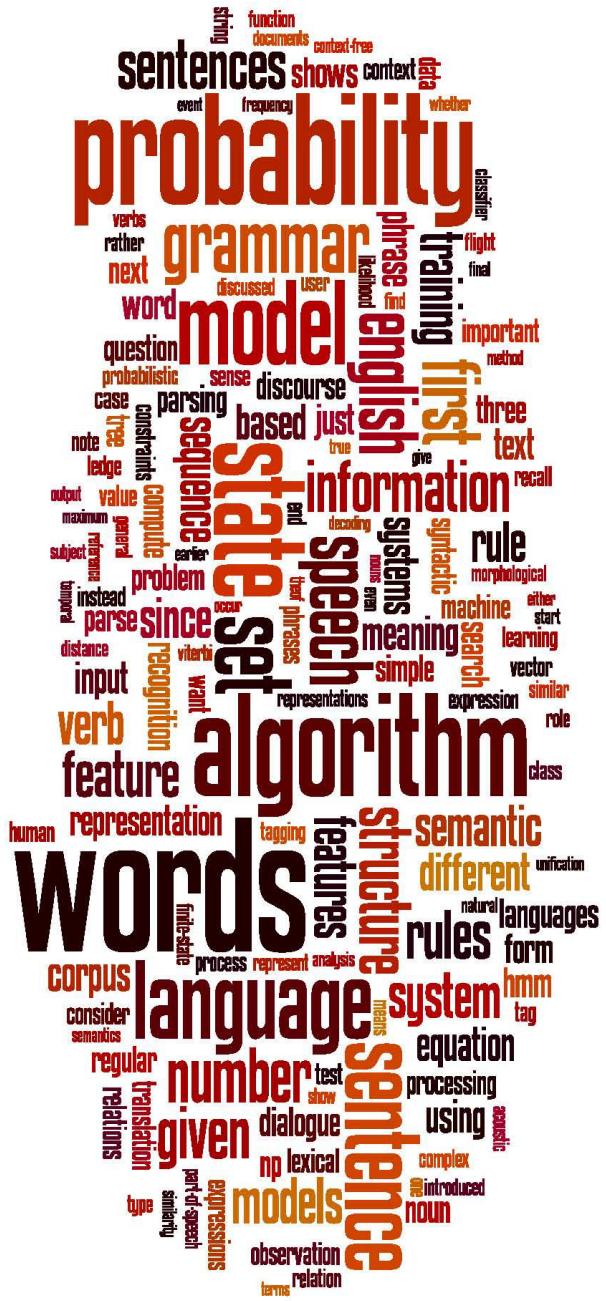


 $t_2:$ 



CFGs and PCFGs

(Probabilistic)
Context-Free
Grammars



Grammar Transforms

Restricting the grammar form for efficient parsing



Chomsky Normal Form

- All rules are of the form $X \rightarrow Y Z$ or $X \rightarrow w$
 - $X, Y, Z \in N$ and $w \in T$
- A transformation to this form doesn't change the weak generative capacity of a CFG
 - That is, it recognizes the same language
 - But maybe with different trees
- Empties and unaries are removed recursively
- n -ary rules are divided by introducing new nonterminals ($n > 2$)



A phrase structure grammar

 $S \rightarrow NP\ VP$ $N \rightarrow people$ $VP \rightarrow V\ NP$ $N \rightarrow fish$ $VP \rightarrow V\ NP\ PP$ $N \rightarrow tanks$ $NP \rightarrow NP\ NP$ $N \rightarrow rods$ $NP \rightarrow NP\ PP$ $V \rightarrow people$ $NP \rightarrow N$ $V \rightarrow fish$ $NP \rightarrow e$ $V \rightarrow tanks$ $PP \rightarrow P\ NP$ $P \rightarrow with$



Chomsky Normal Form steps

 $S \rightarrow NP\ VP$ $S \rightarrow VP$ $VP \rightarrow V\ NP$ $VP \rightarrow V$ $VP \rightarrow V\ NP\ PP$ $VP \rightarrow V\ PP$ $NP \rightarrow NP\ NP$ $NP \rightarrow NP$ $NP \rightarrow NP\ PP$ $NP \rightarrow PP$ $NP \rightarrow N$ $PP \rightarrow P\ NP$ $PP \rightarrow P$ $N \rightarrow people$ $N \rightarrow fish$ $N \rightarrow tanks$ $N \rightarrow rods$ $V \rightarrow people$ $V \rightarrow fish$ $V \rightarrow tanks$ $P \rightarrow with$



Chomsky Normal Form steps

 $S \rightarrow NP\ VP$ $VP \rightarrow V\ NP$ $S \rightarrow V\ NP$ $VP \rightarrow V$ $S \rightarrow V$ $VP \rightarrow V\ NP\ PP$ $S \rightarrow V\ NP\ PP$ $VP \rightarrow V\ PP$ $S \rightarrow V\ PP$ $NP \rightarrow NP\ NP$ $NP \rightarrow NP$ $NP \rightarrow NP\ PP$ $NP \rightarrow PP$ $NP \rightarrow N$ $PP \rightarrow P\ NP$ $PP \rightarrow P$ $N \rightarrow people$ $N \rightarrow fish$ $N \rightarrow tanks$ $N \rightarrow rods$ $V \rightarrow people$ $V \rightarrow fish$ $V \rightarrow tanks$ $P \rightarrow with$



Chomsky Normal Form steps

 $S \rightarrow NP\ VP$ $VP \rightarrow V\ NP$ $S \rightarrow V\ NP$ $VP \rightarrow V$ $VP \rightarrow V\ NP\ PP$ $S \rightarrow V\ NP\ PP$ $VP \rightarrow V\ PP$ $S \rightarrow V\ PP$ $NP \rightarrow NP\ NP$ $NP \rightarrow NP$ $NP \rightarrow NP\ PP$ $NP \rightarrow PP$ $NP \rightarrow N$ $PP \rightarrow P\ NP$ $PP \rightarrow P$ $N \rightarrow people$ $N \rightarrow fish$ $N \rightarrow tanks$ $N \rightarrow rods$ $V \rightarrow people$ $S \rightarrow people$ $V \rightarrow fish$ $S \rightarrow fish$ $V \rightarrow tanks$ $S \rightarrow tanks$ $P \rightarrow with$



Chomsky Normal Form steps

 $S \rightarrow NP\ VP$ $VP \rightarrow V\ NP$ $S \rightarrow V\ NP$ $VP \rightarrow V\ NP\ PP$ $S \rightarrow V\ NP\ PP$ $VP \rightarrow V\ PP$ $S \rightarrow V\ PP$ $NP \rightarrow NP\ NP$ $NP \rightarrow NP$ $NP \rightarrow NP\ PP$ $NP \rightarrow PP$ $NP \rightarrow N$ $PP \rightarrow P\ NP$ $PP \rightarrow P$ $N \rightarrow people$ $N \rightarrow fish$ $N \rightarrow tanks$ $N \rightarrow rods$ $V \rightarrow people$ $S \rightarrow people$ $VP \rightarrow people$ $V \rightarrow fish$ $S \rightarrow fish$ $VP \rightarrow fish$ $V \rightarrow tanks$ $S \rightarrow tanks$ $VP \rightarrow tanks$ $P \rightarrow with$



Chomsky Normal Form steps

$S \rightarrow NP\ VP$	$NP \rightarrow people$
$VP \rightarrow V\ NP$	$NP \rightarrow fish$
$S \rightarrow V\ NP$	$NP \rightarrow tanks$
$VP \rightarrow V\ NP\ PP$	$NP \rightarrow rods$
$S \rightarrow V\ NP\ PP$	$V \rightarrow people$
$VP \rightarrow V\ PP$	$S \rightarrow people$
$S \rightarrow V\ PP$	$VP \rightarrow people$
$NP \rightarrow NP\ NP$	$V \rightarrow fish$
$NP \rightarrow NP\ PP$	$S \rightarrow fish$
$NP \rightarrow P\ NP$	$VP \rightarrow fish$
$PP \rightarrow P\ NP$	$V \rightarrow tanks$
	$S \rightarrow tanks$
	$VP \rightarrow tanks$
	$P \rightarrow with$
	$PP \rightarrow with$



Chomsky Normal Form steps

$S \rightarrow NP\ VP$	$NP \rightarrow people$
$VP \rightarrow V\ NP$	$NP \rightarrow fish$
$S \rightarrow V\ NP$	$NP \rightarrow tanks$
$VP \rightarrow V\ @VP_V$	$NP \rightarrow rods$
$@VP_V \rightarrow NP\ PP$	$V \rightarrow people$
$S \rightarrow V\ @S_V$	$S \rightarrow people$
$@S_V \rightarrow NP\ PP$	$VP \rightarrow people$
$VP \rightarrow V\ PP$	$V \rightarrow fish$
$S \rightarrow V\ PP$	$S \rightarrow fish$
$NP \rightarrow NP\ NP$	$VP \rightarrow fish$
$NP \rightarrow NP\ PP$	$V \rightarrow tanks$
$NP \rightarrow P\ NP$	$S \rightarrow tanks$
$PP \rightarrow P\ NP$	$VP \rightarrow tanks$
	$P \rightarrow with$
	$PP \rightarrow with$



A phrase structure grammar

 $S \rightarrow NP\ VP$ $N \rightarrow people$ $VP \rightarrow V\ NP$ $N \rightarrow fish$ $VP \rightarrow V\ NP\ PP$ $N \rightarrow tanks$ $NP \rightarrow NP\ NP$ $N \rightarrow rods$ $NP \rightarrow NP\ PP$ $V \rightarrow people$ $NP \rightarrow N$ $V \rightarrow fish$ $NP \rightarrow e$ $V \rightarrow tanks$ $PP \rightarrow P\ NP$ $P \rightarrow with$



Chomsky Normal Form steps

$S \rightarrow NP\ VP$	$NP \rightarrow people$
$VP \rightarrow V\ NP$	$NP \rightarrow fish$
$S \rightarrow V\ NP$	$NP \rightarrow tanks$
$VP \rightarrow V\ @VP_V$	$NP \rightarrow rods$
$@VP_V \rightarrow NP\ PP$	$V \rightarrow people$
$S \rightarrow V\ @S_V$	$S \rightarrow people$
$@S_V \rightarrow NP\ PP$	$VP \rightarrow people$
$VP \rightarrow V\ PP$	$V \rightarrow fish$
$S \rightarrow V\ PP$	$S \rightarrow fish$
$NP \rightarrow NP\ NP$	$VP \rightarrow fish$
$NP \rightarrow NP\ PP$	$V \rightarrow tanks$
$NP \rightarrow P\ NP$	$S \rightarrow tanks$
$PP \rightarrow P\ NP$	$VP \rightarrow tanks$
	$P \rightarrow with$
	$PP \rightarrow with$

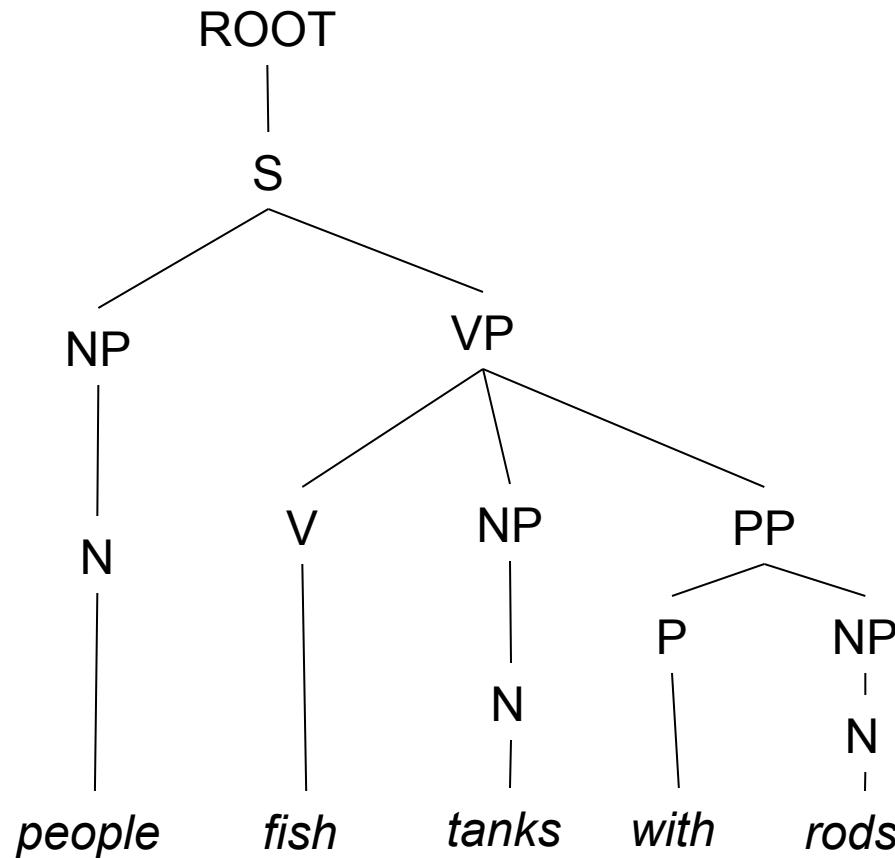


Chomsky Normal Form

- You should think of this as a transformation for efficient parsing
- With some extra book-keeping in symbol names, you can even reconstruct the same trees with a detransform
- In practice full Chomsky Normal Form is a pain
 - Reconstructing n-aries is easy
 - Reconstructing unaries/empties is trickier
- **Binarization** is crucial for cubic time CFG parsing
- The rest isn't necessary; it just makes the algorithms cleaner and a bit quicker

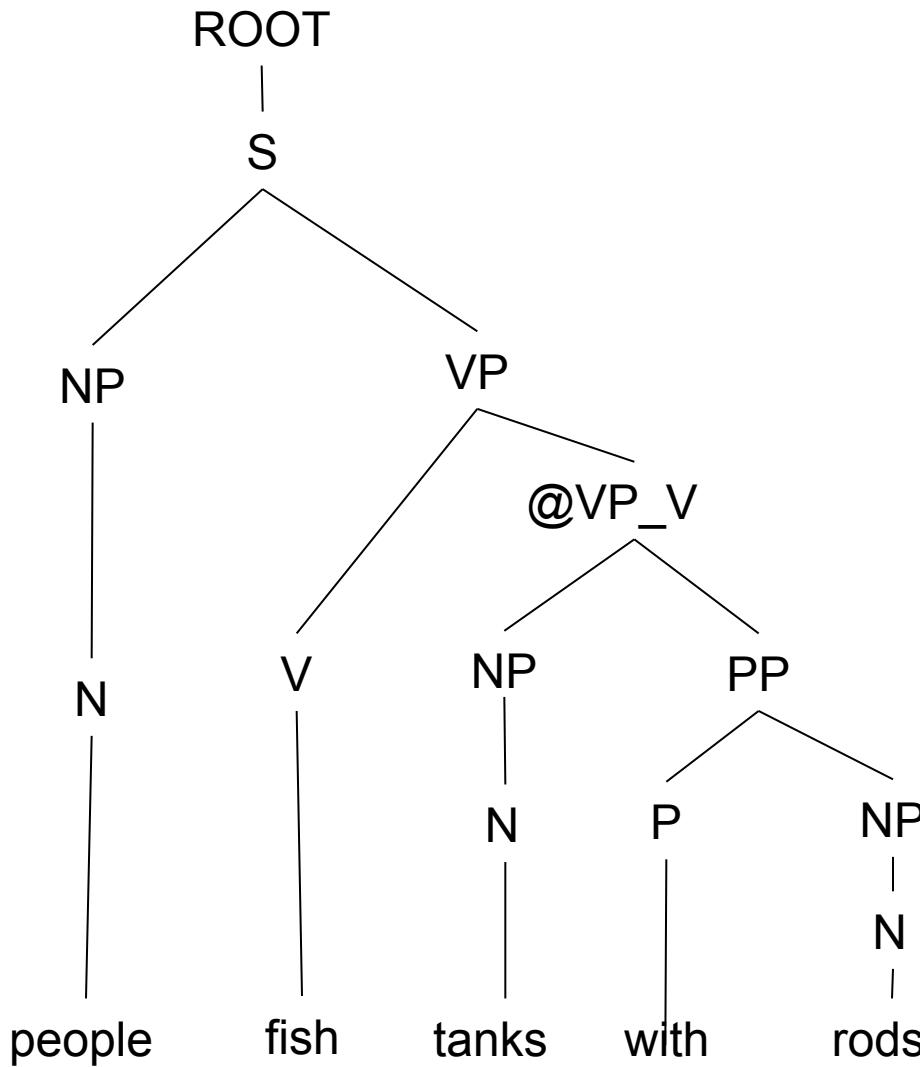


An example: before binarization...



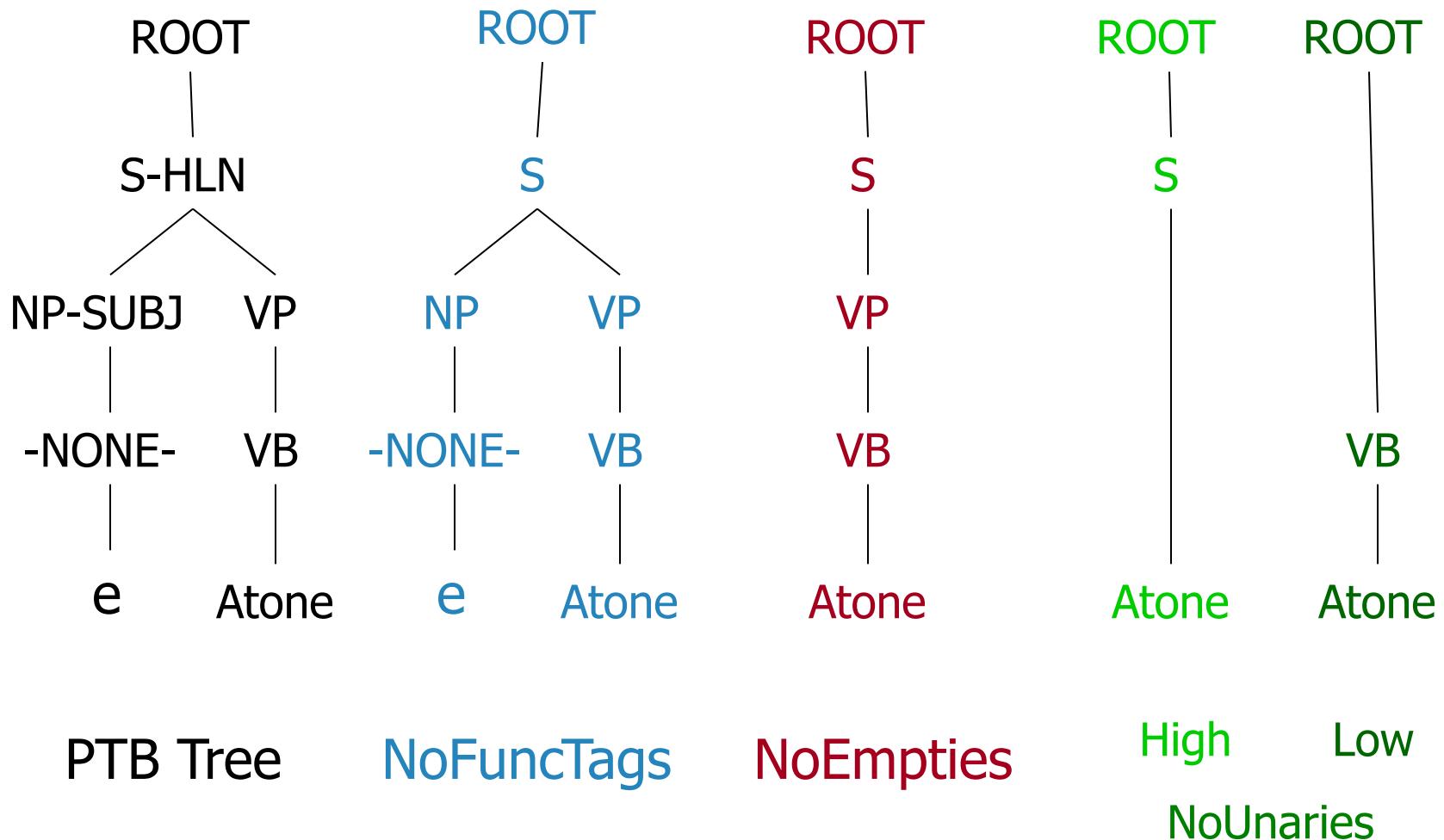


After binarization...



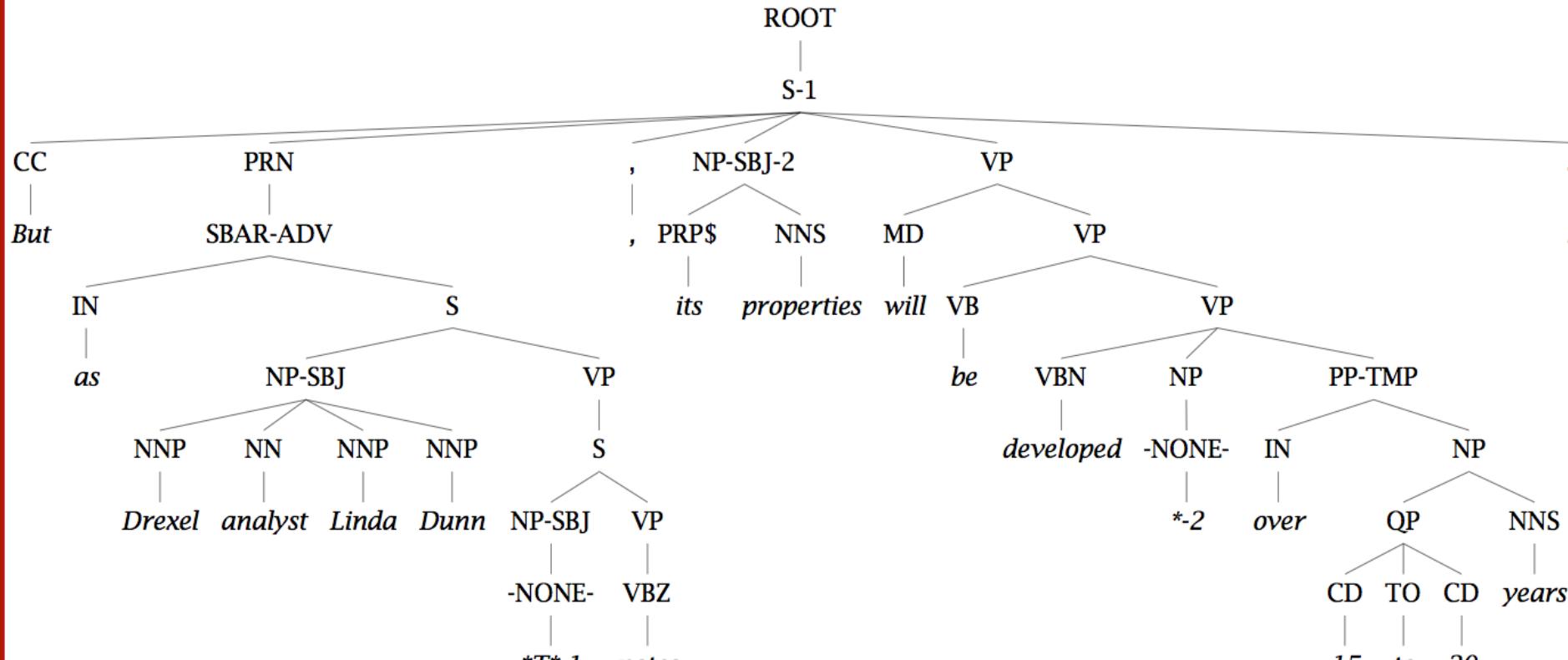


Treebank: empties and unaries



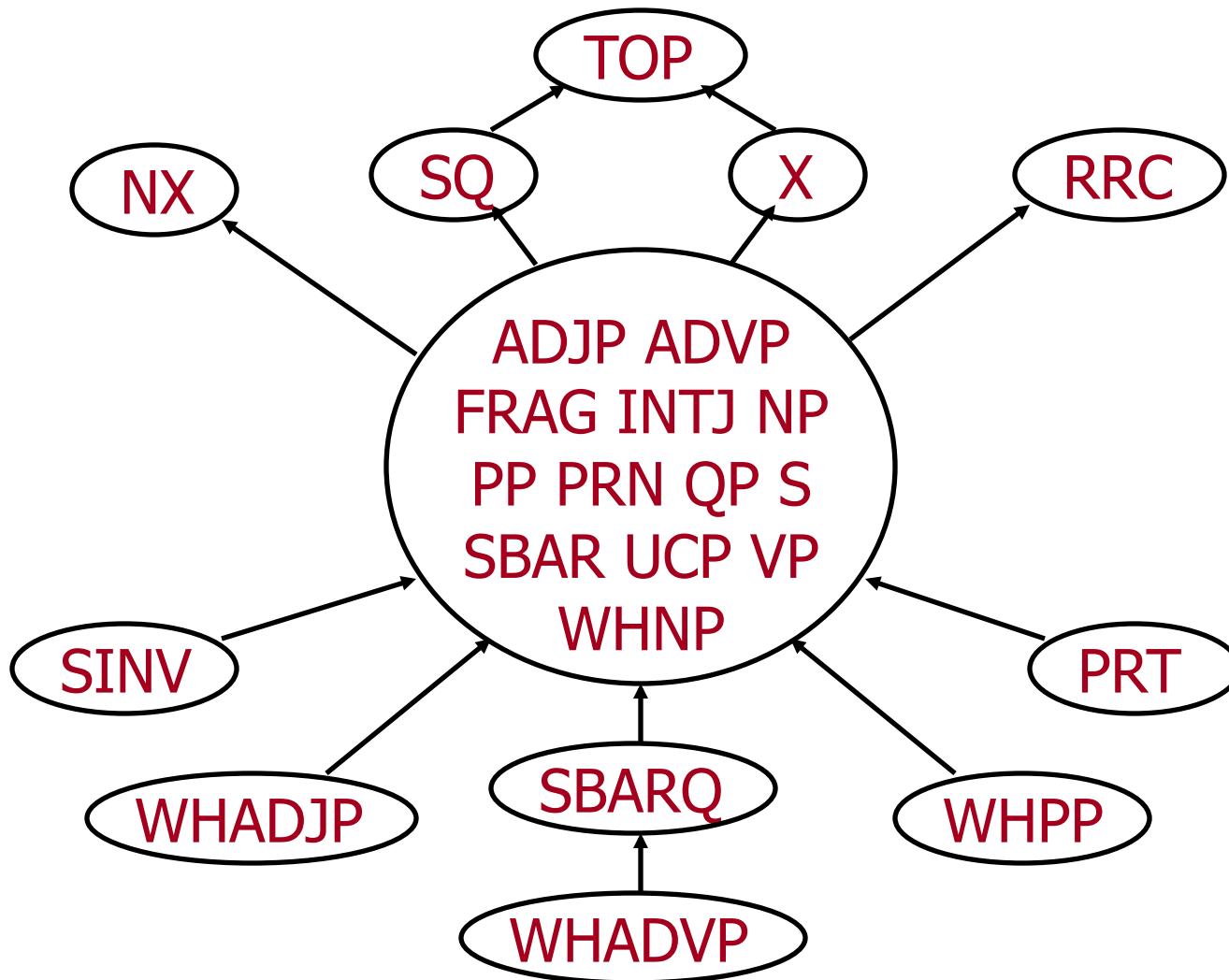


Unary rules: alchemy in the land of treebanks

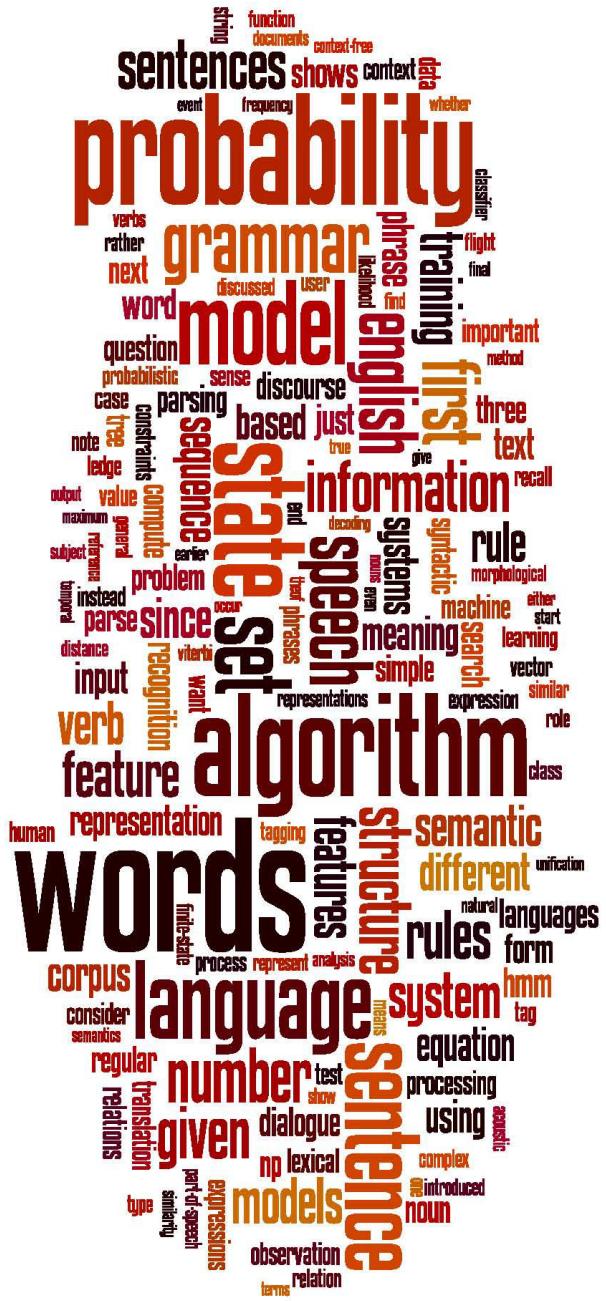




Same-Span Reachability

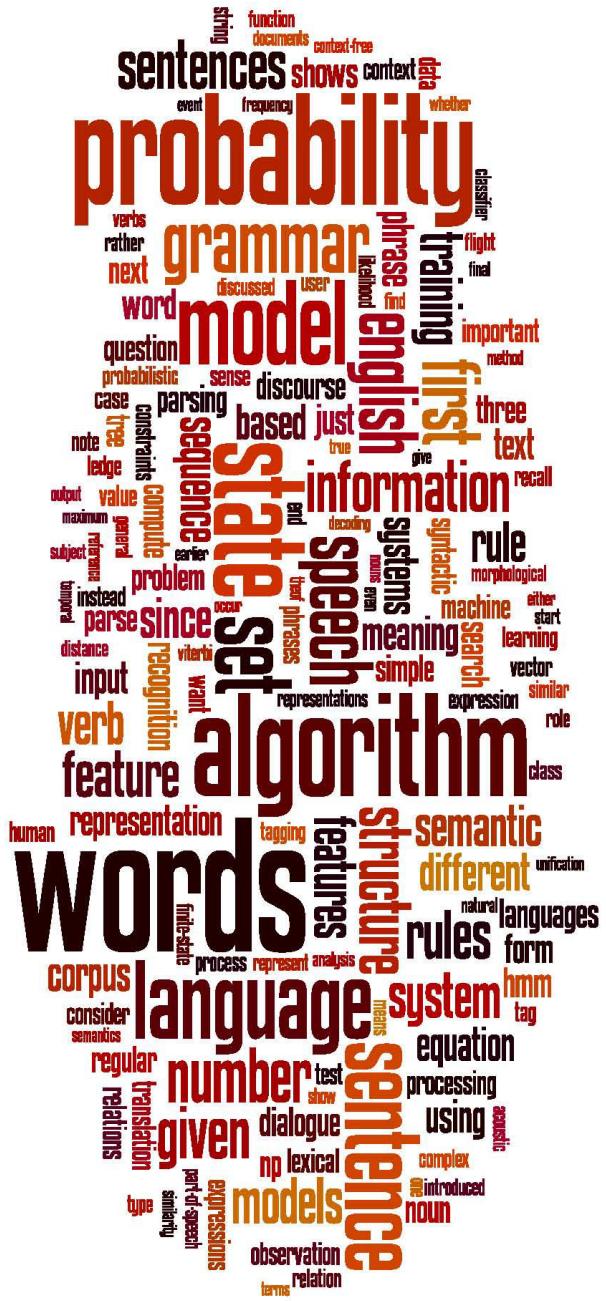


NoEmpties



Grammar Transforms

Restricting the grammar form for efficient parsing

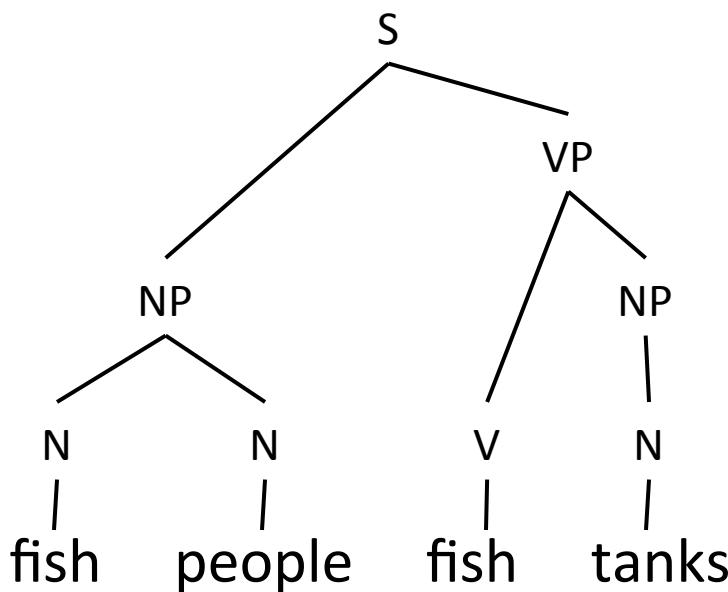


CKY Parsing

Exact polynomial time parsing of (P)CFGs



Constituency Parsing



PCFG

Rule Prob θ_i

$S \rightarrow NP\ VP$ θ_0

$NP \rightarrow NP\ NP$ θ_1

...

$N \rightarrow fish$ θ_{42}

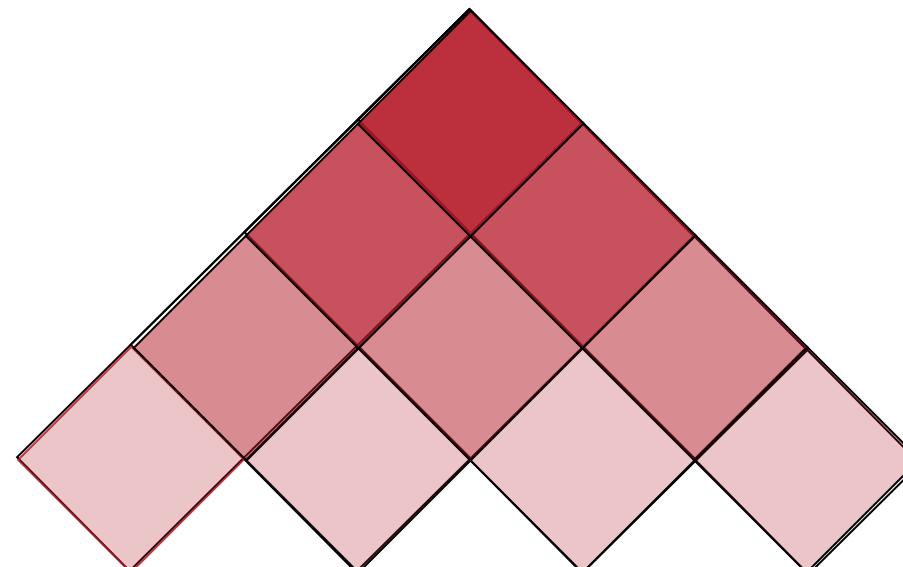
$N \rightarrow people$ θ_{43}

$V \rightarrow fish$ θ_{44}

...



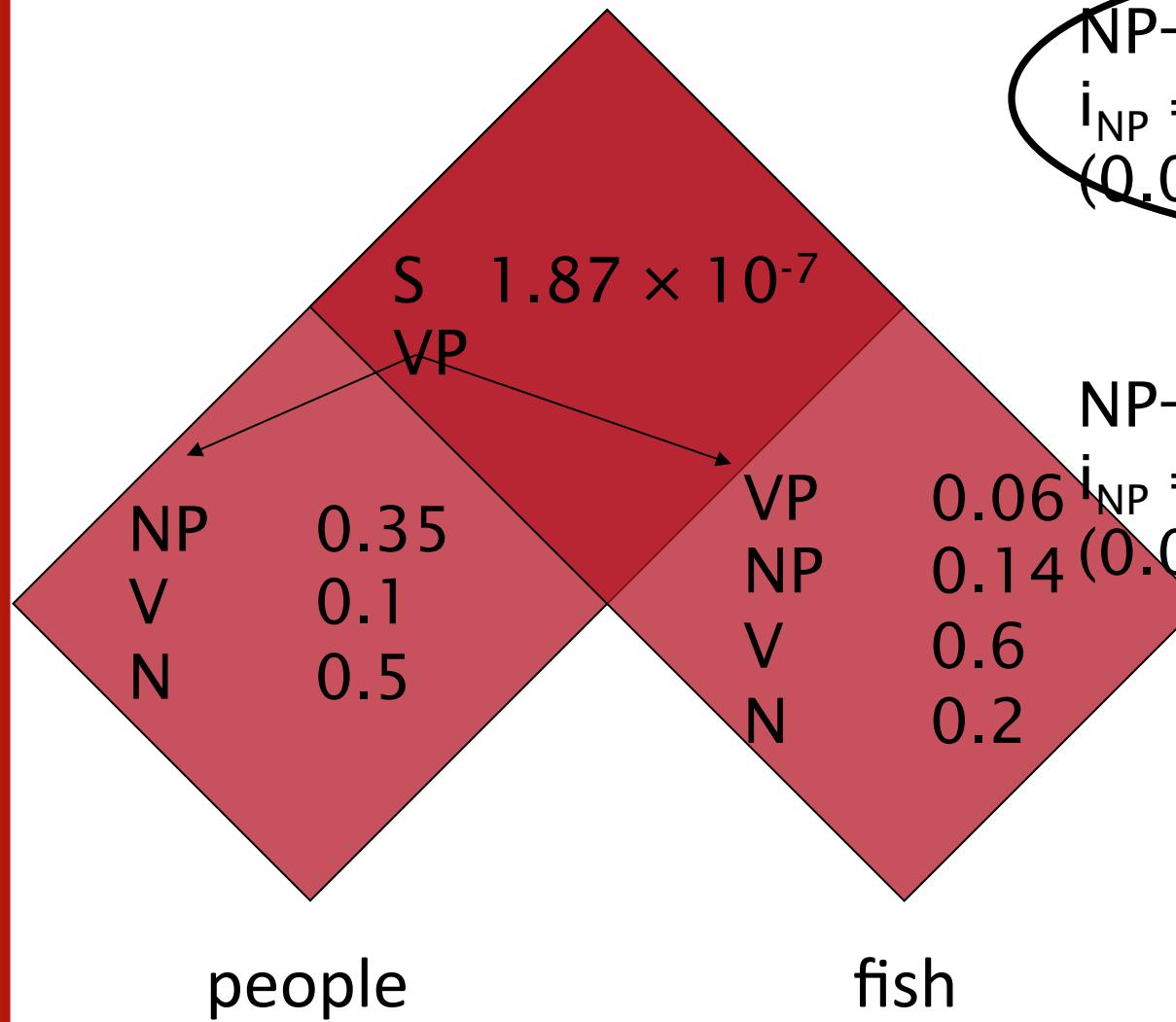
Cocke-Kasami-Younger (CKY) Constituency Parsing



fish people fish tanks



Viterbi (Max) Scores

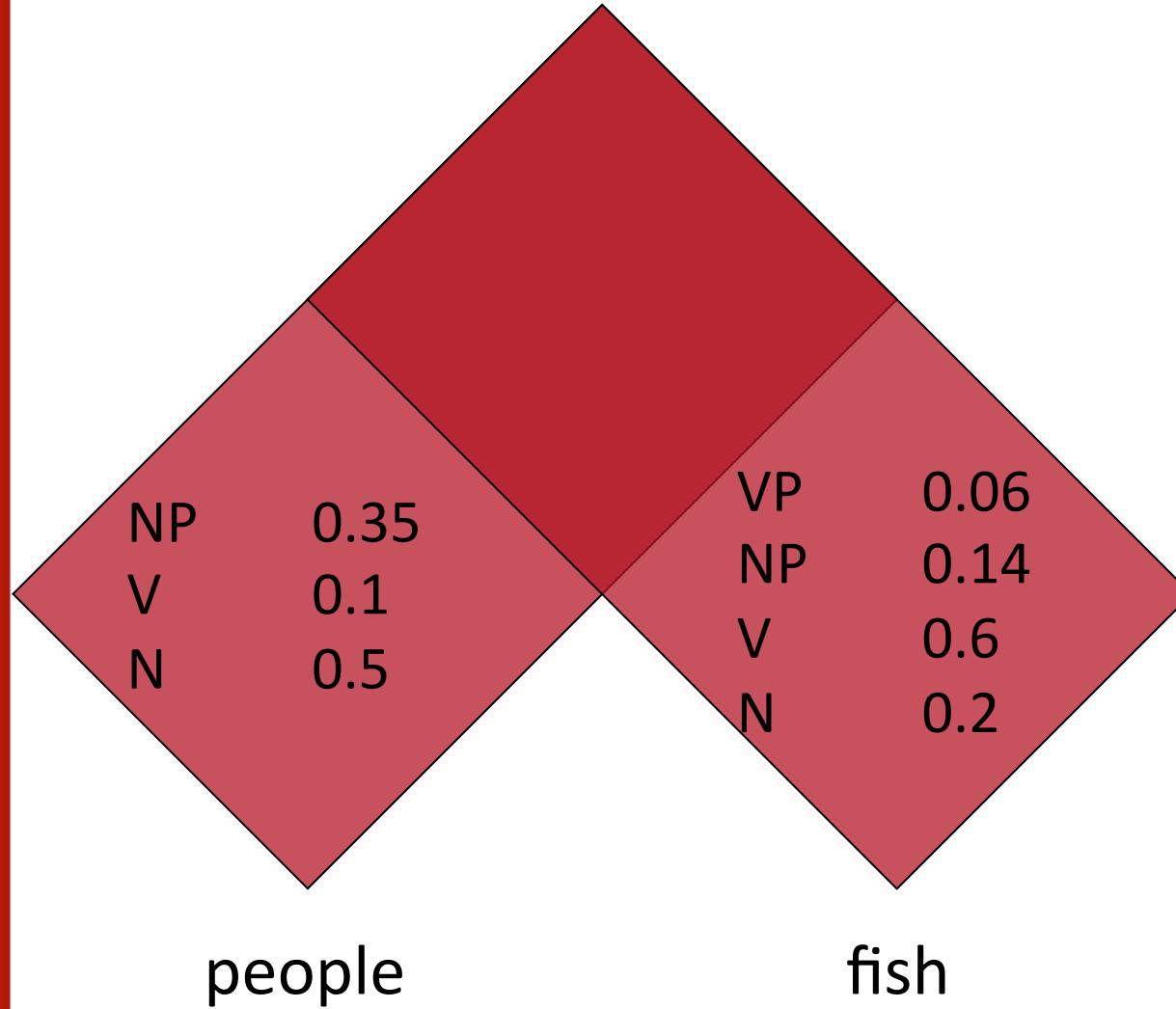


$$\begin{aligned} & \text{NP} \rightarrow \text{NN NNS} & 0.13 \\ & i_{\text{NP}} = (0.13)(0.0023) \\ & (0.0014) \\ & = 1.87 \times 10^{-7} \end{aligned}$$

$$\begin{aligned} & \text{NP} \rightarrow \text{NNP NNS} & 0.056 \\ & i_{\text{NP}} = (0.056)(0.001) \\ & (0.0014) \\ & = 7.84 \times 10^{-8} \end{aligned}$$



Viterbi (Max) Scores



$S \rightarrow NP\ VP$	0.9
$S \rightarrow VP$	0.1
$VP \rightarrow V\ NP$	0.5
$VP \rightarrow V$	0.1
$VP \rightarrow V @VP_V$	0.3
$VP \rightarrow V\ PP$	0.1
$@VP_V \rightarrow NP\ PP$	1.0
$NP \rightarrow NP\ NP$	0.1
$NP \rightarrow NP\ PP$	0.2
$NP \rightarrow N$	0.7
$PP \rightarrow P\ NP$	1.0



Extended CKY parsing

- Unaries can be incorporated into the algorithm
 - Messy, but doesn't increase algorithmic complexity
- Empties can be incorporated
 - Use fenceposts
 - Doesn't increase complexity; essentially like unaries
- Binarization is *vital*
 - Without binarization, you don't get parsing cubic in the length of the sentence and in the number of nonterminals in the grammar
 - Binarization may be an explicit transformation or implicit in how the parser works (Early-style dotted rules), but it's always there.



The CKY algorithm (1960/1965)

... extended to unaries

```
function CKY(words, grammar) returns [most_probable_parse, prob]
    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 -> words[i] in grammar
                score[i][i+1][A] = P(A -> words[i])
        //handle unaries
        boolean added = true
        while added
            added = false
            for A, B in nonterms
                if score[i][i+1][B] > 0 && A->B in grammar
                    prob = P(A->B)*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
```



The CKY algorithm (1960/1965)

... extended to unaries

```
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->BC)
                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->B)*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)
```



Quiz Question!

NNS 0.0023
VB 0.001

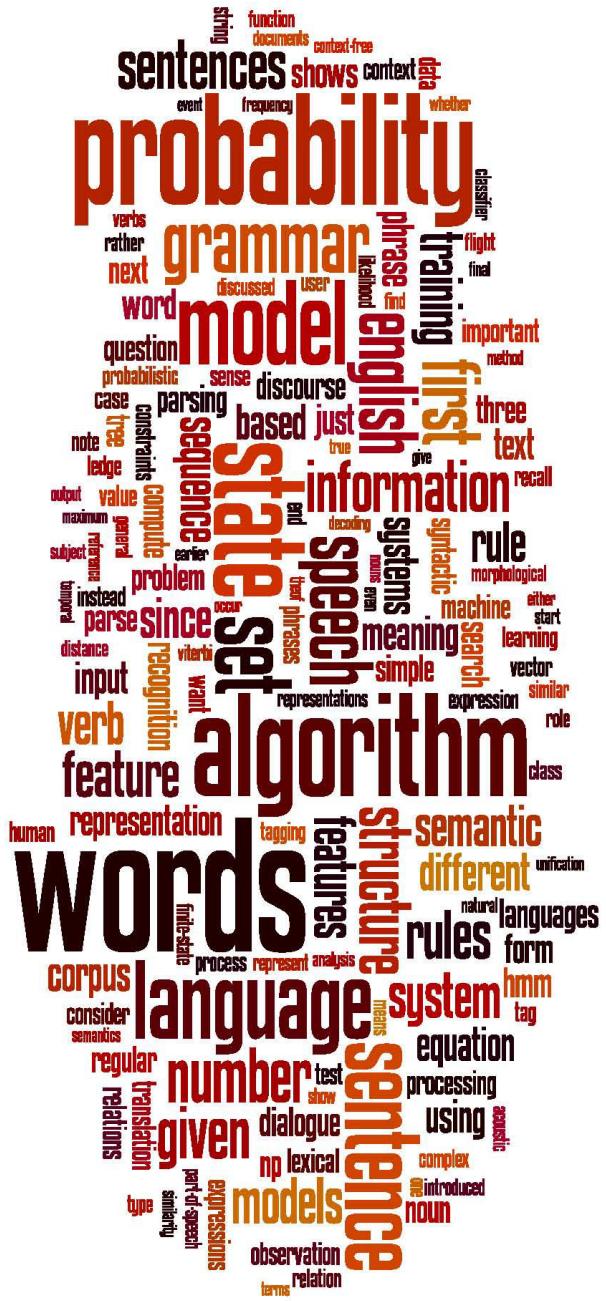
runs

PP 0.2
IN 0.0014
NNS 0.0001

down

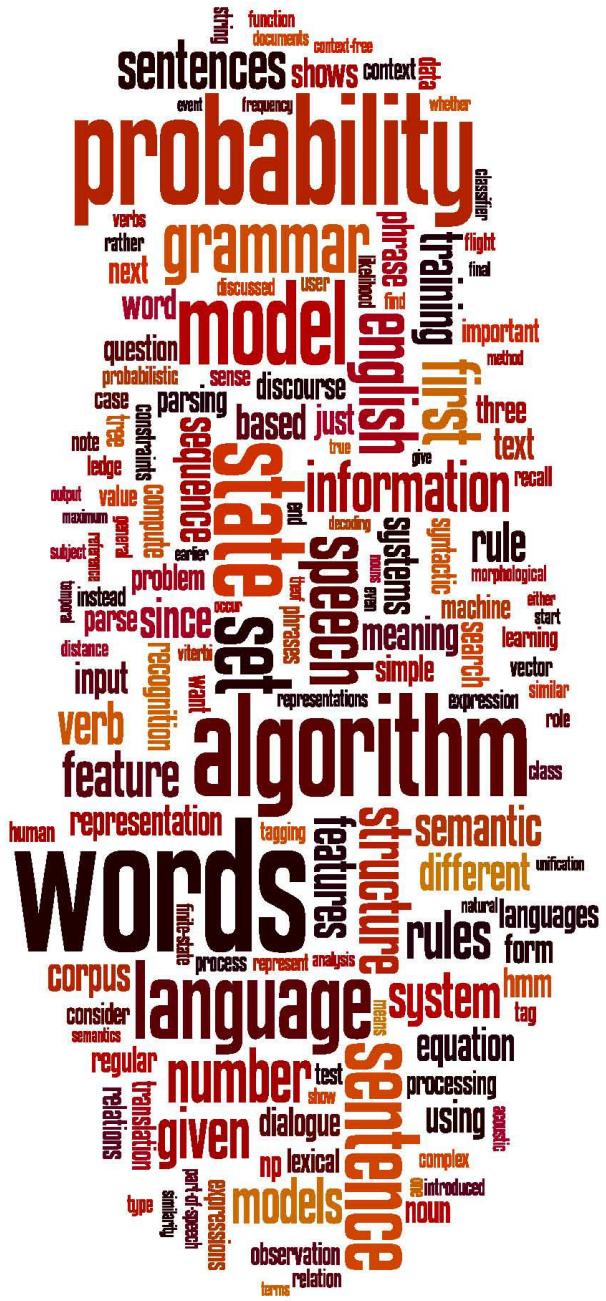
PP → IN	0.002
NP → NNS NNS	0.01
NP → NNS NP	0.005
NP → NNS PP	0.01
VP → VB PP	0.045
VP → VB NP	0.015

What constituents (with what probability can you make?)



CKY Parsing

Exact polynomial time parsing of (P)CFGs



CKY Parsing

A worked example



The grammar: Binary, no epsilons,

$S \rightarrow NP\ VP$	0.9	$N \rightarrow people$	0.5
$S \rightarrow VP$	0.1	$N \rightarrow fish$	0.2
$VP \rightarrow V\ NP$	0.5	$N \rightarrow tanks$	0.2
$VP \rightarrow V$	0.1	$N \rightarrow rods$	0.1
$VP \rightarrow V @VP_V$	0.3	$V \rightarrow people$	0.1
$VP \rightarrow V\ PP$	0.1	$V \rightarrow fish$	0.6
$@VP_V \rightarrow NP\ PP$	1.0	$V \rightarrow tanks$	0.3
$NP \rightarrow NP\ NP$	0.1	$P \rightarrow with$	1.0
$NP \rightarrow NP\ PP$	0.2		
$NP \rightarrow N$	0.7		
$PP \rightarrow P\ NP$	1.0		

	fish	1	people	2	fish	3	tanks	4
0	score[0][1]		score[0][2]		score[0][3]		score[0][4]	
1								
2			score[1][2]		score[1][3]		score[1][4]	
3					score[2][3]		score[2][4]	
4							score[3][4]	

$S \rightarrow NP VP$

0.9

$S \rightarrow VP$

0.1

$VP \rightarrow V NP$

0.5

$VP \rightarrow V$

0.1

$VP \rightarrow V @VP_V$

0.3

$VP \rightarrow V PP$

0.1

$@VP_V \rightarrow NP PP$

1.0

$NP \rightarrow NP NP$

0.1

$NP \rightarrow NP PP$

0.2

$NP \rightarrow N$

0.7

$PP \rightarrow P NP$

1.0

$N \rightarrow people$

0.5

$N \rightarrow fish$

0.2

$N \rightarrow tanks$

0.2

$N \rightarrow rods$

0.1

$V \rightarrow people$

0.1

$V \rightarrow fish$

0.6

$V \rightarrow tanks$

0.3

$P \rightarrow with$

1.0



		0	fish	1	people	2	fish	3	tanks	4
$S \rightarrow NP VP$	0.9									
$S \rightarrow VP$	0.1									
$VP \rightarrow V NP$	0.5									
$VP \rightarrow V$	0.1									
$VP \rightarrow V @VP_V$	0.3									
$VP \rightarrow V PP$	0.1									
$@VP_V \rightarrow NP PP$	1.0									
$NP \rightarrow NP NP$	0.1									
$NP \rightarrow NP PP$	0.2									
$NP \rightarrow N$	0.7	2								
$PP \rightarrow P NP$	1.0									
$N \rightarrow people$	0.5									
$N \rightarrow fish$	0.2									
$N \rightarrow tanks$	0.2									
$N \rightarrow rods$	0.1									
$V \rightarrow people$	0.1									
$V \rightarrow fish$	0.6									
$V \rightarrow tanks$	0.3									
$P \rightarrow with$	1.0									

```

// handle unaries
boolean added = true
while added
    added = false
    for A, B in nonterms
        if score[i][i+1][B] > 0 && A->B in grammar
            prob = P(A->B)*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

```

		fish	people	fish	tanks	
	0	N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006				4
S → NP VP	0.9					
S → VP	0.1					
VP → V NP	0.5					
VP → V	0.1					
VP → V @VP_V	0.3					
VP → V PP	0.1		N → people 0.5 V → people 0.1 NP → N 0.35 VP → V 0.01 S → VP 0.001			
@VP_V → NP PP	1.0					
NP → NP NP	0.1					
NP → NP PP	0.2					
NP → N	0.7					
PP → P NP	1.0			N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006		
N → people	0.5					
N → fish	0.2					
N → tanks	0.2					
N → rods	0.1	prob=score[begin][split][B]*score[split][end][C]*P(A->BC) if (prob > score[begin][end][A]) score[begin][end][A] = prob back[begin][end][A] = new Triple(split,B,C)				
V → people	0.1					
V → fish	0.6					
V → tanks	0.3					
P → with	1.0					

		fish	1	people	2	fish	3	tanks	4
$S \rightarrow NP VP$	0.9	0	$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.105 $S \rightarrow NP VP$ 0.00126					
$S \rightarrow VP$	0.1			$N \rightarrow people 0.5$ $V \rightarrow people 0.1$ $NP \rightarrow N 0.35$ $VP \rightarrow V 0.01$ $S \rightarrow VP 0.001$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.007 $S \rightarrow NP VP$ 0.0189				
$VP \rightarrow V NP$	0.5					$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.00196 $VP \rightarrow V NP$ 0.042 $S \rightarrow NP VP$ 0.00378		
$VP \rightarrow V$	0.1						$N \rightarrow tanks 0.2$ $V \rightarrow tanks 0.1$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.03$ $S \rightarrow VP 0.003$		
$VP \rightarrow V @VP_V$	0.3								
$VP \rightarrow V PP$	0.1								
$@VP_V \rightarrow NP PP$	1.0								
$NP \rightarrow NP NP$	0.1								
$NP \rightarrow NP PP$	0.2								
$NP \rightarrow N$	0.7	2							
$PP \rightarrow P NP$	1.0								
$N \rightarrow people$	0.5								
$N \rightarrow fish$	0.2								
$N \rightarrow tanks$	0.2								
$N \rightarrow rods$	0.1								
$V \rightarrow people$	0.1								
$V \rightarrow fish$	0.6								
$V \rightarrow tanks$	0.3								
$P \rightarrow with$	1.0								
		3	<pre>//handle unaries boolean added = true while added added = false for A, B in nonterms prob = P(A->B)*score[begin][end][B]; if prob > score[begin][end][A] score[begin][end][A] = prob back[begin][end][A] = B added = true </pre>						
		4							

		fish	people	fish	tanks	
	0	N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.0049 VP → V NP 0.105 <i>S → VP 0.0105</i>			4
	1		N → people 0.5 V → people 0.1 NP → N 0.35 VP → V 0.01 S → VP 0.001	NP → NP NP 0.0049 VP → V NP 0.007 S → NP VP 0.0189		
	2			N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.00196 VP → V NP 0.042 <i>S → VP 0.0042</i>	
	3				N → tanks 0.2 V → tanks 0.1 NP → N 0.14 VP → V 0.03 S → VP 0.003	
	4	for split = begin+1 to end-1 for A,B,C in nonterms prob=score[begin][split][B]*score[split][end][C]*P(A->BC) if prob > score[begin][end][A] score[begin][end][A] = prob back[begin][end][A] = new Triple(split,B,C)				
S → NP VP	0.9					
S → VP	0.1					
VP → V NP	0.5					
VP → V	0.1					
VP → V @VP_V	0.3					
VP → V PP	0.1					
@VP_V → NP PP	1.0					
NP → NP NP	0.1					
NP → NP PP	0.2					
NP → N	0.7					
PP → P NP	1.0					
N → people	0.5					
N → fish	0.2					
N → tanks	0.2					
N → rods	0.1					
V → people	0.1					
V → fish	0.6					
V → tanks	0.3					
P → with	1.0					

		fish	1	people	2	fish	3	tanks	4
$S \rightarrow NP VP$	0.9	0	$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.105 $S \rightarrow VP$ 0.0105	$NP \rightarrow NP NP$ 0.0000686 $VP \rightarrow V NP$ 0.00147 $S \rightarrow NP VP$ 0.000882				
$S \rightarrow VP$	0.1			$N \rightarrow people 0.5$ $V \rightarrow people 0.1$ $NP \rightarrow N 0.35$ $VP \rightarrow V 0.01$ $S \rightarrow VP 0.001$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.007 $S \rightarrow NP VP$ 0.0189				
$VP \rightarrow V NP$	0.5				$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.00196 $VP \rightarrow V NP$ 0.042 $S \rightarrow VP$ 0.0042			
$VP \rightarrow V$	0.1								
$VP \rightarrow V @VP_V$	0.3								
$VP \rightarrow V PP$	0.1								
$@VP_V \rightarrow NP PP$	1.0								
$NP \rightarrow NP NP$	0.1								
$NP \rightarrow NP PP$	0.2								
$NP \rightarrow N$	0.7	2							
$PP \rightarrow P NP$	1.0								
$N \rightarrow people$	0.5								
$N \rightarrow fish$	0.2								
$N \rightarrow tanks$	0.2	3							
$N \rightarrow rods$	0.1								
$V \rightarrow people$	0.1								
$V \rightarrow fish$	0.6								
$V \rightarrow tanks$	0.3								
$P \rightarrow with$	1.0	4							

```

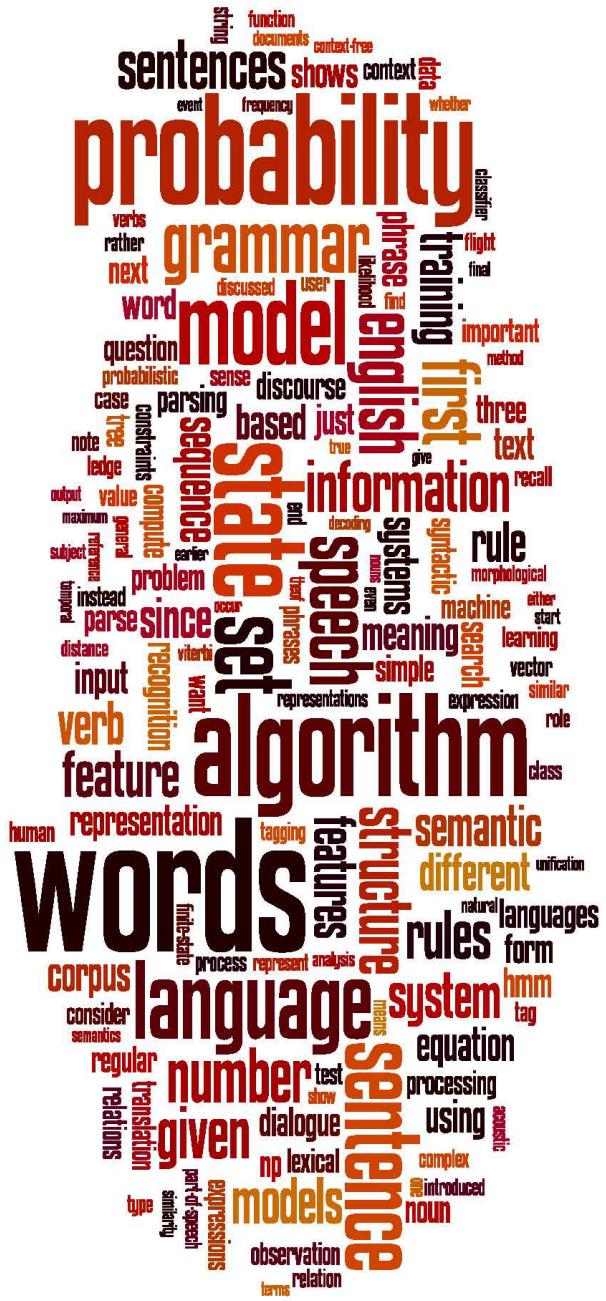
for split = begin+1 to end-1
  for A,B,C in nonterms
    prob=score[begin][split][B]*score[split][end][C]*P(A->BC)
    if prob > score[begin][end][A]
      score[begin][end][A] = prob
      back[begin][end][A] = new Triple(split,B,C)
  
```

		fish	1	people	2	fish	3	tanks	4
$S \rightarrow NP VP$	0.9	0	$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.105 $S \rightarrow VP$ 0.0105	$NP \rightarrow NP NP$ 0.0000686 $VP \rightarrow V NP$ 0.00147 $S \rightarrow NP VP$ 0.000882				
$S \rightarrow VP$	0.1			$N \rightarrow people 0.5$ $V \rightarrow people 0.1$ $NP \rightarrow N 0.35$ $VP \rightarrow V 0.01$ $S \rightarrow VP 0.001$	$NP \rightarrow NP NP$ 0.0049 $VP \rightarrow V NP$ 0.007 $S \rightarrow NP VP$ 0.0189	$NP \rightarrow NP NP$ 0.0000686 $VP \rightarrow V NP$ 0.000098 $S \rightarrow NP VP$ 0.01323			
$VP \rightarrow V NP$	0.5				$N \rightarrow fish 0.2$ $V \rightarrow fish 0.6$ $NP \rightarrow N 0.14$ $VP \rightarrow V 0.06$ $S \rightarrow VP 0.006$	$NP \rightarrow NP NP$ 0.00196 $VP \rightarrow V NP$ 0.042 $S \rightarrow VP$ 0.0042			
$VP \rightarrow V$	0.1								
$VP \rightarrow V @VP_V$	0.3								
$VP \rightarrow V PP$	0.1								
$@VP_V \rightarrow NP PP$	1.0								
$NP \rightarrow NP NP$	0.1								
$NP \rightarrow NP PP$	0.2								
$NP \rightarrow N$	0.7	2							
$PP \rightarrow P NP$	1.0								
$N \rightarrow people$	0.5								
$N \rightarrow fish$	0.2								
$N \rightarrow tanks$	0.2	3							
$N \rightarrow rods$	0.1								
$V \rightarrow people$	0.1								
$V \rightarrow fish$	0.6								
$V \rightarrow tanks$	0.3								
$P \rightarrow with$	1.0	4							

```

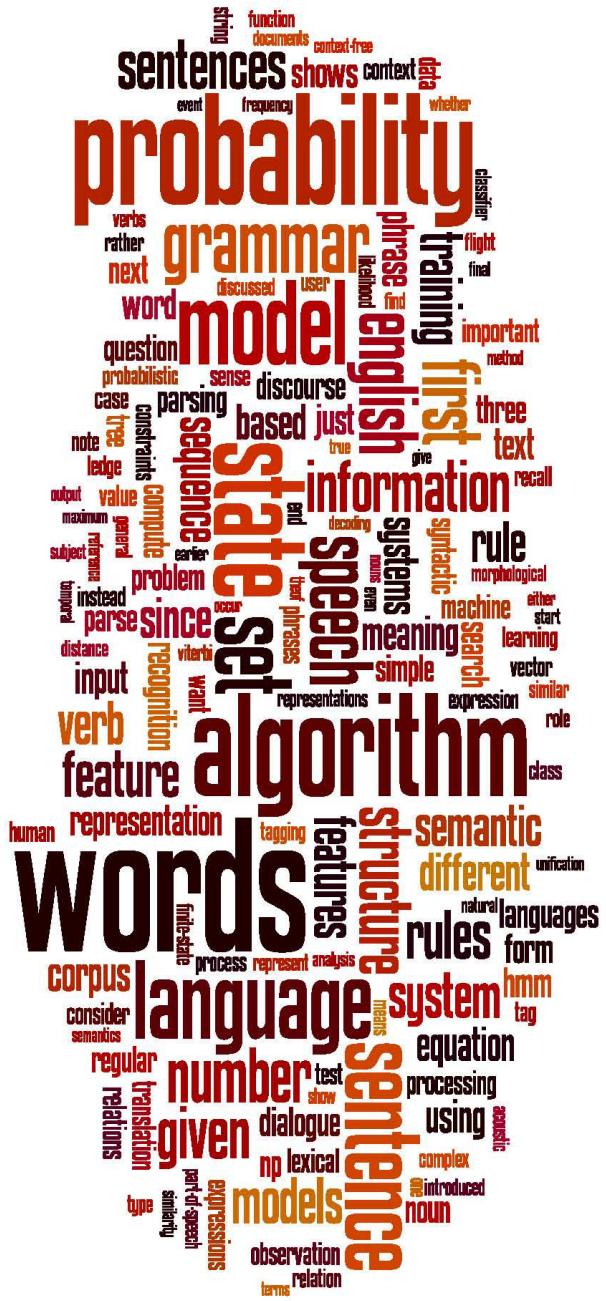
for split = begin+1 to end-1
  for A,B,C in nonterms
    prob=score[begin][split][B]*score[split][end][C]*P(A->BC)
    if prob > score[begin][end][A]
      score[begin][end][A] = prob
      back[begin][end][A] = new Triple(split,B,C)
  
```

		fish	people	fish	tanks	
	0	N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.0049 VP → V NP 0.105 S → VP 0.0105	NP → NP NP 0.0000686 VP → V NP 0.00147 S → NP VP 0.000882	NP → NP NP 0.0000009604 VP → V NP 0.00002058 S → NP VP 0.00018522	4
	1		N → people 0.5 V → people 0.1 NP → N 0.35 VP → V 0.01 S → VP 0.001	NP → NP NP 0.0049 VP → V NP 0.007 S → NP VP 0.0189	NP → NP NP 0.0000686 VP → V NP 0.000098 S → NP VP 0.01323	2
	2			N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.00196 VP → V NP 0.042 S → VP 0.0042	3
	3				N → tanks 0.2 V → tanks 0.1 NP → N 0.14 VP → V 0.03 S → VP 0.003	4
P → with	4		Call buildTree(score, back) to get the best parse			
S → NP VP	0.9					
S → VP	0.1					
VP → V NP	0.5					
VP → V	0.1					
VP → V @VP_V	0.3					
VP → V PP	0.1					
@VP_V → NP PP	1.0					
NP → NP NP	0.1					
NP → NP PP	0.2					
NP → N	0.7					
PP → P NP	1.0					
N → people	0.5					
N → fish	0.2					
N → tanks	0.2					
N → rods	0.1					
V → people	0.1					
V → fish	0.6					
V → tanks	0.3					
P → with	1.0					



CKY Parsing

A worked example

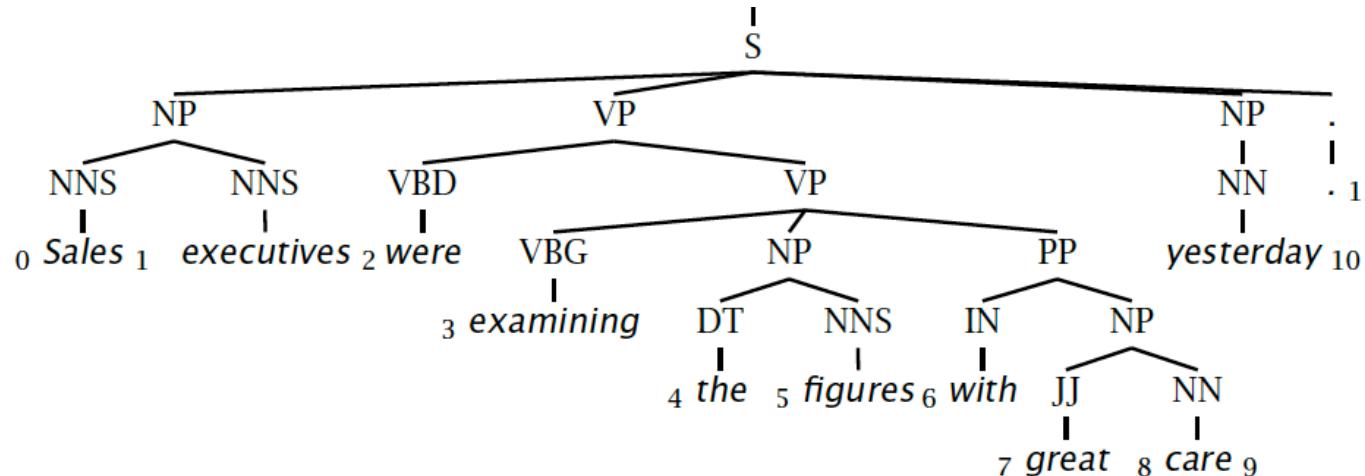


Constituency Parser Evaluation

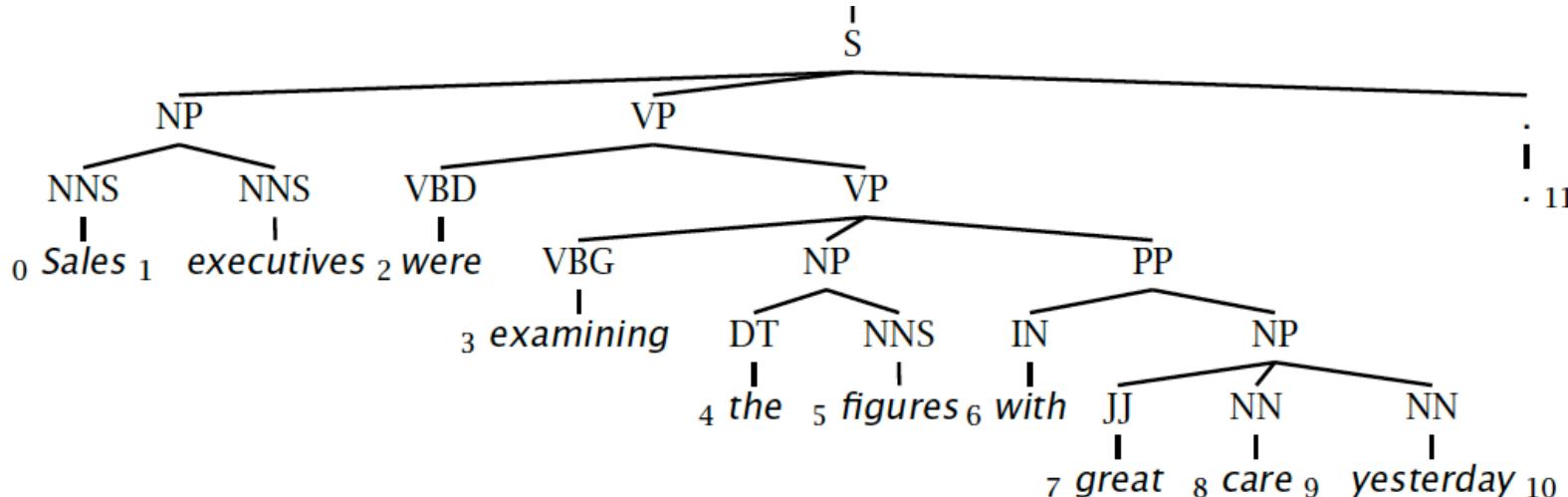


Evaluating constituency parsing

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-(9:10)



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





Evaluating constituency parsing

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-(9:10)

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 $3/8 = 37.5\%$

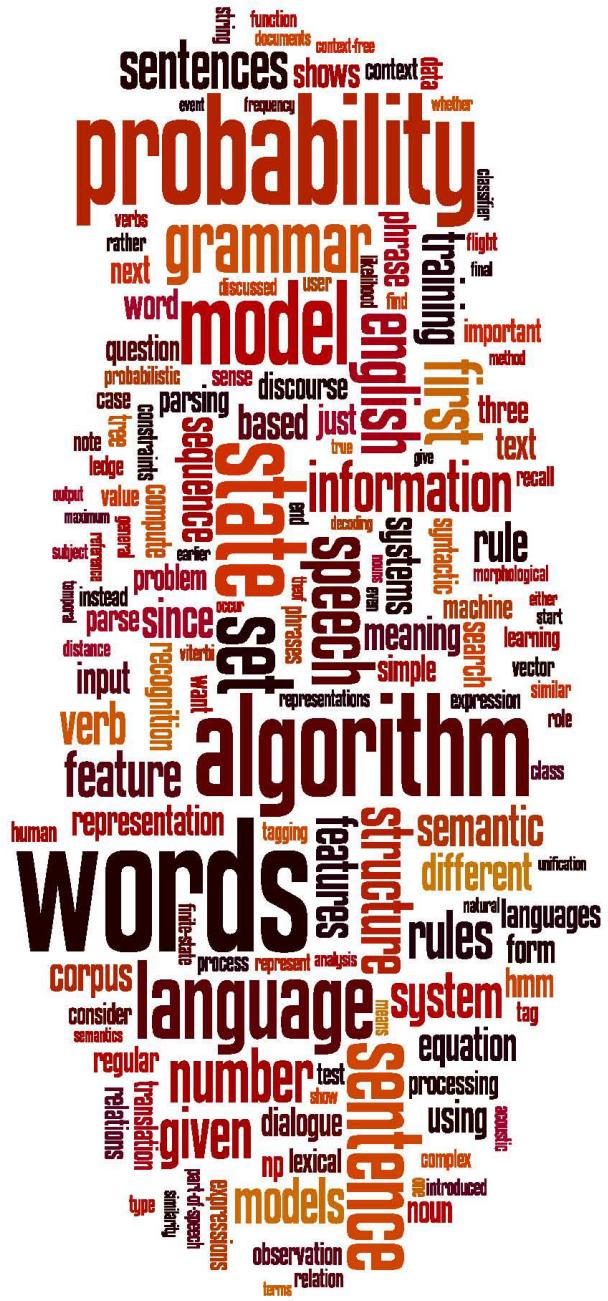
LP/LR F1 40.0%

Tagging Accuracy $11/11 = 100.0\%$

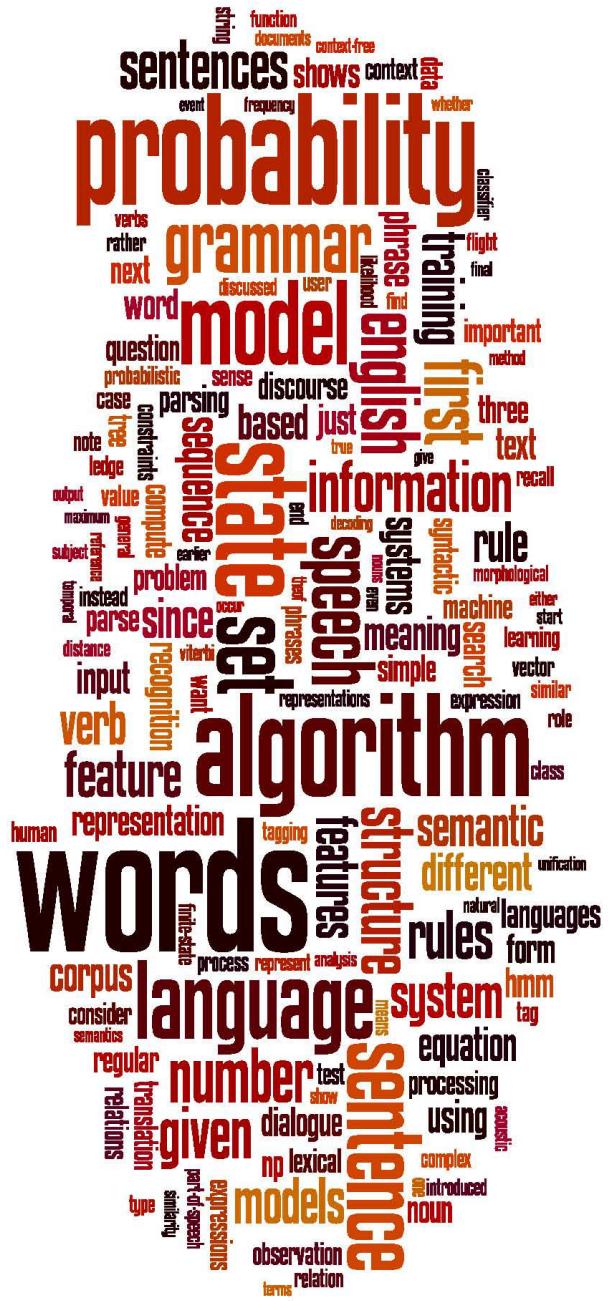


How good are PCFGs?

- Penn WSJ parsing accuracy: about 73% LP/LR F1
- Robust
 - Usually admit everything, but with low probability
- Partial solution for grammar ambiguity
 - A PCFG gives some idea of the plausibility of a parse
 - But not so good because the independence assumptions are too strong
- Give a probabilistic language model
 - But in the simple case it performs worse than a trigram model
- The problem seems to be that PCFGs lack the lexicalization of a trigram model



Constituency Parser Evaluation



Lexicalization of PCFGs

Introduction

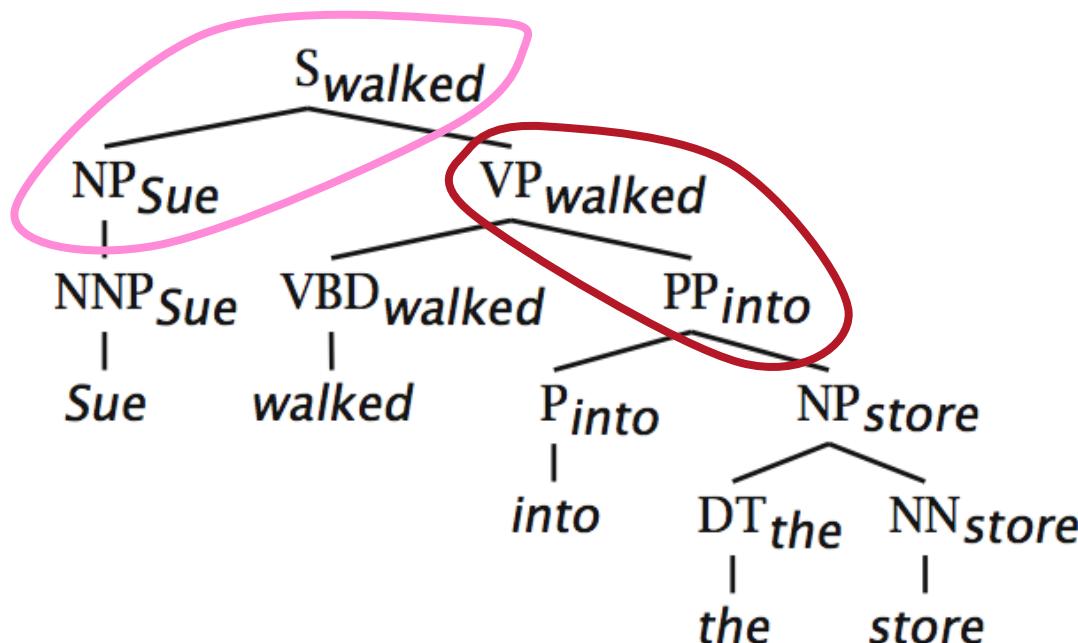
Christopher
Manning



(Head) Lexicalization of PCFGs

[Magerman 1995, 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

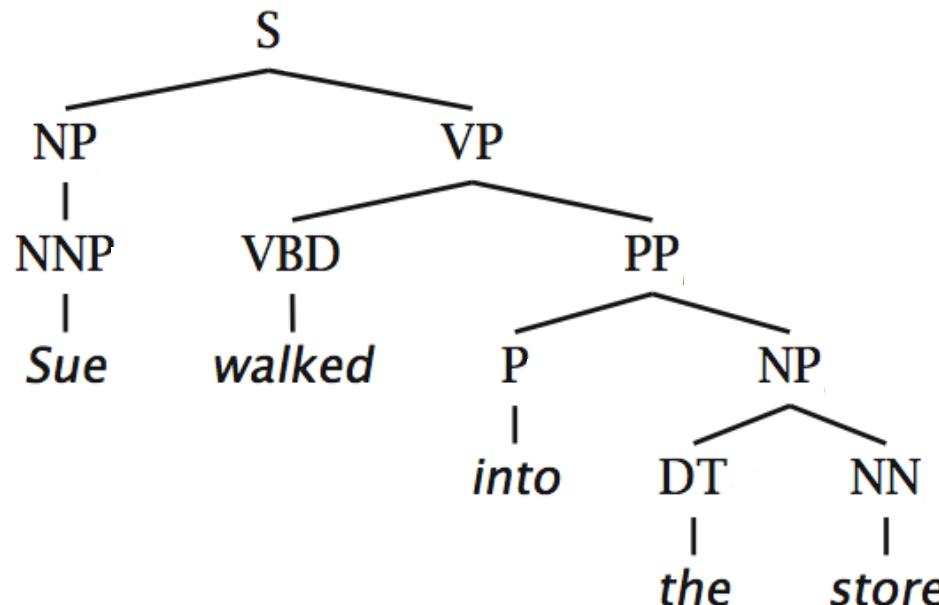




(Head) Lexicalization of PCFGs

[Magerman 1995, 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

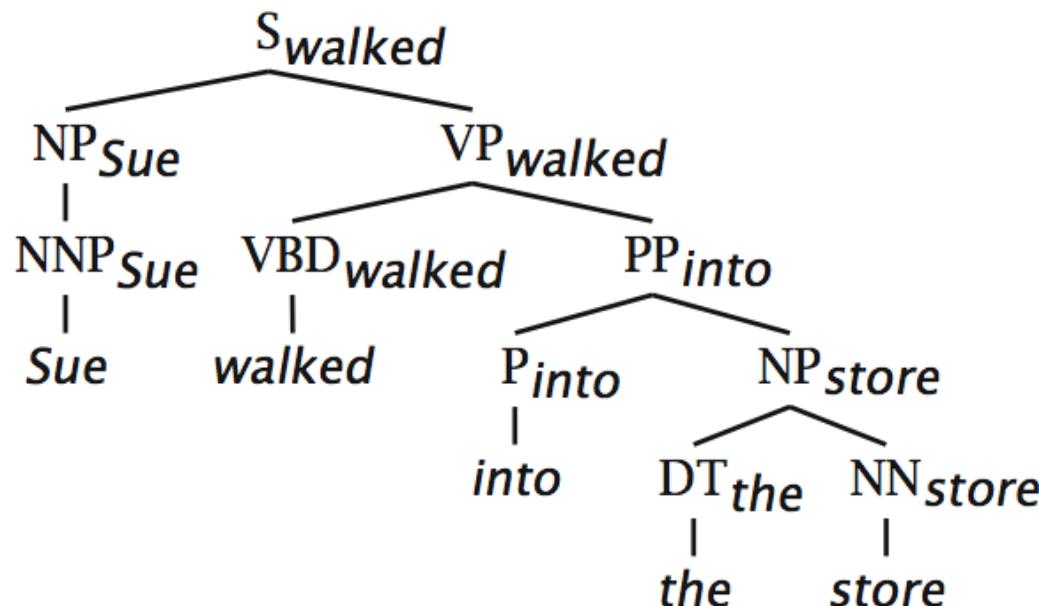




(Head) Lexicalization of PCFGs

[Magerman 1995, 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

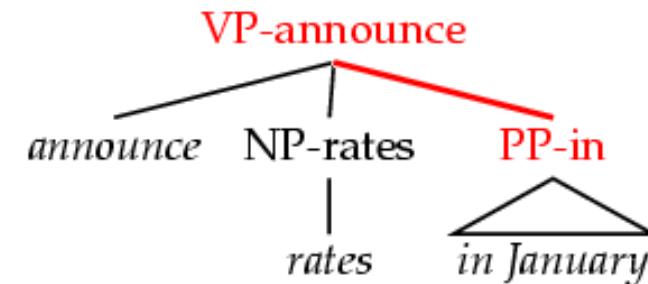
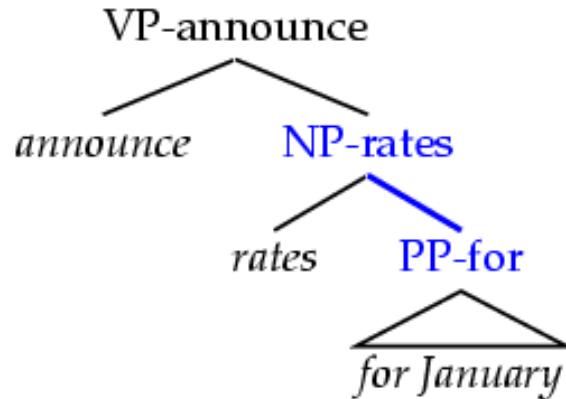




(Head) Lexicalization of PCFGs

[Magerman 1995, Collins 1997; Charniak 1997]

- Word-to-word affinities are useful for certain ambiguities
 - PP attachment is now (partly) captured in a local PCFG rule.
 - Think about: What useful information isn't captured?

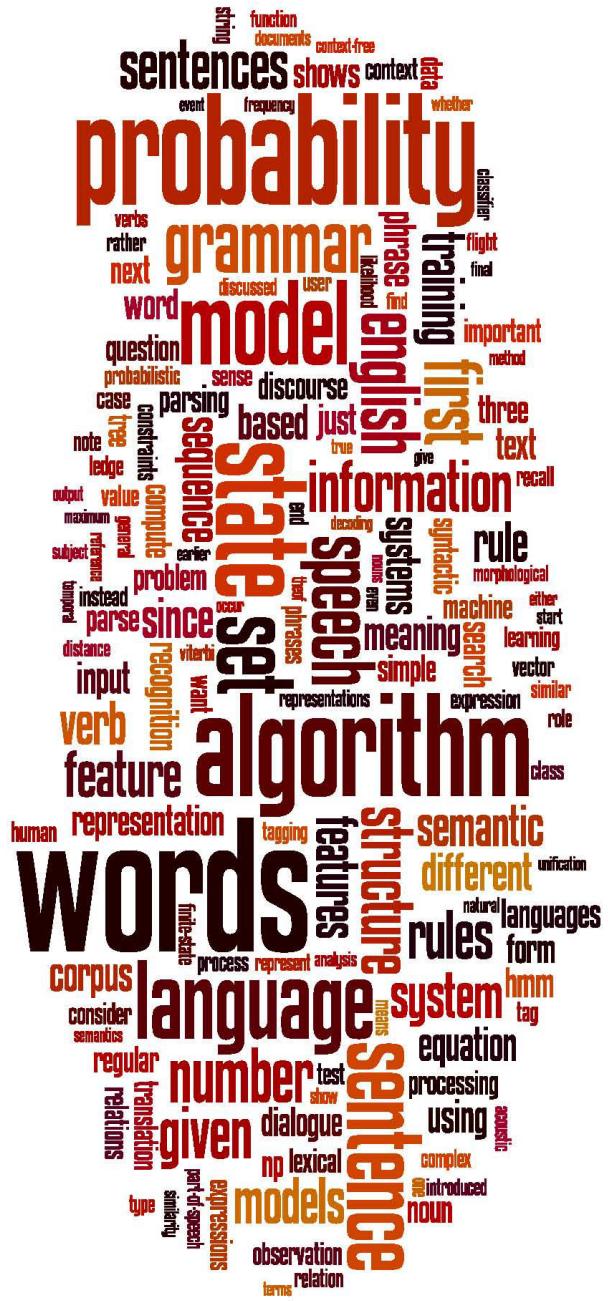


- Also useful for: coordination scope, verb complement patterns



Lexicalized parsing was seen as *the* parsing breakthrough of the late 1990s

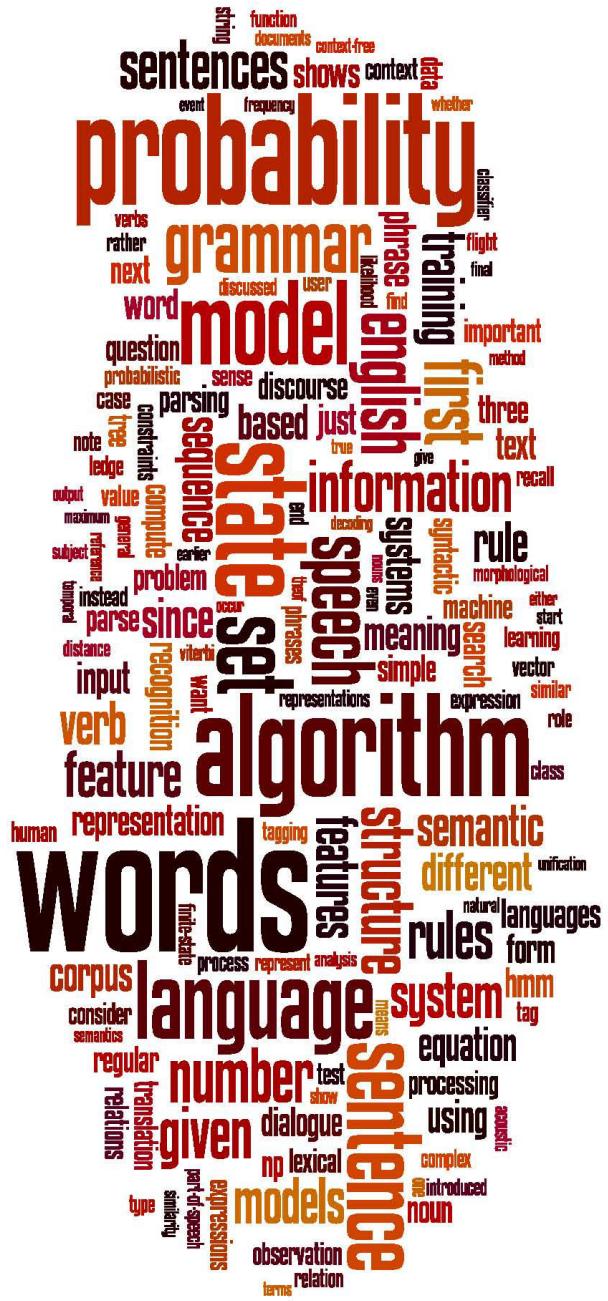
- Eugene Charniak, 2000 JHU workshop: “To do better, it is necessary to condition probabilities on the actual words of the sentence. This makes the probabilities much tighter:
 - $p(\text{VP} \rightarrow \text{V NP NP}) = 0.00151$
 - $p(\text{VP} \rightarrow \text{V NP NP} \mid \text{said}) = 0.00001$
 - $p(\text{VP} \rightarrow \text{V NP NP} \mid \text{gave}) = 0.01980$ ”
- Michael Collins, 2003 COLT tutorial: “Lexicalized Probabilistic Context-Free Grammars ... perform vastly better than PCFGs (88% vs. 73% accuracy)”



Lexicalization of PCFGs

Introduction

Christopher
Manning



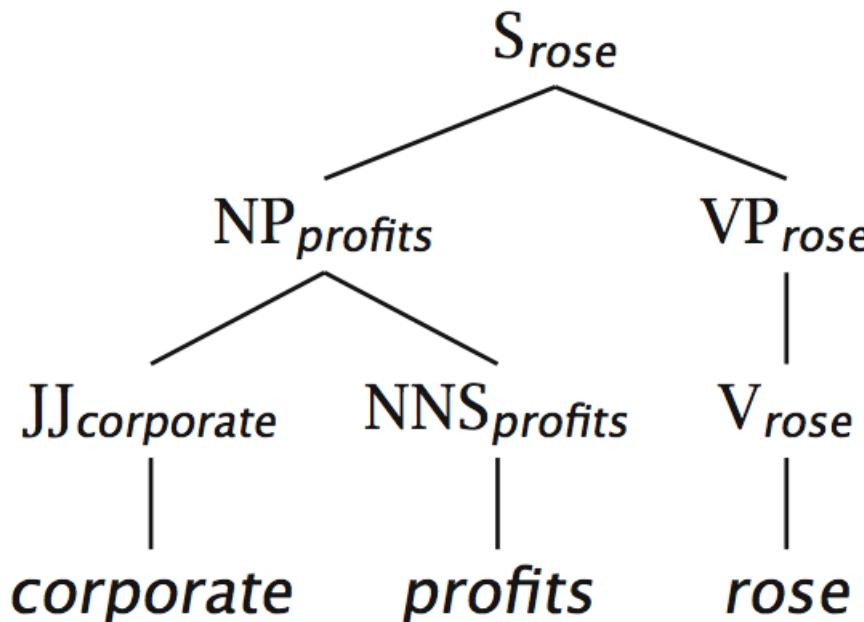
Lexicalization of PCFGs

The model of
Charniak (1997)



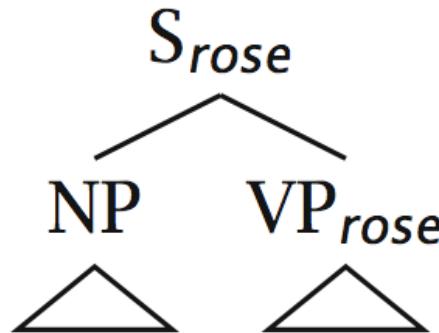
Charniak (1997)

- A very straightforward model of a lexicalized PCFG
- Probabilistic conditioning is “top-down” like a regular PCFG
 - But actual parsing is bottom-up, somewhat like the CKY algorithm we saw

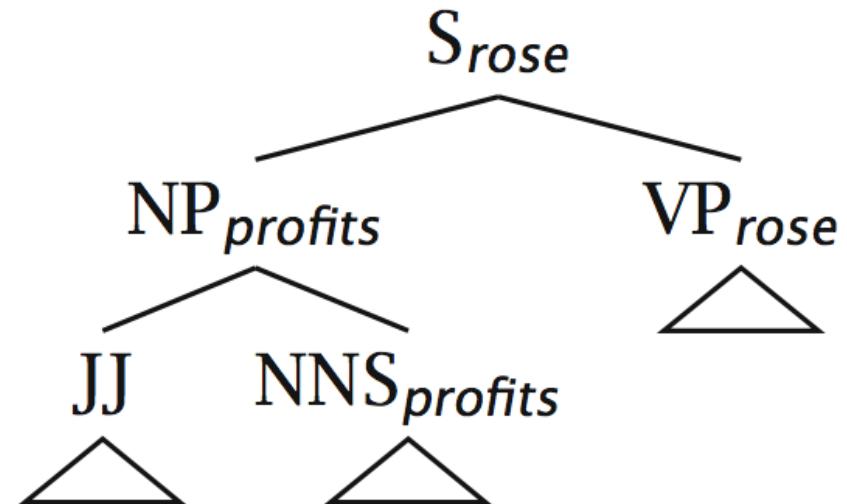
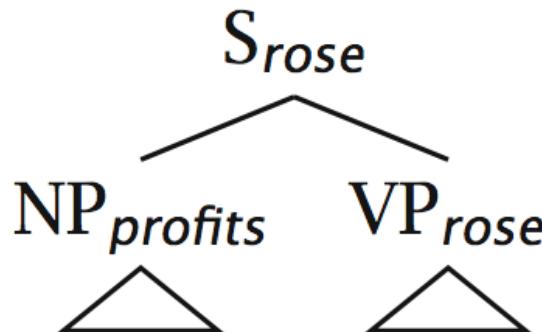




Charniak (1997) example



- a. $h = \text{profits}; c = \text{NP}$
- b. $ph = \text{rose}; pc = \text{S}$
- c. $P(h|ph, c, pc)$
- d. $P(r|h, c, pc)$





Lexicalization models argument selection by sharpening rule expansion probabilities

- The probability of different verbal complement frames (i.e., “subcategorizations”) depends on the verb:

<i>Local Tree</i>	<i>come</i>	<i>take</i>	<i>think</i>	<i>want</i>
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%



“monolexical” probabilities



Lexicalization sharpens probabilities: Predicting heads

“Bilexical probabilities”

- $P(\text{prices} \mid \text{n-plural}) = .013$
- $P(\text{prices} \mid \text{n-plural}, \text{NP}) = .013$
- $P(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}) = .025$
- $P(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}, \text{v-past}) = .052$
- $P(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}, \text{v-past}, \text{fell}) = .146$



Charniak (1997) linear interpolation/ shrinkage

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

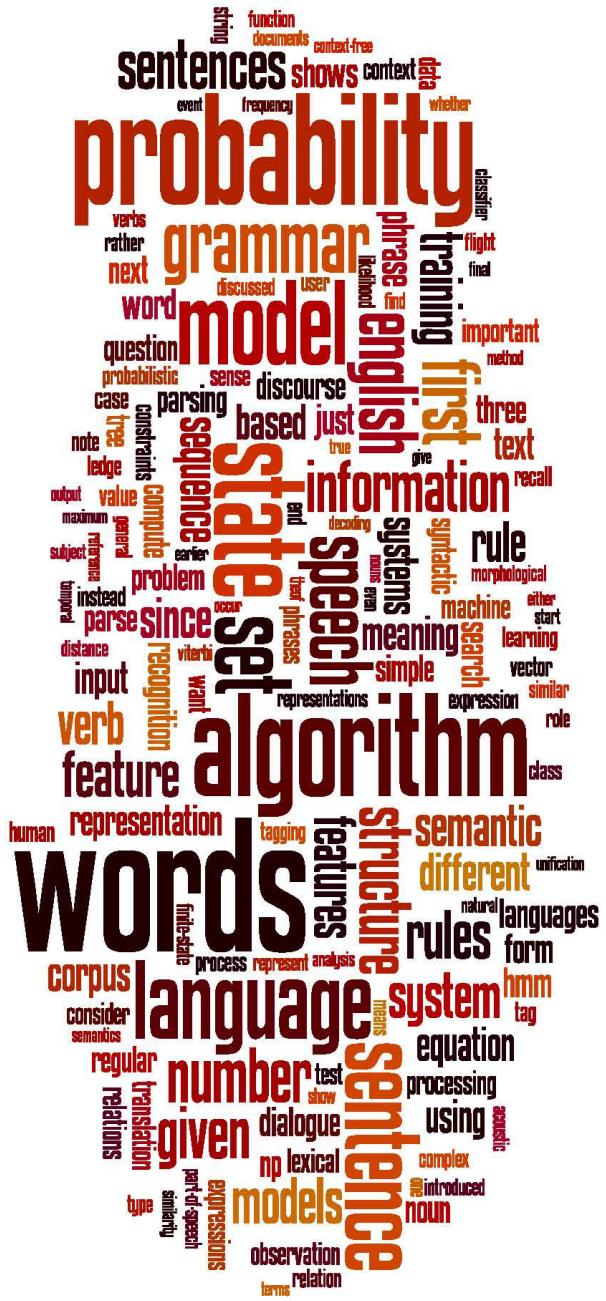
- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction



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.



Lexicalization of PCFGs

The model of
Charniak (1997)



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



Quiz question!

- Classify each of the italic red phrases as a:

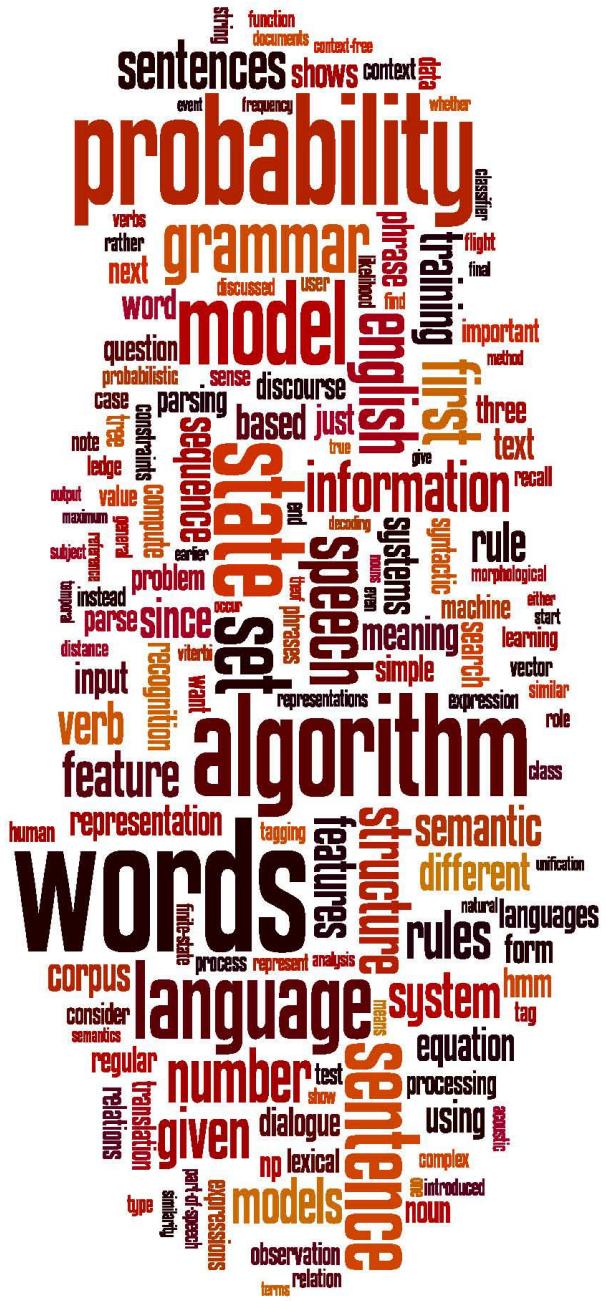
WHNP WHADJP WHADVP WHPP

1. That explains *why* she is succeeding.
2. *Which student* scored highest on the assignment?
3. Nobody knows *how deep* the recession will be.
4. *During which class* did the slide projection not work?
5. *Whose iPhone* was stolen?



Sparseness & the Penn Treebank (2)

- Many parse preferences depend on bilexical statistics: likelihoods of relationships between pairs of words (compound nouns, PP attachments, ...)
- 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
- There has been only modest success in augmenting the Penn Treebank with extra unannotated materials or using semantic classes – given a reasonable amount of annotated training data.
 - Cf. Charniak 1997, Charniak 2000
 - But McClosky et al. 2006 doing self-training and Koo and Collins 2008 semantic classes are rather more successful!

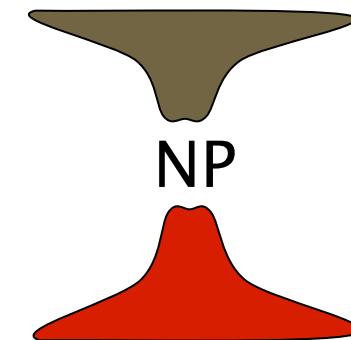
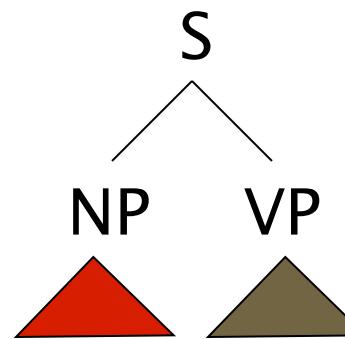


PCFG Independence Assumptions



PCFGs and Independence

- The symbols in a PCFG define independence assumptions:

$$\begin{aligned} S &\rightarrow NP \ VP \\ NP &\rightarrow DT \ NN \end{aligned}$$


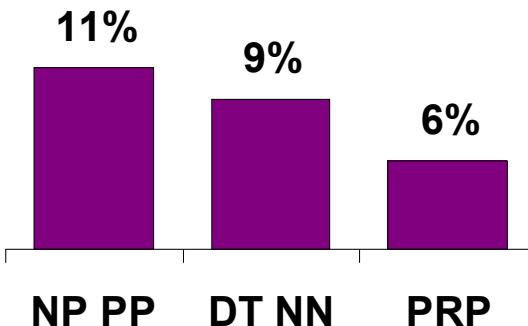
- At any node, **the material inside that node** is independent of the material outside that node, given the label of that node
- Any information that statistically connects behavior **inside** and outside a node must flow through that node's label



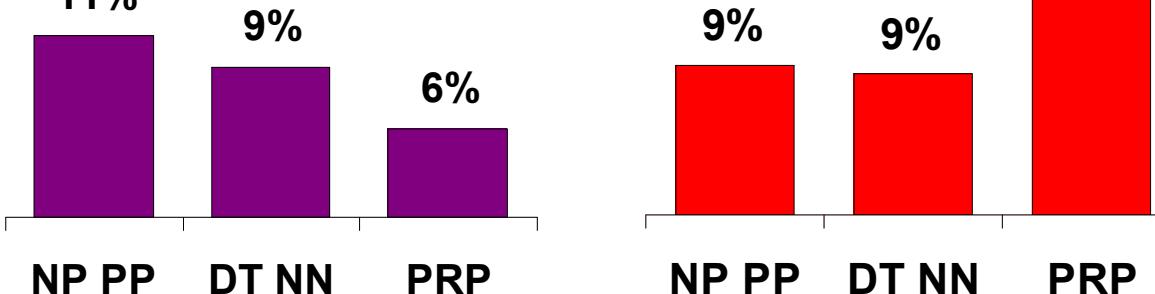
Non-Independence I

- The independence assumptions of a PCFG are often too strong

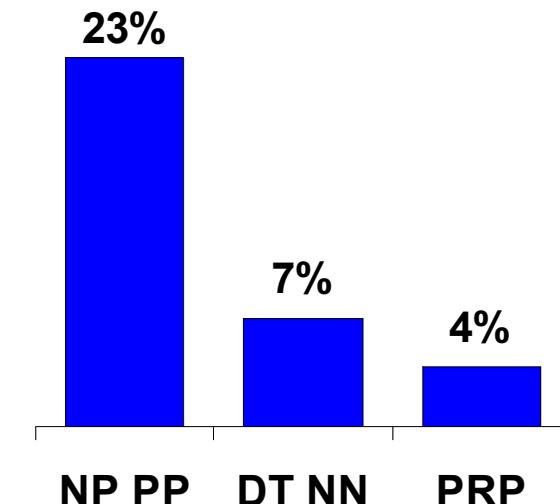
All NPs



NPs under S



NPs under VP

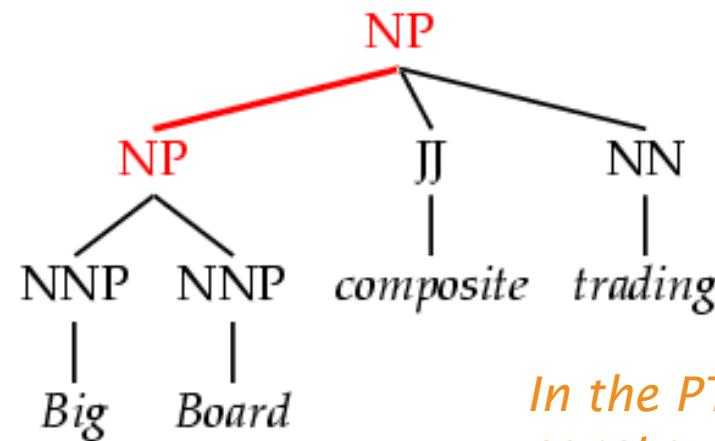
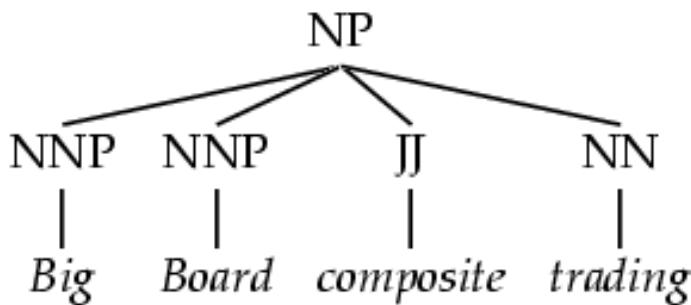
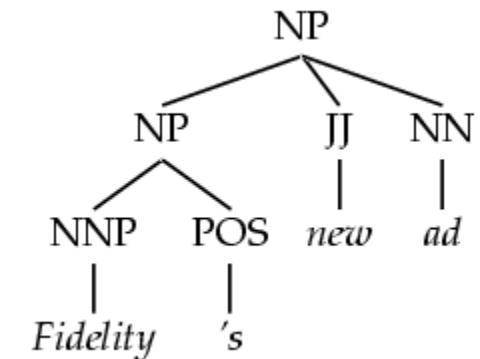


- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects)



Non-Independence II

- Symptoms of overly strong assumptions:
 - Rewrites get used where they don't belong



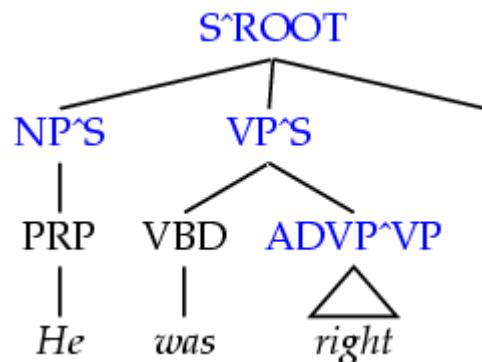
In the PTB, this construction is for possessives



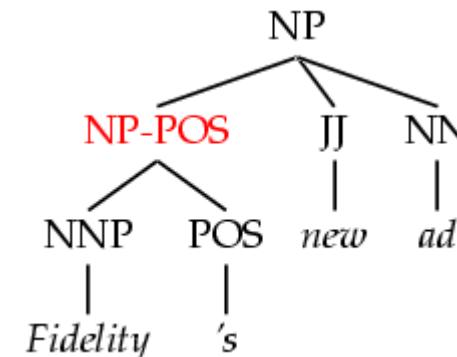
Refining the Grammar Symbols

- We can relax independence assumptions by encoding dependencies into the PCFG symbols, by **state splitting**:

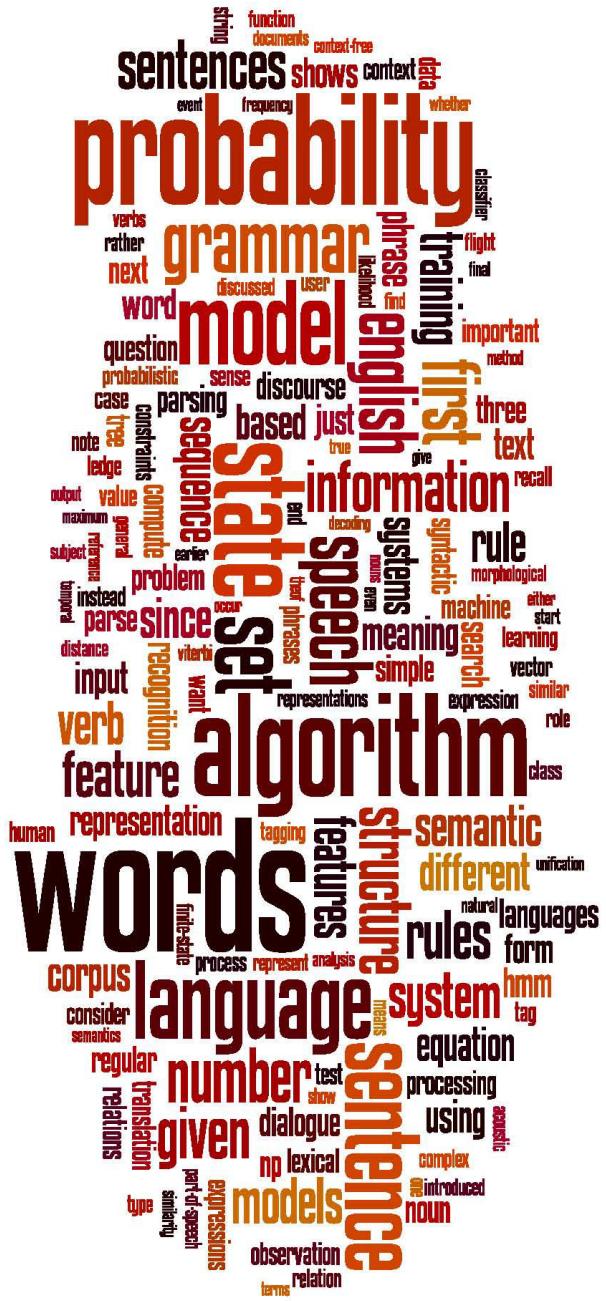
Parent annotation
[Johnson 98]



Marking
possessive NPs



- Too much state-splitting → sparseness (no smoothing used!)
- What are the most useful features to encode?

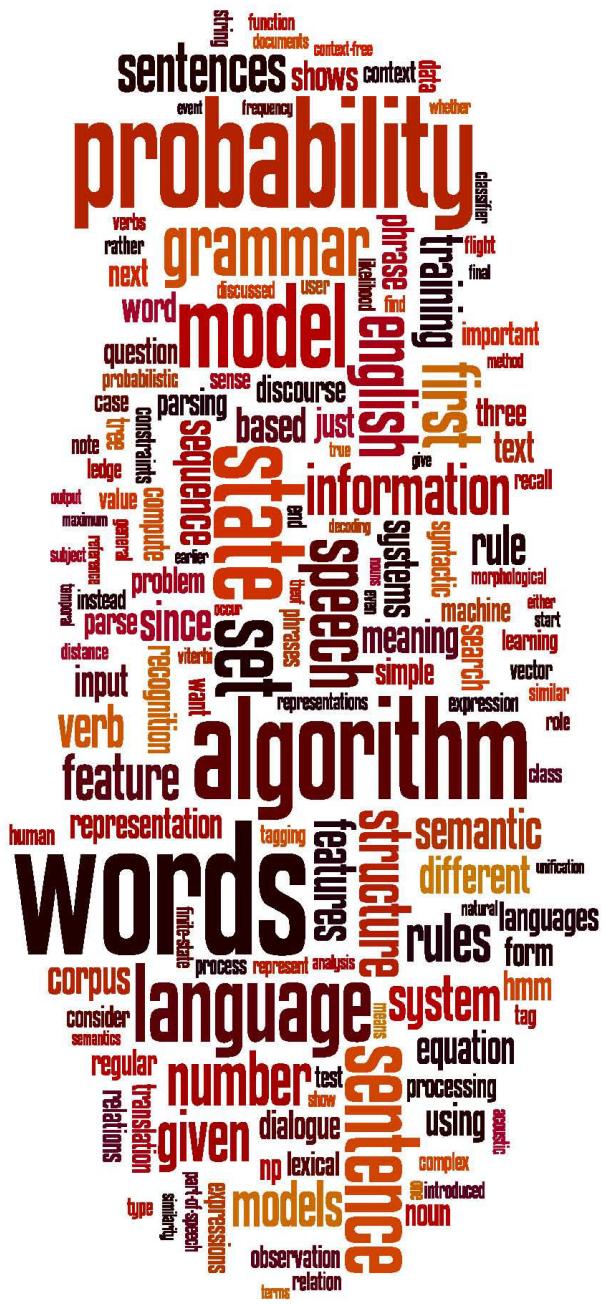


PCFG Independence Assumptions



Annotations

- Annotations split the grammar categories into sub-categories.
- Conditioning on history vs. annotating
 - $P(\text{NP}^{\wedge} S \rightarrow \text{PRP})$ is a lot like $P(\text{NP} \rightarrow \text{PRP} \mid S)$
 - $P(\text{NP-POS} \rightarrow \text{NNP POS})$ isn't history conditioning.
- Feature grammars vs. annotation
 - Can think of a symbol like $\text{NP}^{\wedge} \text{NP-POS}$ as
 $\text{NP} [\text{parent:NP}, \text{+POS}]$
- After parsing with an annotated grammar, the annotations are then stripped for evaluation.



The Return of Unlexicalized PCFGs



Accurate Unlexicalized Parsing

[Klein and Manning 1993]

- What do we mean by an “unlexicalized” PCFG?
 - Grammar rules are not systematically specified down to the level of lexical items
 - NP-stocks is not allowed
 - NP^AS-CC is fine
 - Closed vs. open class words
 - Long tradition in linguistics of using function words as features or markers for selection (VB-have, SBAR-if/whether)
 - Different to the blexical idea of semantic heads
 - Open-class selection is really a proxy for semantics
- Thesis
 - Most of what you need for accurate parsing, and much of what lexicalized PCFGs actually capture *isn't* lexical selection between content words but just basic grammatical features, like verb form, finiteness, presence of a verbal auxiliary, etc.



Experimental Approach

- Corpus: Penn Treebank, WSJ; iterate on small dev set



Training: sections 02-21

Development: section 22 (first 20 files) ←

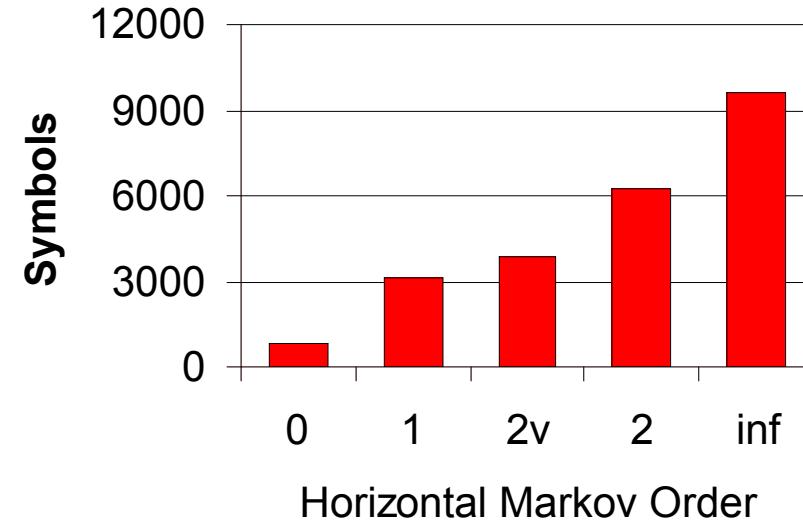
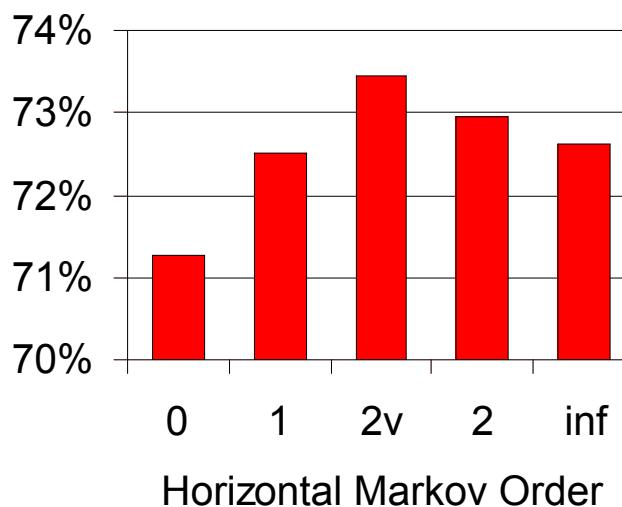
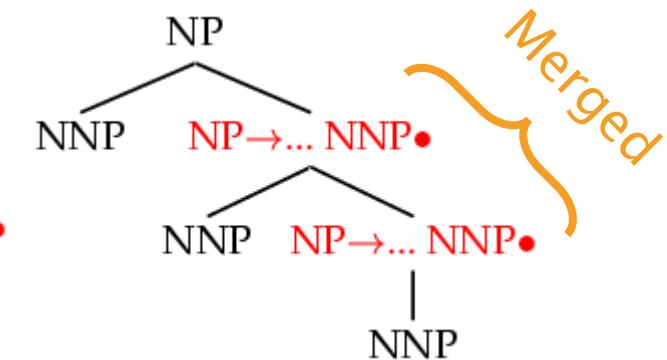
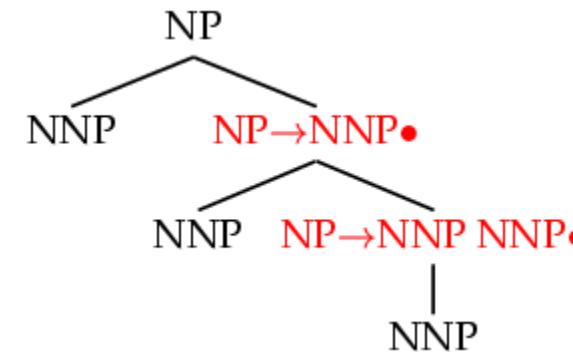
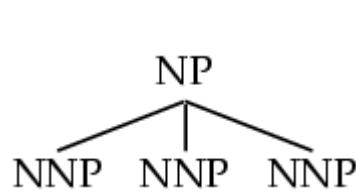
Test: section 23

- Size – number of symbols in grammar.
 - Passive / complete symbols: NP, NP^S
 - Active / incomplete symbols: @NP_NP_CC [from binarization]
- We state-split as sparingly as possible
 - Highest accuracy with fewest symbols
 - Error-driven, manual hill-climb, one annotation at a time



Horizontal Markovization

- Horizontal Markovization: Merges States

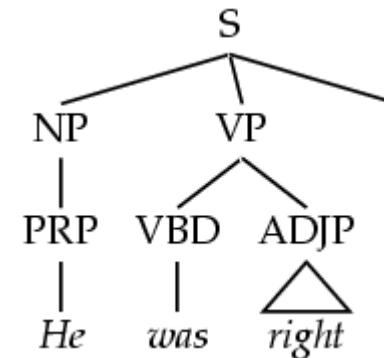




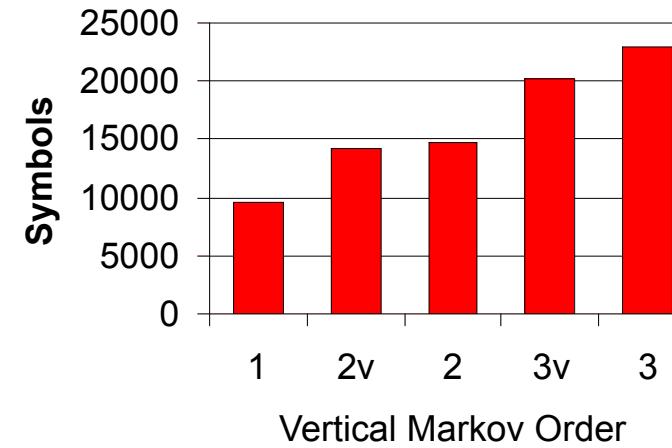
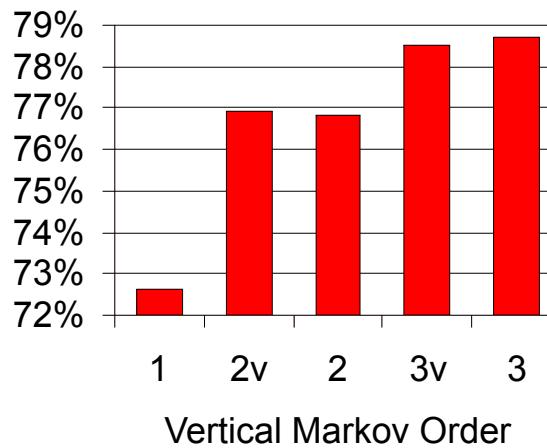
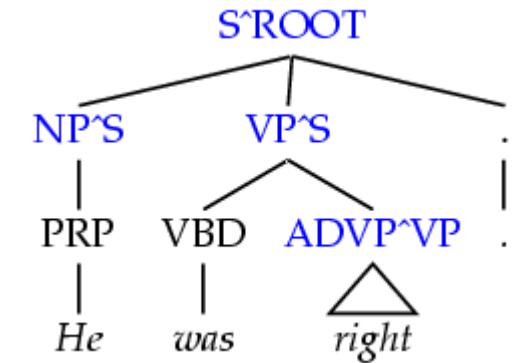
Vertical Markovization

- Vertical Markov order: rewrites depend on past k ancestor nodes.
(i.e., parent annotation)

Order 1



Order 2

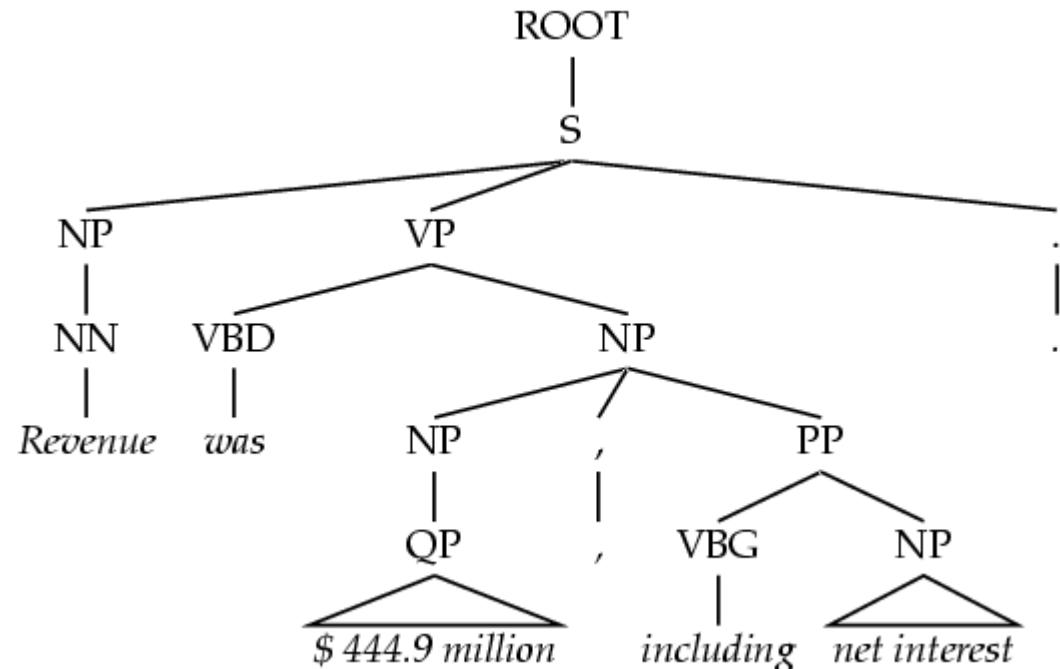


Model	F1	Size
v=h=2v	77.8	7.5K



Unary Splits

- Problem: unary rewrites are used to transmute categories so a high-probability rule can be used.
- Solution: Mark unary rewrite sites with -U

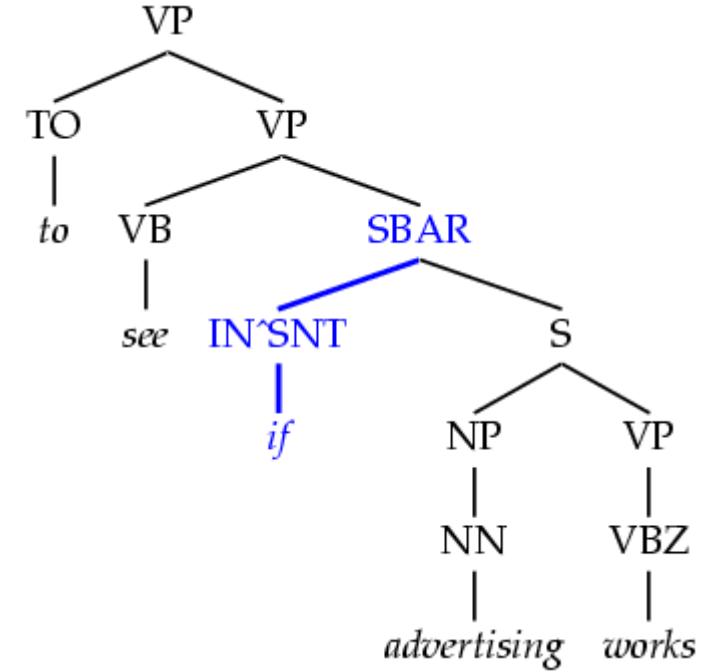


Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K



Tag Splits

- Problem: Treebank tags are too coarse.
- Example: SBAR sentential complementizers (*that*, *whether*, *if*), subordinating conjunctions (*while*, *after*), and true prepositions (*in*, *of*, *to*) are all tagged IN.
- Partial Solution:
 - Subdivide the IN tag.

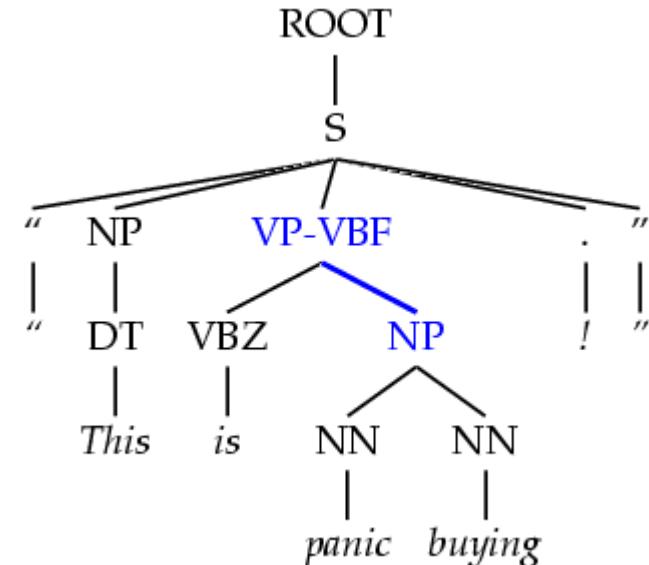


Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K



Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield.
- Examples:
 - Possessive NPs
 - Finite vs. infinite VPs
 - Lexical heads!
- Solution: annotate future elements into nodes.

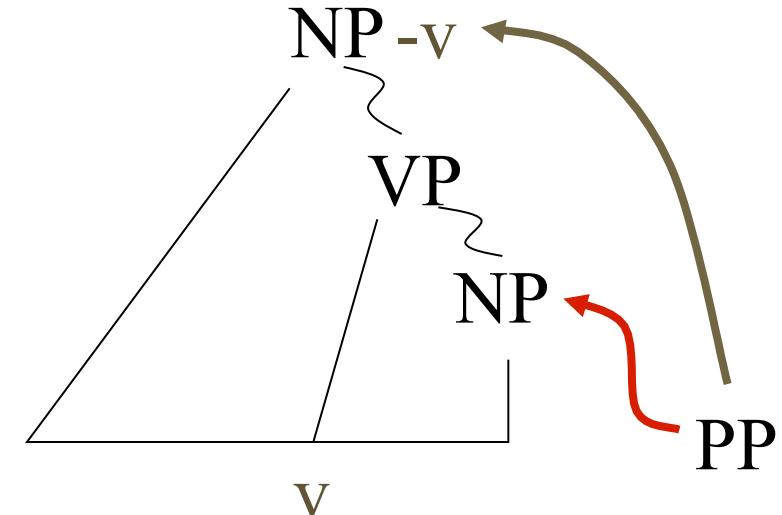


Annotation	F1	Size
tag splits	82.3	9.7K
POSS-NP	83.1	9.8K
SPLIT-VP	85.7	10.5K



Distance / Recursion Splits

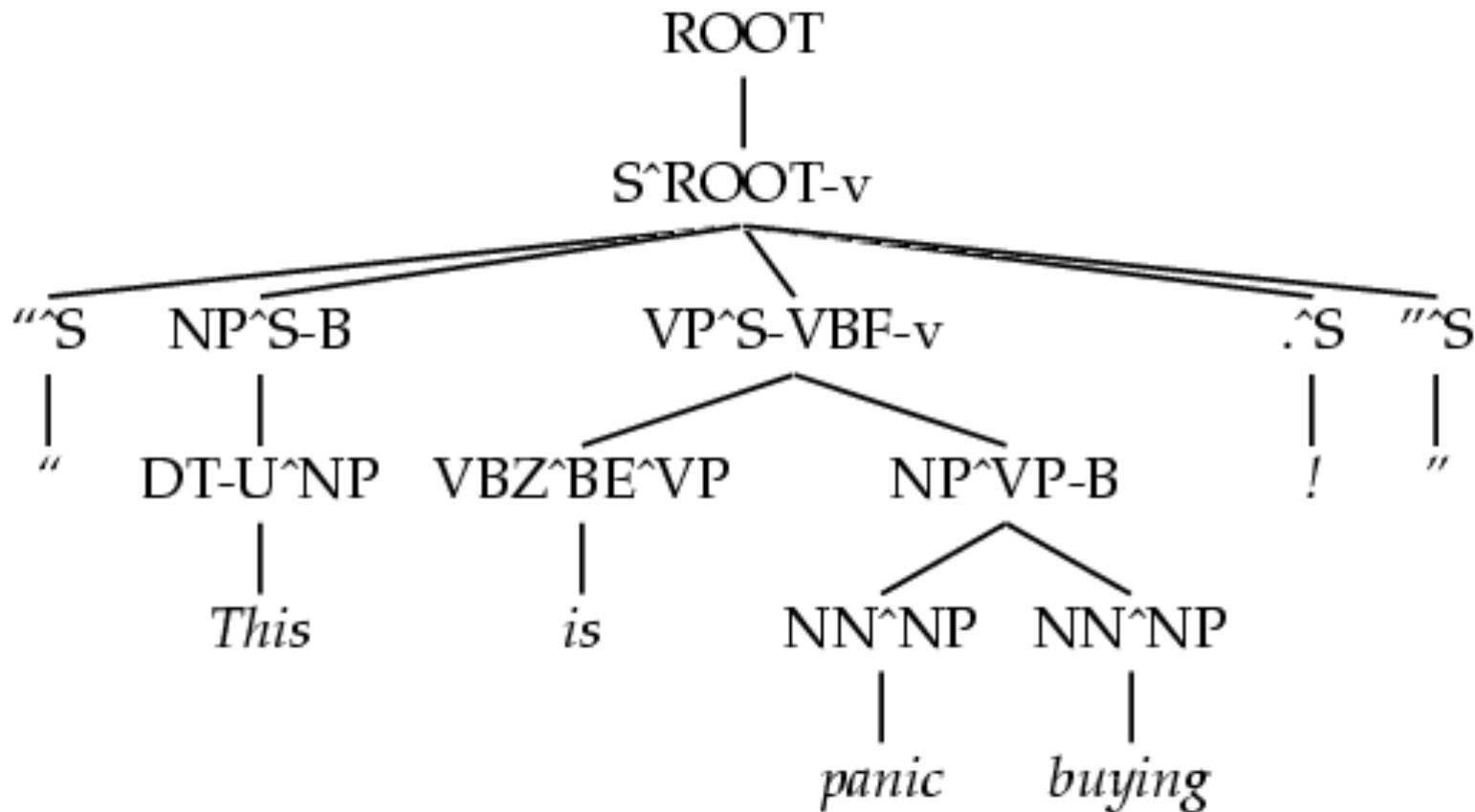
- Problem: vanilla PCFGs cannot distinguish attachment heights.
- Solution: mark a property of higher or lower sites:
 - Contains a verb.
 - Is (non)-recursive.
 - Base NPs [cf. Collins 99]
 - Right-recursiive NPs



Annotation	F1	Size
Previous	85.7	10.5K
BASE-NP	86.0	11.7K
DOMINATES-V	86.9	14.1K
RIGHT-REC-NP	87.0	15.2K



A Fully Annotated Tree

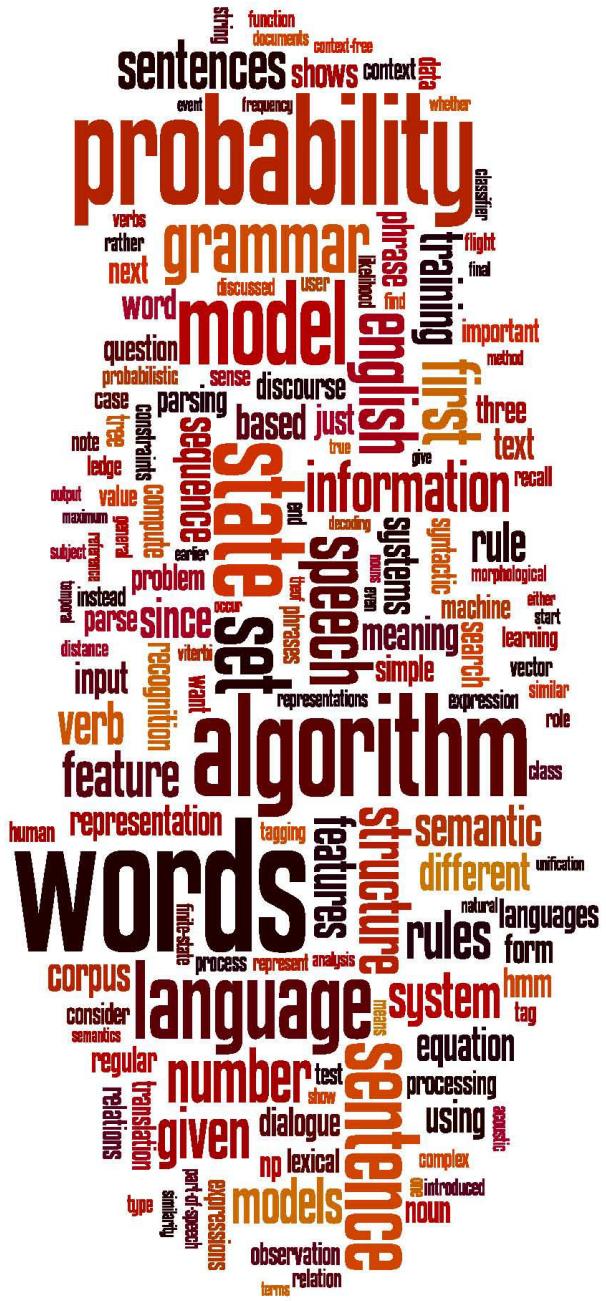




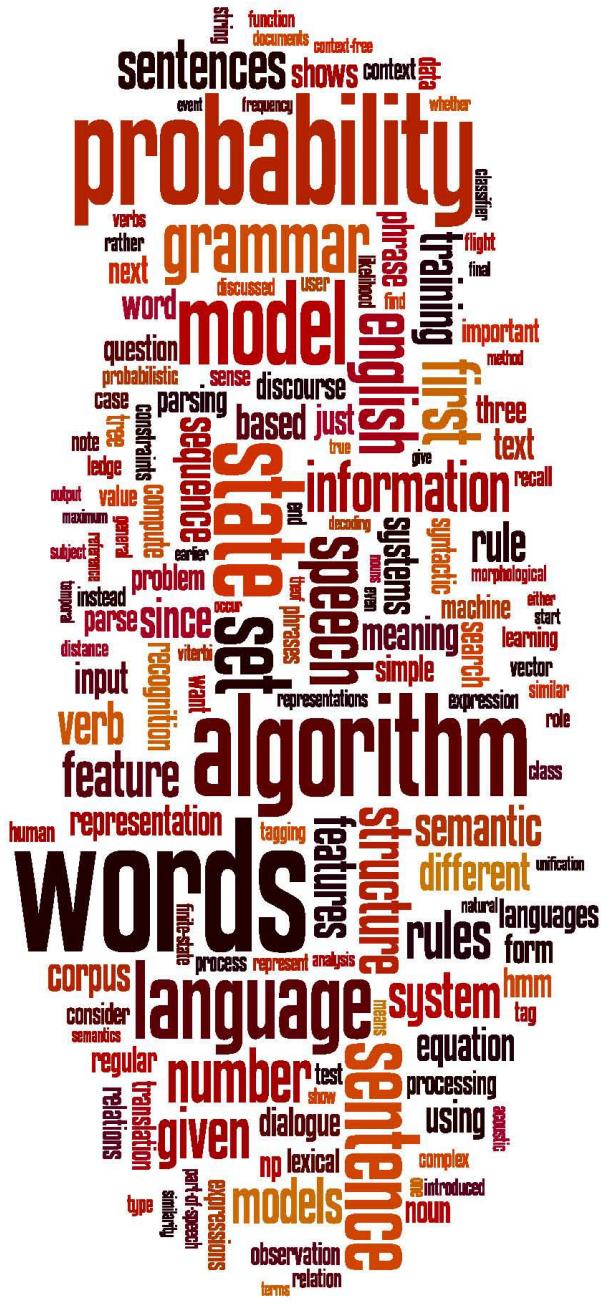
Final Test Set Results

Parser	LP	LR	F1
Magerman 95	84.9	84.6	84.7
Collins 96	86.3	85.8	86.0
Klein & Manning 03	86.9	85.7	86.3
Charniak 97	87.4	87.5	87.4
Collins 99	88.7	88.6	88.6

- Beats “first generation” lexicalized parsers



The Return of Unlexicalized PCFGs



Latent Variable PCFGs

Extending the idea
to induced
syntactico-semantic
classes

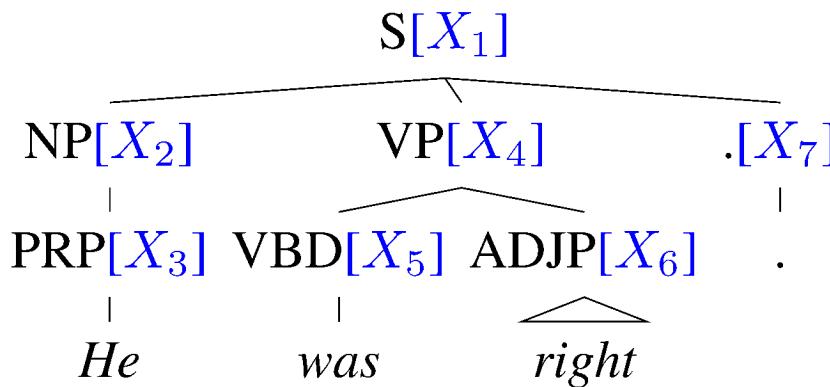


Learning Latent Annotations

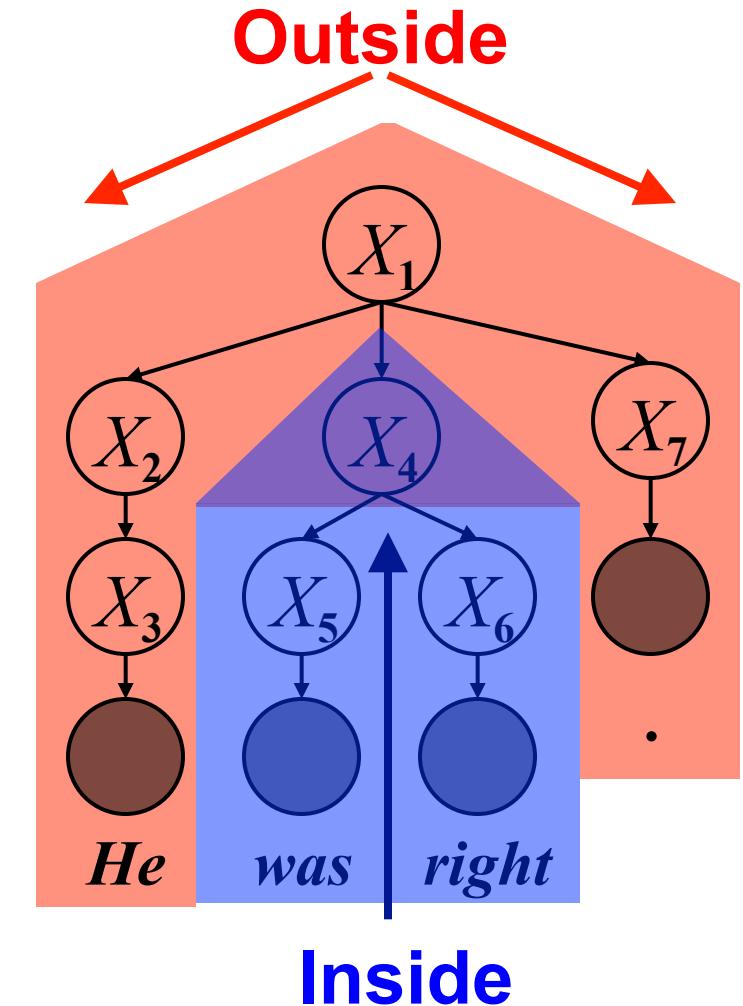
[Petrov and Klein 2006, 2007]

Can you automatically find good symbols?

- Brackets are known
- Base categories are known
- Induce subcategories
- Clever split/merge category refinement



EM algorithm, like Forward-Backward for HMMs, but constrained by tree





POS tag splits' commonest words: effectively a semantic class-based model

- Proper Nouns (NNP):

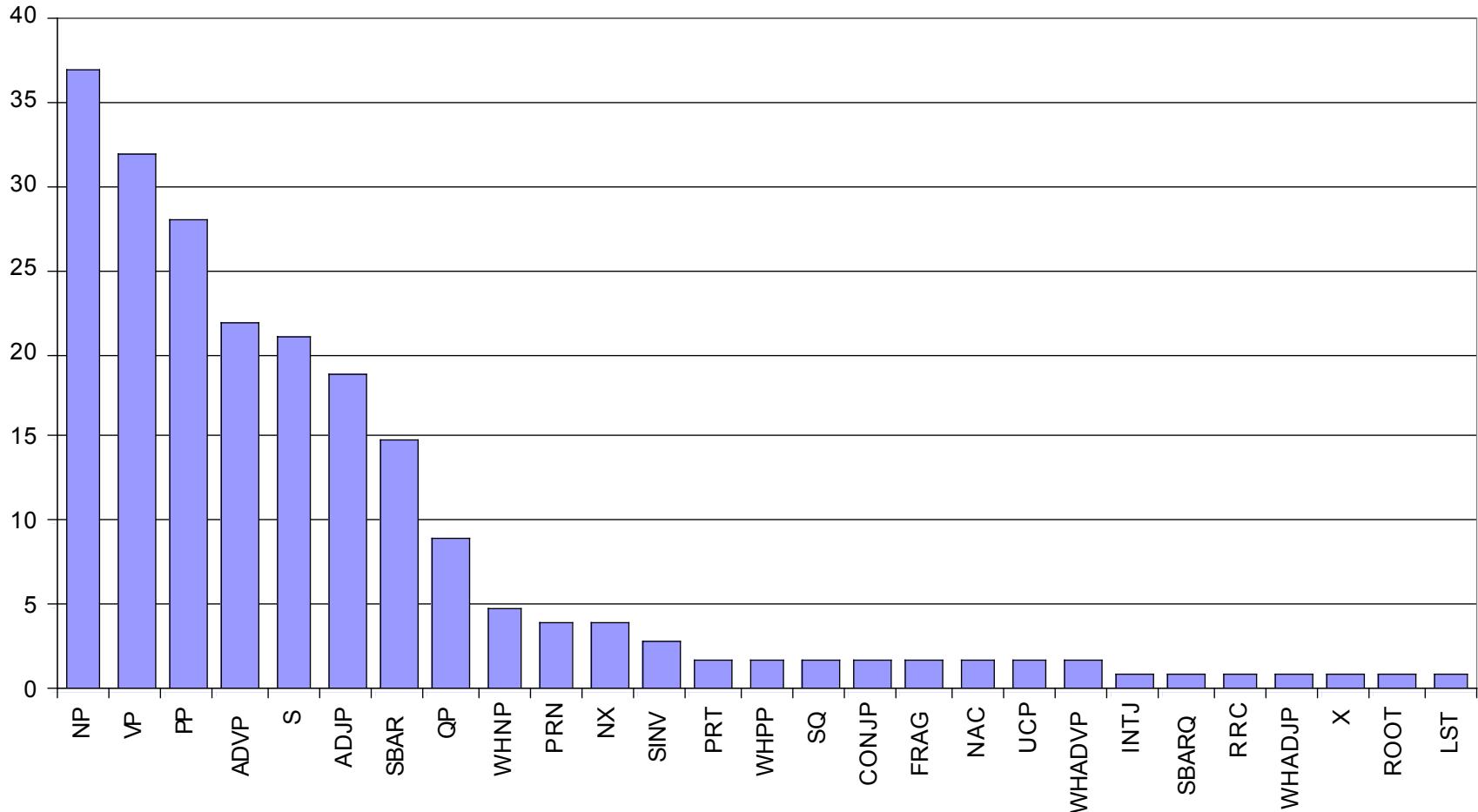
NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Personal pronouns (PRP):

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him



Number of phrasal subcategories

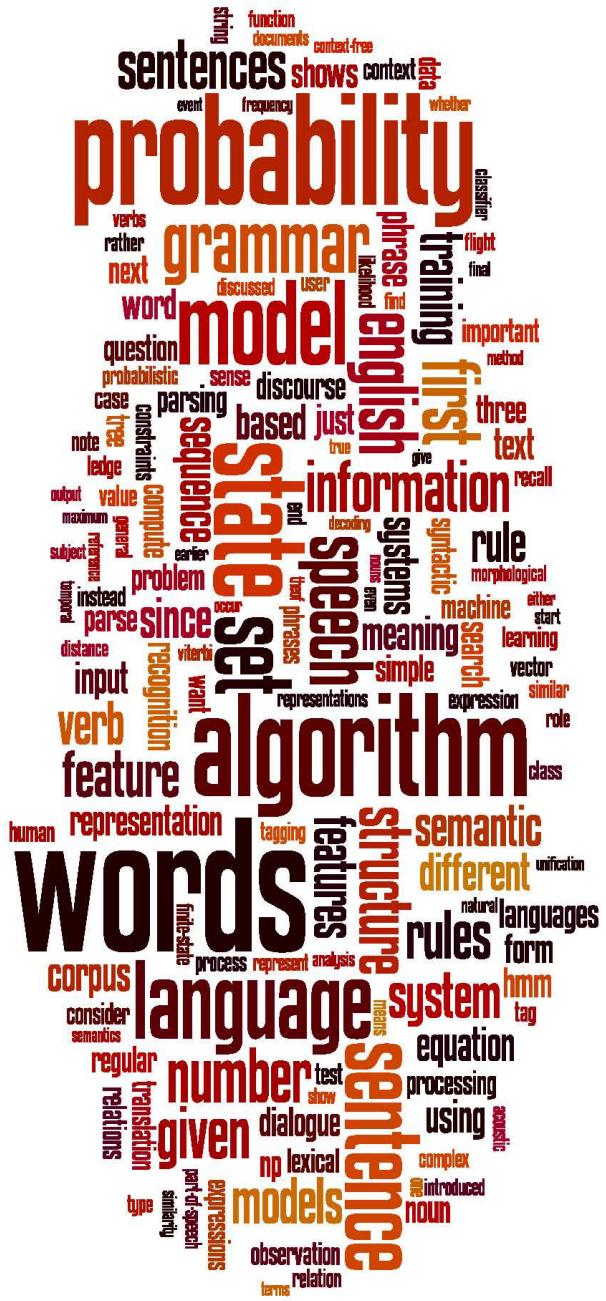




The Latest Parsing Results...

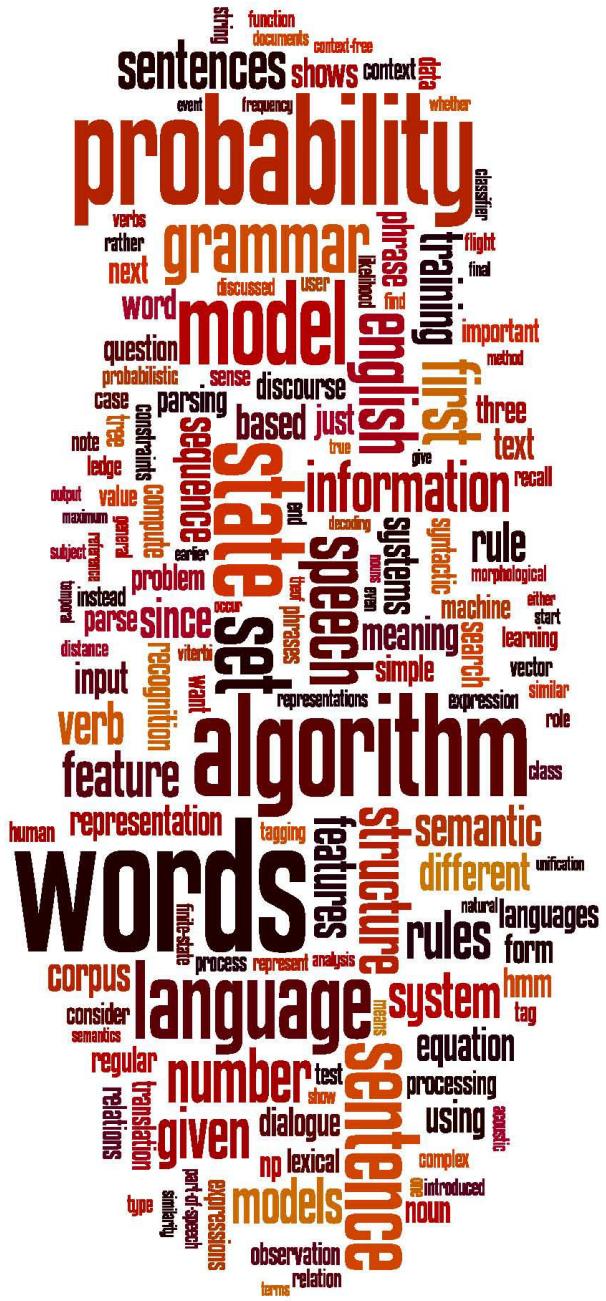
(English PTB3 WSJ train 2-21, test 23)

<i>Parser</i>	<i>F1</i> ≤ 40 words	<i>F1</i> all words
Klein & Manning unlexicalized 2003	86.3	85.7
Matsuzaki et al. simple EM latent states 2005	86.7	86.1
Charniak generative, lexicalized ("maxent inspired") 2000	90.1	89.5
Petrov and Klein NAACL 2007	90.6	90.1
Charniak & Johnson discriminative reranker 2005	92.0	91.4
Fossum & Knight 2009 combining constituent parsers		92.4



Latent Variable PCFGs

Extending the idea to induced syntactico-semantic classes



Dependency Parsing

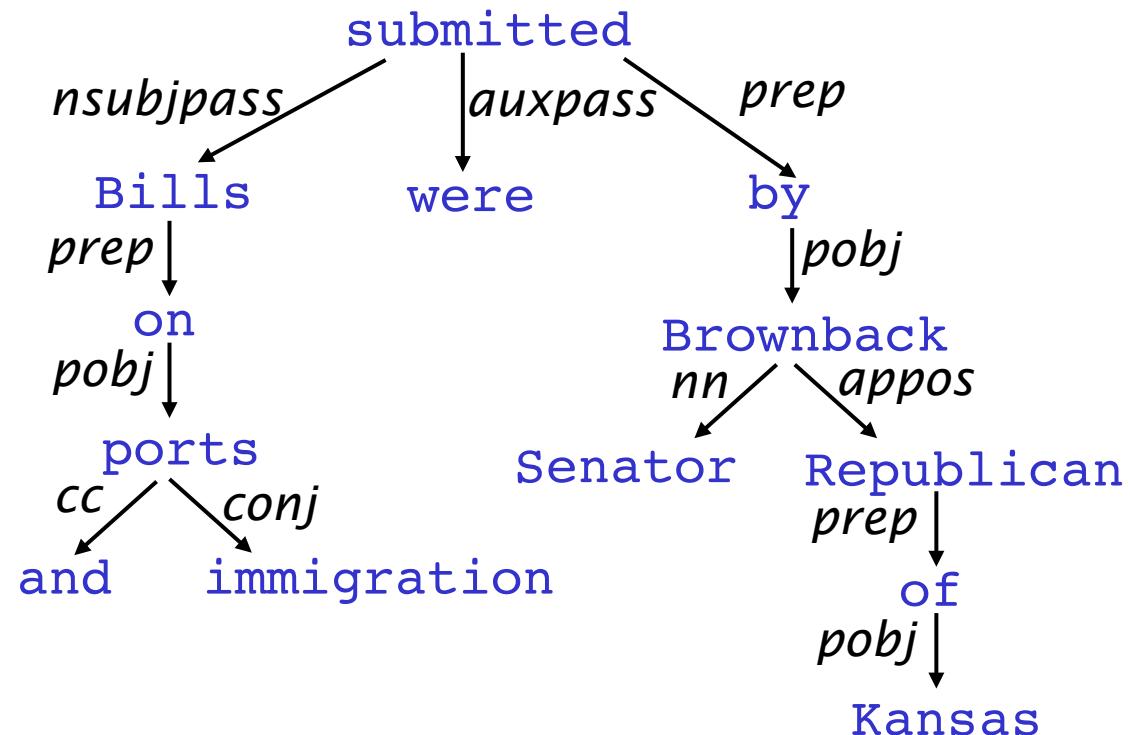
Introduction



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

The arrows are commonly **typed** with the name of grammatical relations (subject, prepositional object, apposition, etc.)



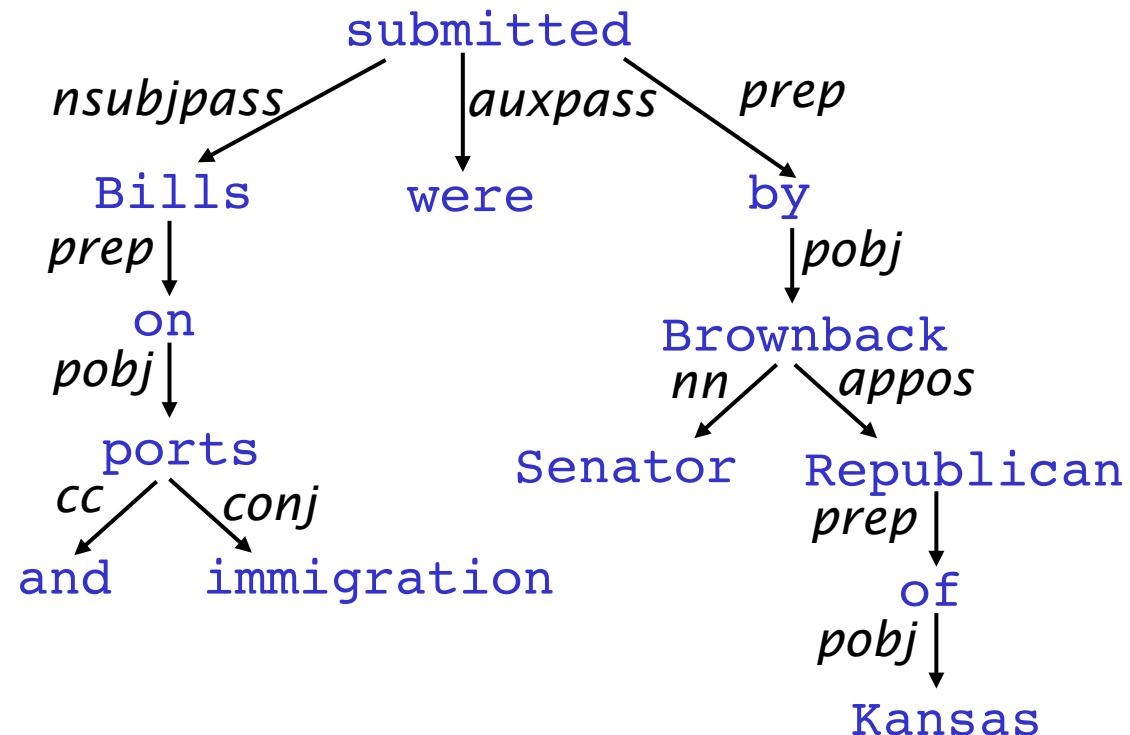


Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

The arrow connects a **head** (governor, superior, regent) with a **dependent** (modifier, inferior, subordinate)

Usually, dependencies form a tree (connected, acyclic, single-head)

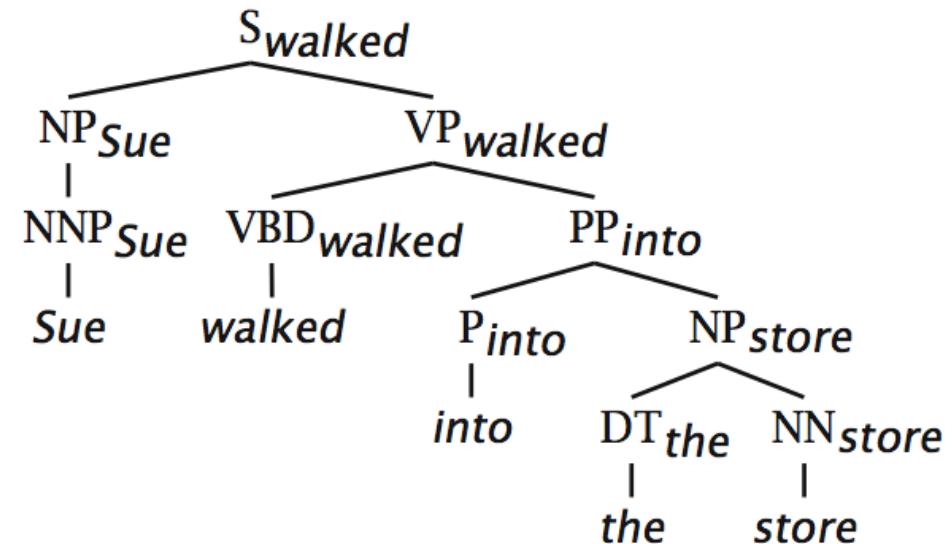




Relation between phrase structure and dependency structure

- A dependency grammar has a notion of a head. Officially, CFGs don't.
- But modern linguistic theory and all modern statistical parsers (Charniak, Collins, Stanford, ...) do, via hand-written phrasal “head rules”:
 - The head of a Noun Phrase is a noun/number/adj/...
 - The head of a Verb Phrase is a verb/modal/....
- The head rules can be used to extract a dependency parse from a CFG parse

- The closure of dependencies give constituency from a dependency tree
- But the dependents of a word must be at the same level (i.e., “flat”) – there can be no VP!





Methods of Dependency Parsing

1. Dynamic programming (like in the CKY algorithm)

You can do it similarly to lexicalized PCFG parsing: an $O(n^5)$ algorithm

Eisner (1996) gives a clever algorithm that reduces the complexity to $O(n^3)$, by producing parse items with heads at the ends rather than in the middle

2. Graph algorithms

You create a Maximum Spanning Tree for a sentence

McDonald et al.'s (2005) MSTParser scores dependencies independently using a ML classifier (he uses MIRA, for online learning, but it could be MaxEnt)

3. Constraint Satisfaction

Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.

4. “Deterministic parsing”

Greedy choice of attachments guided by machine learning classifiers

MaltParser (Nivre et al. 2008) – discussed in the next segment



Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

1. Bilexical affinities [issues → the] is plausible
2. Dependency distance mostly with nearby words
3. Intervening material

Dependencies rarely span intervening verbs or punctuation

4. Valency of heads

How many dependents on which side are usual for a head?





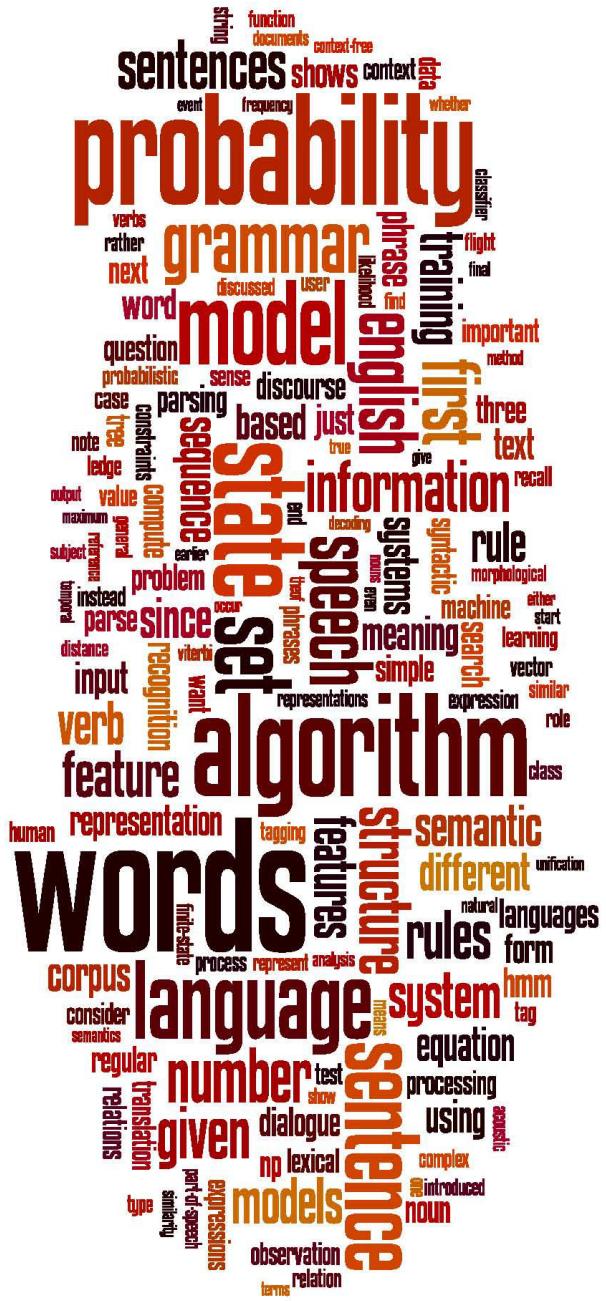
Quiz question!

- Consider this sentence:

Retail sales drop in April cools afternoon market trading.

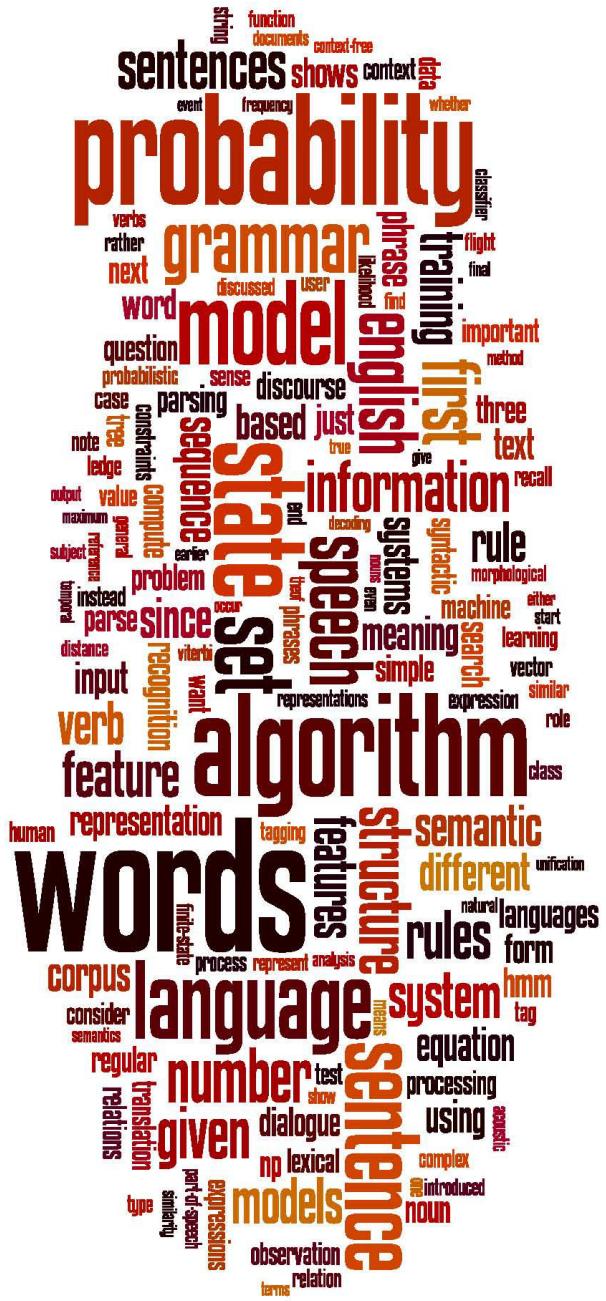
- Which word are these words a dependent of?

1. sales
2. April
3. afternoon
4. trading



Dependency Parsing

Introduction



Greedy Transition-Based Parsing

MaltParser



MaltParser

[Nivre et al. 2008]

- A simple form of greedy discriminative dependency parser
- The parser does a sequence of bottom up actions
 - Roughly like “shift” or “reduce” in a shift-reduce parser, but the “reduce” actions are specialized to create dependencies with head on left or right
- The parser has:
 - a stack σ , written with top to the right
 - which starts with the ROOT symbol
 - a buffer β , written with top to the left
 - which starts with the input sentence
 - a set of dependency arcs A
 - which starts off empty
 - a set of actions



Basic transition-based dependency parser

Start: $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

1. Shift $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
2. Left-Arc_r $\sigma | w_i, w_j | \beta, A \rightarrow \sigma, w_j | \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc_r $\sigma | w_i, w_j | \beta, A \rightarrow \sigma, w_i | \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\beta = \emptyset$

Notes:

- Unlike the regular presentation of the CFG reduce step, dependencies combine one thing from each of stack and buffer



Actions (“arc-eager” dependency parser)

Start: $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

1. Left-Arc_r $\sigma|w_i, w_j|\beta, A \rightarrow \sigma, w_j|\beta, A \cup \{r(w_j, w_i)\}$

Precondition: $r'(w_k, w_i) \notin A, w_i \neq \text{ROOT}$

2. Right-Arc_r $\sigma|w_i, w_j|\beta, A \rightarrow \sigma|w_i|w_j, \beta, A \cup \{r(w_i, w_j)\}$

3. Reduce $\sigma|w_i, \beta, A \rightarrow \sigma, \beta, A$

Precondition: $r'(w_k, w_i) \in A$

4. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$

Finish: $\beta = \emptyset$

This is the common “arc-eager” variant: a head can immediately take a right dependent, before *its* dependents are found



Example

1. Left-Arc_r $\sigma|w_i, w_j|\beta, A \xrightarrow{} \sigma, w_j|\beta, A \cup \{r(w_j, w_i)\}$
Precondition: $(w_k, r', w_i) \notin A, w_i \neq \text{ROOT}$
2. Right-Arc_r $\sigma|w_i, w_j|\beta, A \xrightarrow{} \sigma|w_i|w_j, \beta, A \cup \{r(w_i, w_j)\}$
3. Reduce $\sigma|w_i, \beta, A \xrightarrow{} \sigma, \beta, A$
Precondition: $(w_k, r', w_i) \in A$
4. Shift $\sigma, w_i|\beta, A \xrightarrow{} \sigma|w_i, \beta, A$

Happy children like to play with their friends .

	[ROOT]	[Happy, children, ...]	\emptyset
Shift	[ROOT, Happy]	[children, like, ...]	\emptyset
LA _{amod}	[ROOT]	[children, like, ...]	{amod(children, happy)} = A ₁
Shift	[ROOT, children]	[like, to, ...]	A ₁
LA _{nsubj}	[ROOT]	[like, to, ...]	A ₁ \cup {nsubj(like, children)} = A ₂
RA _{root}	[ROOT, like]	[to, play, ...]	A ₂ \cup {root(ROOT, like)} = A ₃
Shift	[ROOT, like, to]	[play, with, ...]	A ₃
LA _{aux}	[ROOT, like]	[play, with, ...]	A ₃ \cup {aux(play, to)} = A ₄
RA _{xcomp}	[ROOT, like, play]	[with their, ...]	A ₄ \cup {xcomp(like, play)} = A ₅



Example

1. Left-Arc_r $\sigma|w_i, w_j|\beta, A \xrightarrow{} \sigma, w_j|\beta, A \cup \{r(w_j, w_i)\}$
Precondition: $(w_k, r', w_i) \notin A, w_i \neq \text{ROOT}$
2. Right-Arc_r $\sigma|w_i, w_j|\beta, A \xrightarrow{} \sigma|w_i|w_j, \beta, A \cup \{r(w_i, w_j)\}$
3. Reduce $\sigma|w_i, \beta, A \xrightarrow{} \sigma, \beta, A$
Precondition: $(w_k, r', w_i) \in A$
4. Shift $\sigma, w_i|\beta, A \xrightarrow{} \sigma|w_i, \beta, A$

Happy children like to play with their friends .

RA _{xcomp}	[ROOT, like, play]	[with their, ...]	$A_4 \cup \{\text{xcomp}(\text{like}, \text{play})\} = A_5$
RA _{prep}	[ROOT, like, play, with]	[their, friends, ...]	$A_5 \cup \{\text{prep}(\text{play}, \text{with})\} = A_6$
Shift	[ROOT, like, play, with, their]	[friends, .]	A_6
LA _{poss}	[ROOT, like, play, with]	[friends, .]	$A_6 \cup \{\text{poss}(\text{friends}, \text{their})\} = A_7$
RA _{pobj}	[ROOT, like, play, with, friends]	[.]	$A_7 \cup \{\text{pobj}(\text{with}, \text{friends})\} = A_8$
Reduce	[ROOT, like, play, with]	[.]	A_8
Reduce	[ROOT, like, play]	[.]	A_8
Reduce	[ROOT, like]	[.]	A_8
RA _{punc}	[ROOT, like, .]	[]	$A_8 \cup \{\text{punc}(\text{like}, .)\} = A_9$

You terminate as soon as the buffer is empty. Dependencies = A_9



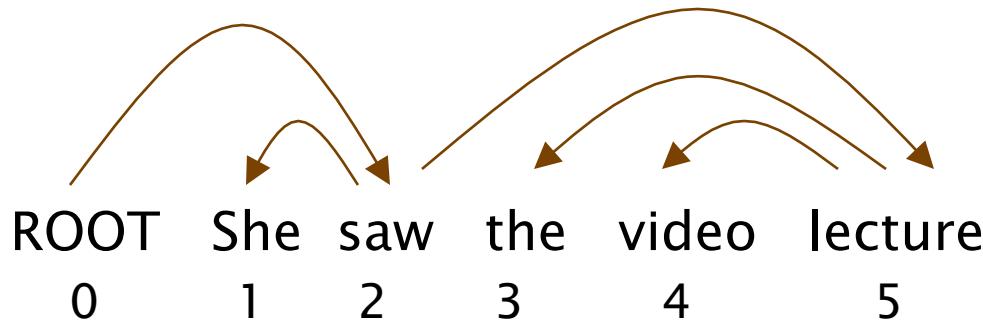
MaltParser

[Nivre et al. 2008]

- We have left to explain how we choose the next action
- Each action is predicted by a discriminative classifier (often SVM, could be maxent classifier) over each legal move
 - Max of 4 untyped choices, max of $|R| \times 2 + 2$ when typed
 - Features: top of stack word, POS; first in buffer word, POS; etc.
- There is NO search (in the simplest and usual form)
 - But you could do some kind of beam search if you wish
- The model's accuracy is *slightly* below the best LPCFGs (evaluated on dependencies), but
- It provides close to state of the art parsing performance
- It provides **VERY** fast linear time parsing



Evaluation of Dependency Parsing: (labeled) dependency accuracy



$$\text{Acc} = \frac{\# \text{ correct deps}}{\# \text{ of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	dobj

Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp



Representative performance numbers

- The CoNLL-X (2006) shared task provides evaluation numbers for various dependency parsing approaches over 13 languages
 - MALT: LAS scores from 65–92%, depending greatly on language/treebank
- Here we give a few UAS numbers for English to allow some comparison to constituency parsing

Parser	UAS%
Sagae and Lavie (2006) ensemble of dependency parsers	92.7
Charniak (2000) generative, constituency	92.2
Collins (1999) generative, constituency	91.7
McDonald and Pereira (2005) – MST graph-based dependency	91.5
Yamada and Matsumoto (2003) – transition-based dependency	90.4



Projectivity

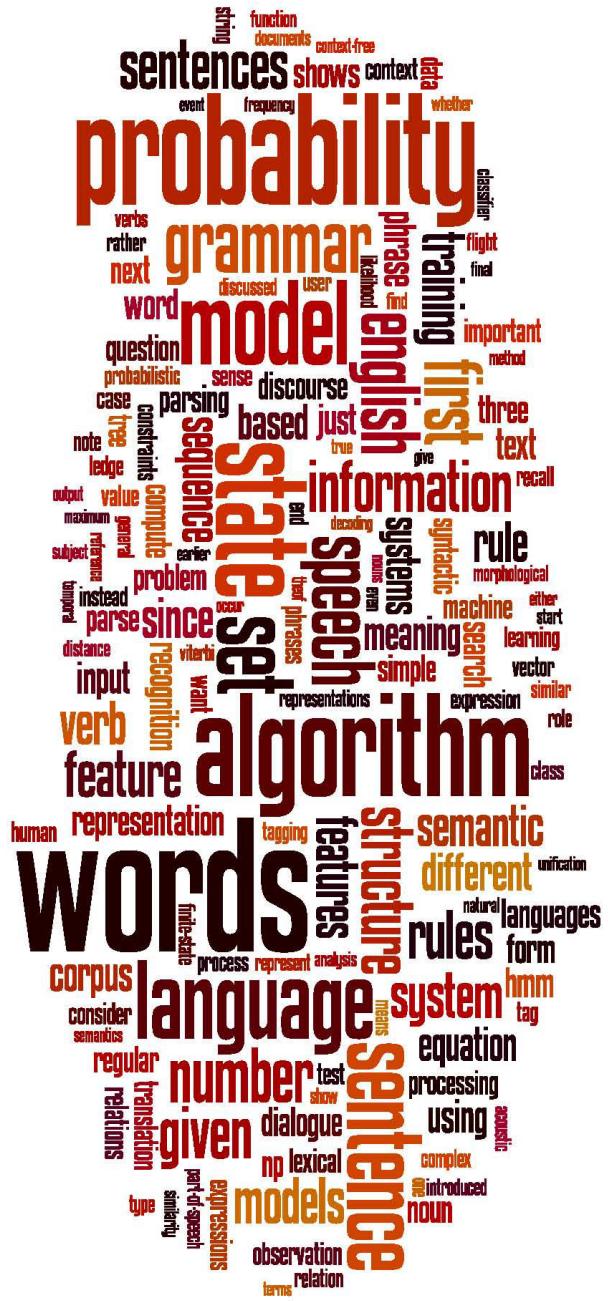
- Dependencies from a CFG tree using heads, must be **projective**
 - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- But dependency theory normally does allow non-projective structures to account for displaced constituents
 - You can't easily get the semantics of certain constructions right without these nonprojective dependencies





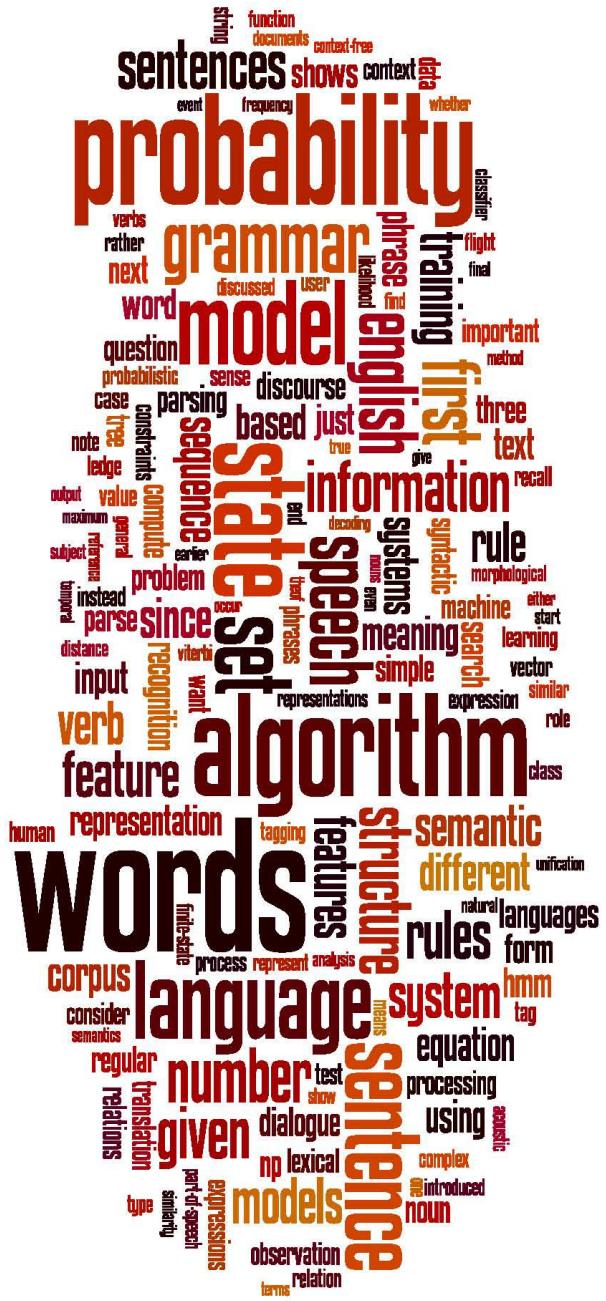
Handling non-projectivity

- The arc-eager algorithm we presented only builds projective dependency trees
- Possible directions to head:
 1. Just declare defeat on nonprojective arcs
 2. Use a dependency formalism which only admits projective representations (a CFG doesn't represent such structures...)
 3. Use a postprocessor to a projective dependency parsing algorithm to identify and resolve nonprojective links
 4. Add extra types of transitions that can model at least most non-projective structures
 5. Move to a parsing mechanism that does not use or require any constraints on projectivity (e.g., the graph-based MSTParser)



Greedy Transition-Based Parsing

MaltParser



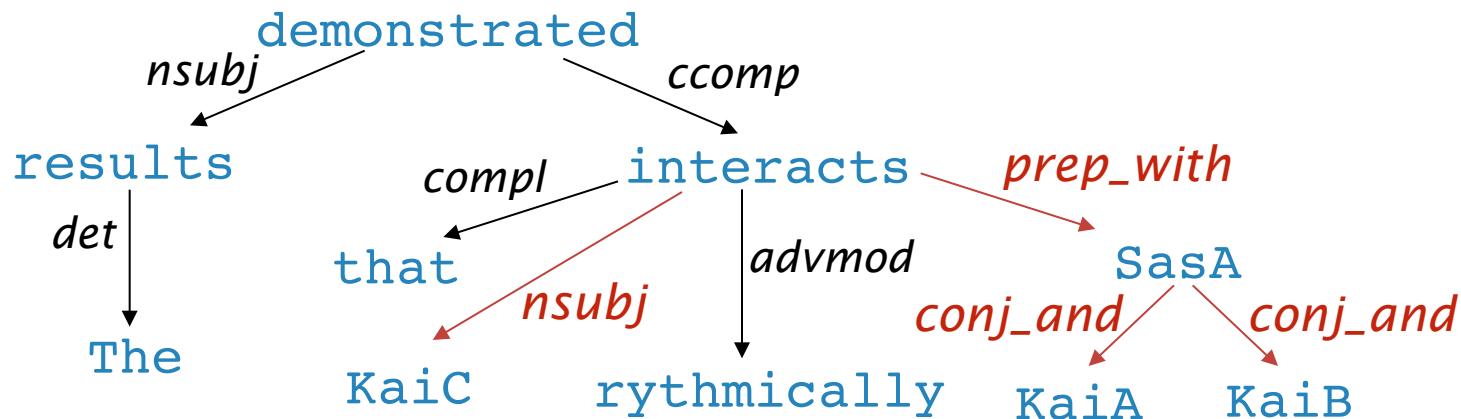
Dependencies encode relational structure

Relation Extraction with Stanford Dependencies



Dependency paths identify relations like protein interaction

[Erkan et al. EMNLP 07, Fundel et al. 2007]



KaiC ←nsubj interacts prep_with→ SasA

KaiC ←nsubj interacts prep_with→ SasA conj_and→ KaiA

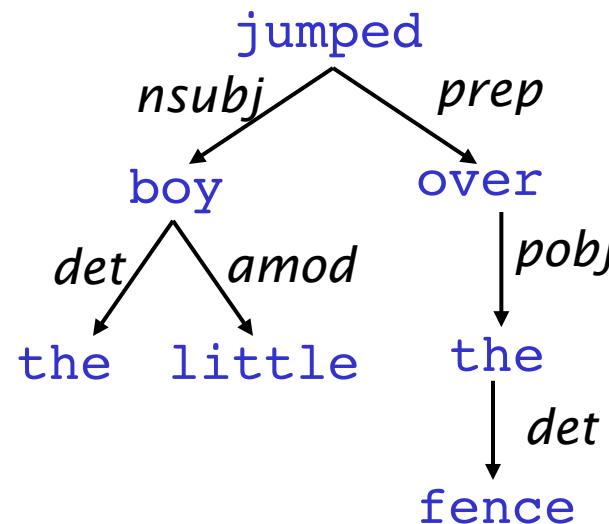
KaiC ←nsubj interacts prep_with→ SasA conj_and→ KaiB



Stanford Dependencies

[de Marneffe et al. LREC 2006]

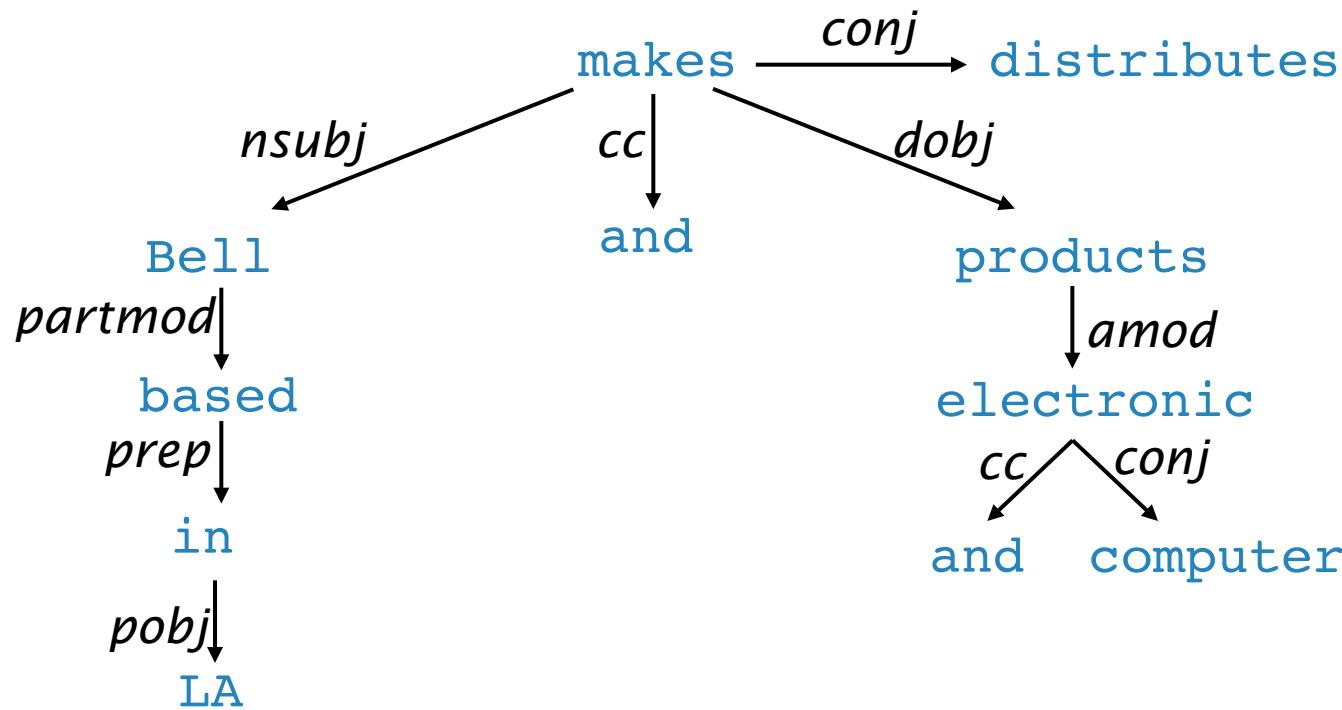
- The basic dependency representation is projective
- It can be generated by postprocessing headed phrase structure parses (Penn Treebank syntax)
- It can also be generated directly by dependency parsers, such as MaltParser, or the Easy-First Parser





Graph modification to facilitate semantic analysis

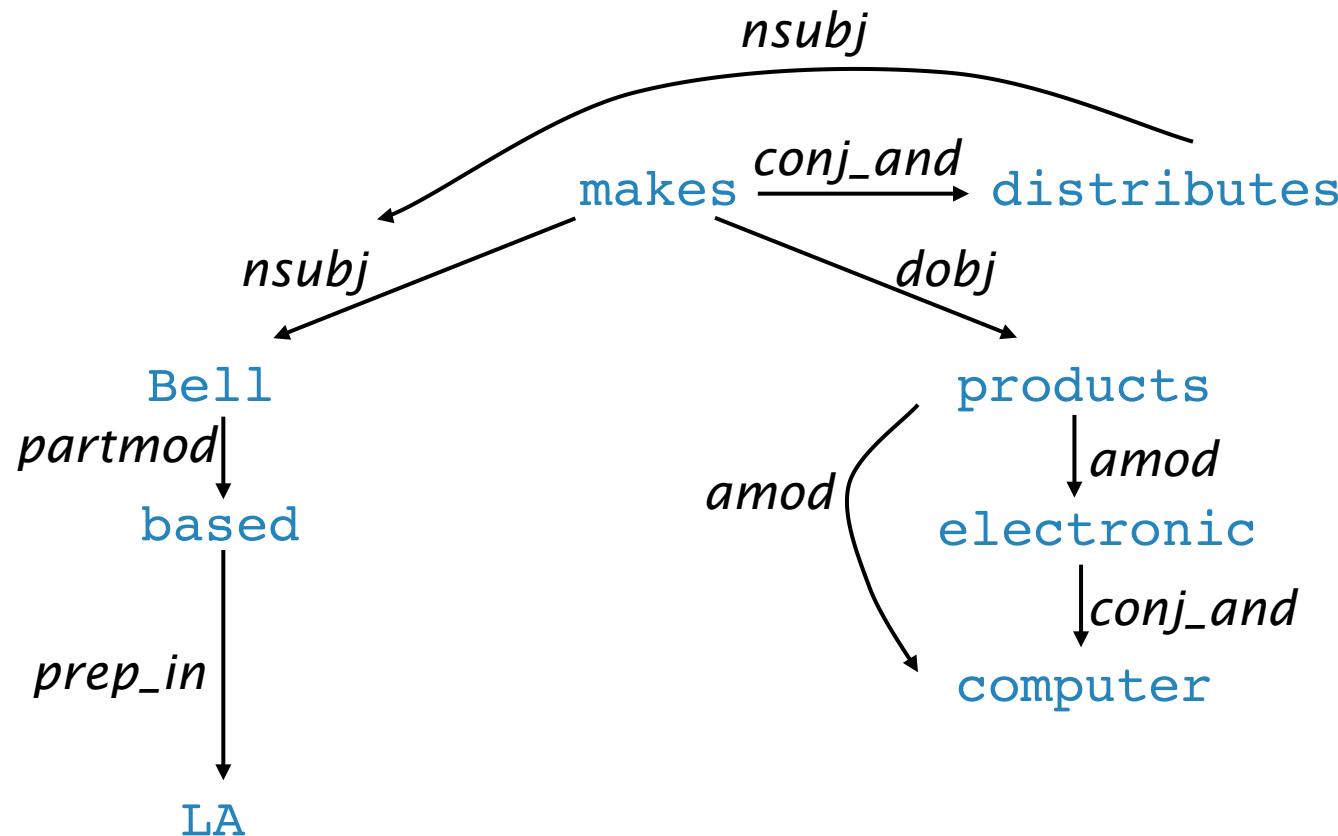
Bell, based in LA, makes and distributes electronic and computer products.





Graph modification to facilitate semantic analysis

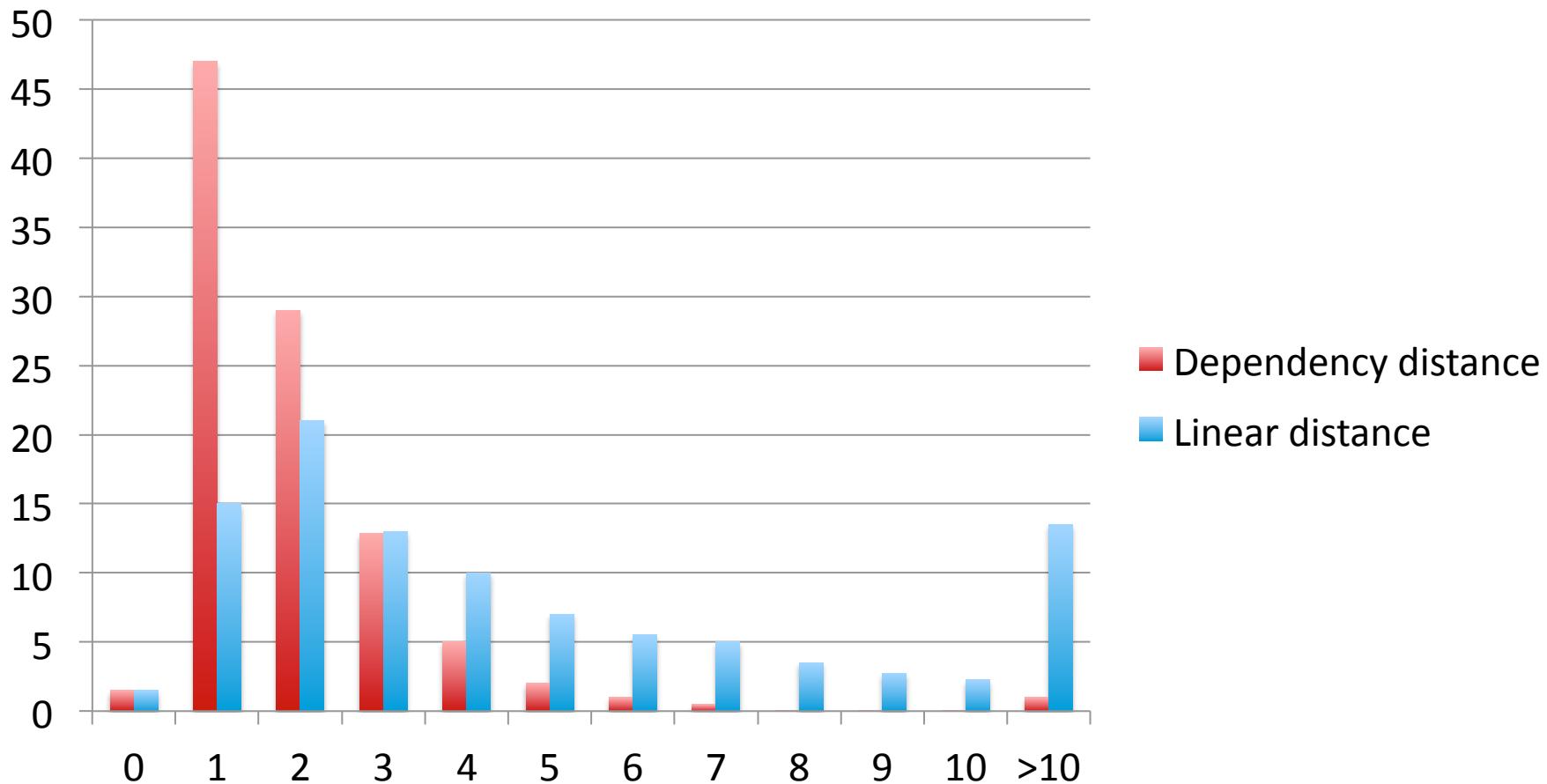
Bell, based in LA, makes and distributes electronic and computer products.

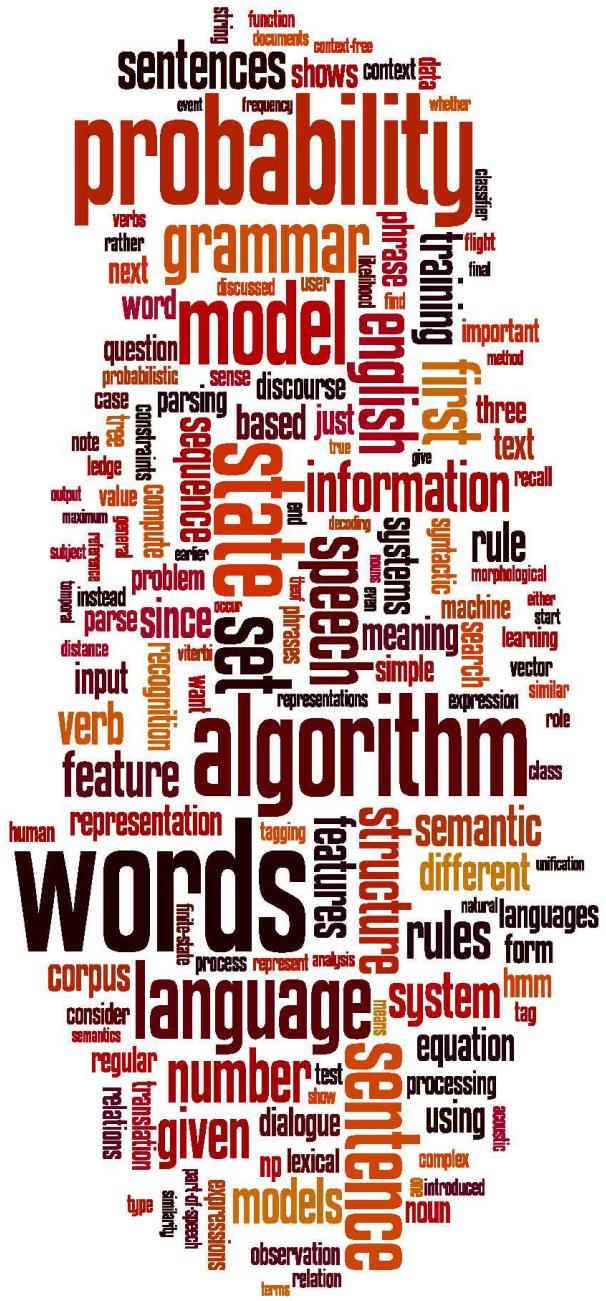




BioNLP 2009/2011 relation extraction shared tasks

[Björne et al. 2009]





Dependencies encode relational structure

Relation Extraction with Stanford Dependencies

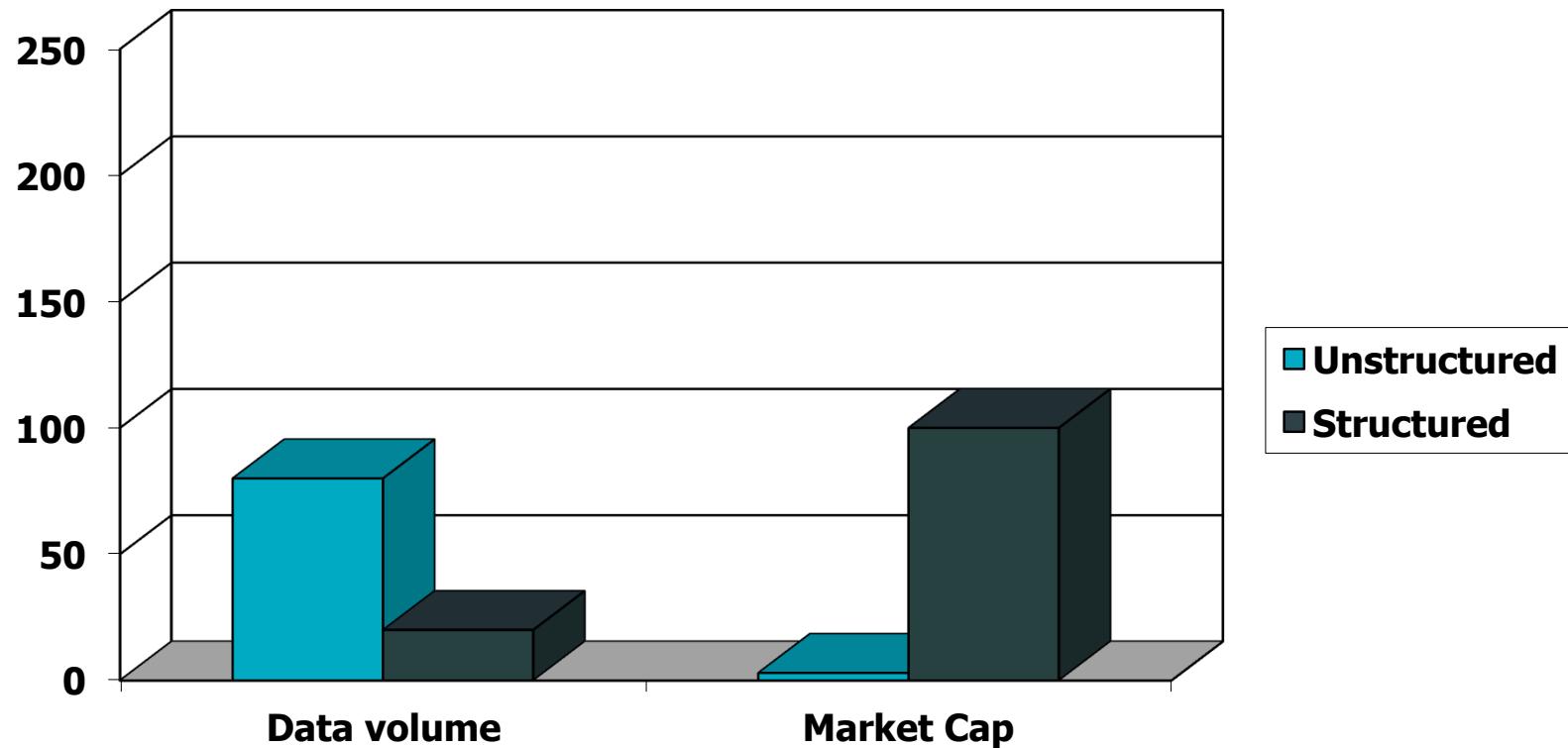
Introduction to **Information Retrieval**

Introducing Information Retrieval
and Web Search

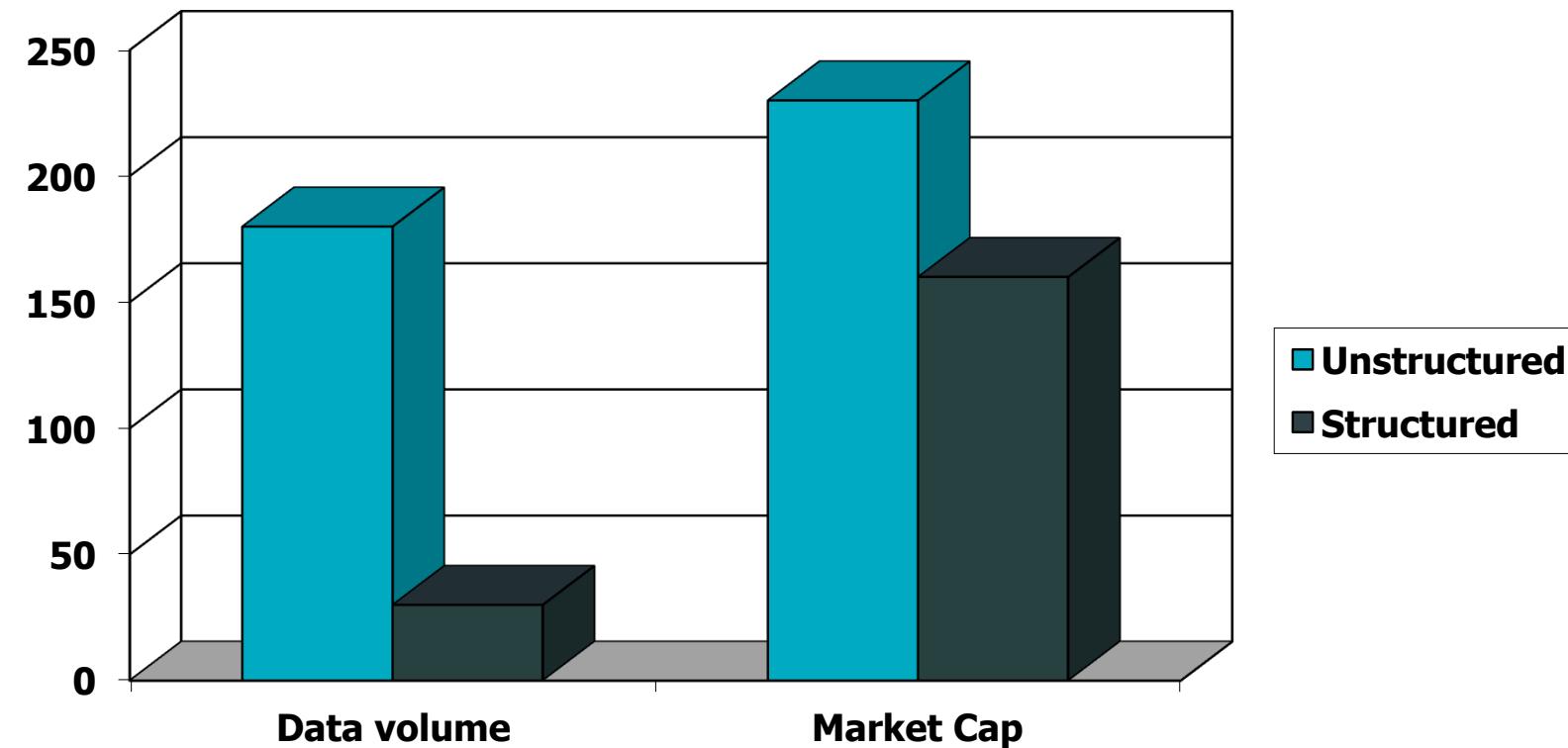
Information Retrieval

- Information Retrieval (IR) is **finding material** (usually documents) of an **unstructured** nature (usually text) that satisfies an **information need** from within **large collections** (usually stored on computers).
- These days we frequently think first of **web search**, but there are many other cases:
 - E-mail search
 - Searching your laptop
 - Corporate knowledge bases
 - Legal information retrieval

Unstructured (text) vs. structured (database) data in the mid-nineties



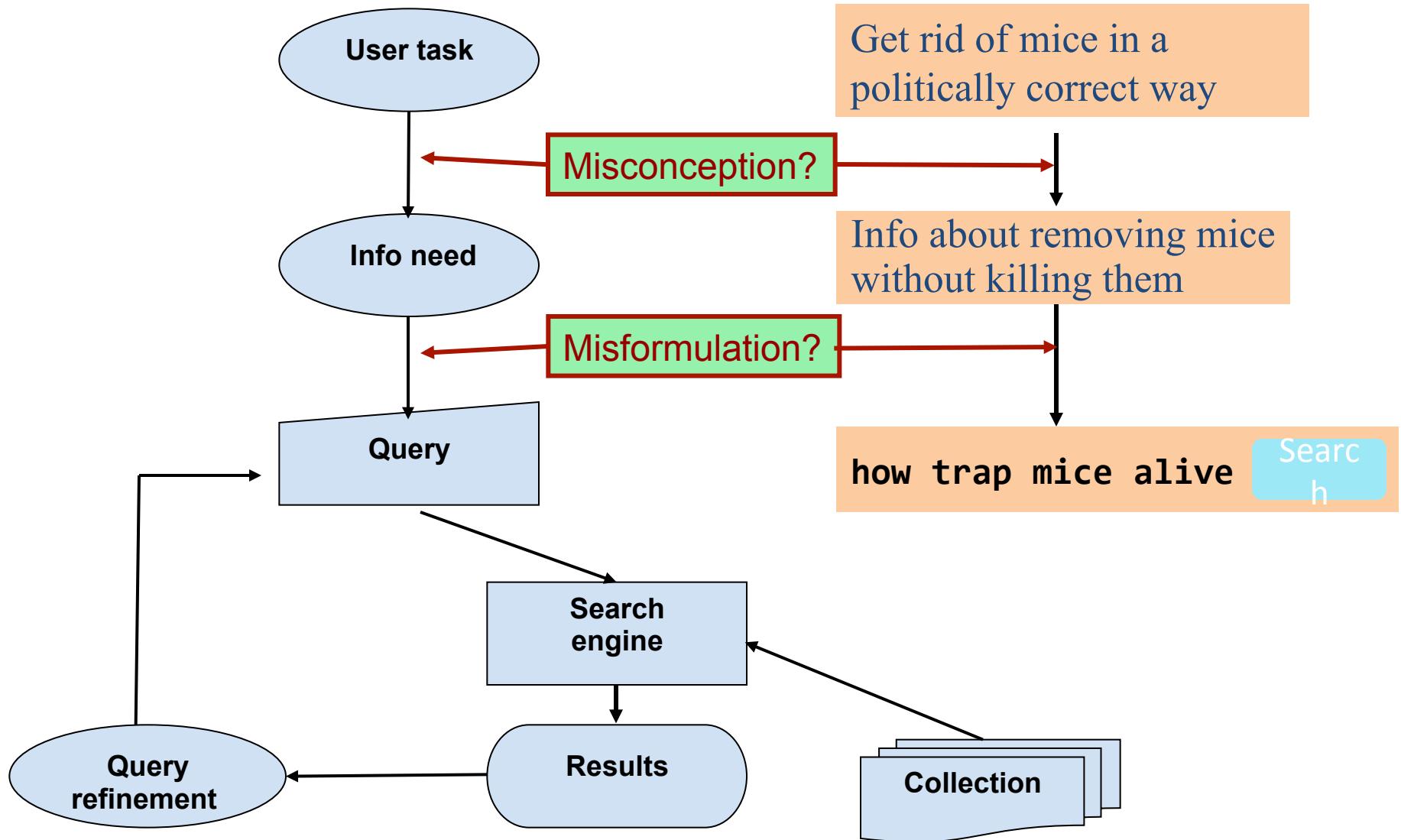
Unstructured (text) vs. structured (database) data today



Basic assumptions of Information Retrieval

- **Collection:** A set of documents
 - Assume it is a static collection for the moment
- **Goal:** Retrieve documents with information that is **relevant** to the user's **information need** and helps the user complete a **task**

The classic search model



How good are the retrieved docs?

- *Precision* : Fraction of retrieved docs that are relevant to the user's **information need**
 - *Recall* : Fraction of relevant docs in collection that are retrieved
-
- More precise definitions and measurements to follow later

Introduction to **Information Retrieval**

Introducing Information Retrieval
and Web Search

Introduction to **Information Retrieval**

Term-document incidence matrices

Unstructured data in 1620

- Which plays of Shakespeare contain the words ***Brutus*** AND ***Caesar*** but ***NOT Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar***, then strip out lines containing ***Calpurnia***?
- Why is that not the answer?
 - Slow (for large corpora)
 - **NOT *Calpurnia*** is non-trivial
 - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
 - Ranked retrieval (best documents to return)
 - Later lectures

Term-document incidence matrices

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

*Brutus AND Caesar BUT NOT
Calpurnia*

1 if play contains
word, 0 otherwise

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for *Brutus*, *Caesar* and *Calpurnia* (complemented) → bitwise AND.
 - 110100 AND
 - 110111 AND
 - 101111 =
 - **100100**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Answers to query

■ Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

■ Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius **Caesar** I was killed i' the
Capitol; **Brutus** killed me.

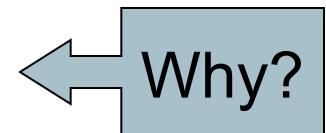


Bigger collections

- Consider $N = 1$ million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
 - 6GB of data in the documents.
- Say there are $M = 500K$ *distinct* terms among these.

Can't build the matrix

- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
 - matrix is extremely sparse.
- What's a better representation?
 - We only record the 1 positions.



Introduction to **Information Retrieval**

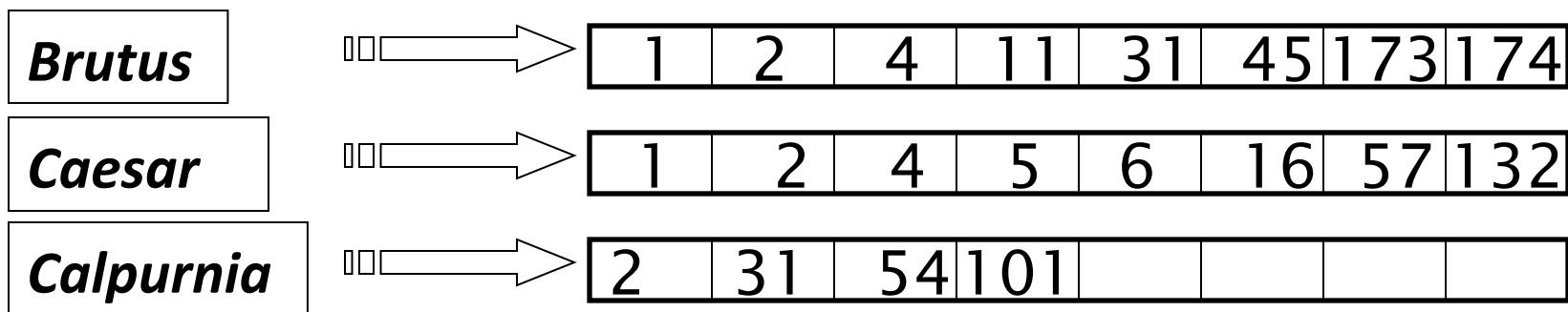
Term-document incidence matrices

Introduction to **Information Retrieval**

The Inverted Index
The key data structure underlying modern IR

Inverted index

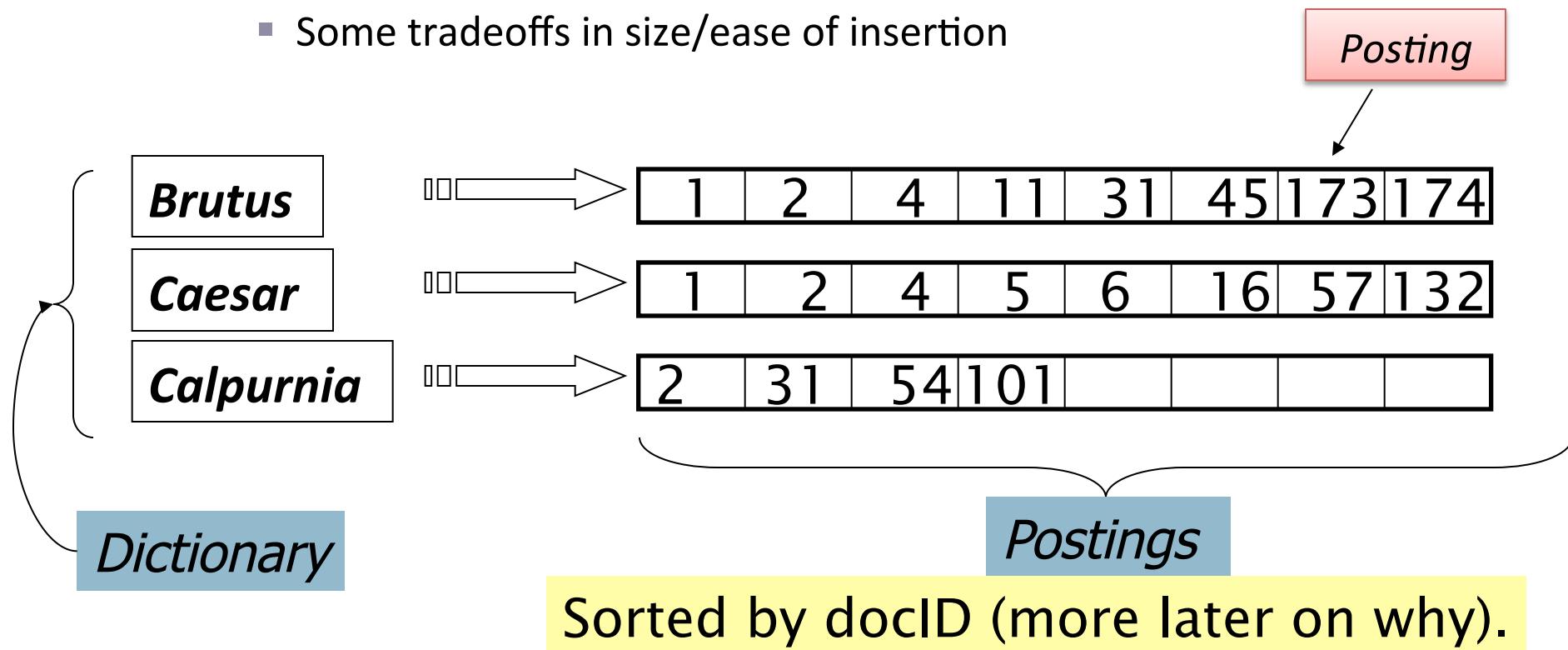
- For each term t , we must store a list of all documents that contain t .
 - Identify each doc by a **docID**, a document serial number
- Can we used fixed-size arrays for this?



What happens if the word **Caesar** is added to document 14?

Inverted index

- We need variable-size **postings lists**
 - On disk, a continuous run of postings is normal and best
 - In memory, can use linked lists or variable length arrays
 - Some tradeoffs in size/ease of insertion



Inverted index construction

Documents to be indexed



Friends, Romans, countrymen.
⋮

Tokenizer

Token stream

Friends

Romans

Countrymen

Linguistic modules

Modified tokens

friend

roman

countryman

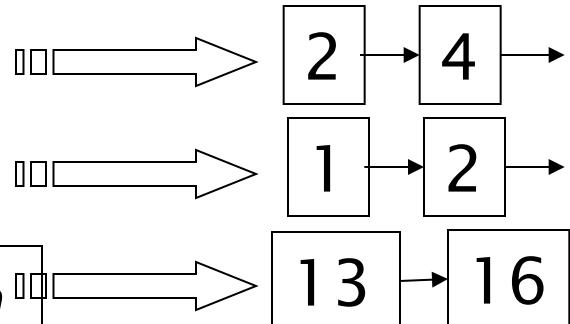
Indexer

Inverted index

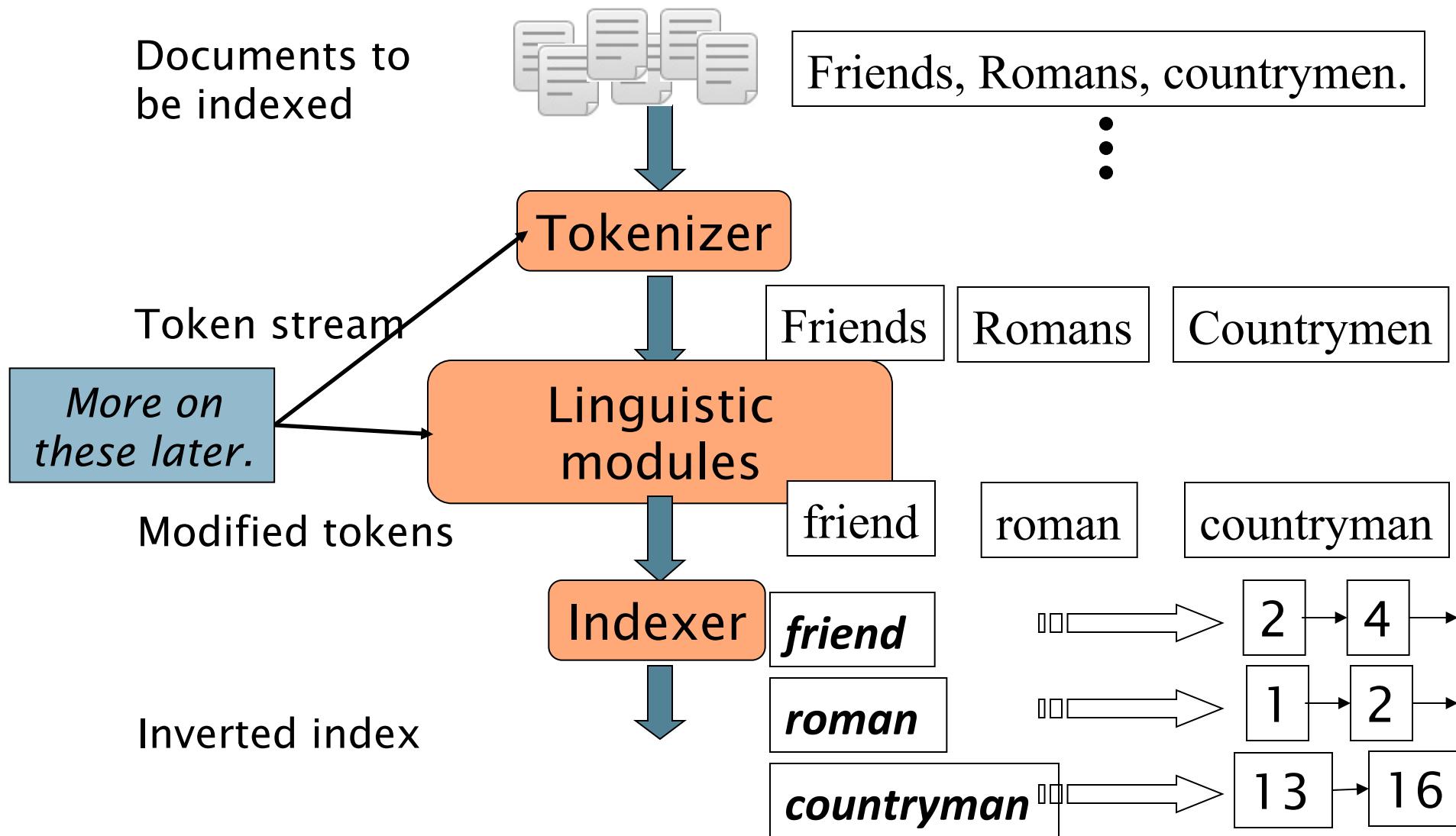
friend

roman

countryman



Inverted index construction



Initial stages of text processing

- Tokenization
 - Cut character sequence into word tokens
 - Deal with "*John's*", *a state-of-the-art solution*
- Normalization
 - Map text and query term to same form
 - You want *U.S.A.* and *USA* to match
- Stemming
 - We may wish different forms of a root to match
 - *authorize*, *authorization*
- Stop words
 - We may omit very common words (or not)
 - *the*, *a*, *to*, *of*

Indexer steps: Token sequence

- Sequence of (Modified token, Document ID) pairs.

Doc 1

I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.

Doc 2

So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indexer steps: Sort

- Sort by terms
 - And then docID

Core indexing step

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

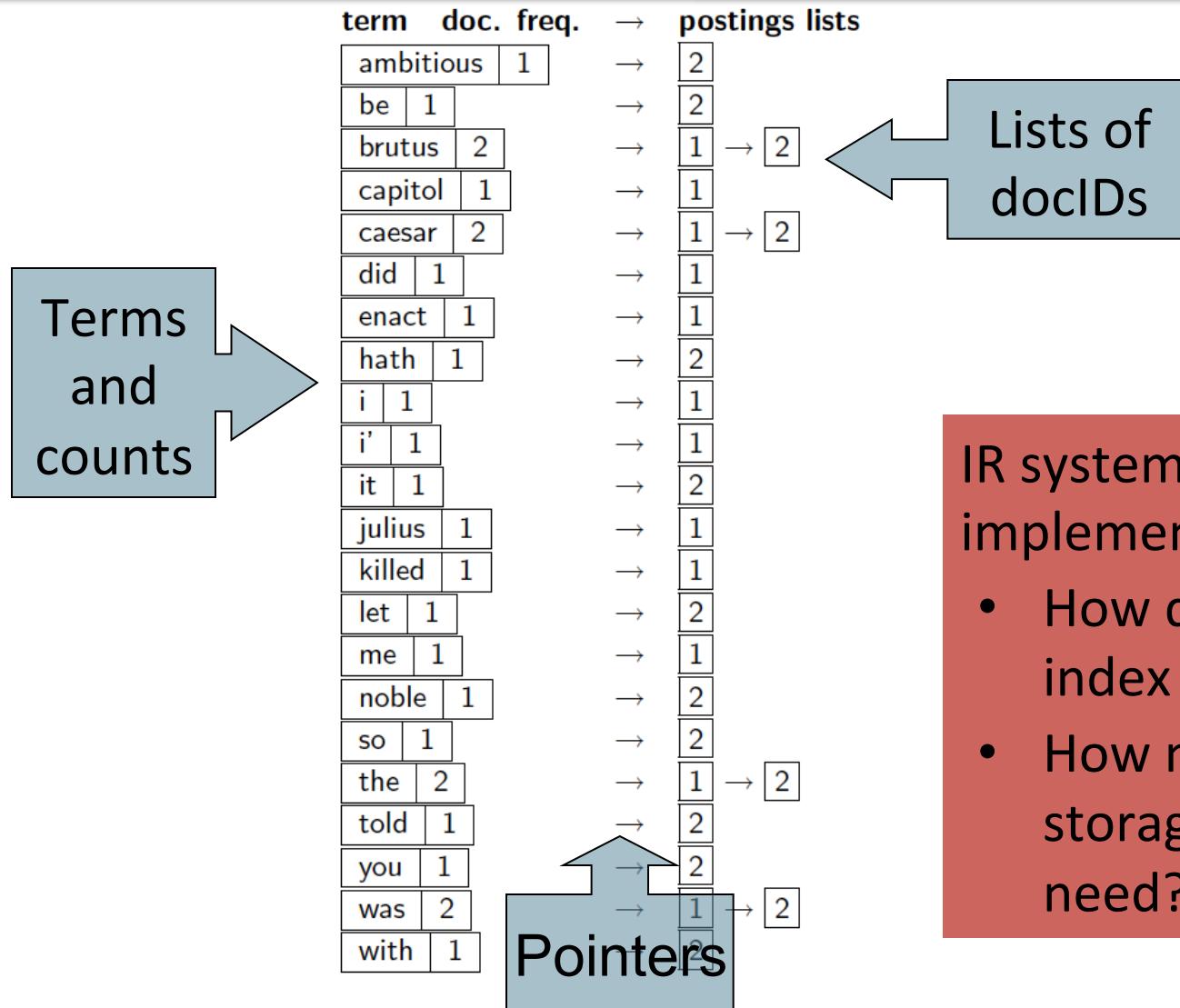
Why frequency?
Will discuss later.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



term	doc.	freq.	→	postings lists
ambitious	1		→	2
be	1		→	2
brutus	2		→	1 → 2
capitol	1		→	1
caesar	2		→	1 → 2
did	1		→	1
enact	1		→	1
hath	1		→	2
i	1		→	1
i'	1		→	1
it	1		→	2
julius	1		→	1
killed	1		→	1
let	1		→	2
me	1		→	1
noble	1		→	2
so	1		→	2
the	2		→	1 → 2
told	1		→	2
you	1		→	2
was	2		→	1 → 2
with	1		→	2

Where do we pay in storage?



IR system implementation

- How do we index efficiently?
- How much storage do we need?

Introduction to **Information Retrieval**

The Inverted Index
The key data structure underlying modern IR

Introduction to **Information Retrieval**

Query processing with an inverted index

The index we just built

- How do we process a query?
 - Later - what kinds of queries can we process?

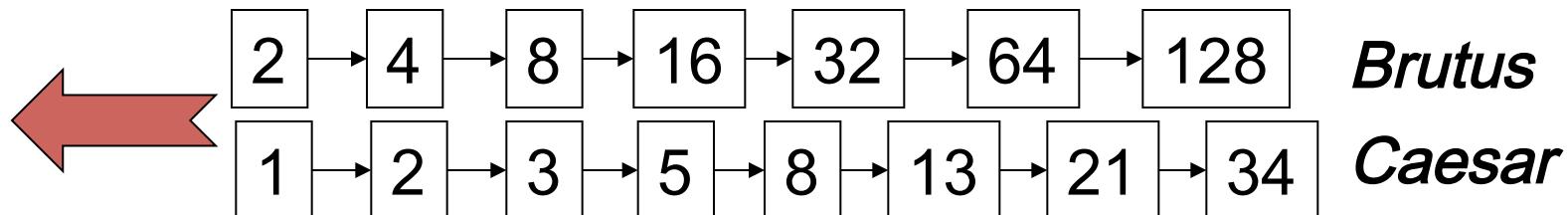


Query processing: AND

- Consider processing the query:

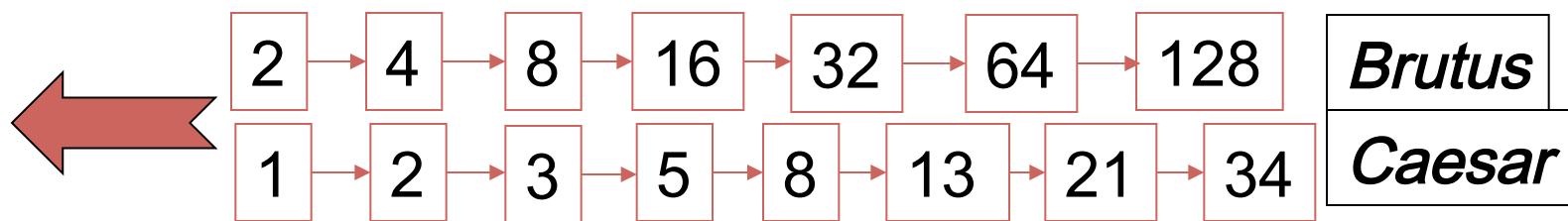
Brutus AND Caesar

- Locate ***Brutus*** in the Dictionary;
 - Retrieve its postings.
- Locate ***Caesar*** in the Dictionary;
 - Retrieve its postings.
- “Merge” the two postings (intersect the document sets):



The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries

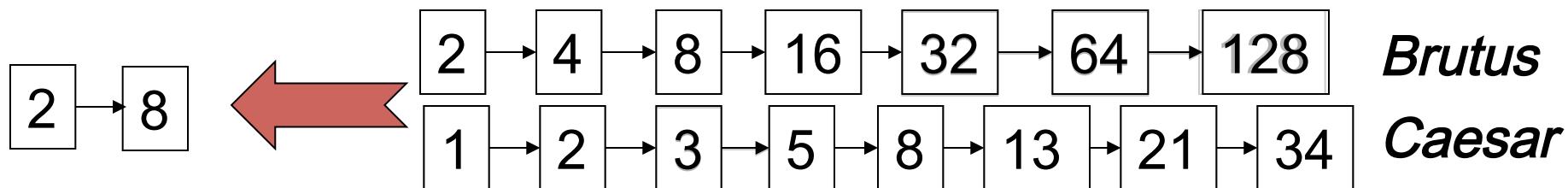


If the list lengths are x and y , the merge takes $O(x+y)$ operations.

Crucial: postings sorted by docID.

The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are x and y , the merge takes $O(x+y)$ operations.

Crucial: postings sorted by docID.

Intersecting two postings lists (a “merge” algorithm)

INTERSECT(p_1, p_2)

```
1  answer ← ⟨ ⟩  
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$   
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$   
4      then ADD(answer,  $\text{docID}(p_1)$ )  
5           $p_1 \leftarrow \text{next}(p_1)$   
6           $p_2 \leftarrow \text{next}(p_2)$   
7      else if  $\text{docID}(p_1) < \text{docID}(p_2)$   
8          then  $p_1 \leftarrow \text{next}(p_1)$   
9          else  $p_2 \leftarrow \text{next}(p_2)$   
10 return answer
```

Introduction to **Information Retrieval**

Query processing with an inverted index

Introduction to **Information Retrieval**

Phrase queries and positional indexes

Phrase queries

- We want to be able to answer queries such as “*stanford university*” – as a phrase
- Thus the sentence “*I went to university at Stanford*” is not a match.
 - The concept of phrase queries has proven easily understood by users; one of the few “advanced search” ideas that works
 - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only $\langle term : docs \rangle$ entries

A first attempt: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example the text “Friends, Romans, Countrymen” would generate the biwords
 - *friends romans*
 - *romans countrymen*
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

Longer phrase queries

- Longer phrases can be processed by breaking them down
- ***stanford university palo alto*** can be broken into the Boolean query on biwords:

stanford university AND university palo AND palo alto

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.



Can have false positives!

Extended biwords

- Parse the indexed text and perform part-of-speech-tagging (POST).
- Bucket the terms into (say) Nouns (N) and articles/prepositions (X).
- Call any string of terms of the form NX^*N an extended biword.
 - Each such extended biword is now made a term in the dictionary.
- Example: *catcher in the rye*

N X X N

- Query processing: parse it into N's and X's
 - Segment query into enhanced biwords
 - Look up in index: *catcher rye*

Issues for biword indexes

- False positives, as noted before
- Index blowup due to bigger dictionary
 - Infeasible for more than biwords, big even for them
- Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy

Solution 2: Positional indexes

- In the postings, store, for each ***term*** the position(s) in which tokens of it appear:

<***term***, number of docs containing ***term***;

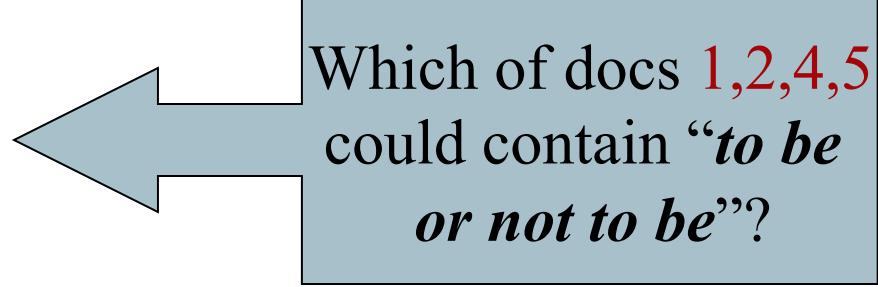
doc1: position1, position2 ... ;

doc2: position1, position2 ... ;

etc.>

Positional index example

<**be**: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, ...>



- For phrase queries, we use a merge algorithm recursively at the document level
- But we now need to deal with more than just equality

Processing a phrase query

- Extract inverted index entries for each distinct term:
to, be, or, not.
- Merge their *doc:position* lists to enumerate all positions with “***to be or not to be***”.
 - ***to:***
 - 2:1,17,74,222,551; **4:8,16,190,429,433;** 7:13,23,191; ...
 - ***be:***
 - 1:17,19; **4:17,191,291,430,434;** 5:14,19,101; ...
- Same general method for proximity searches

Proximity queries

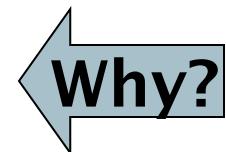
- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
 - Again, here, $/k$ means “within k words of”.
- Clearly, positional indexes can be used for such queries; biword indexes cannot.
- Exercise: Adapt the linear merge of postings to handle proximity queries. Can you make it work for any value of k ?
 - This is a little tricky to do correctly and efficiently
 - See Figure 2.12 of *IIR*

Positional index size

- A positional index expands postings storage *substantially*
 - Even though indices can be compressed
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries ... whether used explicitly or implicitly in a ranking retrieval system.

Positional index size

- Need an entry for each occurrence, not just once per document
- Index size depends on average document size
 - Average web page has <1000 terms
 - SEC filings, books, even some epic poems ... easily 100,000 terms
- Consider a term with frequency 0.1%



Document size	Postings	Positional postings
1000	1	1
100,000	1	100

Rules of thumb

- A positional index is 2–4 as large as a non-positional index
- Positional index size 35–50% of volume of original text
 - Caveat: all of this holds for “English-like” languages

Combination schemes

- These two approaches can be profitably combined
 - For particular phrases (“*Michael Jackson*”, “*Britney Spears*”) it is inefficient to keep on merging positional postings lists
 - Even more so for phrases like “*The Who*”
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme
 - A typical web query mixture was executed in $\frac{1}{4}$ of the time of using just a positional index
 - It required 26% more space than having a positional index alone

Introduction to **Information Retrieval**

Phrase queries and positional indexes

Introduction to **Information Retrieval**

Introducing ranked retrieval

Ranked retrieval

- Thus far, our queries have all been Boolean.
 - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
 - Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
 - Most users incapable of writing Boolean queries (or they are, but they think it's too much work).
 - Most users don't want to wade through 1000s of results.
 - This is particularly true of web search.

Problem with Boolean search: feast or famine

- Boolean queries often result in either too few (≈ 0) or too many (1000s) results.
 - Query 1: “*standard user dlink 650*” \rightarrow 200,000 hits
 - Query 2: “*standard user dlink 650 no card found*” \rightarrow 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
 - AND gives too few; OR gives too many

Ranked retrieval models

- Rather than a set of documents satisfying a query expression, in **ranked retrieval models**, the system returns an ordering over the (top) documents in the collection with respect to a query
- **Free text queries**: Rather than a query language of operators and expressions, the user's query is just one or more words in a human language
- In principle, there are two separate choices here, but in practice, ranked retrieval models have normally been associated with free text queries and vice versa

Feast or famine: not a problem in ranked retrieval

- When a system produces a ranked result set, large result sets are not an issue
 - Indeed, the size of the result set is not an issue
 - We just show the top k (≈ 10) results
 - We don't overwhelm the user
- Premise: the ranking algorithm works

Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in $[0, 1]$ – to each document
- This score measures how well document and query “match”.

Query-document matching scores

- We need a way of assigning a score to a query/document pair
- Let's start with a one-term query
- If the query term does not occur in the document: score should be 0
- The more frequent the query term in the document, the higher the score (should be)
- We will look at a number of alternatives for this

Introduction to **Information Retrieval**

Introducing ranked retrieval

Introduction to **Information Retrieval**

Scoring with the Jaccard coefficient

Take 1: Jaccard coefficient

- A commonly used measure of overlap of two sets A and B is the Jaccard coefficient
- $\text{jaccard}(A,B) = |A \cap B| / |A \cup B|$
- $\text{jaccard}(A,A) = 1$
- $\text{jaccard}(A,B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Scoring example

- What is the query-document match score that the Jaccard coefficient computes for each of the two documents below?
- Query: *ides of march*
- Document 1: *caesar died in march*
- Document 2: *the long march*

Issues with Jaccard for scoring

- It doesn't consider *term frequency* (how many times a term occurs in a document)
 - Rare terms in a collection are more informative than frequent terms
 - Jaccard doesn't consider this information
- We need a more sophisticated way of normalizing for length
 - Later in this lecture, we'll use $|A \cap B| / \sqrt{|A \cup B|}$
... instead of $|A \cap B| / |A \cup B|$ (Jaccard) for length normalization.

Introduction to **Information Retrieval**

Scoring with the Jaccard coefficient

Introduction to **Information Retrieval**

Term frequency weighting

Recall: Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

Term-document count matrices

- Consider the number of occurrences of a term in a document:
 - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Term-document count matrices

- Consider the number of occurrences of a term in a document:
 - Each document is a **count vector** in $\mathbb{N}^{|V|}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
- *John is quicker than Mary and Mary is quicker than John have the same vectors*
- This is called the **bag of words** model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents
 - We will look at “recovering” positional information later on
 - For now: bag of words model

Term frequency tf

- The term frequency $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d .
- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
 - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
 - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

NB: frequency = count in IR

Log-frequency weighting

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \text{tf}_{t,d}, & \text{if } \text{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- Score for a document-query pair: sum over terms t in both q and d :
- $\text{score} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Log-frequency weighting

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \text{tf}_{t,d}, & \text{if } \text{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- $0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms t in both q and d :
- $\text{score} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Introduction to **Information Retrieval**

Term frequency weighting

Introduction to **Information Retrieval**

(Inverse) Document frequency weighting

Document frequency

- Rare terms are more informative than frequent terms
 - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric*.

Document frequency, continued

- Frequent terms are less informative than rare terms
- Consider a query term that is frequent in the collection (e.g., *high*, *increase*, *line*)
- A document containing such a term is more likely to be relevant than a document that doesn't
- But it's not a sure indicator of relevance.
- → For frequent terms, we want positive weights for words like *high*, *increase*, and *line*
- But lower weights than for rare terms.
- We will use document frequency (df) to capture this.

idf weight

- df_t is the document frequency of t : the number of documents that contain t
 - df_t is an inverse measure of the informativeness of t
 - $\text{df}_t \leq N$
- We define the idf (inverse document frequency) of t by

$$\text{idf}_t = \log_{10} (N/\text{df}_t)$$

- We use $\log (N/\text{df}_t)$ instead of N/df_t to “dampen” the effect of idf.

Will turn out the base of the log is immaterial.

idf example, suppose $N = 1$ million

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$\text{idf}_t = \log_{10} (N/\text{df}_t)$$

There is one idf value for each term t in a collection.

Effect of idf on ranking

- Question: Does idf have an effect on ranking for one-term queries, like
 - iPhone

Effect of idf on ranking

- Question: Does idf have an effect on ranking for one-term queries, like
 - iPhone
- idf has no effect on ranking one term queries
 - idf affects the ranking of documents for queries with at least two terms
 - For the query **capricious person**, idf weighting makes occurrences of **capricious** count for much more in the final document ranking than occurrences of **person**.

Collection vs. Document frequency

- The collection frequency of t is the number of occurrences of t in the collection, counting multiple occurrences.
- Example:

Word	Collection frequency	Document frequency
<i>insurance</i>	10440	3997
<i>try</i>	10422	8760

- Which word is a better search term (and should get a higher weight)?

Introduction to **Information Retrieval**

(Inverse) Document frequency weighting

Introduction to **Information Retrieval**

tf-idf weighting

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \times \log_{10}(N / \text{df}_t)$$

- Best known weighting scheme in information retrieval
 - Note: the “-” in tf-idf is a hyphen, not a minus sign!
 - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Final ranking of documents for a query

$$\text{Score}(q, d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

Introduction to **Information Retrieval**

tf-idf weighting

Introduction to **Information Retrieval**

The Vector Space Model (VSM)

Documents as vectors

- Now we have a $|V|$ -dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors – most entries are zero

Queries as vectors

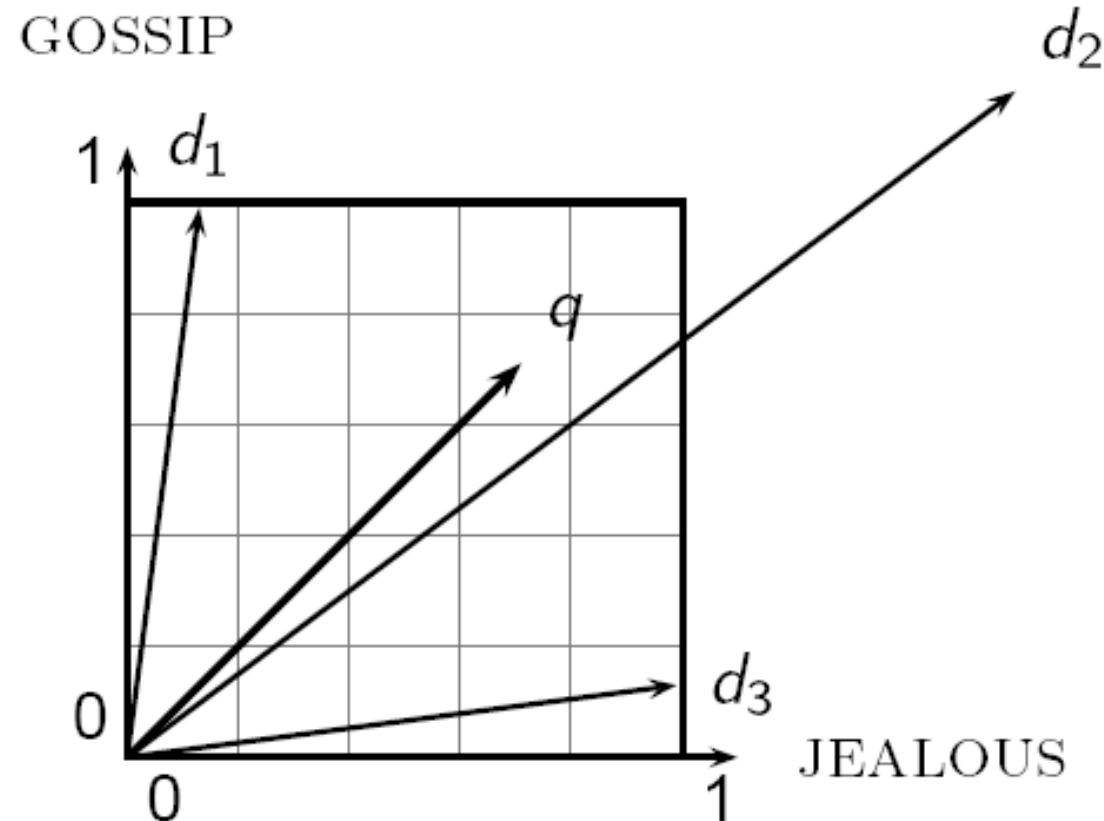
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors
- proximity \approx inverse of distance
- Recall: We do this because we want to get away from the you're-either-in-or-out Boolean model
- Instead: rank more relevant documents higher than less relevant documents

Formalizing vector space proximity

- First cut: distance between two points
 - (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

The Euclidean distance between \vec{q} and $\vec{d_2}$ is large even though the distribution of terms in the query \vec{q} and the distribution of terms in the document $\vec{d_2}$ are very similar.



Use angle instead of distance

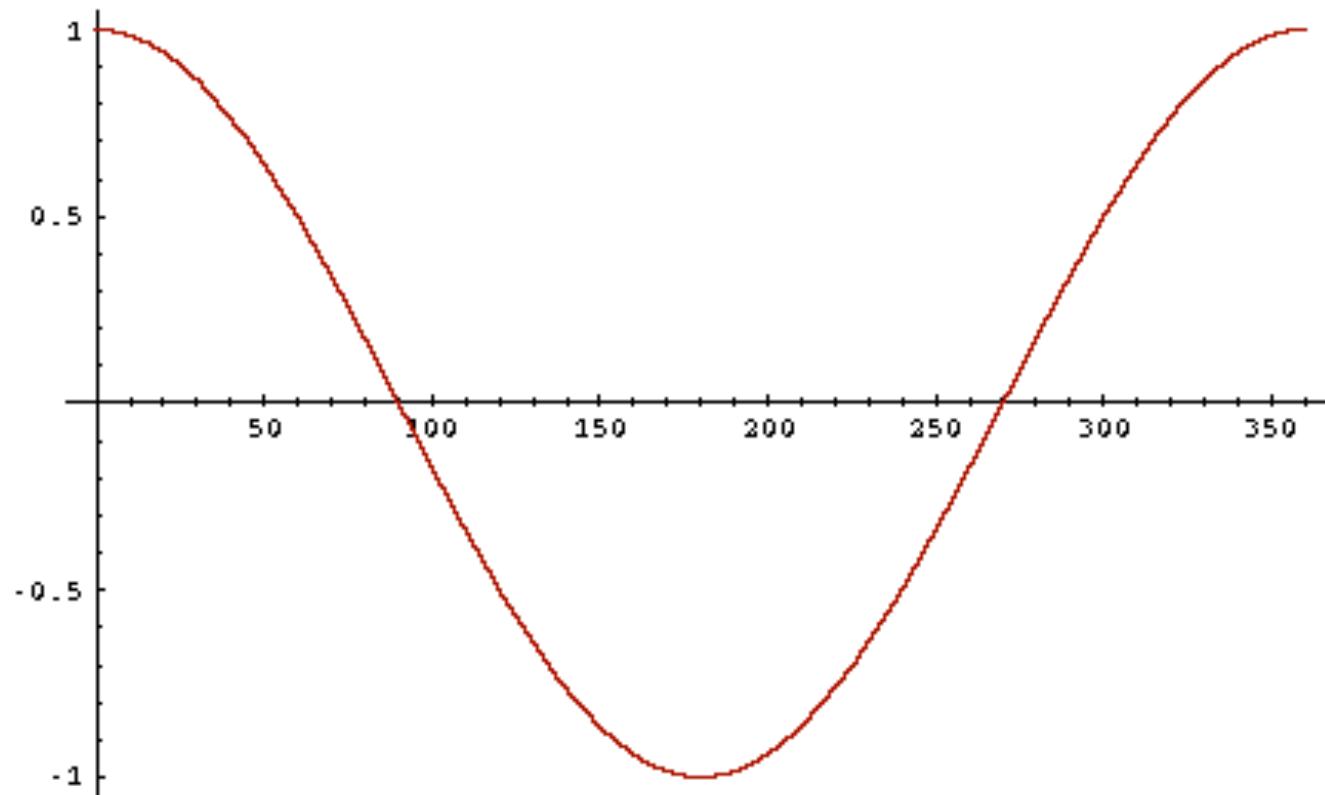
- Thought experiment: take a document d and append it to itself. Call this document d' .
- “Semantically” d and d' have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- Key idea: Rank documents according to angle with query.

From angles to cosines

- The following two notions are equivalent.
 - Rank documents in decreasing order of the angle between query and document
 - Rank documents in increasing order of $\text{cosine}(\text{query}, \text{document})$
- Cosine is a monotonically decreasing function for the interval $[0^\circ, 180^\circ]$

From angles to cosines



- But how – *and why* – should we be computing cosines?

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d' (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
 - Long and short documents now have comparable weights

cosine(query,document)

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|} \cdot \frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Dot product Unit vectors

q_i is the tf-idf weight of term i in the query
 d_i is the tf-idf weight of term i in the document

$\cos(\vec{q}, \vec{d})$ is the cosine similarity of \vec{q} and \vec{d} ... or,
equivalently, the cosine of the angle between \vec{q} and \vec{d} .

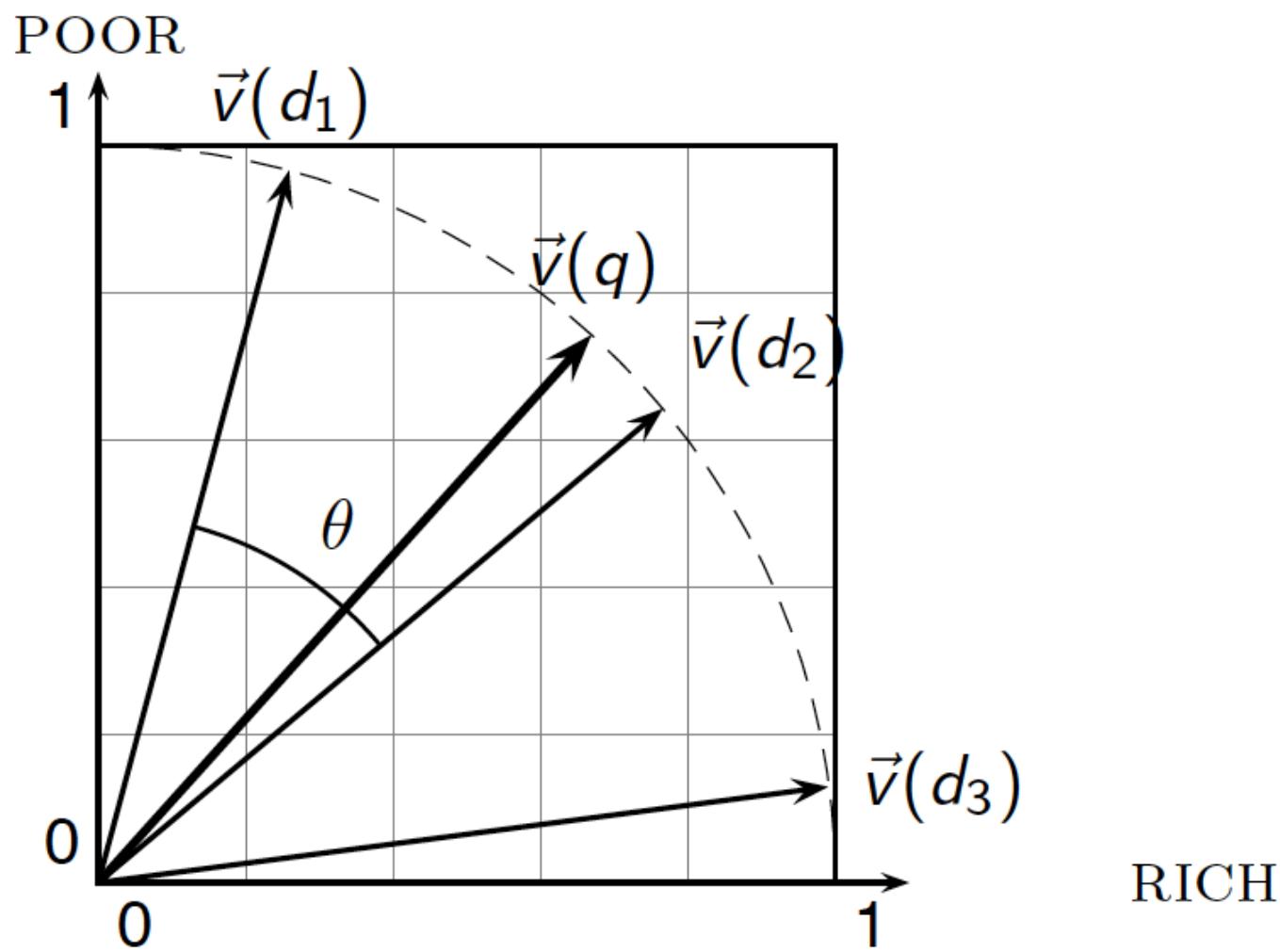
Cosine for length-normalized vectors

- For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

$$\cos(\vec{q}, \vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

for q, d length-normalized.

Cosine similarity illustrated



Cosine similarity amongst 3 documents

How similar are
the novels

SaS: *Sense and*

Sensibility

PaP: *Pride and*

Prejudice, and

WH: *Wuthering*
Heights?

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

Note: To simplify this example, we don't do idf weighting.

3 documents example contd.

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

$$\cos(\text{SaS}, \text{PaP}) \approx$$

$$0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94$$

$$\cos(\text{SaS}, \text{WH}) \approx 0.79$$

$$\cos(\text{PaP}, \text{WH}) \approx 0.69$$

Why do we have $\cos(\text{SaS}, \text{PaP}) > \cos(\text{SaS}, \text{WH})$?

Introduction to **Information Retrieval**

The Vector Space Model (VSM)

Introduction to **Information Retrieval**

Calculating tf-idf cosine scores
in an IR system

tf-idf weighting has many variants

Term frequency	Document frequency	Normalization
n (natural) $tf_{t,d}$	n (no) 1	n (none) 1
I (logarithm) $1 + \log(tf_{t,d})$	t (idf) $\log \frac{N}{df_t}$	c (cosine) $\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$
a (augmented) $0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf) $\max\{0, \log \frac{N - df_t}{df_t}\}$	u (pivoted unique) $1/u$
b (boolean) $\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		b (byte size) $1/CharLength^\alpha, \alpha < 1$
L (log ave) $\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$		

Columns headed ‘n’ are acronyms for weight schemes.

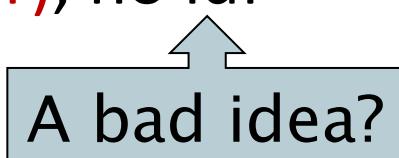
Why is the base of the log in idf immaterial?

tf-idf weighting has many variants

Term frequency	Document frequency	Normalization
n (natural) $tf_{t,d}$	n (no) 1	n (none) 1
I (logarithm) $1 + \log(tf_{t,d})$	t (idf) $\log \frac{N}{df_t}$	c (cosine) $\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$
a (augmented) $0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf) $\max\{0, \log \frac{N - df_t}{df_t}\}$	u (pivoted unique) $1/u$
b (boolean) $\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		b (byte size) $1/CharLength^\alpha, \alpha < 1$
L (log ave) $\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$		

Weighting may differ in queries vs documents

- Many search engines allow for different weightings for queries vs. documents
- SMART Notation: denotes the combination in use in an engine, with the notation $ddd.ooo$, using the acronyms from the previous table
- A very standard weighting scheme is: Inc.ltc
- Document: logarithmic tf (l as first character), no idf and cosine normalization
- Query: logarithmic tf (l in leftmost column), idf (t in second column), cosine normalization ...



A bad idea?

tf-idf example: Inc.ltc

Document: *car insurance auto insurance*

Query: *best car insurance*

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n' liz e	tf-raw	tf-wt	wt	n' liz e	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

Exercise: what is N , the number of docs?

$$\text{Doc length} = \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$$

$$\text{Score} = 0+0+0.27+0.53 = 0.8$$

Computing cosine scores

COSINESCORE(q)

- 1 *float Scores[N] = 0*
- 2 *float Length[N]*
- 3 **for each** query term t
- 4 **do** calculate $w_{t,q}$ and fetch postings list for t
- 5 **for each** pair($d, tf_{t,d}$) in postings list
- 6 **do** $Scores[d] += w_{t,d} \times w_{t,q}$
- 7 Read the array $Length$
- 8 **for each** d
- 9 **do** $Scores[d] = Scores[d]/Length[d]$
- 10 **return** Top K components of $Scores[]$

Summary – vector space ranking

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user

Introduction to **Information Retrieval**

Calculating tf-idf cosine scores
in an IR system

Introduction to **Information Retrieval**

Evaluating search engines

Measures for a search engine

- How fast does it index
 - Number of documents/hour
 - (Average document size)
- How fast does it search
 - Latency as a function of index size
- Expressiveness of query language
 - Ability to express complex information needs
 - Speed on complex queries
- Uncluttered UI
- Is it free?

Measures for a search engine

- All of the preceding criteria are *measurable*: we can quantify speed/size
 - we can make expressiveness precise
- The key measure: user happiness
 - What is this?
 - Speed of response/size of index are factors
 - But blindingly fast, useless answers won't make a user happy
- Need a way of quantifying user happiness with the results returned
 - Relevance of results to user's information need

Evaluating an IR system

- An **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**
- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: **wine red white heart attack effective**
- You evaluate whether the doc addresses the information need, not whether it has these words

Evaluating ranked results

- Evaluation of a result set:
 - If we have
 - a benchmark document collection
 - a benchmark set of queries
 - assessor judgments of whether documents are relevant to queries
- Then we can use Precision/Recall/F measure as before
- Evaluation of ranked results:
 - The system can return any number of results
 - By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a *precision-recall curve*

Recall/Precision

R P

- 1 R
- 2 N
- 3 N
- 4 R
- 5 R
- 6 N
- 7 R
- 8 N
- 9 N
- 10 N

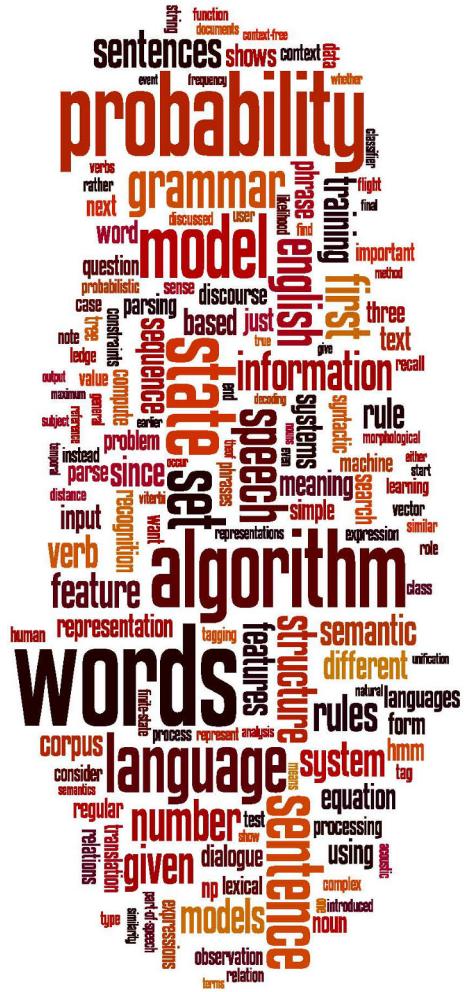
Assume 10 rel docs
in collection

Two current evaluation measures...

- Mean average precision (MAP)
 - AP: Average of the precision value obtained for the top k documents, each time a relevant doc is retrieved
 - Avoids interpolation, use of fixed recall levels
 - Does weight most accuracy of top returned results
 - MAP for set of queries is arithmetic average of APs
 - Macro-averaging: each query counts equally

Introduction to **Information Retrieval**

Evaluating search engines



Word Meaning and Similarity

Word Senses and Word Relations



Reminder: lemma and wordform

- A **lemma** or **citation form**
 - Same stem, part of speech, rough semantics
- A **wordform**
 - The “inflected” word as it appears in text

Wordform	Lemma
banks	bank
sung	sing
duermes	dormir



Lemmas have senses

- One lemma “bank” can have many meanings:

Sense 1: • ...a **bank** can hold the investments in a custodial account¹...

Sense 2: • "...as agriculture burgeons on the east **bank** ² the river will shrink even more"

- **Sense (or word sense)**

- A discrete representation of an aspect of a word’s meaning.

- The lemma **bank** here has two senses



Homonymy

Homonyms: words that share a form but have unrelated, distinct meanings:

- bank_1 : financial institution, bank_2 : sloping land
- bat_1 : club for hitting a ball, bat_2 : nocturnal flying mammal

1. Homographs (bank/bank, bat/bat)

2. Homophones:

1. Write and right
2. Piece and peace



Homonymy causes problems for NLP applications

- Information retrieval
 - “bat care”
- Machine Translation
 - bat: *murciélagos* (animal) or *bate* (for baseball)
- Text-to-Speech
 - bass (stringed instrument) vs. bass (fish)



Polysemy

- 1. The **bank** was constructed in 1875 out of local red brick.
- 2. I withdrew the money from the **bank**
- Are those the same sense?
 - Sense 2: “A financial institution”
 - Sense 1: “The building belonging to a financial institution”
- A **polysemous** word has **related** meanings
 - Most non-rare words have multiple meanings



Metonymy or Systematic Polysemy: A systematic relationship between senses

- Lots of types of polysemy are systematic
 - School, university, hospital
 - All can mean the institution or the building.
- A systematic relationship:
 - Building ↔ Organization
- Other such kinds of systematic polysemy:

Author (Jane Austen wrote Emma)

↔ Works of Author (I love Jane Austen)

Tree (Plums have beautiful blossoms)

↔ Fruit (I ate a preserved plum)



How do we know when a word has more than one sense?

- The “zeugma” test: Two senses of **serve**?
 - Which flights **serve** breakfast?
 - Does Lufthansa **serve** Philadelphia?
 - ?Does Lufthansa serve breakfast and San Jose?
- Since this conjunction sounds weird,
 - we say that these are **two different senses of “serve”**



Synonyms

- Word that have the same meaning in some or all contexts.
 - filbert / hazelnut
 - couch / sofa
 - big / large
 - automobile / car
 - vomit / throw up
 - Water / H₂O
- Two lexemes are synonyms
 - if they can be substituted for each other in all situations
 - If so they have the same **propositional meaning**



Synonyms

- But there are few (or no) examples of perfect synonymy.
 - Even if many aspects of meaning are identical
 - Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.
- Example:
 - Water/H₂O
 - Big/large
 - Brave/courageous



Synonymy is a relation between senses rather than words

- Consider the words *big* and *large*
- Are they synonyms?
 - How **big** is that plane?
 - Would I be flying on a **large** or small plane?
- How about here:
 - Miss Nelson became a kind of **big** sister to Benjamin.
 - ?Miss Nelson became a kind of **large** sister to Benjamin.
- Why?
 - *big* has a sense that means being older, or grown up
 - *large* lacks this sense



Antonyms

- Senses that are opposites with respect to one feature of meaning
- Otherwise, they are very similar!

dark/light	short/long	fast/slow	rise/fall
hot/cold	up/down	in/out	
- More formally: antonyms can
 - define a binary opposition
or be at opposite ends of a scale
 - long/short, fast/slow
 - Be **reversives**:
 - rise/fall, up/down



Hyponymy and Hypernymy

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other
 - *car* is a hyponym of *vehicle*
 - *mango* is a hyponym of *fruit*
- Conversely **hypernym/superordinate** (“hyper is super”)
 - *vehicle* is a **hypernym** of *car*
 - *fruit* is a hypernym of *mango*

Superordinate/hyper	vehicle	fruit	furniture
Subordinate/hyponym	car	mango	chair



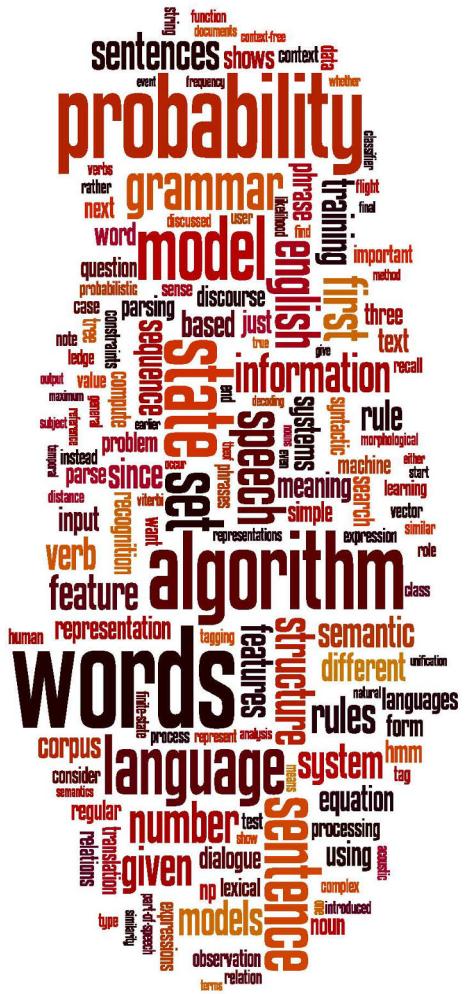
Hyponymy more formally

- Extensional:
 - The class denoted by the superordinate extensionally includes the class denoted by the hyponym
- Entailment:
 - A sense A is a hyponym of sense B if *being an A* entails *being a B*
- Hyponymy is usually transitive
 - (A hypo B and B hypo C entails A hypo C)
- Another name: the **IS-A hierarchy**
 - A **IS-A** B (or A **ISA** B)
 - B **subsumes** A



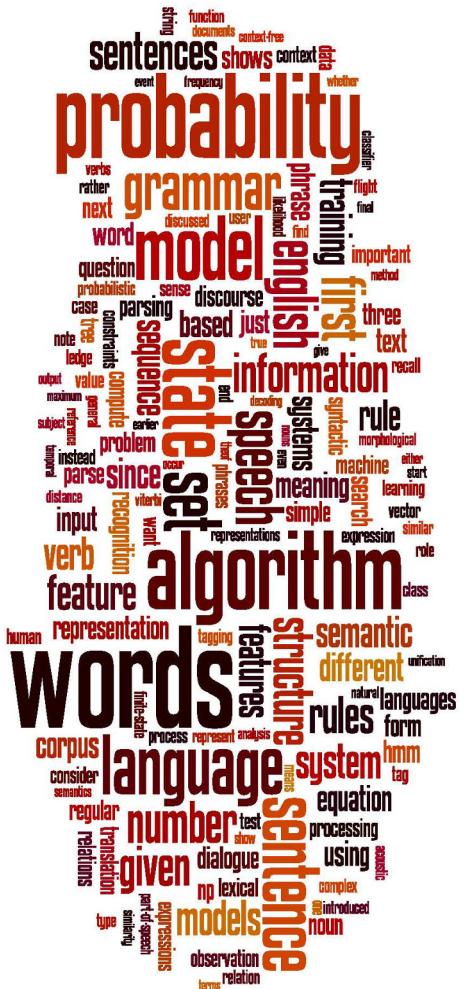
Hyponyms and Instances

- WordNet has both **classes** and **instances**.
- An **instance** is an individual, a proper noun that is a unique entity
 - San Francisco is an **instance** of city
 - But city is a class
 - city is a **hyponym** of municipality...location...



Word Meaning and Similarity

Word Senses and
Word Relations



Word Meaning and Similarity

WordNet and other
Online Thesauri



Applications of Thesauri and Ontologies

- Information Extraction
- Information Retrieval
- Question Answering
- Bioinformatics and Medical Informatics
- Machine Translation



WordNet 3.0

- A hierarchically organized lexical database
- On-line thesaurus + aspects of a dictionary
 - Some other languages available or under development
 - (Arabic, Finnish, German, Portuguese...)

Category	Unique Strings
Noun	117,798
Verb	11,529
Adjective	22,479
Adverb	4,481



Senses of “bass” in Wordnet

Noun

- S: (n) **bass** (the lowest part of the musical range)
- S: (n) **bass**, [bass part](#) (the lowest part in polyphonic music)
- S: (n) **bass**, [basso](#) (an adult male singer with the lowest voice)
- S: (n) [sea bass](#), **bass** (the lean flesh of a saltwater fish of the family Serranidae)
- S: (n) [freshwater bass](#), **bass** (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- S: (n) **bass**, [bass voice](#), [basso](#) (the lowest adult male singing voice)
- S: (n) **bass** (the member with the lowest range of a family of musical instruments)
- S: (n) **bass** (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Adjective

- S: (adj) **bass**, [deep](#) (having or denoting a low vocal or instrumental range) "a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"



How is “sense” defined in WordNet?

- The **synset (synonym set)**, the set of near-synonyms, instantiates a sense or concept, with a **gloss**
- Example: **chump** as a noun with the **gloss**:
“a person who is gullible and easy to take advantage of”
- This sense of “chump” is shared by 9 words:
chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²
- Each of **these** senses have this same gloss
 - (Not **every** sense; sense 2 of gull is the aquatic bird)



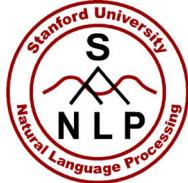
WordNet Hyponym Hierarchy for “bass”

- [S: \(n\) bass, basso](#) (an adult male singer with the lowest voice)
 - [direct hypernym / inherited hypernym / sister term](#)
 - [S: \(n\) singer, vocalist, vocalizer, vocaliser](#) (a person who sings)
 - [S: \(n\) musician, instrumentalist, player](#) (someone who plays a musical instrument (as a profession))
 - [S: \(n\) performer, performing artist](#) (an entertainer who performs a dramatic or musical work for an audience)
 - [S: \(n\) entertainer](#) (a person who tries to please or amuse)
 - [S: \(n\) person, individual, someone, somebody, mortal, soul](#) (a human being) *"there was too much for one person to do"*
 - [S: \(n\) organism, being](#) (a living thing that has (or can develop) the ability to act or function independently)
 - [S: \(n\) living thing, animate thing](#) (a living (or once living) entity)
 - [S: \(n\) whole, unit](#) (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
 - [S: \(n\) object, physical object](#) (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
 - [S: \(n\) physical entity](#) (an entity that has physical existence)
 - [S: \(n\) entity](#) (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))



WordNet Noun Relations

Relation	Also called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	<i>breakfast</i> ¹ → <i>meal</i> ¹
Hyponym	Subordinate	From concepts to subtypes	<i>meal</i> ¹ → <i>lunch</i> ¹
Member Meronym	Has-Member	From groups to their members	<i>faculty</i> ² → <i>professor</i> ¹
Has-Instance		From concepts to instances of the concept	<i>composer</i> ¹ → <i>Bach</i> ¹
Instance		From instances to their concepts	<i>Austen</i> ¹ → <i>author</i> ¹
Member Holonym	Member-Of	From members to their groups	<i>copilot</i> ¹ → <i>crew</i> ¹
Part Meronym	Has-Part	From wholes to parts	<i>table</i> ² → <i>leg</i> ³
Part Holonym	Part-Of	From parts to wholes	<i>course</i> ⁷ → <i>meal</i> ¹
Antonym		Opposites	<i>leader</i> ¹ → <i>follower</i> ¹



WordNet 3.0

- Where it is:
 - <http://wordnetweb.princeton.edu/perl/webwn>
- Libraries
 - Python: WordNet from NLTK
 - <http://www.nltk.org/Home>
 - Java:
 - JWNL, extJWNL on sourceforge



MeSH: Medical Subject Headings thesaurus from the National Library of Medicine

- **MeSH (Medical Subject Headings)**
 - 177,000 entry terms that correspond to 26,142 biomedical “headings”

- **Hemoglobins**

Synset

Entry Terms: Eryhem, Ferrous Hemoglobin, Hemoglobin

Definition: The oxygen-carrying proteins of ERYTHROCYTES.

They are found in all vertebrates and some invertebrates.

The number of globin subunits in the hemoglobin quaternary structure differs between species. Structures range from monomeric to a variety of multimeric arrangements



The MeSH Hierarchy

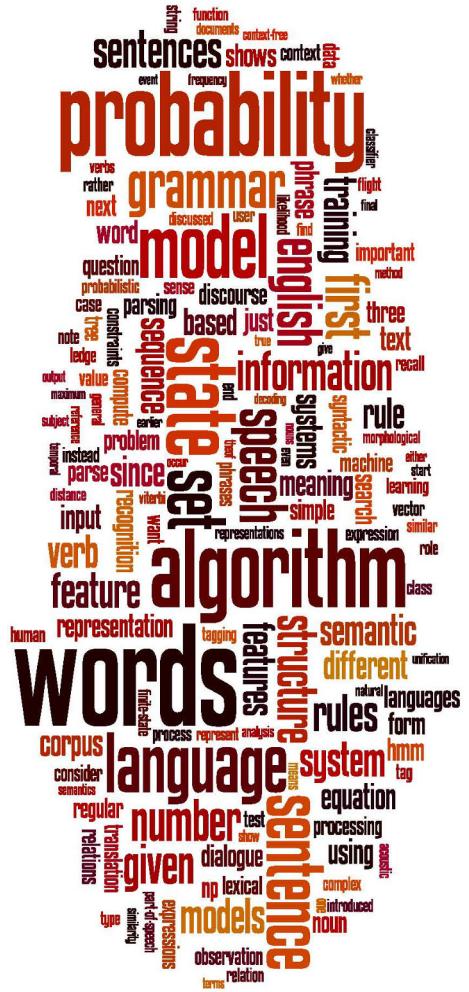
1. + Anatomy [A]
2. + Organisms [B]
3. + Diseases [C]
4. - Chemicals and Drugs [D]
 - [Inorganic Chemicals \[D01\]](#) +
 - [Organic Chemicals \[D02\]](#) +
 - [Heterocyclic Compounds \[D03\]](#) +
 - [Polycyclic Compounds \[D04\]](#) +
 - [Macromolecular Substances \[D05\]](#) +
 - [Hormones, Hormone Substitutes, and Hormone Antagonists \[D06\]](#) +
 - [Enzymes and Coenzymes \[D08\]](#) +
 - [Carbohydrates \[D09\]](#) +
 - [Lipids \[D10\]](#) +
 - [Amino Acids, Peptides, and Proteins \[D12\]](#) +
 - [Nucleic Acids, Nucleotides, and Nucleosides \[D13\]](#) +
 - [Complex Mixtures \[D20\]](#) +
 - [Biological Factors \[D23\]](#) +
 - [Biomedical and Dental Materials \[D25\]](#) +
 - [Pharmaceutical Preparations \[D26\]](#) +

[Amino Acids, Peptides, and Proteins \[D12\]](#)
[Proteins \[D12.776\]](#)
[Blood Proteins \[D12.776.124\]](#)
[Acute-Phase Proteins \[D12.776.124.050\]](#) +
[Anion Exchange Protein 1, Erythrocyte \[D12.776.124.078\]](#)
[Ankyrins \[D12.776.124.080\]](#)
[beta 2-Glycoprotein I \[D12.776.124.117\]](#)
[Blood Coagulation Factors \[D12.776.124.125\]](#) +
[Cholesterol Ester Transfer Proteins \[D12.776.124.197\]](#)
[Fibrin \[D12.776.124.270\]](#) +
[Glycophorin \[D12.776.124.300\]](#)
[Hemocyanin \[D12.776.124.337\]](#)
► [Hemoglobins \[D12.776.124.400\]](#)
[Carboxyhemoglobin \[D12.776.124.400.141\]](#)
[Erythrocytins \[D12.776.124.400.220\]](#)



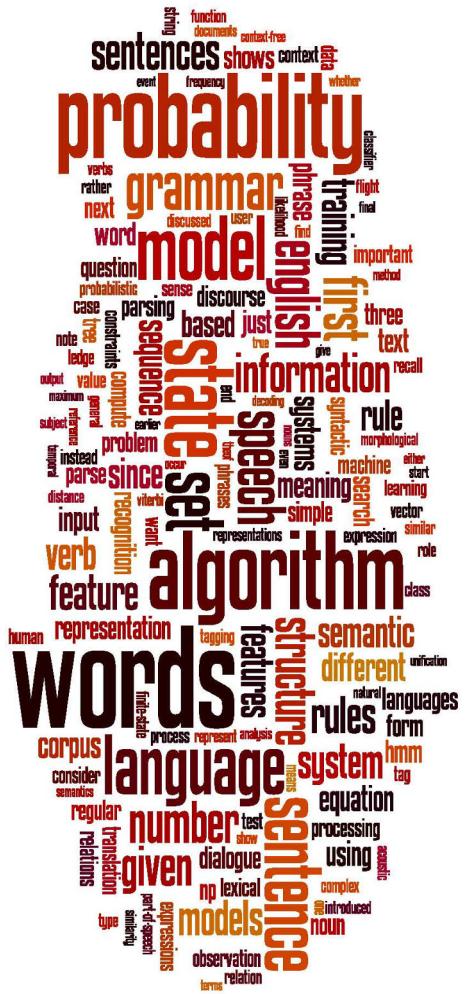
Uses of the MeSH Ontology

- Provide synonyms (“entry terms”)
 - E.g., glucose and dextrose
- Provide hypernyms (from the hierarchy)
 - E.g., glucose ISA monosaccharide
- Indexing in MEDLINE/PubMED database
 - NLM’s bibliographic database:
 - 20 million journal articles
 - Each article hand-assigned 10-20 MeSH terms



Word Meaning and Similarity

WordNet and other Online Thesauri



Word Meaning and Similarity

Word Similarity:
Thesaurus Methods



Word Similarity

- **Synonymy:** a binary relation
 - Two words are either synonymous or not
- **Similarity (or distance):** a looser metric
 - Two words are more similar if they share more features of meaning
- Similarity is properly a relation between **senses**
 - The word “bank” is not similar to the word “slope”
 - Bank¹ is similar to fund³
 - Bank² is similar to slope⁵
- But we’ll compute similarity over both words and senses



Why word similarity

- Information retrieval
- Question answering
- Machine translation
- Natural language generation
- Language modeling
- Automatic essay grading
- Plagiarism detection
- Document clustering



Word similarity and word relatedness

- We often distinguish **word similarity** from **word relatedness**
 - **Similar words:** near-synonyms
 - **Related words:** can be related any way
 - car, bicycle: **similar**
 - car, gasoline: **related**, not similar

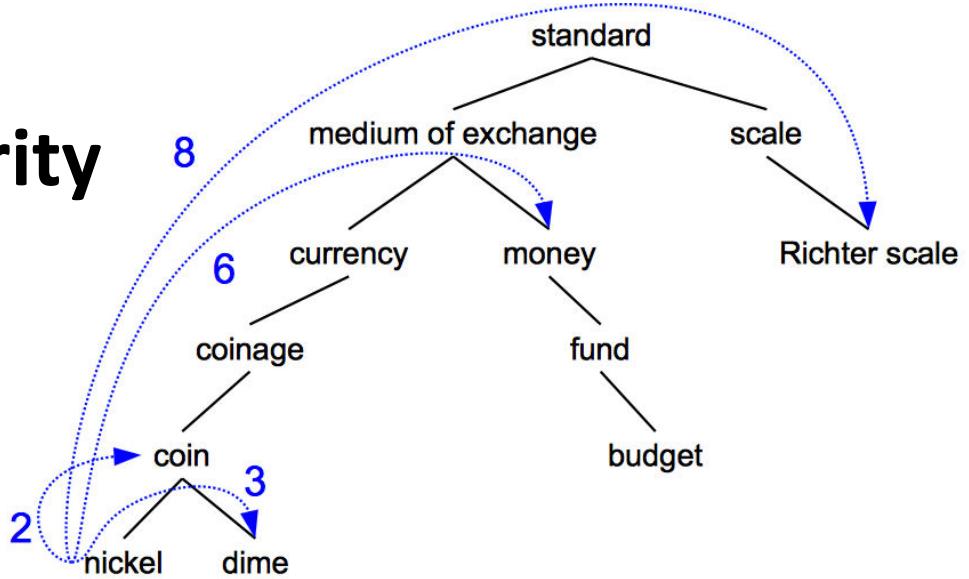


Two classes of similarity algorithms

- Thesaurus-based algorithms
 - Are words “nearby” in hypernym hierarchy?
 - Do words have similar glosses (definitions)?
- Distributional algorithms
 - Do words have similar distributional contexts?



Path based similarity



- Two concepts (senses/synsets) are similar if they are near each other in the thesaurus hierarchy
 - =have a short path between them
 - concepts have path 1 to themselves



Refinements to path-based similarity

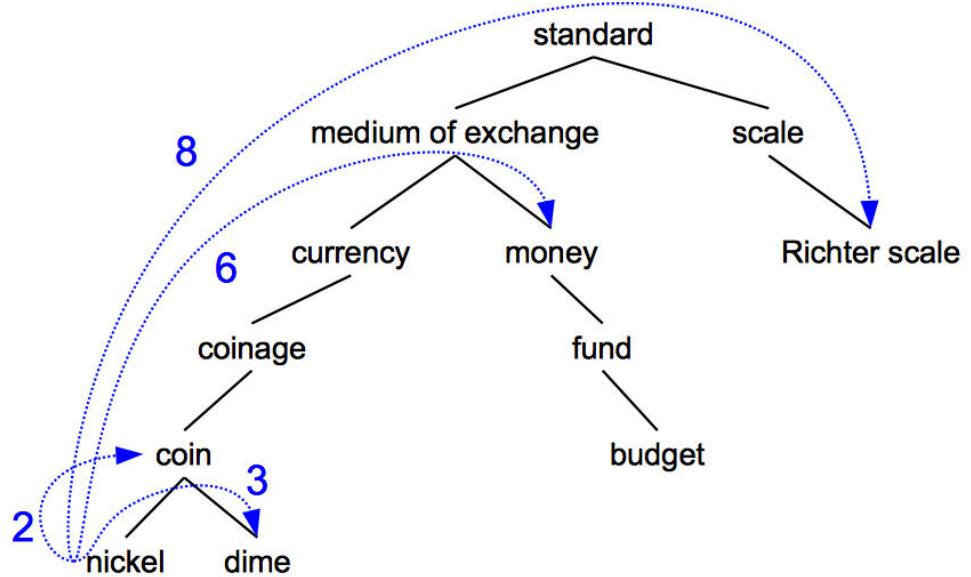
- $\text{pathlen}(c_1, c_2) = 1 + \text{number of edges in the shortest path in the hypernym graph between sense nodes } c_1 \text{ and } c_2$
- ranges from 0 to 1 (identity)
- $\text{simpath}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}$
- $\text{wordsim}(w_1, w_2) = \max_{c_1 \in \text{senses}(w_1), c_2 \in \text{senses}(w_2)} \text{sim}(c_1, c_2)$



Example: path-based similarity

$$\text{simpath}(c_1, c_2) = 1/\text{pathlen}(c_1, c_2)$$

- $\text{simpath}(\text{nickel}, \text{coin}) = 1/2 = .5$
- $\text{simpath}(\text{fund}, \text{budget}) = 1/2 = .5$
- $\text{simpath}(\text{nickel}, \text{currency}) = 1/4 = .25$
- $\text{simpath}(\text{nickel}, \text{money}) = 1/6 = .17$
- $\text{simpath}(\text{coinage}, \text{Richter scale}) = 1/6 = .17$





Problem with basic path-based similarity

- Assumes each link represents a uniform distance
 - But *nickel* to *money* seems to us to be closer than *nickel* to *standard*
 - Nodes high in the hierarchy are very abstract
- We instead want a metric that
 - Represents the cost of each edge independently
 - Words connected only through abstract nodes
 - are less similar



Information content similarity metrics

Resnik 1995. Using information content to evaluate semantic similarity in a taxonomy. IJCAI

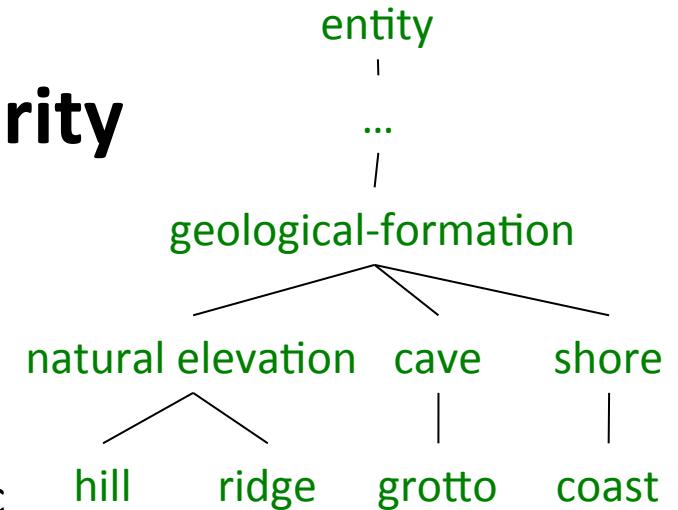
- Let's define $P(c)$ as:
 - The probability that a randomly selected word in a corpus is an instance of concept c
 - Formally: there is a distinct random variable, ranging over words, associated with each concept in the hierarchy
 - for a given concept, each observed noun is either
 - a member of that concept with probability $P(c)$
 - not a member of that concept with probability $1-P(c)$
 - All words are members of the root node (Entity)
 - $P(\text{root})=1$
 - The lower a node in hierarchy, the lower its probability



Information content similarity

- Train by counting in a corpus
 - Each instance of `hill` counts toward frequency of *natural elevation*, *geological formation*, *entity*, etc
 - Let `words(c)` be the set of all words that are children of node `c`
 - `words("geo-formation") = {hill,ridge,grotto,coast,cave,shore,natural elevation}`
 - `words("natural elevation") = {hill, ridge}`

$$P(c) = \frac{\sum_{w \in \text{words}(c)} \text{count}(w)}{N}$$

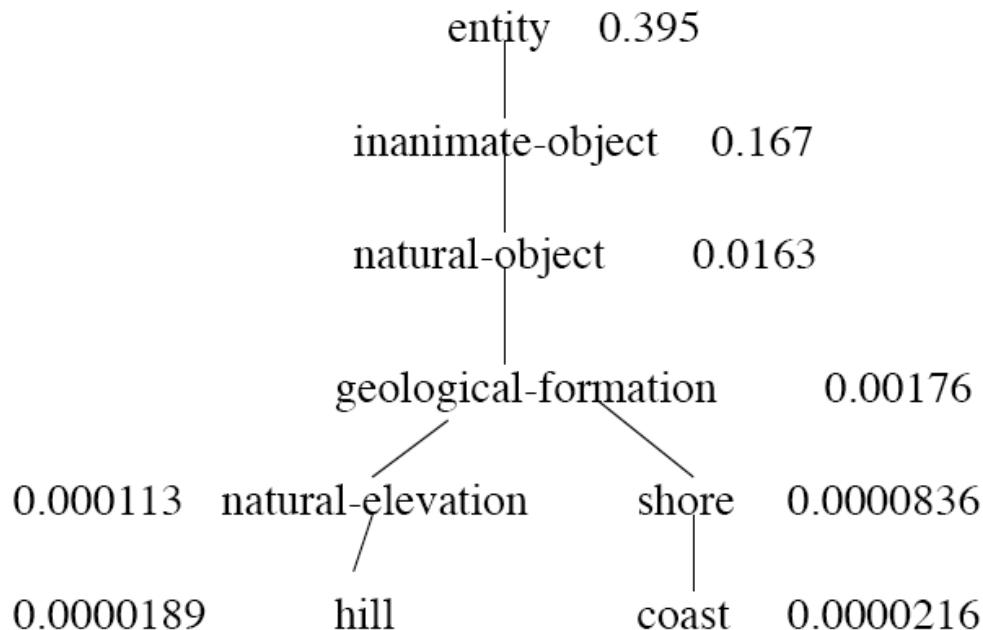




Information content similarity

- WordNet hierarchy augmented with probabilities $P(c)$

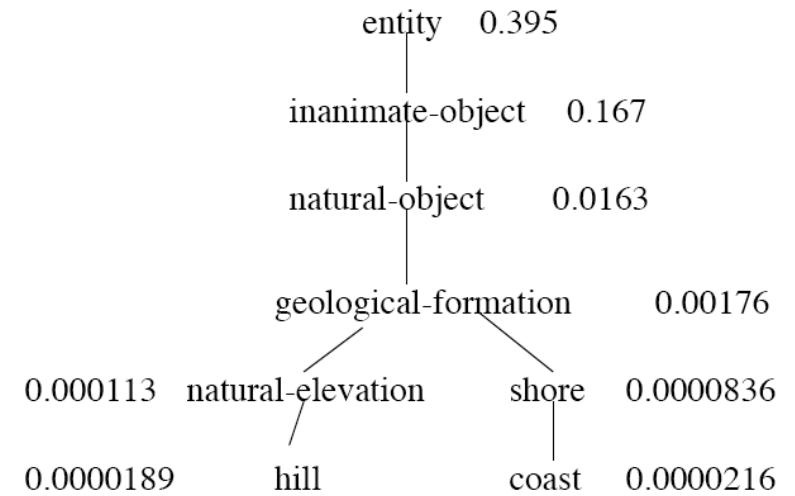
D. Lin. 1998. An Information-Theoretic Definition of Similarity. ICML 1998





Information content: definitions

- Information content:
 $IC(c) = -\log P(c)$
- Most informative subsumer
 (Lowest common subsumer)
 $LCS(c_1, c_2) =$
 The most informative (lowest)
 node in the hierarchy
 subsuming both c_1 and c_2





Using information content for similarity: the Resnik method

Philip Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. IJCAI 1995.
Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. JAIR 11, 95-130.

- The similarity between two words is related to their common information
- The more two words have in common, the more similar they are
- Resnik: measure common information as:
 - The information content of the most informative (lowest) subsumer (MIS/LCS) of the two nodes
 - $\text{sim}_{\text{resnik}}(c_1, c_2) = -\log P(\text{LCS}(c_1, c_2))$



Dekang Lin method

Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. ICML

- Intuition: Similarity between A and B is not just what they have in common
- The more **differences** between A and B, the less similar they are:
 - Commonality: the more A and B have in common, the more similar they are
 - Difference: the more differences between A and B, the less similar
- Commonality: $\text{IC}(\text{common}(A,B))$
- Difference: $\text{IC}(\text{description}(A,B)) - \text{IC}(\text{common}(A,B))$



Dekang Lin similarity theorem

- The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are

$$sim_{Lin}(A, B) \propto \frac{IC(common(A, B))}{IC(description(A, B))}$$

- Lin (altering Resnik) defines $IC(common(A, B))$ as $2 \times$ information of the LCS

$$sim_{Lin}(c_1, c_2) = \frac{2 \log P(LCS(c_1, c_2))}{\log P(c_1) + \log P(c_2)}$$



Lin similarity function

	geological-formation	0.00176
0.000113	natural-elevation	0.0000836
0.0000189	hill	0.0000216
	shore	
	coast	

$$sim_{Lin}(A, B) = \frac{2 \log P(LCS(c_1, c_2))}{\log P(c_1) + \log P(c_2)}$$

$$sim_{Lin}(\text{hill}, \text{coast}) = \frac{2 \log P(\text{geological-formation})}{\log P(\text{hill}) + \log P(\text{coast})}$$

$$\begin{aligned}
 &= \frac{2 \ln 0.00176}{\ln 0.0000189 + \ln 0.0000216} \\
 &= .59
 \end{aligned}$$



The (extended) Lesk Algorithm

- A thesaurus-based measure that looks at **glosses**
- Two concepts are similar if their glosses contain similar words
 - *Drawing paper*: paper that is **specially prepared** for use in drafting
 - *Decal*: the art of transferring designs from **specially prepared paper** to a wood or glass or metal surface
- For each n -word phrase that's in both glosses
 - Add a score of n^2
 - Paper and **specially prepared** for $1 + 2^2 = 5$
 - Compute overlap also for other relations
 - glosses of hypernyms and hyponyms



Summary: thesaurus-based similarity

$$\text{sim}_{\text{path}}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}$$

$$\text{sim}_{\text{resnik}}(c_1, c_2) = -\log P(\text{LCS}(c_1, c_2)) \quad \text{sim}_{\text{lin}}(c_1, c_2) = \frac{2 \log P(\text{LCS}(c_1, c_2))}{\log P(c_1) + \log P(c_2)}$$

$$\text{sim}_{\text{jiangconrath}}(c_1, c_2) = \frac{1}{\log P(c_1) + \log P(c_2) - 2 \log P(\text{LCS}(c_1, c_2))}$$

$$\text{sim}_{e\text{Lesk}}(c_1, c_2) = \sum_{r, q \in \text{RELS}} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2)))$$



Libraries for computing thesaurus-based similarity

- NLTK
 - http://nltk.github.com/api/nltk.corpus.reader.html?highlight=similarity-nltk.corpus.reader.WordNetCorpusReader.res_similarity
- WordNet::Similarity
 - <http://wn-similarity.sourceforge.net/>
 - Web-based interface:
 - <http://marimba.d.umn.edu/cgi-bin/similarity/similarity.cgi>

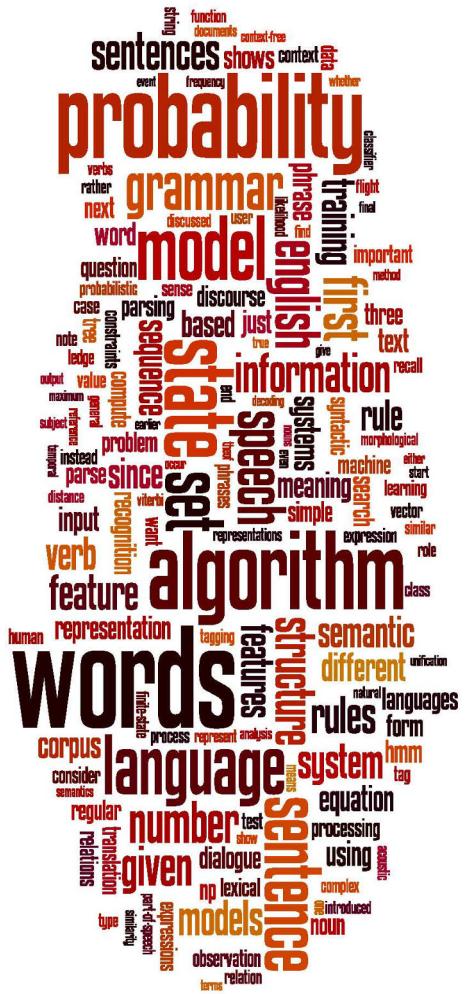


Evaluating similarity

- Intrinsic Evaluation:
 - Correlation between algorithm and human word similarity ratings
- Extrinsic (task-based, end-to-end) Evaluation:
 - Malapropism (spelling error) detection
 - WSD
 - Essay grading
 - Taking TOEFL multiple-choice vocabulary tests

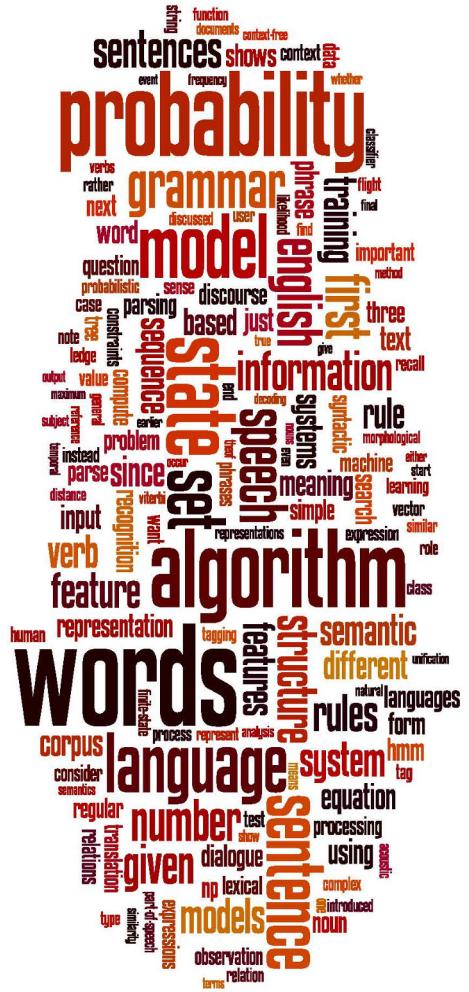
Levied is closest in meaning to:

imposed, believed, requested, correlated



Word Meaning and Similarity

Word Similarity:
Thesaurus Methods



Word Meaning and Similarity

Word Similarity: Distributional Similarity (I)



Problems with thesaurus-based meaning

- We don't have a thesaurus for every language
- Even if we do, they have problems with **recall**
 - Many words are missing
 - Most (if not all) phrases are missing
 - Some connections between senses are missing
 - Thesauri work less well for verbs, adjectives
 - Adjectives and verbs have less structured hyponymy relations



Distributional models of meaning

- Also called vector-space models of meaning
- Offer much higher recall than hand-built thesauri
 - Although they tend to have lower precision
- Zellig Harris (1954): “**oculist** and **eye-doctor** ... occur in almost the same environments....
If A and B have almost identical environments we say that they are synonyms.
- Firth (1957): “You shall know a word by the company it keeps!”



Intuition of distributional word similarity

- Nida example:

A bottle of ***tesgüino*** is on the table
Everybody likes ***tesgüino***
Tesgüino makes you drunk
We make ***tesgüino*** out of corn.

- From context words humans can guess ***tesgüino*** means
 - an alcoholic beverage like **beer**
- Intuition for algorithm:
 - Two words are similar if they have similar word contexts.



Reminder: Term-document matrix

- Each cell: count of term t in a document d : $\text{tf}_{t,d}$:
 - Each document is a **count vector** in \mathbb{N}^v : a column below

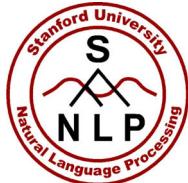
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0



Reminder: Term-document matrix

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0



The words in a term-document matrix

- Each word is a **count vector** in \mathbb{N}^D : a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0



The words in a term-document matrix

- Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0



The Term-Context matrix

- Instead of using entire documents, use smaller contexts
 - Paragraph
 - Window of 10 words
- A word is now defined by a vector over counts of context words



Sample contexts: 20 words (Brown corpus)

- equal amount of sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of clove and nutmeg,
 - on board for their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened to that of
 - of a recursive type well suited to programming on the **digital** computer. In finding the optimal R-stage policy from that of
 - substantially affect commerce, for the purpose of gathering data and **information** necessary for the
- 60 study authorized in the first section of this



Term-context matrix for word similarity

- Two **words** are similar in meaning if their context vectors are similar

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	



Should we use raw counts?

- For the term-document matrix
 - We used **tf-idf** instead of raw term counts
- For the term-context matrix
 - Positive Pointwise Mutual Information (**PPMI**) is common



Pointwise Mutual Information

- **Pointwise mutual information:**
 - Do events x and y co-occur more than if they were independent?
$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$
- **PMI between two words:** (Church & Hanks 1989)
 - Do words x and y co-occur more than if they were independent?
$$\text{PMI}(\textit{word}_1, \textit{word}_2) = \log_2 \frac{P(\textit{word}_1, \textit{word}_2)}{P(\textit{word}_1)P(\textit{word}_2)}$$
- **Positive PMI between two words** (Niwa & Nitta 1994)
 - Replace all PMI values less than 0 with zero



Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{i^*} = \frac{\sum_{j=1}^C f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^W f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i^*} p_{*j}}$$

$$ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$



$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

apricot
pineapple
digital
information

	Count(w,context)				
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p(w=\text{information}, c=\text{data}) = 6/19 = .32$$

$$p(w=\text{information}) = 11/19 = .58$$

$$p(c=\text{data}) = 7/19 = .37$$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N}$$

$$p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

p(w,context)

p(w)

computer data pinch result sugar

apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58

p(context) 0.16 0.37 0.11 0.26 0.11



$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i^*} p_{*j}}$$

		computer	data	pinch	result	sugar	p(w)
	apricot	0.00	0.00	0.05	0.00	0.05	0.11
	pineapple	0.00	0.00	0.05	0.00	0.05	0.11
	digital	0.11	0.05	0.00	0.05	0.00	0.21
	information	0.05	0.32	0.00	0.21	0.00	0.58
	p(context)	0.16	0.37	0.11	0.26	0.11	

- $pmi(\text{information}, \text{data}) = \log_2 (.32 / (.37 * .58)) = .58$

(.57 using full precision)

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-



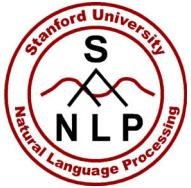
Weighing PMI

- PMI is biased toward infrequent events
- Various weighting schemes help alleviate this
 - See Turney and Pantel (2010)
- Add-one smoothing can also help



	Add-2 Smoothed Count(w,context)				
	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

	p(w,context) [add-2]					p(w)
	computer	data	pinch	result	sugar	
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(context)	0.19	0.25	0.17	0.22	0.17	

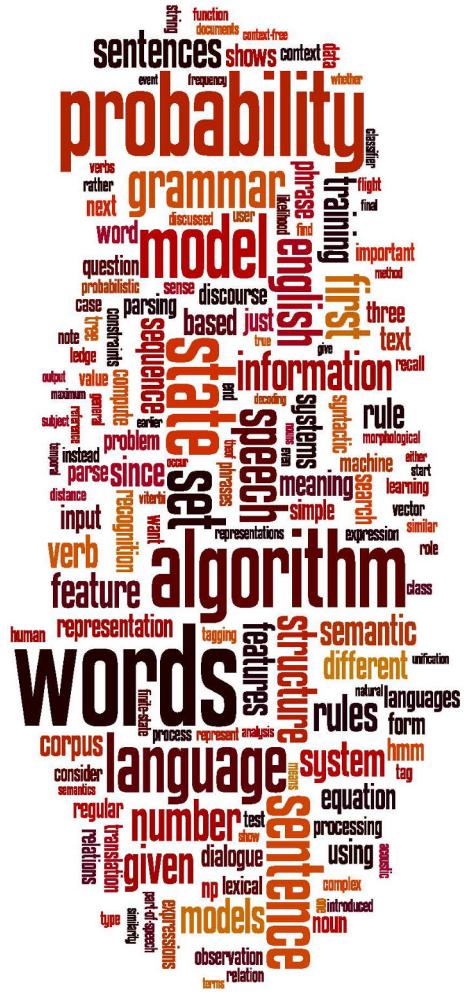


PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

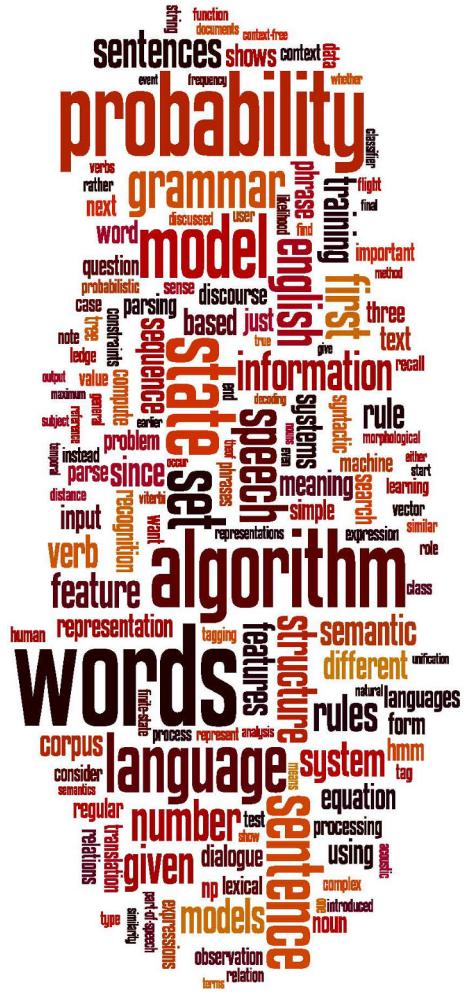
PPMI(w,context) [add-2]

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00



Word Meaning and Similarity

Word Similarity: Distributional Similarity (I)



Word Meaning and Similarity

Word Similarity: Distributional Similarity (II)



Using syntax to define a word's context

- Zellig Harris (1968)
 - “The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities”
- Two words are similar if they have similar parse contexts
- **Duty** and **responsibility** (Chris Callison-Burch’s example)

Modified by adjectives	additional, administrative, assumed, collective, congressional, constitutional ...
Objects of verbs	assert, assign, assume, attend to, avoid, become, breach ...



Co-occurrence vectors based on syntactic dependencies

Dekang Lin, 1998 “Automatic Retrieval and Clustering of Similar Words”

- The contexts C are different dependency relations
 - Subject-of- “absorb”
 - Prepositional-object of “inside”
- Counts for the word cell:

	subj-of, absorb	subj-of, adapt	subj-of, behave	...	pobj-of, inside	pobj-of, into	...	nmod-of, abnormality	nmod-of, anemia	nmod-of, architecture	...	obj-of, attack	obj-of, call	obj-of, come from	obj-of, decorate	...	nmod, bacteria	nmod, body	nmod, bone marrow
cell	1	1	1	...	16	30	...	3	8	1	...	6	11	3	2	...	3	2	2



PMI applied to dependency relations

Hindle, Don. 1990. Noun Classification from Predicate-Argument Structure. ACL

Object of “drink”	Count	PMI
tea	2	11.8
liquid	2	10.5
wine	2	9.3
anything	3	5.2
it	3	1.3

- “Drink it” more common than “drink wine”
- But “wine” is a better “drinkable” thing than “it”



Reminder: cosine for computing similarity

Dot product

Unit vectors

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\vec{v}}{\|\vec{v}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

v_i is the PPMI value for word v in context i

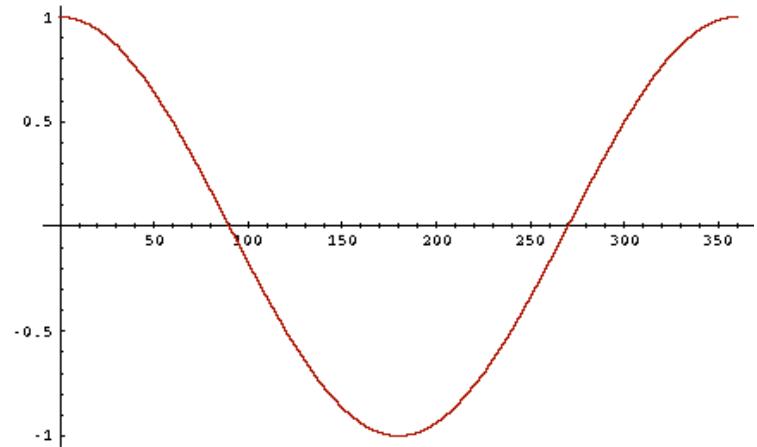
w_i is the PPMI value for word w in context i .

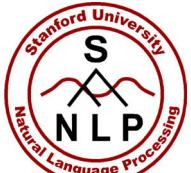
$\text{Cos}(\vec{v}, \vec{w})$ is the cosine similarity of \vec{v} and \vec{w}



Cosine as a similarity metric

- -1: vectors point in opposite directions
 - +1: vectors point in same directions
 - 0: vectors are orthogonal
-
- Raw frequency or PPMI are non-negative, so cosine range 0-1





$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\vec{v}}{\|\vec{v}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Which pair of words is more similar?

$$\text{cosine(apricot,information)} = \frac{\frac{1+0+0}{\sqrt{1+0+0} \sqrt{1+36+1}}}{\sqrt{38}} = \frac{1}{\sqrt{38}} = .16$$

$$\text{cosine(digital,information)} = \frac{\frac{0+6+2}{\sqrt{0+1+4} \sqrt{1+36+1}}}{\sqrt{38} \sqrt{5}} = \frac{8}{\sqrt{38} \sqrt{5}} = .58$$

$$\text{cosine(apricot,digital)} = \frac{\frac{0+0+0}{\sqrt{1+0+0} \sqrt{0+1+4}}}{\sqrt{38}} = 0$$



Other possible similarity measures

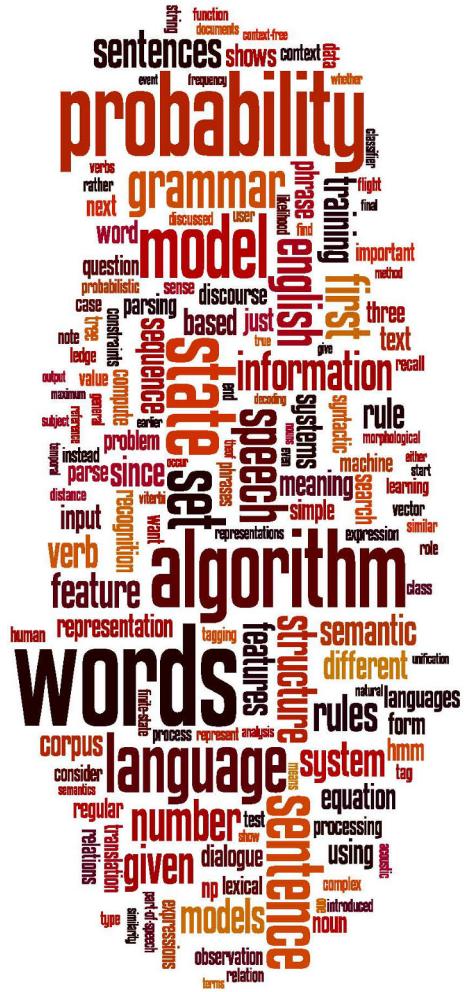
$$\begin{aligned}
 \text{sim}_{\text{cosine}}(\vec{v}, \vec{w}) &= \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i \times w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}} \\
 \text{sim}_{\text{Jaccard}}(\vec{v}, \vec{w}) &= \frac{\sum_{i=1}^N \min(v_i, w_i)}{\sum_{i=1}^N \max(v_i, w_i)} \\
 \text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) &= \frac{2 \times \sum_{i=1}^N \min(v_i, w_i)}{\sum_{i=1}^N (v_i + w_i)} \\
 \text{sim}_{\text{JS}}(\vec{v} || \vec{w}) &= D(\vec{v} \mid \frac{\vec{v} + \vec{w}}{2}) + D(\vec{w} \mid \frac{\vec{v} + \vec{w}}{2})
 \end{aligned}$$



Evaluating similarity (the same as for thesaurus-based)

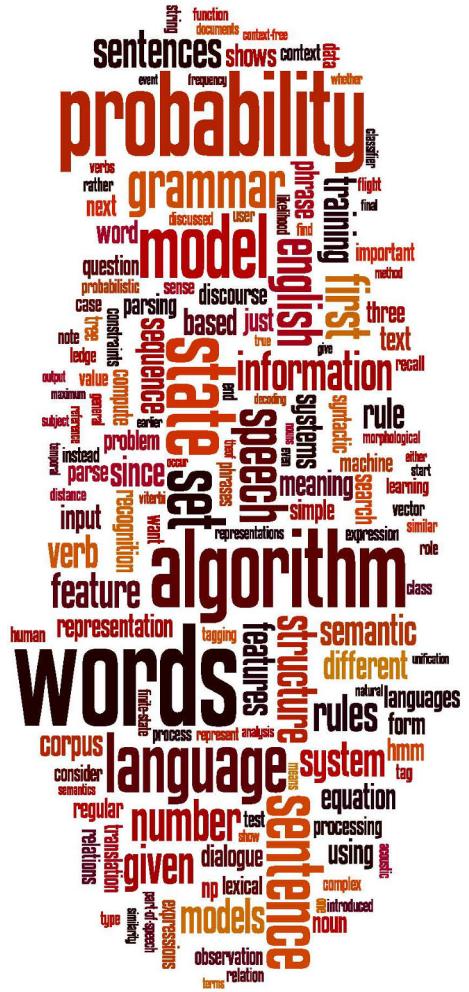
- Intrinsic Evaluation:
 - Correlation between algorithm and human word similarity ratings
- Extrinsic (task-based, end-to-end) Evaluation:
 - Spelling error detection, WSD, essay grading
 - Taking TOEFL multiple-choice vocabulary tests

Levied is closest in meaning to which of these:
imposed, believed, requested, correlated



Word Meaning and Similarity

Word Similarity: Distributional Similarity (II)



Question Answering

What is Question Answering?

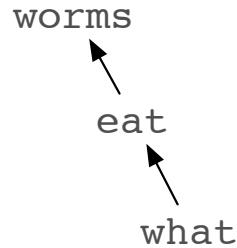


Question Answering

One of the oldest NLP tasks (punched card systems in 1961)

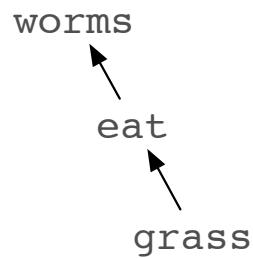
Question:

What do worms eat?

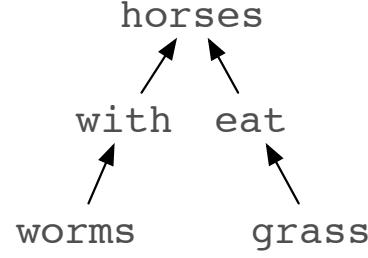


Potential Answers:

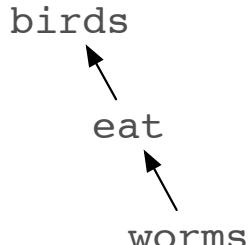
Worms eat grass



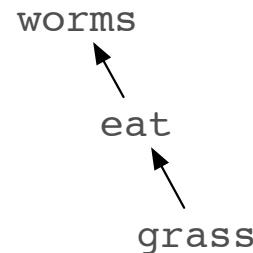
Horses with worms eat grass



Birds eat worms



Grass is eaten by worms



Simmons, Klein, McConlogue. 1964. Indexing and
Dependency Logic for Answering English Questions.
American Documentation 15:30, 196-204



Question Answering: IBM's Watson

- Won Jeopardy on February 16, 2011!

WILLIAM WILKINSON'S
“AN ACCOUNT OF THE PRINCIPALITIES OF
WALLACHIA AND MOLDOVIA”
INSPIRED THIS AUTHOR’S
MOST FAMOUS NOVEL



Bram Stoker



Apple's Siri





WolframAlpha™ computational knowledge engine

how many calories are in two slices of banana cream pie?



[Examples](#) [Random](#)

Assuming any type of pie, banana cream | Use pie, banana cream, prepared from recipe or pie, banana cream, no-bake type, prepared from mix instead

Input interpretation:

pie	amount	2 slices	total calories
	type	banana cream	

Average result:

702 Cal (dietary Calories)

[Show details](#)



Types of Questions in Modern Systems

- Factoid questions
 - *Who wrote “The Universal Declaration of Human Rights”?*
 - *How many calories are there in two slices of apple pie?*
 - *What is the average age of the onset of autism?*
 - *Where is Apple Computer based?*
- Complex (narrative) questions:
 - *In children with an acute febrile illness, what is the efficacy of acetaminophen in reducing fever?*
 - *What do scholars think about Jefferson’s position on dealing with pirates?*



Commercial systems: mainly factoid questions

Where is the Louvre Museum located?	In Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	The yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What is the telephone number for Stanford University?	650-723-2300



Paradigms for QA

- IR-based approaches
 - TREC; IBM Watson; Google
- Knowledge-based and Hybrid approaches
 - IBM Watson; Apple Siri; Wolfram Alpha; True Knowledge Evi



Many questions can already be answered by web search

Google

Search

About 214,000 results (0.38 seconds)

Everything

[Huginn and Muninn - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Huginn_and_Muninn

Images

The **names** of the **ravens** are sometimes modernly anglicized as Hugin and Munin. In the Poetic Edda, a disguised **Odin** expresses that he fears that they may ...

Maps

[Attestations](#) - [Archaeological record](#) - [Theories](#) - [See also](#)

...

Dan Jurafsky



IR-based Question Answering



Search

About 904,000 results (0.30 seconds)

Everything

Best guess for Louvre Museum Location is Paris, France

Images

Mentioned on at least 7 websites including [wikipedia.org](#), [answers.com](#) and [east-buc.k12.ia.us](#) - Show sources - Feedback

Maps

[Musée du Louvre - Wikipedia, the free encyclopedia](#)

Videos

[en.wikipedia.org/wiki/Musée_du_Louvre](#)

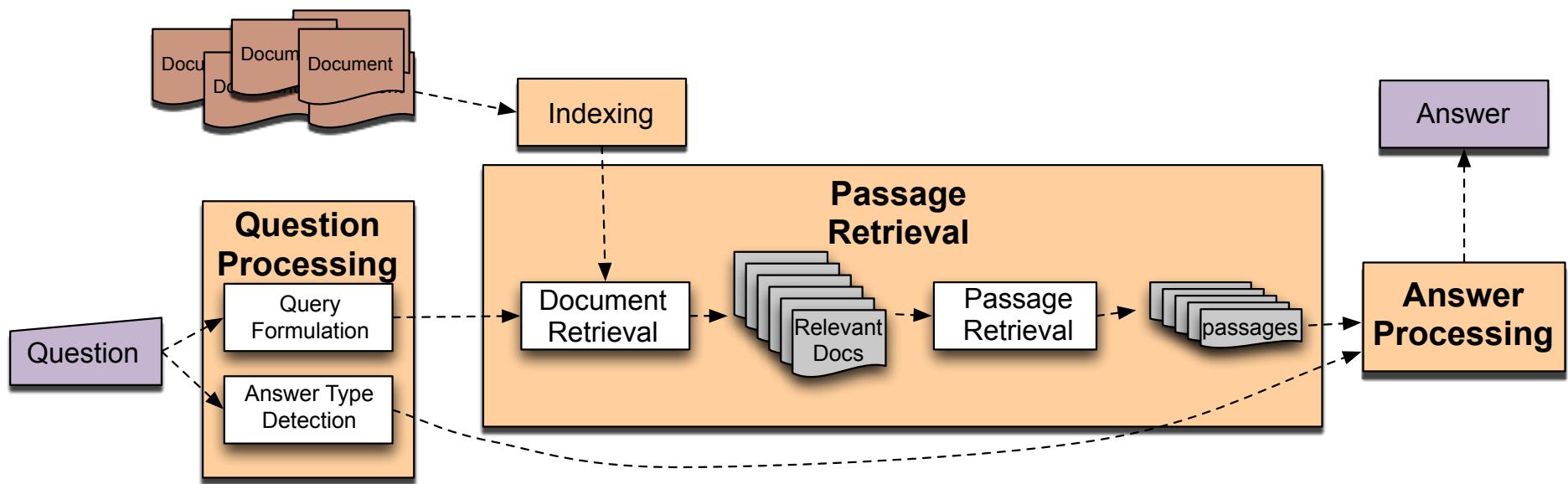
News

Musée du **Louvre** is located in Paris. **Location** within Paris. Established, 1793. **Location**, **Palais Royal**, Musée du **Louvre**, 75001 Paris, France. Type, Art **museum** ...

[Louvre Palace - List of works in the Louvre - Category:Musée du Louvre](#)



IR-based Factoid QA





IR-based Factoid QA

- **QUESTION PROCESSING**
 - Detect question type, answer type, focus, relations
 - Formulate queries to send to a search engine
- **PASSAGE RETRIEVAL**
 - Retrieve ranked documents
 - Break into suitable passages and rerank
- **ANSWER PROCESSING**
 - Extract candidate answers
 - Rank candidates
 - using evidence from the text and external sources



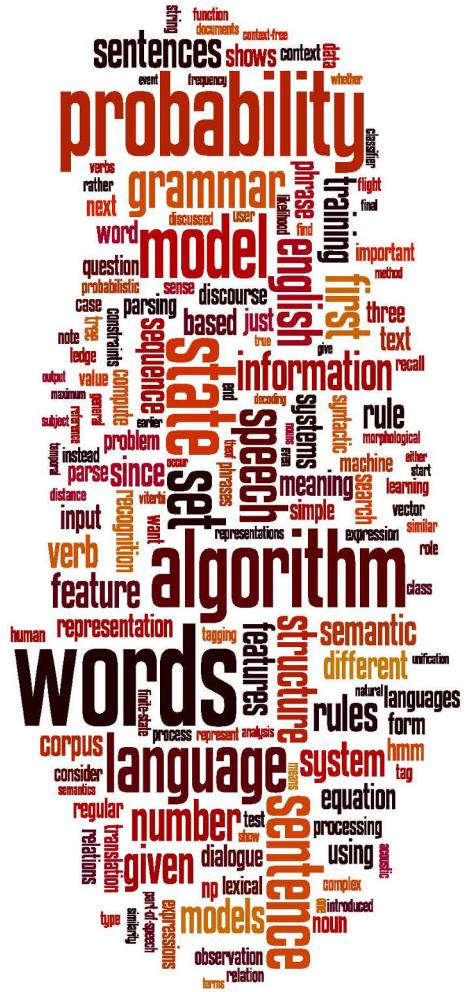
Knowledge-based approaches (Siri)

- Build a semantic representation of the query
 - Times, dates, locations, entities, numeric quantities
- Map from this semantics to query structured data or resources
 - Geospatial databases
 - Ontologies (Wikipedia infoboxes, dbPedia, WordNet, Yago)
 - Restaurant review sources and reservation services
 - Scientific databases



Hybrid approaches (IBM Watson)

- Build a shallow semantic representation of the query
- Generate answer candidates using IR methods
 - Augmented with ontologies and semi-structured data
- Score each candidate using richer knowledge sources
 - Geospatial databases
 - Temporal reasoning
 - Taxonomical classification



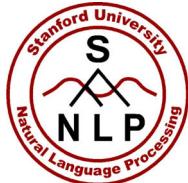
Question Answering

What is Question Answering?

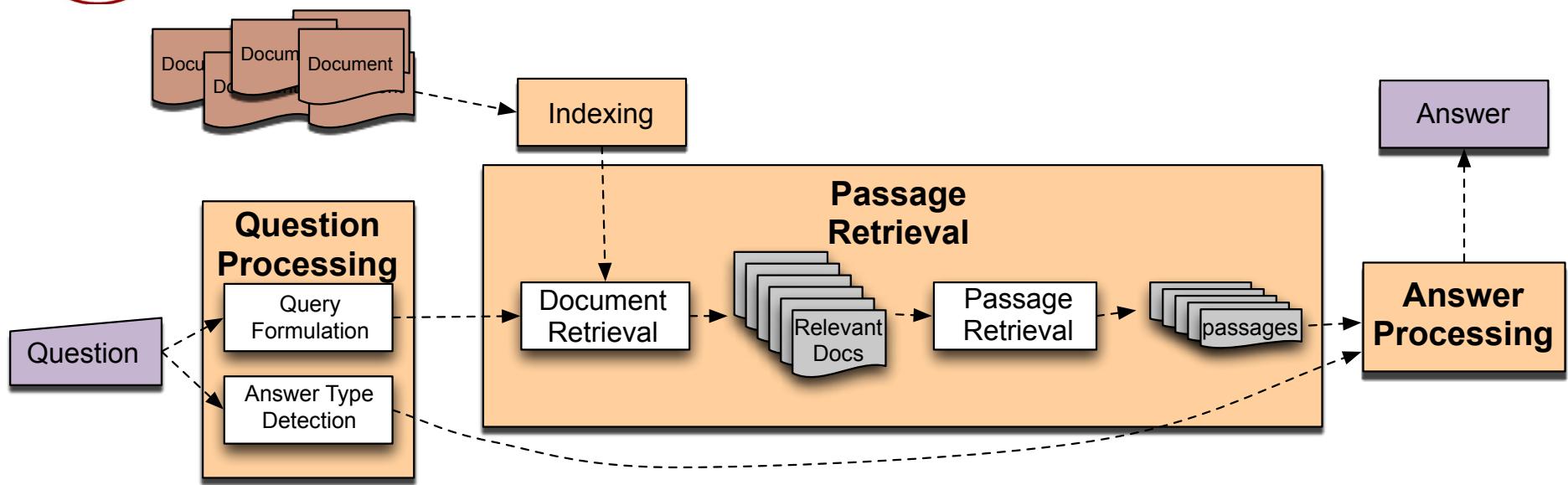


Question Answering

Answer Types and
Query Formulation



Factoid Q/A





Question Processing

Things to extract from the question

- Answer Type Detection
 - Decide the **named entity type** (person, place) of the answer
- Query Formulation
 - Choose **query keywords** for the IR system
- Question Type classification
 - Is this a definition question, a math question, a list question?
- Focus Detection
 - Find the question words that are replaced by the answer
- Relation Extraction
 - Find relations between entities in the question



Question Processing

They're the two states you could be reentering if you're crossing Florida's northern border

- Answer Type: US state
- Query: two states, border, Florida, north
- Focus: the two states
- Relations: borders(Florida, ?x, north)



Answer Type Detection: Named Entities

- *Who founded Virgin Airlines?*
 - PERSON
- *What Canadian city has the largest population?*
 - CITY.



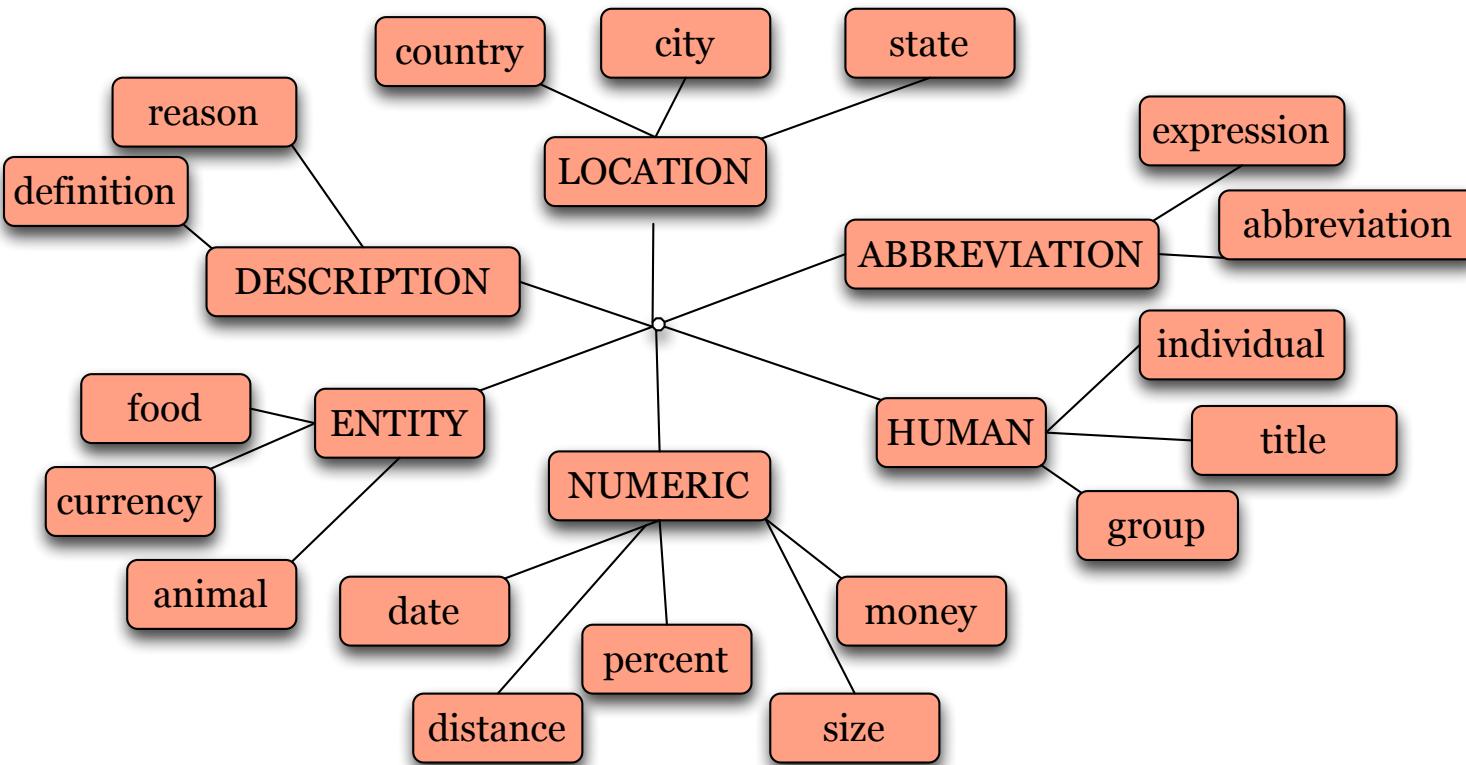
Answer Type Taxonomy

Xin Li, Dan Roth. 2002. Learning Question Classifiers. COLING'02

- 6 coarse classes
 - ABBEVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC
- 50 finer classes
 - LOCATION: city, country, mountain...
 - HUMAN: group, individual, title, description
 - ENTITY: animal, body, color, currency...



Part of Li & Roth's Answer Type Taxonomy





Answer Types

ENTITY

animal	What are the names of Odin's ravens?
body	What part of your body contains the corpus callosum?
color	What colors make up a rainbow ?
creative	In what book can I find the story of Aladdin?
currency	What currency is used in China?
disease/medicine	What does Salk vaccine prevent?
event	What war involved the battle of Chapultepec?
food	What kind of nuts are used in marzipan?
instrument	What instrument does Max Roach play?
lang	What's the official language of Algeria?
letter	What letter appears on the cold-water tap in Spain?
other	What is the name of King Arthur's sword?
plant	What are some fragrant white climbing roses?
product	What is the fastest computer?
religion	What religion has the most members?
sport	What was the name of the ball game played by the Mayans?
substance	What fuel do airplanes use?
symbol	What is the chemical symbol for nitrogen?
technique	What is the best way to remove wallpaper?
term	How do you say " Grandma " in Irish?
vehicle	What was the name of Captain Bligh's ship?
word	What's the singular of dice?



More Answer Types

HUMAN	
description	Who was Confucius?
group	What are the major companies that are part of Dow Jones?
ind	Who was the first Russian astronaut to do a spacewalk?
title	What was Queen Victoria's title regarding India?
LOCATION	
city	What's the oldest capital city in the Americas?
country	What country borders the most others?
mountain	What is the highest peak in Africa?
other	What river runs through Liverpool?
state	What states do not have state income tax?
NUMERIC	
code	What is the telephone number for the University of Colorado?
count	About how many soldiers died in World War II?
date	What is the date of Boxing Day?
distance	How long was Mao's 1930s Long March?
money	How much did a McDonald's hamburger cost in 1963?
order	Where does Shanghai rank among world cities in population?
other	What is the population of Mexico?
period	What was the average life expectancy during the Stone Age?
percent	What fraction of a beaver's life is spent swimming?
speed	What is the speed of the Mississippi River?
temp	How fast must a spacecraft travel to escape Earth's gravity?
size	What is the size of Argentina?
weight	How many pounds are there in a stone?



Answer types in Jeopardy

Ferrucci et al. 2010. Building Watson: An Overview of the DeepQA Project. AI Magazine. Fall 2010. 59-79.

- 2500 answer types in 20,000 Jeopardy question sample
- The most frequent 200 answer types cover < 50% of data
- The 40 most frequent Jeopardy answer types

he, country, city, man, film, state, she, author, group, here, company, president, capital, star, novel, character, woman, river, island, king, song, part, series, sport, singer, actor, play, team, show, actress, animal, presidential, composer, musical, nation, book, title, leader, game

Dan Jurafsky



Answer Type Detection

- Hand-written rules
- Machine Learning
- Hybrids



Answer Type Detection

- Regular expression-based rules can get some cases:
 - Who {is|was|are|were} PERSON
 - PERSON (YEAR – YEAR)
- Other rules use the **question headword**:
(the headword of the first noun phrase after the wh-word)
 - Which **city** in China has the largest number of foreign financial companies?
 - What is the state **flower** of California?



Answer Type Detection

- Most often, we treat the problem as machine learning classification
 - **Define** a taxonomy of question types
 - **Annotate** training data for each question type
 - **Train** classifiers for each question class using a rich set of features.
 - features include those hand-written rules!

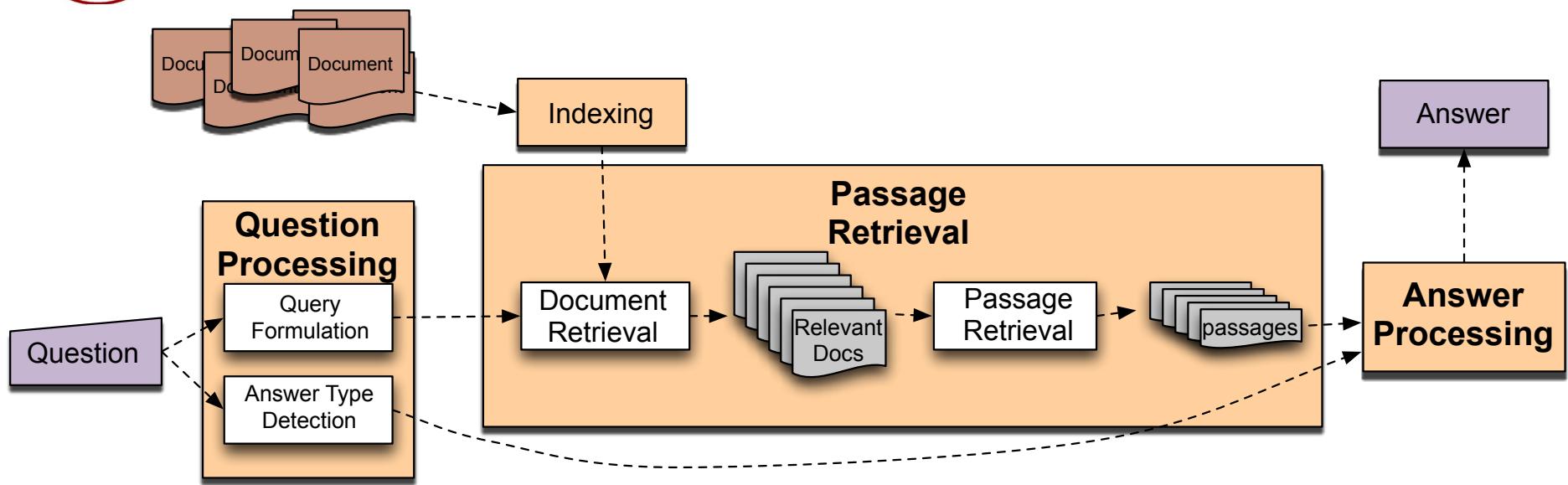


Features for Answer Type Detection

- Question words and phrases
- Part-of-speech tags
- Parse features (headwords)
- Named Entities
- Semantically related words



Factoid Q/A





Keyword Selection Algorithm

Dan Moldovan, Sanda Harabagiu, Marius Paca, Rada Mihalcea, Richard Goodrum, Roxana Girju and Vasile Rus. 1999. Proceedings of TREC-8.

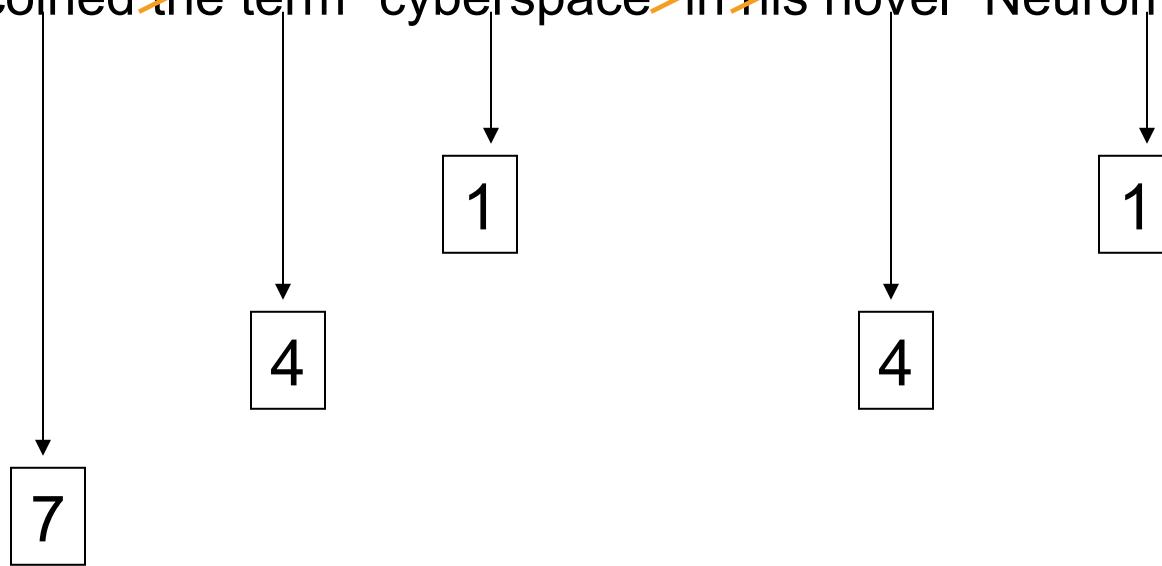
1. Select all non-stop words in quotations
2. Select all NNP words in recognized named entities
3. Select all complex nominals with their adjectival modifiers
4. Select all other complex nominals
5. Select all nouns with their adjectival modifiers
6. Select all other nouns
7. Select all verbs
8. Select all adverbs
9. Select the QFW word (skipped in all previous steps)
10. Select all other words



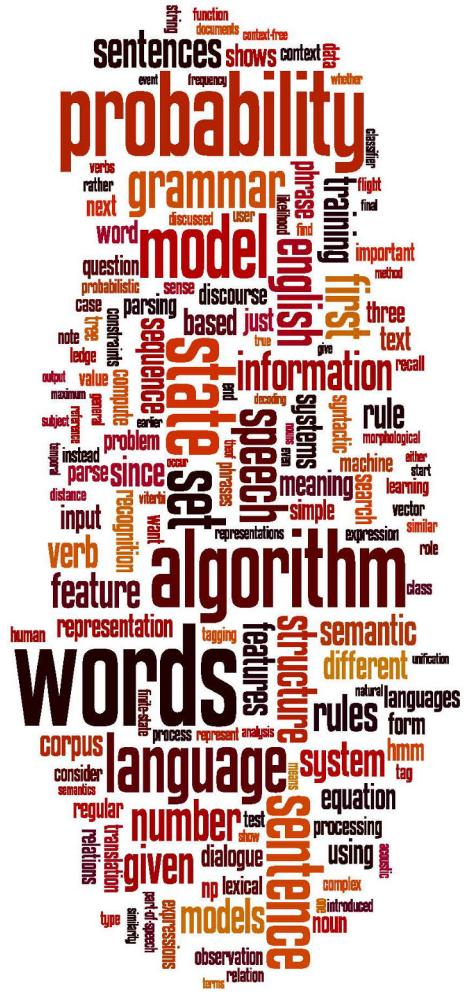
Choosing keywords from the query

Slide from Mihai Surdeanu

~~Who coined the term “cyberspace” in his novel “Neuromancer”?~~

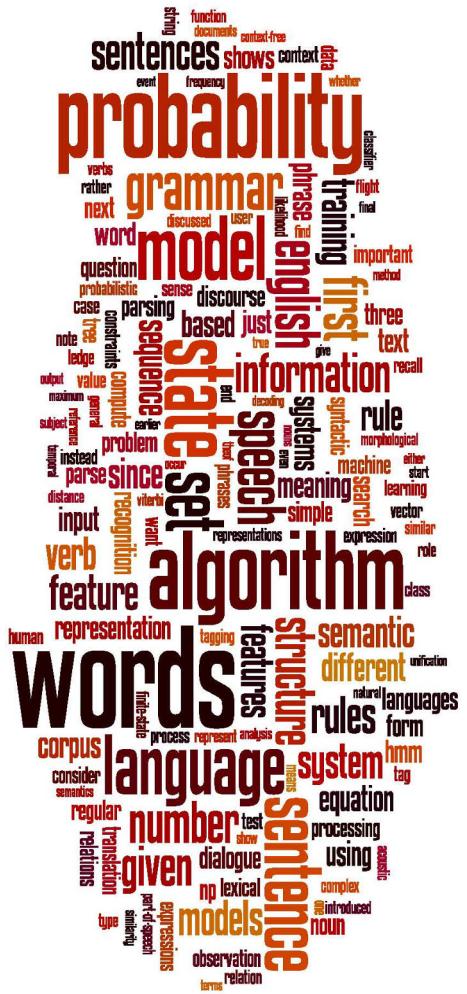


cyberspace/1 Neuromancer/1 term/4 novel/4 coined/7



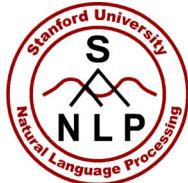
Question Answering

Answer Types and Query Formulation

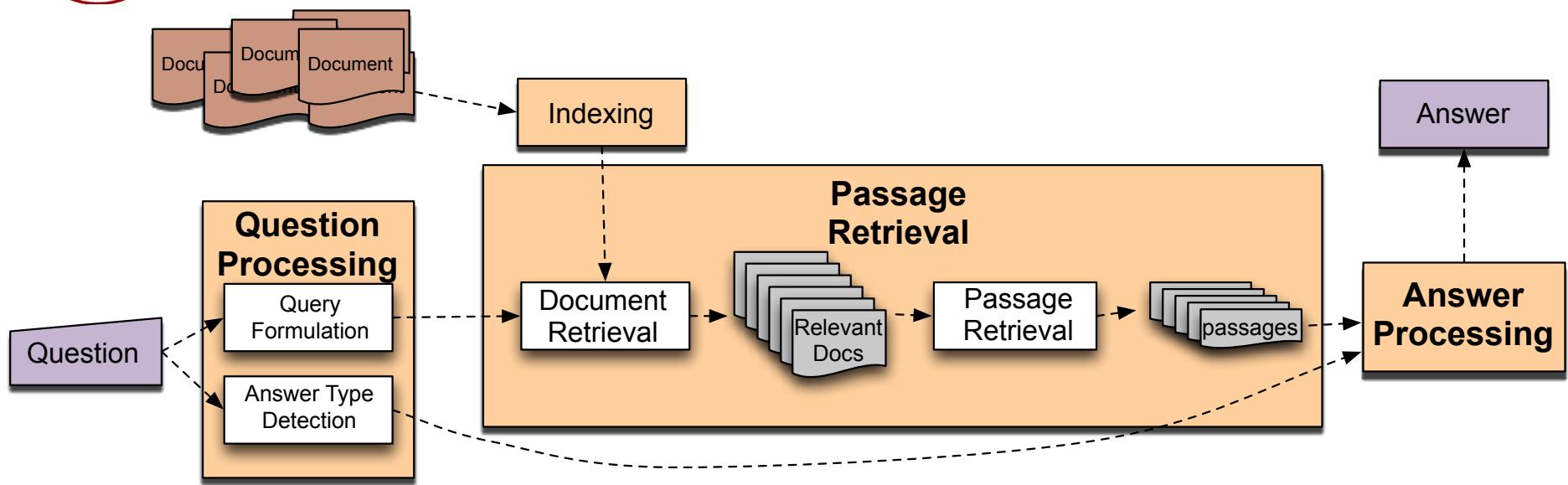


Question Answering

Passage Retrieval and
Answer Extraction



Factoid Q/A





Passage Retrieval

- Step 1: IR engine retrieves documents using query terms
- Step 2: Segment the documents into shorter units
 - something like paragraphs
- Step 3: Passage ranking
 - Use answer type to help rerank passages



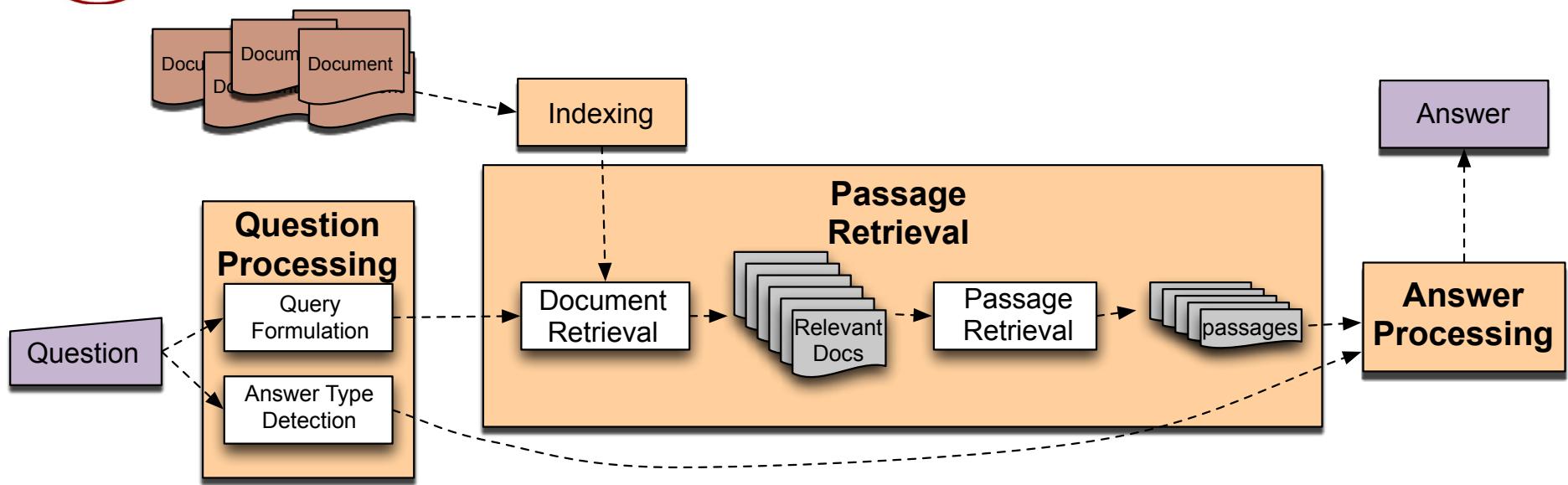
Features for Passage Ranking

Either in rule-based classifiers or with supervised machine learning

- Number of Named Entities of the right type in passage
- Number of query words in passage
- Number of question N-grams also in passage
- Proximity of query keywords to each other in passage
- Longest sequence of question words
- Rank of the document containing passage



Factoid Q/A





Answer Extraction

- Run an answer-type named-entity tagger on the passages
 - Each answer type requires a named-entity tagger that detects it
 - If answer type is CITY, tagger has to tag CITY
 - Can be full NER, simple regular expressions, or hybrid
- Return the string with the right type:
 - Who is the prime minister of India (**PERSON**)
Manmohan Singh, Prime Minister of India, had told left leaders that the deal would not be renegotiated.
 - How tall is Mt. Everest? (**LENGTH**)
The official height of Mount Everest is **29035 feet**



Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: Person
- Passage:

The Marie biscuit is named after Marie Alexandrovna, the daughter of Czar Alexander II of Russia and wife of Alfred, the second son of Queen Victoria and Prince Albert



Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: Person
- Passage:

The Marie biscuit is named after **Marie Alexandrovna**,
the daughter of **Czar Alexander II of Russia** and wife of
Alfred, the second son of **Queen Victoria** and **Prince
Albert**



Use machine learning: Features for ranking candidate answers

Answer type match: Candidate contains a phrase with the correct answer type.

Pattern match: Regular expression pattern matches the candidate.

Question keywords: # of question keywords in the candidate.

Keyword distance: Distance in words between the candidate and query keywords

Novelty factor: A word in the candidate is not in the query.

Apposition features: The candidate is an appositive to question terms

Punctuation location: The candidate is immediately followed by a comma, period, quotation marks, semicolon, or exclamation mark.

Sequences of question terms: The length of the longest sequence of question terms that occurs in the candidate answer.



Candidate Answer scoring in IBM Watson

- Each candidate answer gets scores from >50 components
 - (from unstructured text, semi-structured text, triple stores)
 - logical form (parse) match between question and candidate
 - passage source reliability
 - geospatial location
 - California is “southwest of Montana”
 - temporal relationships
 - taxonomic classification

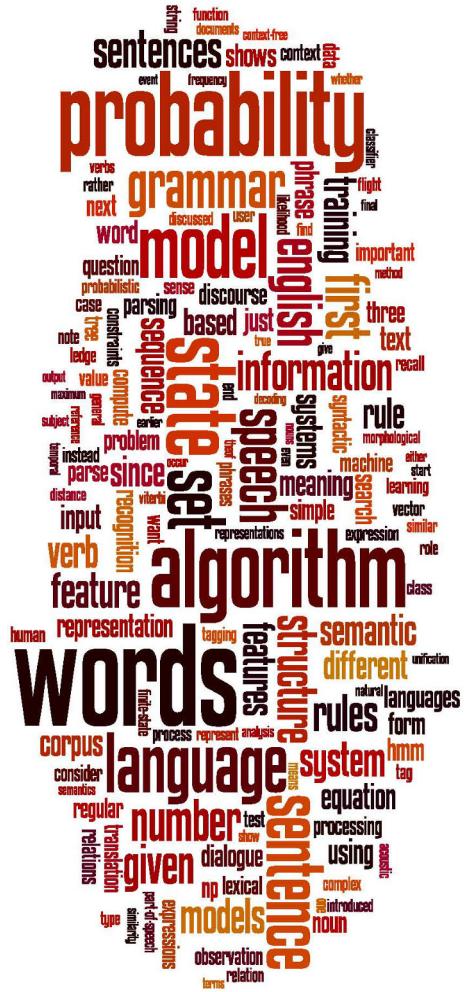


Common Evaluation Metrics

1. *Accuracy* (does answer match gold-labeled answer?)
2. *Mean Reciprocal Rank*

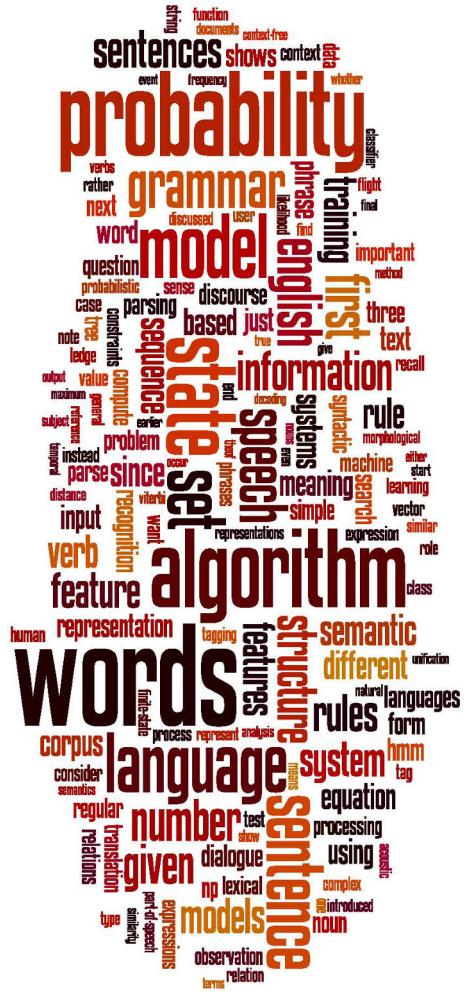
- For each query return a ranked list of M candidate answers.
- Query score is 1/Rank of the first correct answer
 - *If first answer is correct: 1*
 - *else if second answer is correct: ½*
 - *else if third answer is correct: ⅓, etc.*
 - *Score is 0 if none of the M answers are correct*
- Take the mean over all N queries

$$MRR = \frac{\sum_{i=1}^N \frac{1}{rank_i}}{N}$$



Question Answering

Passage Retrieval and Answer Extraction



Question Answering

Using Knowledge in QA



Relation Extraction

- Answers: Databases of Relations
 - born-in("Emma Goldman", "June 27 1869")
 - author-of("Cao Xue Qin", "Dream of the Red Chamber")
 - Draw from Wikipedia infoboxes, DBpedia, FreeBase, etc.
- Questions: Extracting Relations in Questions

Whose granddaughter starred in E.T.?

(acted-in ?x "E.T.")

(granddaughter-of ?x ?y)



Temporal Reasoning

- Relation databases
 - (and obituaries, biographical dictionaries, etc.)
- IBM Watson

"In 1594 he took a job as a tax collector in Andalusia"

Candidates:

- Thoreau is a bad answer (born in 1817)
- Cervantes is possible (was alive in 1594)



Geospatial knowledge (containment, directionality, borders)

- Beijing is a good answer for “Asian city”
- California is “southwest of Montana”
- geonames.org:

www.geonames.org/search.html?q=palo+alto&country=

[GeoNames Home](#) | [Postal Codes](#) | [Download](#) / [Webservice](#) | [About](#)

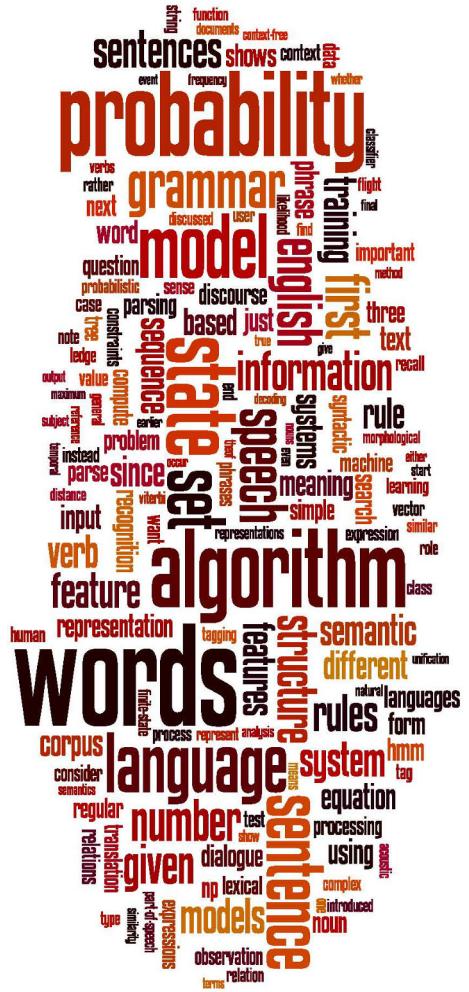
[login](#)

	<input type="text" value="palo alto"/>	<input type="button" value="all countries"/>	<input type="button" value="search"/>	<input type="button" value="show on map"/>	[advanced search]
459 records found for "palo alto"					
	Name	Country	Feature class	Latitude	Longitude
1	Palo Alto	United States , California Santa Clara County	populated place population 64,403, elevation 9m	N 37° 26' 30"	W 122° 8' 34"
2	Palo Alto Township	United States , Iowa Jasper County	administrative division elevation 256m	N 41° 38' 15"	W 93° 2' 57"
3	Borough of Palo Alto	United States , Pennsylvania Schuylkill County	administrative division population 1,032, elevation 210m	N 40° 41' 21"	W 76° 10' 2"



Context and Conversation in Virtual Assistants like Siri

- Coreference helps resolve ambiguities
 - U: “Book a table at Il Fornaio at 7:00 with **my mom**”
 - U: “Also send **her** an email reminder”
- Clarification questions:
 - U: “Chicago pizza”
 - S: “Did you mean pizza restaurants in Chicago or Chicago-style pizza?”



Question Answering

Using Knowledge in QA