

Spelling Correction: Edit Distance

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 1

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
-

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?
- Need a **distance metric**
- The simplest metric: **edit distance**

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
 -) Insertion
 -) Deletion
 -) Substitution

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

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

Minimum Edit Distance

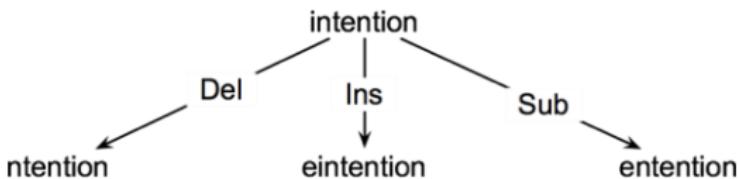
I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has a cost of 1 (Levenshtein)
 -) Distance between these is 5
- If substitution costs 2 (alternate version)
 -) Distance between these is 8

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits



Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them
- Keep track of the shortest path to each state

Defining Minimum Edit Distance Matrix

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

Thus, the edit distance between X and Y is $D(n,m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
 -) Compute $D(i,j)$ for small i,j
 -) Compute larger $D(i,j)$ based on previously computed smaller values
 -) Compute $D(i,j)$ for all i and j till you get to $D(n,m)$

Dynamic Programming Algorithm

Initialization

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

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

Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

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

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	

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

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

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 -) We often need to align characters 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

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	

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

The Edit Distance Table

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

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

Minimum Edit with Backtrace

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	

Adding Backtrace to Minimum Edit

Base conditions:

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

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

Termination:

$D(N, M)$ is distance

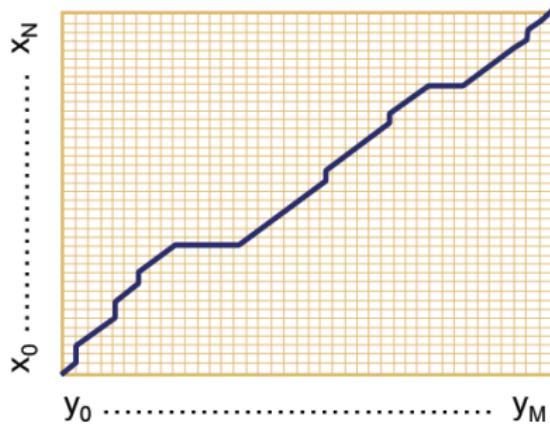
Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

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

The distance matrix



Every non-decreasing path from $(0,0)$ to (M,N) corresponds to an alignment of two sequences.

An optimal alignment is composed of optimal sub-alignments.

Result of Backtrace

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

$O(n+m)$

Weighted Edit Distance, Other variations

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 2

Weighted Edit Distance

Why to add weights to the computation?

- Some letters are more likely to be mistyped.

Confusion Matrix for Spelling Errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)
Y (correct)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
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	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
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

Keyboard Design



Weighted Minimum 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

How to modify the algorithm with transpose?

Transpose

- $\text{transpose}(x,y) = (y,x)$
- Also known as metathesis

Modification to the dynamic programmic algorithm

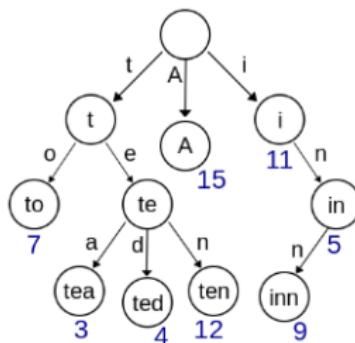
$$D[i][j] = \min \begin{cases} D[i-1, j] + 1 & (\text{deletion}) \\ D[i, j-1] + 1 & (\text{insertion}) \\ 1 & \text{if } (x[i] \neq y[j]) \text{ (substitution)} \\ 0 & \text{otherwise} \\ D(i-2, j-2) + 1 & (x[i] = y[j-1] \text{ and } x[i-1] = y[j]) \\ & (\text{transposition}) \end{cases}$$

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit ditance from the query term to each dictionary term - an exhaustive search

Can be made efficient if we do it over a trie structure



How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance ≤ 2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for
- For Chinese alphabet size is 70,000 (Unicode Han Characters)

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Number of deletes within edit distance ≤ 2 for a word of length 9 will be 45

A further check is required to remove the false positives

Spelling Correction

Types of spelling errors: Non-word Errors

- behaf → behalf

Types of spelling errors: Real-word Errors

- **Typographical errors:** three → there
- **Cognitive errors (homophones):** piece → peace, too → two

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 the error word
- Choose the best one:
 -) 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

Choosing best candidate

- Noisy Channel

Noisy Channel Model for Spelling Correction

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 3

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\begin{aligned}\hat{w} &= \arg \max_{w \in V} P(w|x) \\ &= \arg \max_{w \in V} \frac{P(x|w)P(w)}{P(x)} \\ &= \arg \max_{w \in V} P(x|w)P(w)\end{aligned}$$

Non-word spelling error: across

Words with similar spelling

Small edit distance to error

Words with similar pronunciation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:

Insertion, Deletion, Substitution,

Transposition of two adjacent letters

Words within edit distance 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

Allow deletion of space or hyphen

- thisidea → this idea
- inlaw → in-law

Computing error probability: confusion matrix

- $\text{del}[x,y]$: count (xy typed as x)
- $\text{ins}[x,y]$: count (x typed as xy)
- $\text{sub}[x,y]$: count (x typed as y)
- $\text{trans}[x,y]$: count(xy typed as yx)

Insertion and deletion are conditioned on previous character

Channel model

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

Channel model for across

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

Using a bigram language model

- “... versatile across whose ...”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress} \mid \text{versatile}) = 0.000021, P(\text{across} \mid \text{versatile}) = 0.000021$
- $P(\text{whose} \mid \text{actress}) = 0.0010, P(\text{whose} \mid \text{across}) = 0.000006$
- $P(\text{"versatile actress whose"}) = 0.000021 * 0.0010 = 210 \times 10^{-10}$
- $P(\text{"versatile across whose"}) = 0.000021 * 0.000006 = 1 \times 10^{-10}$

Real-word spelling errors

- The study was conducted mainly **be** John Black
- The design **an** construction of the system ...

25-40% of spelling errors are real words

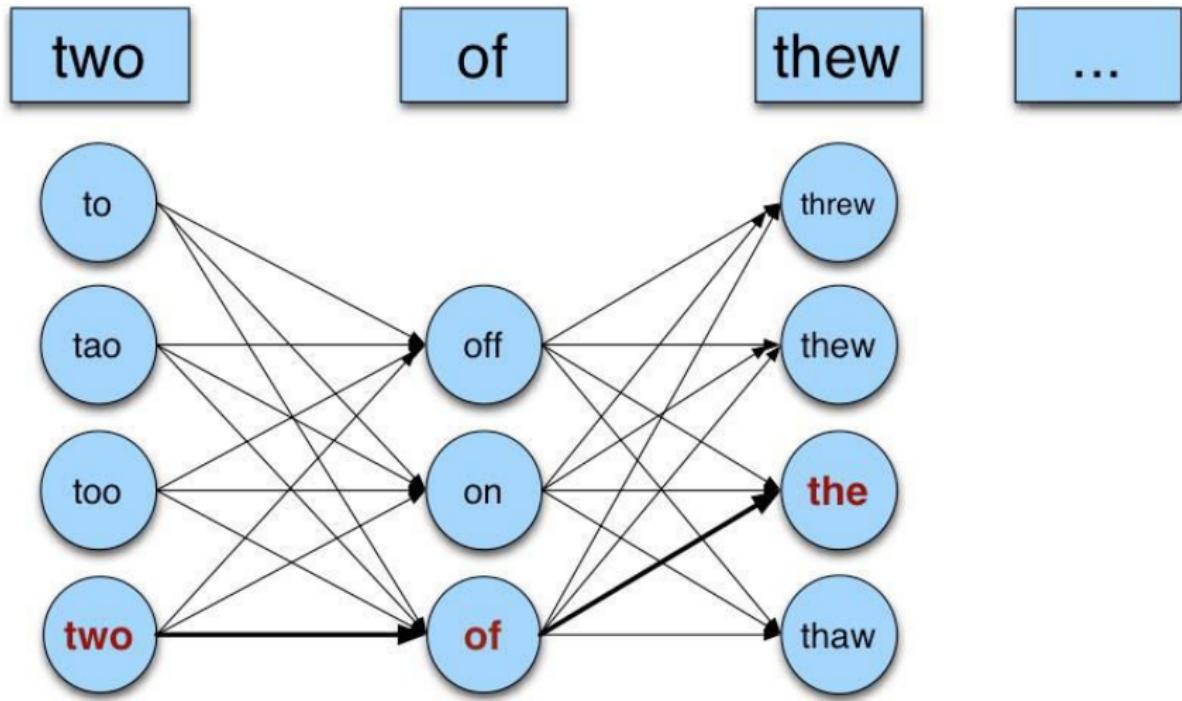
Noisy channel for real-word spell correction

Given a sentence $X = w_1, w_2, w_3 \dots, w_n$

- 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_3) = { $w_3, w'_3, w''_3, w'''_3, \dots$ }

Choose the sequence W that maximizes $P(W|X)$

Noisy channel for real-world spell correction



Simplification: One error per sentence

Choose among all possible sentences with one word replaced

two of thew

- w_1, w''_2, w_3 two off thew
- w_1, w_2, w'_3 two of the
- w'''_1, w_2, w_3 too of thew

Choose the sequence W that maximizes $P(W|X)$

Getting the probability values

Noisy Channel

$$\hat{W} = \arg \max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg \max_{W \in S} P(X|W)P(W)$$

$P(X|W)$

- Same as for non-word spelling correction
- Also require probability for no error $P(w|w)$

Probability of no error

What is the probability for a correctly typed word? $P(\text{"the"} | \text{"the"})$

It may depend on the source text under consideration

- 1 error in 10 words → 0.9
- 1 error in 100 words → 0.99

Computing $P(W)$

Use Language Model

- Unigram
- Bigram
- ...

N-gram Language Models

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 4

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

min·u·et  noun \min-yə-'wet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Use a Language Model

$P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Machine Translation

Which sentence is more plausible in the target language?

- $P(\text{high winds}) > P(\text{large winds})$

Other Applications

- Context Sensitive Spelling Correction
- Natural Language Generation
- ...

Completion Prediction

- Language model also supports predicting the completion of a sentence.
 -) Please turn off your cell ...
 -) Your program does not ...
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

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

- **Related Task:** probability of an upcoming word:

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

- A model that computes either of these is called a **language model**

Computing $P(W)$

How to compute the joint probability

$P(\text{about, fifteen, minutes, from})$

Basic Idea

Rely on the Chain Rule of Probability

The Chain Rule

Conditional Probabilities

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

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

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

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

Probability of words in sentences

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

P("about fifteen minutes from") =

$P(\text{about}) \times P(\text{fifteen} | \text{about}) \times P(\text{minutes} | \text{about fifteen}) \times P(\text{from} | \text{about fifteen minutes})$

Estimating These Probability Values

Count and divide

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\text{Count (about fifteen minutes from office)}}{\text{Count (about fifteen minutes from)}}$$

What is the problem

We may never see enough data for estimating these

Markov Assumption

Simplifying Assumption: Use only the previous word

$$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{from})$$

Or the couple previous words

$$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{minutes from})$$

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

We approximate each component in the product

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

N-Gram Models

P(office | about fifteen minutes from)

An N -gram model uses only $N - 1$ words of prior context.

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} \mid \text{from})$
- Trigram: $P(\text{office} \mid \text{minutes from})$

Markov model and Language Model

An N -gram model is an $N - 1$ -order Markov Model

N-Gram Models

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

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the “most probable”

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

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

An Example

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

< s > I am here < /s >

< s > who am I < /s >

< s > I would like to know < /s >

Estimating bigrams

$$P(I | < s >) = 2/3$$

$$P(< /s > | \text{here}) = 1$$

$$P(\text{would} | I) = 1/3$$

$$P(\text{here} | \text{am}) = 1/2$$

$$P(\text{know} | \text{like}) = 0$$

Bigram counts from 9222 Restaurant 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

Computing bigram probabilities

Normalize by unigrams

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

Bigram Probabilities

	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

Computing Sentence Probabilities

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

$$\begin{aligned} &= P(I | < s >) \times P(\text{want} | I) \times P(\text{english} | \text{want}) \times P(\text{food} | \text{english}) \times P(</ s > | \text{food}) \\ &= 0.000031 \end{aligned}$$

What knowledge does n-gram represent?

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

Practical Issues

Everything in log space

- Avoids underflow
- Adding is faster than multiplying

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

Handling zeros

Use smoothing

Language Modeling Toolkit

SRILM

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

Google N-grams

Number of tokens: 1,024,908,267,229

Number of sentences: 95,119,665,584

Number of unigrams: 13,588,391

Number of bigrams: 314,843,401

Number of trigrams: 977,069,902

Number of fourgrams: 1,313,818,354

Number of fivegrams: 1,176,470,663

[http://googleresearch.blogspot.in/2006/08/
all-our-n-gram-are-belong-to-you.html](http://googleresearch.blogspot.in/2006/08/all-our-n-gram-are-belong-to-you.html)

Example from the 4-gram data

serve as the inspector 66

serve as the inspiration 1390

serve as the installation 136

serve as the institute 187

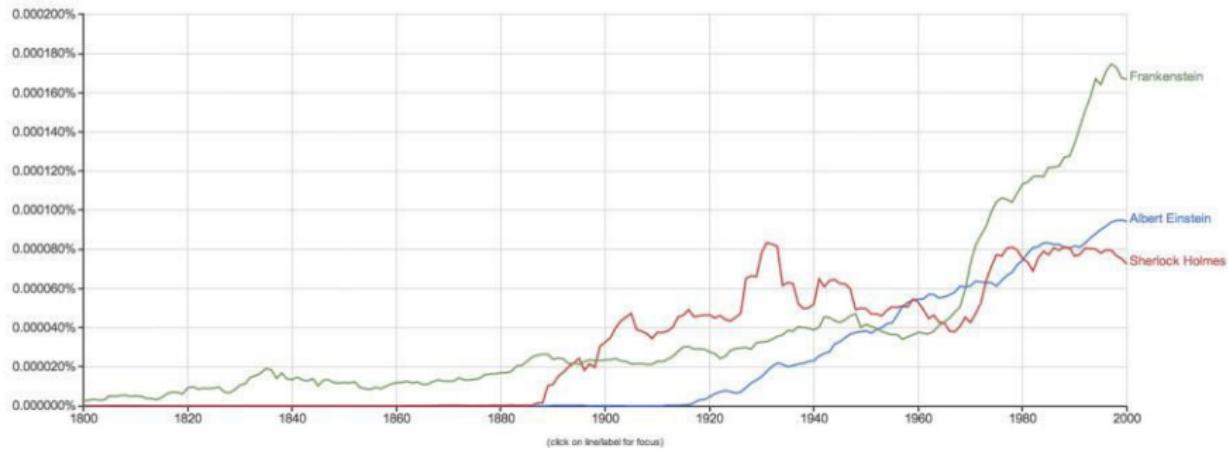
serve as the institution 279

serve as the institutional 461

Google books Ngram Data

Google books Ngram Viewer

Graph these comma-separated phrases: case-insensitive
between 1800 and 2000 from the corpus English with smoothing of 3



Evaluation of Language Models, Basic Smoothing

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 5

Evaluating Language Model

Does it prefer good sentences to bad sentences?

Assign higher probability to real (or frequently observed) sentences than ungrammatical (or rarely observed) ones

Training and Test Corpora

- Parameters of the model are trained on a large corpus of text, called **training set**.
- Performance is tested on a disjoint (held-out) **test data** using an **evaluation metric**

Extrinsic evaluation of N-grams models

Comparison of two models, A and B

- Use each model for one or more tasks: *spelling corrector, speech recognizer, machine translation*
- Get accuracy values for A and B
- Compare accuracy for A and B

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and ...
- The president of India is ...
- I wrote a ...

Unigram model doesn't work for this game.

A better model of text

is one which assigns a higher probability to the actual word

Perplexity

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

Perplexity ($PP(W)$)

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

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Applying chain Rule

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

For bigrams

$$PP(W) = \prod \frac{1}{P(w_i | w_{i-1})}^{\frac{1}{N}}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

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

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Unigram perplexity: 962?

The model is as confused on test data as if it had to choose uniformly and independently among 962 possibilities for each word.

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram $(\langle s \rangle, w)$ as per its probability
- Choose a random bigram (w, x) as per its probability
- And so on until we choose $\langle /s \rangle$

```
<s> I
I want
want to
to eat
eat Chinese
Chinese food
food </s>
I want to eat Chinese food
```

Shakespeare as Corpus

- $N = 884,647$ tokens, $V = 29,066$
- Shakespeare produced 300,000 bigram types out of $V^2 = 844$ million possible bigrams.

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.

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- $P(\text{offer} \mid \text{denied the}) = 0$
- The test set will be assigned a probability 0
- And the perplexity can't be computed

Language Modeling: Smoothing

With sparse statistics

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

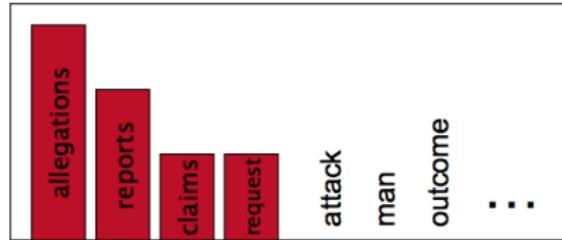
3 allegations

2 reports

1 claims

1 request

7 total



Steal probability mass to generalize better

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

2.5 allegations

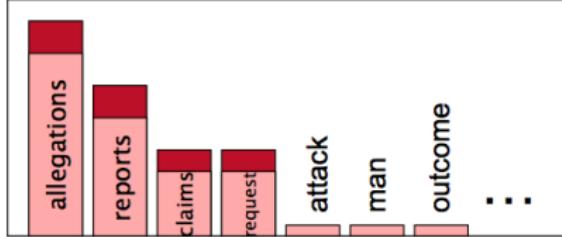
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time than we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)+1}{c(w_{i-1})+V}$

Reconstituted counts as effect of smoothing

Effective bigram count ($c^*(w_{n-1}w_n)$)

$$\frac{c^*(w_{n-1}w_n)}{c(w_{n-1})} = \frac{c(\underline{w_{n-1}}\underline{w_n}) + 1}{c(w_{n-1}) + V}$$

Comparing with bigrams: Restaurant corpus

	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

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:

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

A good value of k or m ?

Can be optimized on held-out set

Language Modelling: Advanced Smoothing Models

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 1

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Good-Turing: Basic Intuition

Use the count of things we have seen once

- to help estimate the count of things we have never seen

N_c : Frequency of frequency c

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

Computing N_c

I	3	
am	2	
here	1	$N_1 = 4$
who	1	$N_2 = 1$
would	1	$N_3 = 1$
like	1	

Good Turing Estimation

Idea

- Reallocate the probability mass of n -grams that occur $r + 1$ times in the training data to the n -grams that occur r times
- In particular, reallocate the probability mass of n -grams that were seen once to the n -grams that were never seen

Adjusted count

For each count c , an adjusted count c^* is computed as:

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

where N_c is the number of n -grams seen exactly c times

Good Turing Estimation

Good Turing Smoothing

$$P_{GT}^*(\text{things with frequency } c) = \frac{c^*}{N}$$

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

What if $c = 0$

$P_{GT}^*(\text{things with frequency } c) = \frac{N_1}{N}$ where N denotes the total number of bigrams that actually occur in training

Complications

What about words with high frequency?

- For small c , $N_c > N_{c+1}$
- For large c , too jumpy

Simple Good-Turing

Replace empirical N_k with a best-fit power law once counts get unreliable

Good-Turing numbers: Example

22 million words of AP Neswire

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

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

Good-Turing numbers: Example

22 million words of AP Neswire

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

It looks like $c^* = c - 0.75$

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

Why don't we just subtract 0.75 (or some d)?

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

We may keep some more values of d for counts 1 and 2

But can we do better than using the regular unigram correct?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- “Francisco” more common than “glasses”
- But “Francisco” mostly follows “San”

$P(w)$: “How likely is w ?”

Instead, $P_{continuation}(w)$: “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

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

Kneser-Ney Smoothing

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

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

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

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

Kneser-Ney Smoothing

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

λ is a normalizing constant

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

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

A general approach is to combine the results of multiple N-gram models.

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Interpolation

mix unigram, bigram, trigram

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probability $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

$P_{bo}(w_i|w_{i-2}w_{i-1}) =$

- $\hat{P}(w_i|w_{i-2}w_{i-1})$, if $c(w_{i-2}w_{i-1}w_i) > 0$
- $\lambda(w_{i-1}w_{i-2})P_{bo}(w_i|w_{i-1})$, otherwise

where $P_{bo}(w_i|w_{i-1}) =$

- $\hat{P}(w_i|w_{i-1})$ if $c(w_{i-1}w_i) > 0$
- $\lambda(w_{n-1})\hat{P}(w_n)$, otherwise

Example Problem

In a corpus, suppose there are 4 words, a , b , c , and d . You are provided with the following counts.

n-gram	count	n-gram	count	n-gram	count
aba	4	ba	5	a	8
abb	0	bb	3	b	9
abc	0	bc	0	c	8
abd	0	bd	0	d	7

Use the recursive definition of backoff smoothing to obtain the probability distribution, $P_{backoff}(w_n | w_{n-2}w_{n-1})$, where $w_{n-1} = b$ and $w_{n-2} = a$. Also assume that $\hat{P}(x) = P(x) - 1/8$.

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Lambdas conditional on context

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

Setting the lambda values

Use a held-out corpus

Choose λ to maximize the probability of held-out data:

- Find the N-gram probabilities on the training data
- Search for λ s that give the largest probability to held-out data

Computational Morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 2

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

dogs

- 2 morphemes, ‘dog’ and ‘s’
- ‘s’ is a plural marker on nouns

unladylike

3 morphemes

- un- ‘not’
- lady ‘well-behaved woman’
- -like ‘having the characteristic of’

Allomorphs

Variants of the same morpheme, but cannot be replaced by one another

Example

- opposite: un-happy, in-comprehensible, im-possible, ir-rational

Bound and Free Morphemes

Bound

Cannot appear as a word by itself.

-s (dog-s), -ly (quick-ly), -ed (walk-ed)

Free

Can appear as a word by itself; often can combine with other morphemes too.

house (house-s), walk (walk-ed), of, the, or

Stems and Affixes

Stems and Affixes

- Stems (roots): The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

Mostly, stems are free morphemes and affixes are bound morphemes

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
talk-ing, quick-ly
- Infix: 'n' in 'vindati' (he knows), as contrasted with *vid* (to know).
Philippines: basa 'read' → b-um-asa 'read'
English: abso-bloody-lutely (emphasis)
- Circumfixes - precedes and follows the stem
Dutch: berg 'mountain', ge-berg-te 'mountains'

Content and functional morphemes

Content morphemes

Carry some semantic content

car, -able, un-

Functional morphemes

Provide grammatical information

-s (plural), -s (3rd singular)

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Derivational morphology

Creates new words by changing part-of-speech: *logic, logical, illogical, illogicality, logician*

Fairly systematic but some derivations missing: *sincere - sincerity, scarce - scarcity, curious - curiosity, fierce - fiercity?*

Morphological processes

Concatenation

Adding continuous affixes - the most common process:

- hope+less, un+happy, anti+capital+ist+s

Often, there are phonological/graphemic changes on morpheme boundaries:

- book + s [s], shoe + s [z]
- happy +er → happier

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: ‘go’ (look), ‘go-go’ (examine with attention)
- Tagalog: ‘basa’ (read), ‘ba-basa’(will read)
- Sanskrit: ‘pac’ (cook), ‘papa⁻ca’ (perfect form, cooked)
- Phrasal reduplication (Telugu): *pillava⁻du nad[.] ustū⁻ nad[.] ustū⁻ pad[.]i po⁻ya⁻du*
(The child fell down while walking)

Morphological processes

Suppletion

‘irregular’ relation between the words
go - went, good - better

Morpheme internal changes

The word changes internally
sing - sang - sung, man - men, goose - geese

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj → Adj: bitter-sweet
- N + N → N: rain-bow
- V + N → V: pick-pocket
- P + V → V: over-do

Particular to languages

room-temperature: Hindi translation?

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Clipping

Longer words are shortened

doctor, laboratory, advertisement, dormitory, examination, bicycle, refrigerator

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see, saw}\}$
- Morphological analysis : word → setOf(lemma +tag)
 $\text{saw} \rightarrow \{ \langle\text{see, verb.past}\rangle, \langle\text{saw, noun.sg}\rangle \}$
- Tagging: word → tag, considers context
 $\text{Peter saw her} \rightarrow \{ \langle\text{see, verb.past}\rangle \}$
- Morpheme segmentation: de-nation-al-iz-ation
- Generation: see + verb.past → saw

What are the applications?

- Text-to-speech synthesis:
lead: verb or noun?
read: present or past?
- Search and information retrieval
- Machine translation, grammar correction

Morphological Analysis

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Goal

To take input forms like those in the first column and produce output forms like those in the second column.

Output contains stem and additional information; +N for noun, +SG for singular, +PL for plural, +V for verb etc.

Issues involved

boy → boys

fly → flies → flies (y → i rule)

Toiling → toil

Duckling → duckl?

- Getter → get + er
- Doer → do + er
- Beer → be + er?

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not duckl.

We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?

Ex: plural morpheme follows the noun

Only some endings go on some words

- Do+er: ok
- Be+er: not so

Spelling change rules

Adjust the surface form using spelling change rules

- Get + er → getter

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1
- Sanskrit: 11 million forms from a lexicon of 170,000 entries, a ratio of 64.7:1
- New forms can be created, compounding etc.

One of the most common methods is finite-state-machines

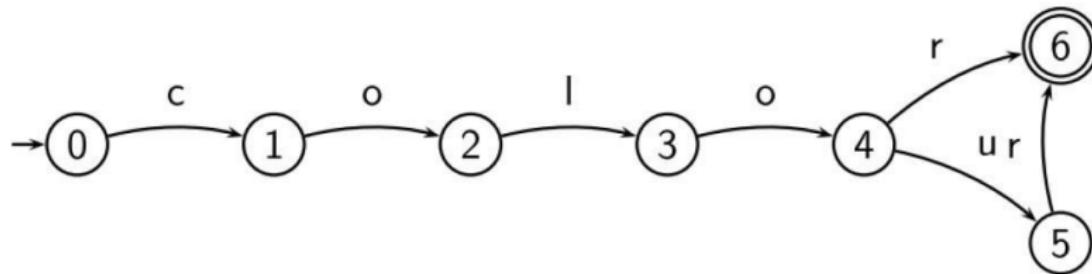
Finite-state methods for morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 3

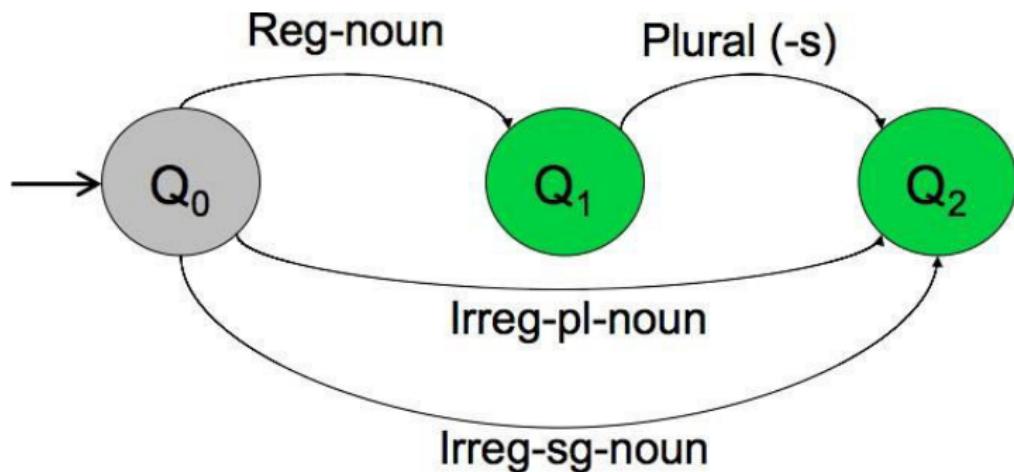
Finite State Automaton (FSA)



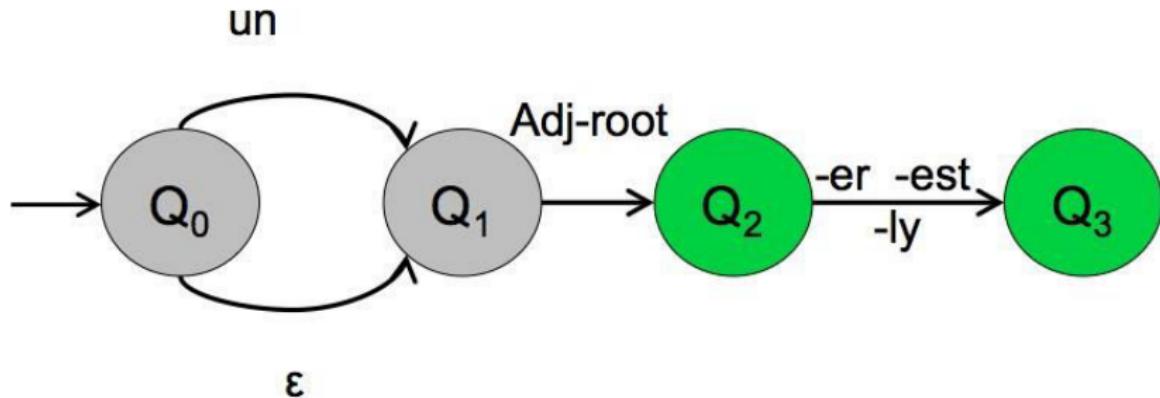
What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty sets)
- Start state and accepting states
- Recognizes regular languages, i.e., languages specified by regular expressions

FSA for nominal inflection in English



FSA for English Adjectives



Word modeled

happy, happier, happiest, real, unreal, cool, coolly, clear, clearly, unclear, unclearly, ...

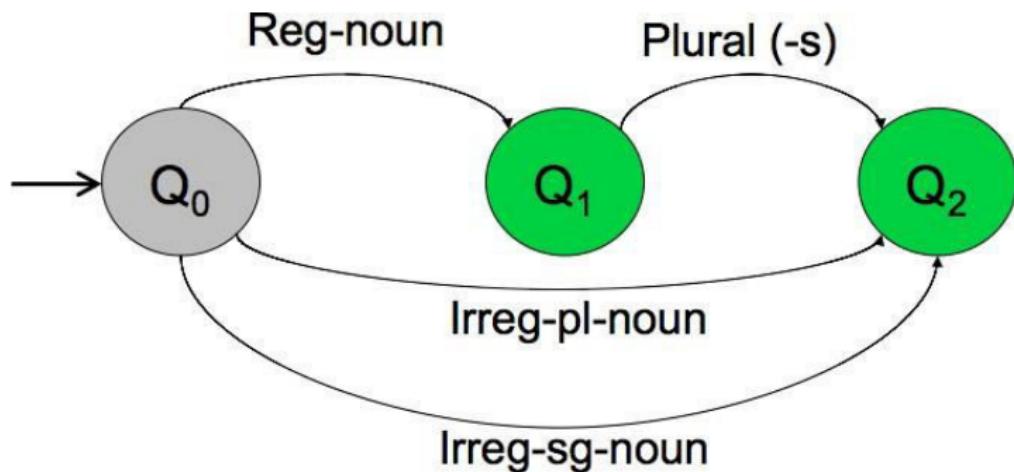
Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

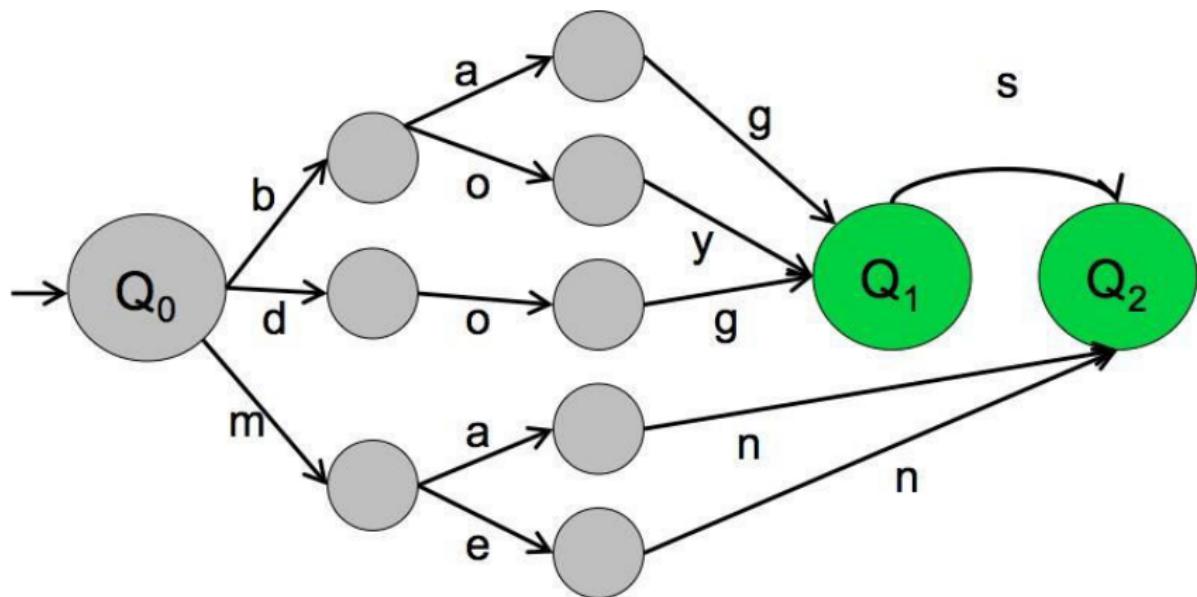
Lexicon

Can we include the lexicon in the FSA?

FSA for nominal inflection in English



After adding a mini-lexicon



Some properties of FSAs: Elegance

- Recognizing problem can be solved in linear time (independent of the size of the automaton)
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states
- An FSA is deterministic iff it has no empty (ϵ) transition and for each state and each symbol, there is at most one applicable transition
- Every non-deterministic automaton can be transformed into a deterministic one

But...

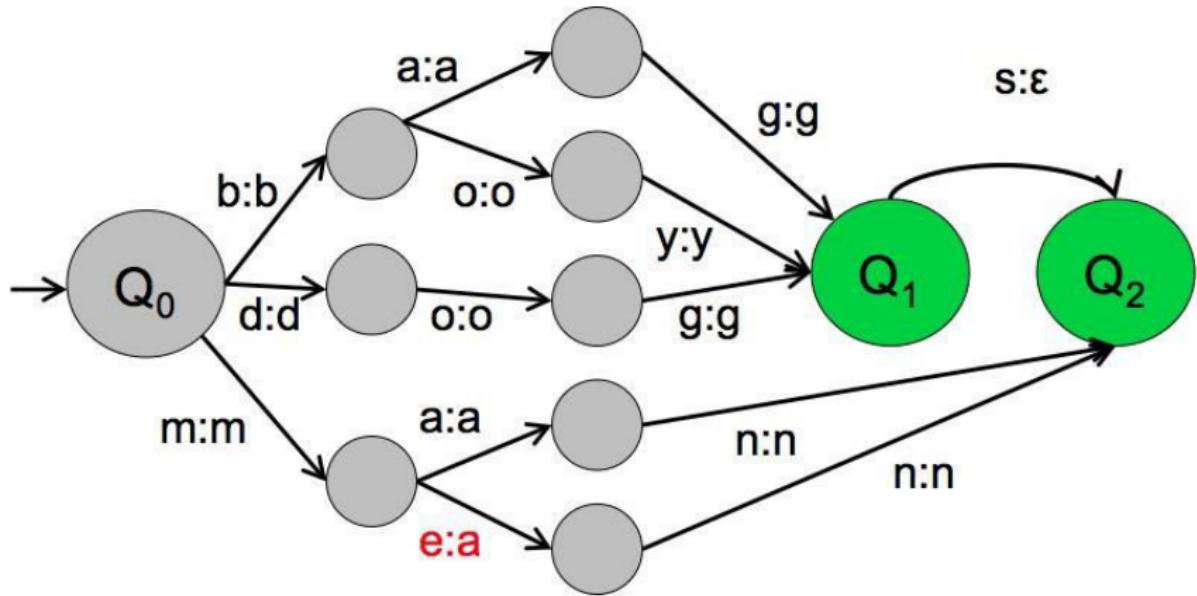
FSA are language recognizers/generators.

We need transducers to build Morphological Analyzers

Finite State Transducers

- Translate strings from one language to strings in another language
- Like FSA, but each edge is associated with two strings

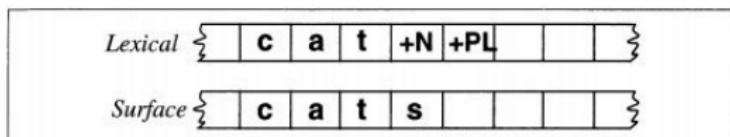
An example FST



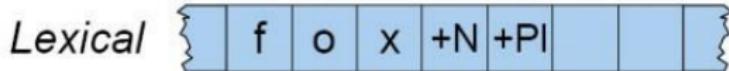
Two-level morphology

Given the input *cats*, we would like to output *cat+N+PL*, telling us that cat is a plural noun.

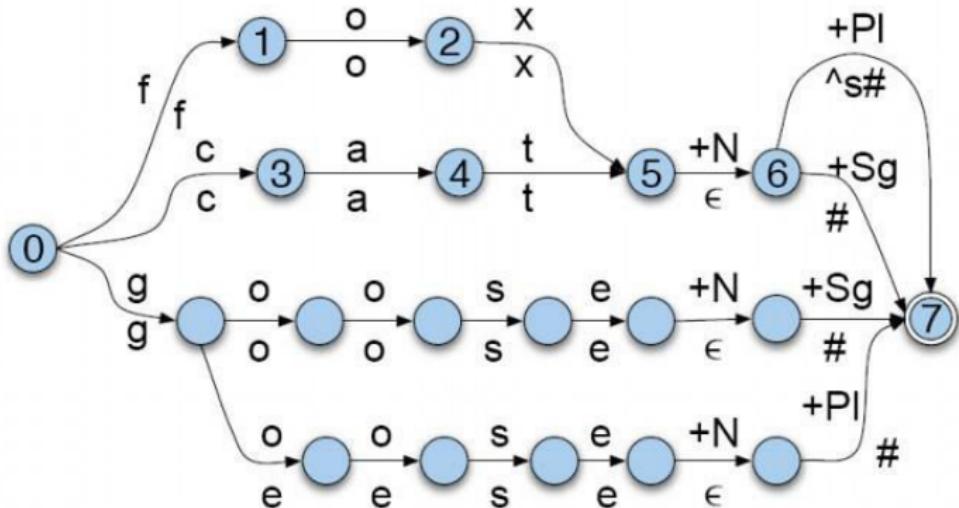
We do this via a version of **two-level morphology**, a correspondence between a lexical level (morphemes and features) to a surface level (actual spelling).



Intermediate tape for Spelling change rules

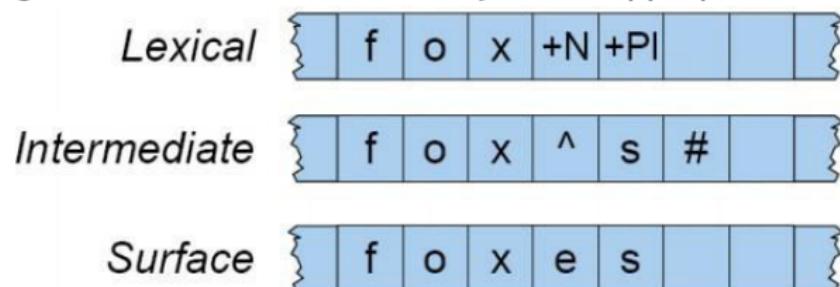


English Nominal Inflection FST



Spelling Handling

A spelling change rule would insert an e only in the appropriate environment.



Rule Handling

Rule Notation

$a \rightarrow b/c_d$: “rewrite a as b when it occurs between c and d .”

Morphological Analysis: Approaches

Two different ways to address phonological/graphemic variations

- Linguistic approach: A phonological component accompanying the simple concatenative process of attaching an ending
- Engineering approach: Phonological changes and irregularities are factored into endings and a higher number of paradigms

Different Approaches: Example from Czech

	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pář-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kř-e	pář-e	mouš-e
:						
P2	žen-0	sov-0	skic-0	ker-0	par-0	much-0

A linguistic approach

$$\begin{array}{llllll} \text{žen} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{sov} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{skic} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{kr} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{pár} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{mouch} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} \end{array}$$

An engineering approach

$$\begin{array}{llllll} \text{žen} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{sov} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{skic} + \begin{cases} \text{a} \\ \text{i} \\ \text{e} \\ 0 \end{cases} & \text{k} + \begin{cases} \text{ra} \\ \text{ry} \\ \check{\text{e}}\text{r} \\ \text{er} \end{cases} & \text{p} + \begin{cases} \text{ára} \\ \text{áry} \\ \check{\text{a}}\text{r}\text{e} \\ \text{ar} \end{cases} & \text{m} + \begin{cases} \text{oucha} \\ \text{ouchy} \\ \text{ouše} \\ \text{uch} \end{cases} \end{array}$$

Tools Available

- AT&T FSM Library and Lextools

<http://www2.research.att.com/~fsmttools/fsm/>

- OpenFST (Google and NYU)

<http://www.openfst.org/>

Introduction to POS Tagging

Pawan Goyal

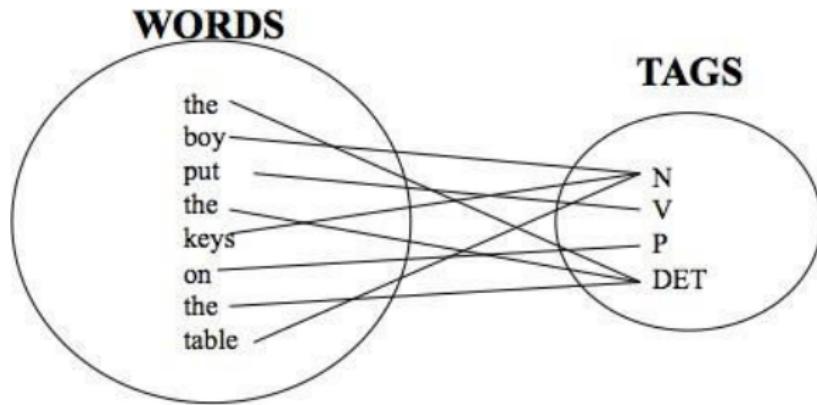
CSE, IITKGP

Week 3: Lecture 4

Part-of-Speech (POS) tagging

Task

Given a text of English, identify the parts of speech of each word



Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Closed class words

- pronouns, determiners, prepositions, connectives, ...
- there is a limited number of these
- *mostly functional*: to tie the concepts of a sentence together

POS examples

- N noun chair, bandwidth, pacing
- V verb study, debate, munch
- ADJ adj purple, tall, ridiculous
- ADV adverb unfortunately, slowly,
- P preposition of, by, to
- PRO pronoun I, me, mine
- DET determiner the, a, that, those

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
N, V, Adj, Adv
- More commonly used set is finer grained, “UPenn TreeBank tagset”, 45 tags

A Nice Tutorial on POS tags

<https://sites.google.com/site/partofspeechhelp/>

UPenn TreeBank POS tag set

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	+%, &
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	\$
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	#
PDT	Predeterminer	<i>all, both</i>	"	Left quote	(‘ or “)
POS	Possessive ending	<i>'s</i>	"	Right quote	(‘ or ”)
PRP	Personal pronoun	<i>I, you, he</i>	(Left parenthesis	([, (, {, <)
PRP\$	Possessive pronoun	<i>your, one's</i>)	Right parenthesis	(],), }, >)
RB	Adverb	<i>quickly, never</i>	,	Comma	,
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	(: ; ... --)
RP	Particle	<i>up, off</i>			

Using the UPenn tagset

Example Sentence

The grand jury commented on a number of other topics.

POS tagged sentence

The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill: *back/VB*

POS tagging problem

To determine the POS tag for a particular instance of a word

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous
- Breakdown of ambiguous word types:

Unambiguous (1 tag)	35,340
Ambiguous (2–7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 ("still")

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/_ to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/_ the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/_ 2500/CD.

Deciding the correct POS

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 2500/CD.

Relevant knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Local context

- Two determiners rarely follow each other
- Two base form verbs rarely follow each other
- Determiner is almost always followed by adjective or noun

POS tagging: Two approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

Statistical tagging

- Get a training corpus of tagged text, learn the transformation rules from the most frequent tags (TBL tagger)
- Probabilistic: Find the most likely sequence of tags T for a sequence of words W

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

Add transformation rules to reduce training mistakes

- MD →NN: DT_
- VBD→VBN: VBD_

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:** words are observed and tags are hidden.
- **Text Classification:** sentences/documents are observed and the category is hidden.
Categories can be positive/negative for sentiments ..
sports/politics/business for documents ...

What gives rise to the two families?

Whether they generate the observed data from hidden stuff or the hidden structure given the data?

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$

e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

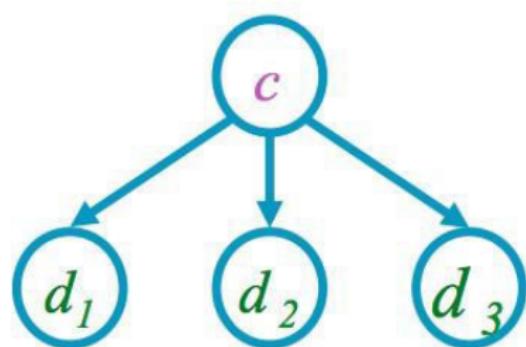
Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$

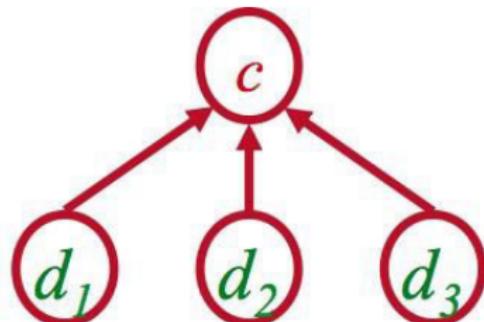
e.g. Logistic regression, maximum entropy models, conditional random fields

SVMs, perceptron, etc. are discriminative classifiers but not directly probabilistic

Generative vs. Discriminative Models



Naive Bayes



Logistic Regression

Joint vs. conditional likelihood

- A *joint* model gives probabilities $P(d, c)$ and tries to maximize this joint likelihood.
- A *conditional* model gives probabilities $P(c|d)$, taking the data as given and modeling only the conditional probability of the class.

Hidden Markov Models for POS Tagging

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 5

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)} \\ &= \operatorname{argmax}_T P(W|T)P(T) \\ &= \operatorname{argmax}_T \underset{i}{\cdot} P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i)P(t_i|t_1 \dots t_{i-1})\end{aligned}$$

Further simplifications

$$\hat{T} = \underset{i}{\operatorname{argmax}}_T P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
 $P(t_i | t_1 \dots t_{i-1}) \approx P(t_i | t_{i-1})$
- Using these simplifications:

$$\hat{T} = \underset{i}{\operatorname{argmax}}_T P(w_i | t_i) P(t_i | t_{i-1})$$

Computing the probability values

Tag Transition probabilities $p(t_i|t_{i-1})$

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

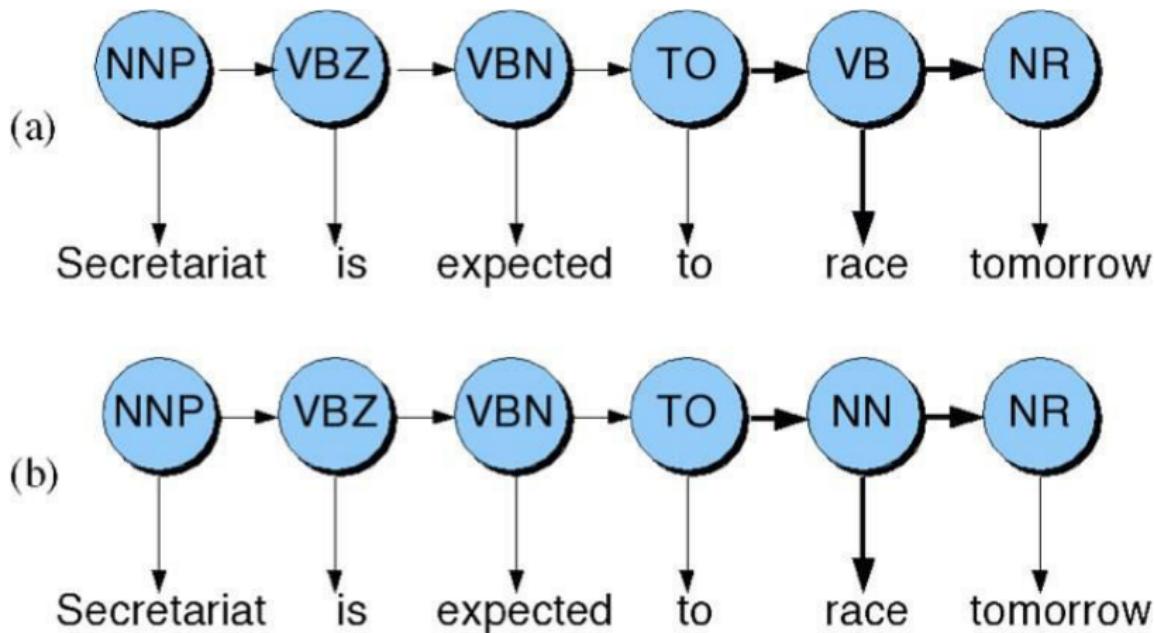
$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

Word Likelihood probabilities $p(w_i|t_i)$

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = 0.47$$

Disambiguating "race"



Disambiguating “race”

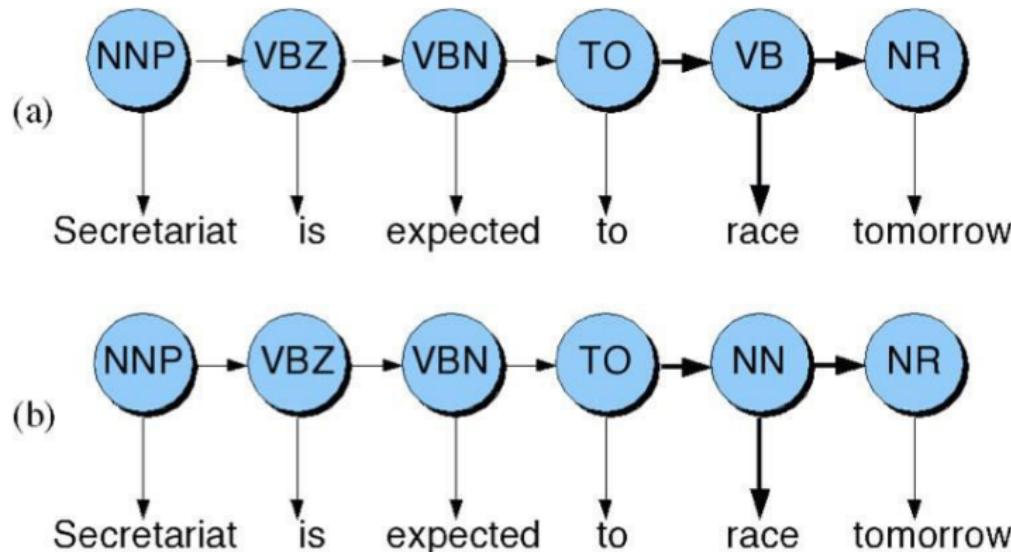
Difference in probability due to

- $P(VB|TO)$ vs. $P(NN|TO)$
- $P(race|VB)$ vs. $P(race|NN)$
- $P(NR|VB)$ vs. $P(NR|NN)$

After computing the probabilities

- $P(NN|TO)P(NR|NN)P(race|NN) = 0.0047 \times 0.0012 \times 0.00057 = 0.00000000032$
- $P(VB|TO)P(NR|VB)P(race|VB) = 0.83 \times 0.0027 \times 0.00012 = 0.00000027$

What is this model?



This is a Hidden Markov Model

Hidden Markov Models

- Tag Transition probabilities $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions) $p(w_i|t_i)$
- What we have described with these probabilities is a hidden markov model.
- Let us quickly introduce the Markov Chain, or observable Markov Model.

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n : variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

$$P(q_n | q_{n-1}, q_{n-2}, \dots, q_1)$$

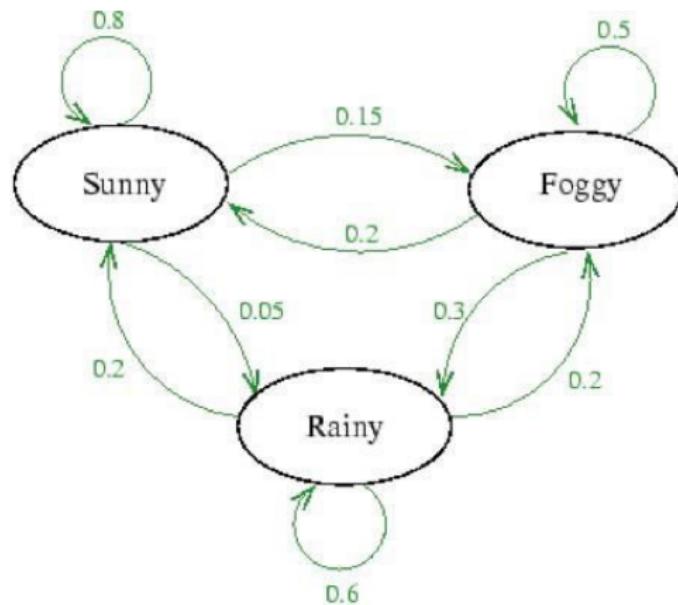
First-order Markov Assumption

$$P(q_n | q_{n-1}, q_{n-2}, \dots, q_1) = P(q_n | q_{n-1})$$

Markov Chain Transition Table

Table 1: Probabilities $p(q_{n+1}|q_n)$ of tomorrow's weather based on today's weather

		Tomorrow's weather		
		Sunny	Rainy	Foggy
Today's weather	Sunny	0.8	0.05	0.15
	Rainy	0.2	0.6	0.2
	Foggy	0.2	0.3	0.5



Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

$$\begin{aligned}&= P(q_3 = \text{rainy} | q_2 = \text{sunny}, q_1 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny}) \\&= P(q_3 = \text{rainy} | q_2 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny}) \\&= 0.05 \times 0.8 \\&= 0.04\end{aligned}$$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
'sunny' weather is both observable and state
- But in POS tagging
The output symbols are words
But the hidden states are POS tags
- A Hidden Markov Model is an extension of a Markov chain in which the output symbols are not the same as the states
- We don't know which state we are in

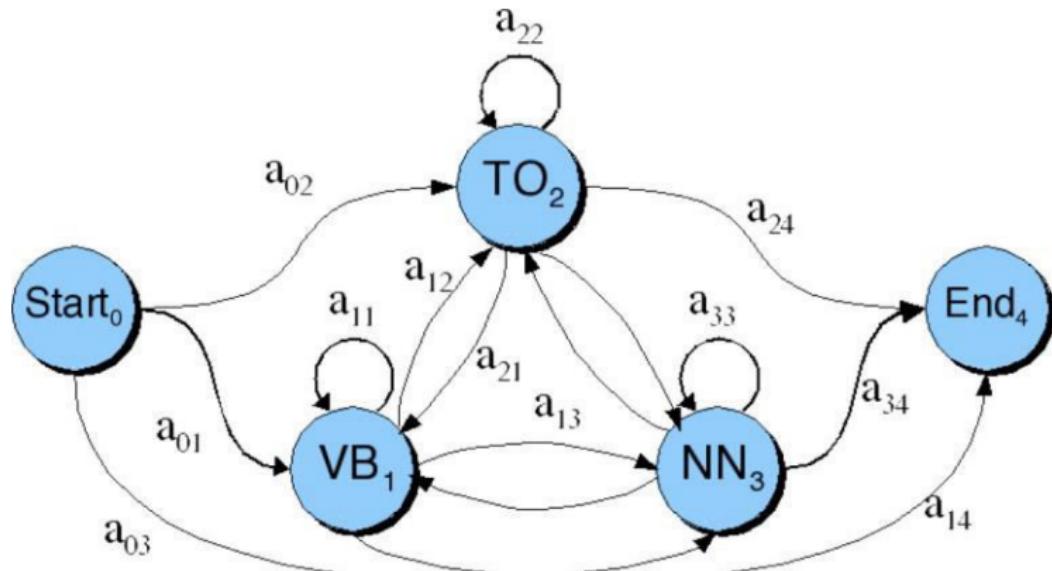
Hidden Markov Models (HMMs)

Elements of an HMM model

- A set of states (here: the tags)
- An output alphabet (here: words)
- Initial state (here: beginning of sentence)
- State transition probabilities (here $p(t_n|t_{n-1})$)
- Symbol emission probabilities (here $p(w_i|t_i)$)

Graphical Representation

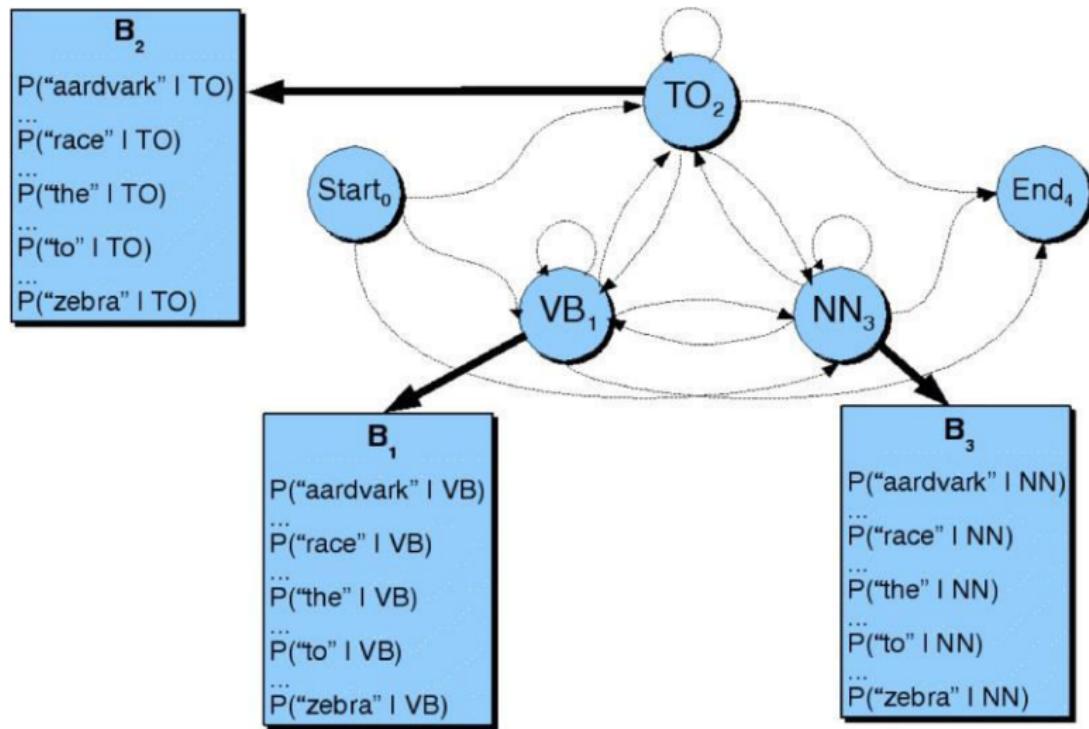
When tagging a sentence, we are walking through the state graph:



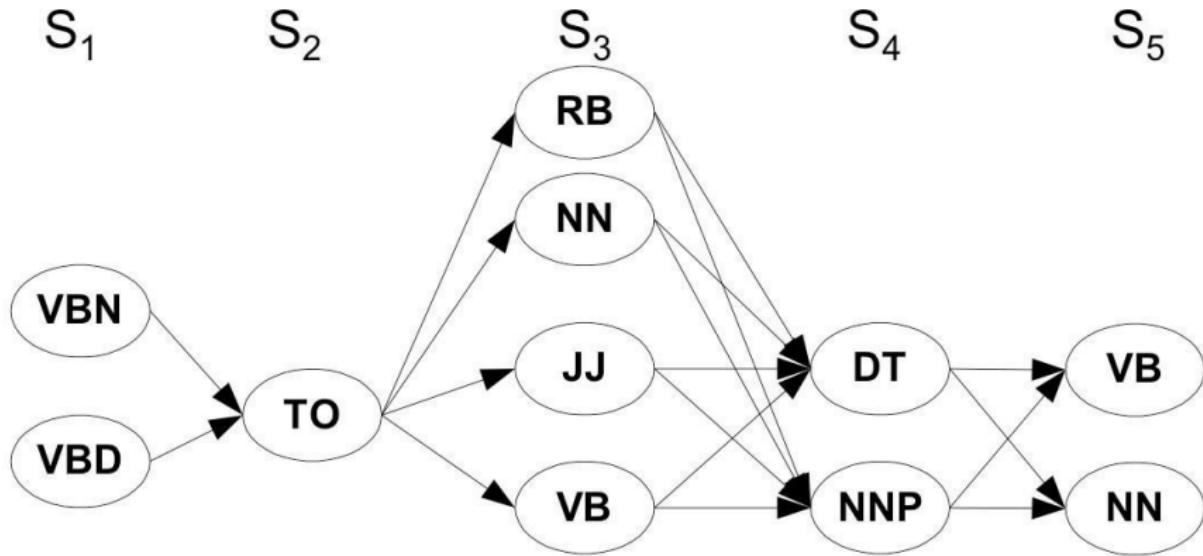
Edges are labeled with the state transition probabilities: $p(t_n | t_{n-1})$

Graphical Representation

At each state we emit a word: $P(w_n | t_n)$

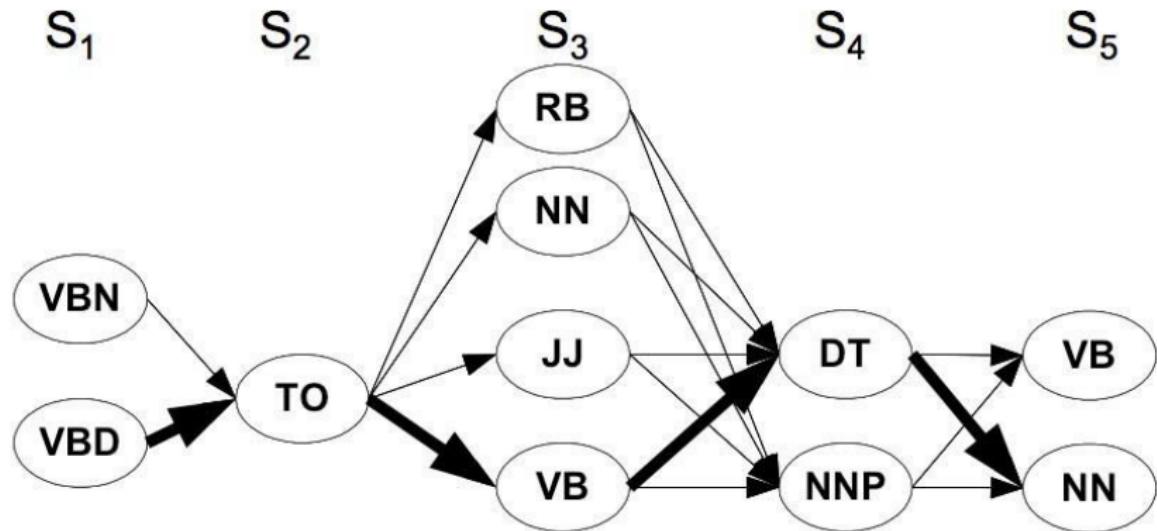


Walking through the states: best path



promised to back the bill

Walking through the states: best path



promised to back the bill

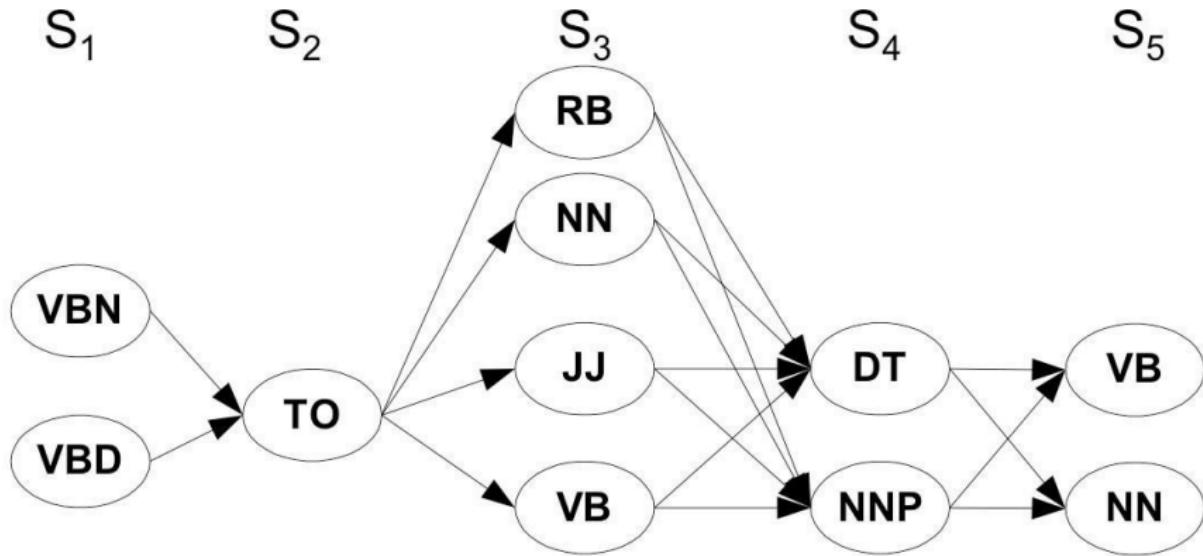
Viterbi Decoding for HMM, Parameter Learning

Pawan Goyal

CSE, IIT Kharagpur

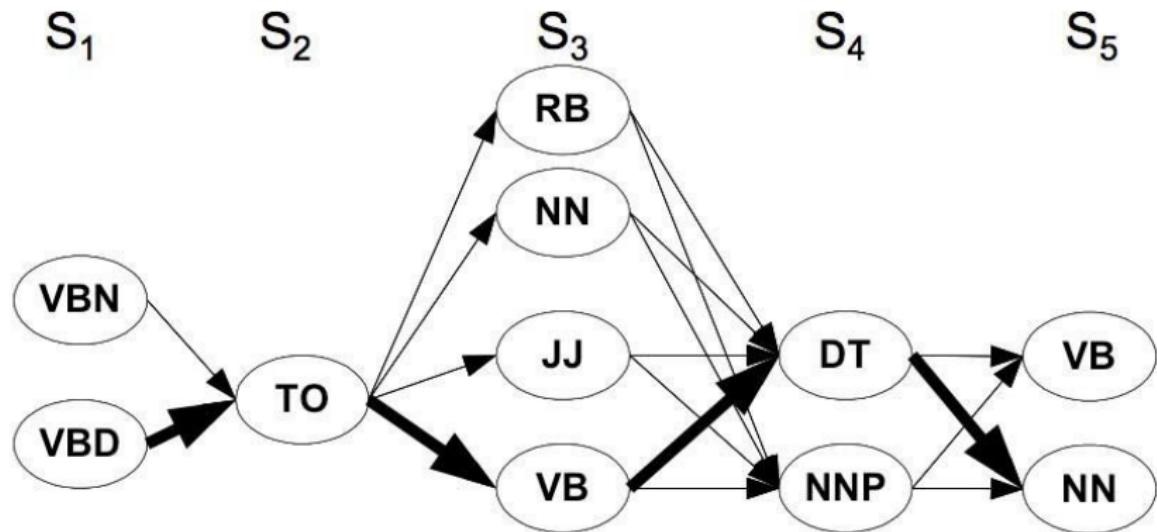
Week 4, Lecture 1

Walking through the states: best path



promised to back the bill

Walking through the states: best path



promised to back the bill

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s : $\delta_j(s)$
- Backtrace from that state to best predecessor $\Psi_j(s)$

Computing these values

- $\delta_j(s+1) = \max_{1 \leq i \leq N} \delta_i(s) p(t_j | t_i) p(w_{s+1} | t_j)$
- $\Psi_j(s+1) = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(s) p(t_j | t_i) p(w_{s+1} | t_j)$

Best final state is $\operatorname{argmax}_{1 \leq i \leq N} \delta_i(|S|)$, we can backtrack from there

Practice Question

- Suppose you want to use a HMM tagger to tag the phrase, “the light book”, where we have the following probabilities:
- $P(\text{the}|\text{Det}) = 0.3$, $P(\text{the}|\text{Noun}) = 0.1$, $P(\text{light}|\text{Noun}) = 0.003$, $P(\text{light}|\text{Adj}) = 0.002$, $P(\text{light}|\text{Verb}) = 0.06$, $P(\text{book}|\text{Noun}) = 0.003$, $P(\text{book}|\text{Verb}) = 0.01$
- $P(\text{Verb}|\text{Det}) = 0.00001$, $P(\text{Noun}|\text{Det}) = 0.5$, $P(\text{Adj}|\text{Det}) = 0.3$,
 $P(\text{Noun}|\text{Noun}) = 0.2$, $P(\text{Adj}|\text{Noun}) = 0.002$, $P(\text{Noun}|\text{Adj}) = 0.2$,
 $P(\text{Noun}|\text{Verb}) = 0.3$, $P(\text{Verb}|\text{Noun}) = 0.3$, $P(\text{Verb}|\text{Adj}) = 0.001$,
 $P(\text{Verb}|\text{Verb}) = 0.1$
- Work out in details the steps of the Viterbi algorithm. You can use a Table to show the steps. Assume all other conditional probabilities, not mentioned to be zero. Also, assume that all tags have the same probabilities to appear in the beginning of a sentence.

Learning the Parameters

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

Methods for these scenarios

- For the first scenario, parameters can be directly estimated using maximum likelihood estimate from the labeled dataset
- For the second scenario, *Baum-Welch Algorithm* is used to estimate the parameters of the hidden markov model.

Baum Welch Algorithm

Pawan Goyal

CSE, IIT Kharagpur

Week 4, Lecture 2

Baum Welch Algorithm

Uses the well-known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden markov model

Parameters of HMM

Let X_t be the random variable denoting hidden state at time t , and Y_t be the observation variable at time T . HMM parameters are given by $\vartheta = (A, B, \pi)$ where

- $A = \{a_{ij}\} = P(X_t = j | X_{t-1} = i)$ is the state transition matrix
- $\pi = \{\pi_i\} = P(X_1 = i)$ is the initial state distribution
- $B = \{b_j(y_t)\} = P(Y_t = y_t | X_t = j)$ is the emission matrix

Given observation sequences $Y = (Y_1 = y_1, Y_2 = y_2, \dots, Y_T = y_T)$, the algorithm tries to find the parameters ϑ that maximise the probability of the observation.

The Algorithm

The basic idea is to start with some random initial conditions on the parameters ϑ , estimate best values of state paths X_t using these, then re-estimate the parameters ϑ using the just-computed values of X_t , iteratively.

Intuition

- Choose some initial values for $\vartheta = (A, B, \pi)$.
- *Repeat the following step until convergence:*
- Determine probable (state) paths $\dots X_{t-1} = i, X_t = j \dots$
- Count the expected number of transitions a_{ij} as well as the expected number of times, various emissions $b_j(y_t)$ are made
- Re-estimate $\vartheta = (A, B, \pi)$ using a_{ij} and $b_j(y_t)$ s.

A forward-backward algorithm is used for finding probable paths.

Forward-Backward Algorithm

Forward Procedure

$\alpha_i(t) = P(Y_1 = y_1, \dots, Y_t = y_t, X_t = i | \vartheta)$ be the probability of seeing y_1, \dots, y_t and being in state i at time t . Found recursively using:

- $\alpha_i(1) = \pi_i b_i(y_1)$
- $\alpha_j(t+1) = b_j(y_{t+1}) \prod_{i=1}^{N_v} \alpha_i(t) a_{ij}$

Backward Procedure

$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \dots, Y_T = y_T | X_t = i, \vartheta)$ be the probability of ending partial sequence y_{t+1}, \dots, y_T given starting state i at time t . $\beta_i(t)$ is computed recursively as:

- $\beta_i(T) = 1$
- $\beta_i(t) = \sum_{j=1}^{N_v} \beta_j(t+1) a_{ij} b_j(y_{t+1})$

Finding probabilities of paths

We compute the following variables:

- Probability of being in state i at time t given the observation Y and parameters ϑ

$$\gamma_i(t) = P(X_t = i | Y, \vartheta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^N \alpha_j(t)\beta_j(t)}$$

- Probability of being in state i and j at time t and $t+1$ respectively given the observation Y and parameters ϑ

$$\zeta_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \vartheta) = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}$$

Updating the parameters

- $\pi_i = \gamma_i(1)$, expected number of times state i was seen at time 1
- $a_{ij} = \frac{\sum_{t=1}^T \zeta_{ij}(t)}{\sum_{t=1}^T \gamma_i(t)}$, expected number of transitions from state i to state j , compared to the total number of transitions away from state i
- $b_i(v_k) = \frac{\sum_{t=1}^T 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$ with $1_{y_t=v_k}$ being an indicator function, is the expected number of times the output observations are v_k while being in state i compared to the expected total number of times in state i .

Maximum Entropy Models

Pawan Goyal

CSE, IIT Kharagpur

Week 4, Lecture 3

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Limited Context

- “is clearly **marked**” → verb, past participle
- “he clearly **marked**” → verb, past tense

Possible solution: Use higher order model, combine various n-gram models to avoid sparseness problem

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.
 - ▶ Whether it is the first word in the article
 - ▶ Whether the next word is *to*
 - ▶ Whether one of the last 5 words is a preposition, etc.
- MaxEnt combines these features in a probabilistic model

Maximum Entropy: The Model

$$p_{\lambda}(y|x) = \frac{1}{Z_{\lambda}(x)} \exp \sum_i \lambda_i f_i(x,y)$$

where

- $Z_{\lambda}(x)$ is a normalizing constant given by

$$Z_{\lambda}(x) = \sum_y \exp \sum_i \lambda_i f_i(x,y)$$

- λ_i is a weight given to a feature f_i
- x denotes an observed datum and y denotes a class

What is the form of the features?

Features in Maximum Entropy Models

- Features encode elements of the context x for predicting tag y
- Context x is taken around the word w , for which a tag y is to be predicted
- Features are binary values functions, e.g.,

$$f(x,y) = \begin{cases} 1 & \text{if } isCapitalized(w) \& y = NNP \\ 0 & \text{otherwise} \end{cases}$$

Example Features

Example: Named Entities

- LOCATION (in Arcadia)
- LOCATION (in Québec)
- DRUG (taking Zantac)
- PERSON (saw Sue)

Example Features

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

Tagging with Maximum Entropy Model

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tag sequence candidate $\{t_1, \dots, t_n\}$ has conditional probability:

$$P(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n p(t_i | x_i)$$

- The context x_i also includes previously assigned tags for a fixed history.
- Beam search is used to find the most probable sequence

Beam 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

But what is a MaxEnt model?

Let's go to the basics now!

Maximum Entropy Model

Intuitive Principle

Model all that is known and assume nothing about that which is unknown.

Given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible.

Maximum Entropy: Overview

- Suppose we wish to model an expert translator's decisions concerning the proper French rendering of the English word 'in'.
- Each French word or phrase f is assigned an estimate $p(f)$, probability that the expert would choose f as a translation of 'in'.
- Collect a large sample of instances of the expert's decisions
- **Goal:** extract a set of facts about the decision-making process (first task) that will aid in constructing a model of this process (second task)

Maximum Entropy Model: Overview

First clue: list of allowed translations

- Suppose the translator always chooses among {dans, en, á, au cours de, pendant}.
- First constraint: $p(\text{dans})+p(\text{en})+p(\text{\'a})+p(\text{au cours de})+p(\text{pendant}) = 1$.
- Infinite number of models p for which this identity holds, the most intuitive model?
- *allocate the total probability evenly among the five possible phrases* → most uniform model subject to our knowledge.
- *Is it the most uniform model overall?* → No, that would grant an equal probability to every possible French phrase.

Maximum Entropy Model: Overview

More clues from the expert's decision

- **Second clue:** Suppose the expert chose either ‘dans’ or ‘en’ 30% of the time.
- **Third clue:** In half of the cases, the expert chose either ‘dans’ or ‘á’

How do we measure uniformity of a model?

As we add complexity to the model, we face two difficulties:

- What exactly is meant by “uniform”?
- How can one measure the uniformity of a model?

Maximum Entropy Modeling

Entropy: measures the uncertainty of a distribution.

Quantifying uncertainty ("surprise")

- Event x
- Probability p_x
- Surprise: $\log(1/p_x)$

Entropy: expected surprise (over p)

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

Coin Tossing

Maximum Entropy Modeling

Distribution required

- Minimize commitment = maximize entropy
- Resemble some reference distribution

Solution

Maximize entropy H , subject to feature-based constraints:

$$E_p[f_i] = E_{\tilde{p}}[f_i]$$

Adding constraints

- Lowers maximum entropy
- Brings the distribution further from uniform and closer to data

Maximum Entropy Principle

Given n feature functions f_i , we would like p to lie in the subset C of P defined by

$$C = \{p \in P \mid p(f_i) = \tilde{p}(f_i), i \in \{1, 2, \dots, n\}\}$$

Empirical count (expectation) of a feature

X

$$\tilde{p}(f_i) = \frac{\tilde{p}(x,y)f_i(x,y)}{x,y}$$

Model expectation of a feature

X

$$p(f_i) = \frac{\tilde{p}(x)p(y|x)f_i(x,y)}{x,y}$$

Select the distribution which is most uniform (conditional probability):

$$p^* = \operatorname{argmax}_{p \in C} H(p) = H(Y|X) \approx - \sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x)$$

Maximum Entropy Principle

$$p^* = \operatorname{argmax}_{p \in \mathcal{C}} H(p)$$

Constraint Optimization

Introduce a parameter λ_i for each feature f_i . Lagrangian is given by

$$\Lambda(p, \lambda) = H(p) + \sum_i \lambda_i (p(f_i) - \tilde{p}(f_i))$$

Solving, we get

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp \sum_i \lambda_i f_i(x, y)$$

where $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y \exp \sum_i \lambda_i f_i(x, y)$$

Maximum Entropy Models

Pawan Goyal

CSE, IIT Kharagpur

Week 4, Lecture 4

Practice Question

Consider the maximum entropy model for POS tagging, where you want to estimate $P(\text{tag}|\text{word})$. In a hypothetical setting, assume that *tag* can take the values *D*, *N* and *V* (short forms for Determiner, Noun and Verb). The variable *word* could be any member of a set V of possible words, where V contains the words *a*, *man*, *sleeps*, as well as additional words. The distribution should give the following probabilities

- $P(D|a) = 0.9$
- $P(N|\text{man}) = 0.9$
- $P(V|\text{sleeps}) = 0.9$
- $P(D|\text{word}) = 0.6$ for any word other than *a*, *man* or *sleeps*
- $P(N|\text{word}) = 0.3$ for any word other than *a*, *man* or *sleeps*
- $P(V|\text{word}) = 0.1$ for any word other than *a*, *man* or *sleeps*

It is assumed that all other probabilities, not defined above could take any values such that

$$\underset{\text{tag}}{P(\text{tag}|\text{word})} = 1 \text{ is satisfied for any word in } V.$$

- Define the features of your maximum entropy model that can model this distribution. Mark your features as f_1, f_2 and so on. Each feature should have the same format as explained in the class.
[Hint: 6 Features should make the analysis easier]
- For each feature f_i , assume a weight λ_i . Now, write expression for the following probabilities in terms of your model parameters
 - $P(D|\text{cat})$
 - $P(N|\text{laughs})$
 - $P(D|\text{man})$
- What value do the parameters in your model take to give the distribution as described above. (i.e. $P(D|a) = 0.9$ and so on. You may leave the final answer in terms of equations)

Features for POS Tagging (Ratnaparakhi, 1996)

The specific word and tag context available to a feature is

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

Example: $f_j(h_i, t_i) = 1$ if $\text{suffix}(w_i) = "ing"$ & $t_i = VBG$

Example Features

Condition	Features
w_i is not rare	$w_i = X$ & $t_i = T$
w_i is rare	X is prefix of w_i , $ X \leq 4$ & $t_i = T$
	X is suffix of w_i , $ X \leq 4$ & $t_i = T$
	w_i contains number & $t_i = T$
	w_i contains uppercase character & $t_i = T$
	w_i contains hyphen & $t_i = T$
$\forall w_i$	$t_{i-1} = X$ & $t_i = T$
	$t_{i-2}t_{i-1} = XY$ & $t_i = T$
	$w_{i-1} = X$ & $t_i = T$
	$w_{i-2} = X$ & $t_i = T$
	$w_{i+1} = X$ & $t_i = T$
	$w_{i+2} = X$ & $t_i = T$

Example Features

<i>Word:</i>	the	stories	about	well-heeled	communities	and	developers
<i>Tag:</i>	DT	NNS	IN	JJ	NNS	CC	NNS
<i>Position:</i>	1	2	3	4	5	6	7

$w_i = \text{about}$	$\& t_i = \text{IN}$	$w_{i-1} = \text{about}$	$\& t_i = \text{JJ}$
$w_{i-1} = \text{stories}$	$\& t_i = \text{IN}$	$w_{i-2} = \text{stories}$	$\& t_i = \text{JJ}$
$w_{i-2} = \text{the}$	$\& t_i = \text{IN}$	$w_{i+1} = \text{communities}$	$\& t_i = \text{JJ}$
$w_{i+1} = \text{well-heeled}$	$\& t_i = \text{IN}$	$w_{i+2} = \text{and}$	$\& t_i = \text{JJ}$
$w_{i+2} = \text{communities}$	$\& t_i = \text{IN}$	$t_{i-1} = \text{IN}$	$\& t_i = \text{JJ}$
$t_{i-1} = \text{NNS}$	$\& t_i = \text{IN}$	$t_{i-2}t_{i-1} = \text{NNS IN}$	$\& t_i = \text{JJ}$
$t_{i-2}t_{i-1} = \text{DT NNS}$	$\& t_i = \text{IN}$	$\text{prefix}(w_i) = w$	$\& t_i = \text{JJ}$
		$\text{prefix}(w_i) = we$	$\& t_i = \text{JJ}$
		$\text{prefix}(w_i) = wel$	$\& t_i = \text{JJ}$
		$\text{prefix}(w_i) = well$	$\& t_i = \text{JJ}$
		$\text{suffix}(w_i) = d$	$\& t_i = \text{JJ}$
		$\text{suffix}(w_i) = ed$	$\& t_i = \text{JJ}$
		$\text{suffix}(w_i) = led$	$\& t_i = \text{JJ}$
		$\text{suffix}(w_i) = eled$	$\& t_i = \text{JJ}$
		$w_i \text{ contains hyphen}$	$\& t_i = \text{JJ}$

Search Algorithm

Conditional Probability

Given a sentence $\{w_1, \dots, w_n\}$, a tag sequence candidate $\{t_1, \dots, t_n\}$ has conditional probability:

$$P(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n p(t_i | x_i)$$

NPTEV

Search Algorithm

Conditional Probability

Given a sentence $\{w_1, \dots, w_n\}$, a tag sequence candidate $\{t_1, \dots, t_n\}$ has conditional probability:

$$P(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n p(t_i | x_i)$$

A *Tag Dictionary* is used, which, for each known word, lists the tags that it has appeared with in the training set.

Search Algorithm

Let $W = \{w_1, \dots, w_n\}$ be a test sentence, s_{ij} be the j th highest probability tag sequence up to and including word w_i .

Search description

- Generate tags for w_1 , find top N , set s_{1j} , $1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
 - ▶ Initialize $j = 1$
 - ▶ Generate tags for w_i , given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
 - ▶ $j = j + 1$, repeat if $j \leq N$
- Find N highest probability sequences generated by above loop, set s_{ij} accordingly
- $i = i + 1$, repeat if $i \leq n$
- Return highest probability sequence s_{n1}

A Good Reference

Berger et al., *A Maximum Entropy Approach to Natural Language Processing*,
Computational Linguistics, Vol. 22, No. 1.

Conditional Random Fields

Pawan Goyal

CSE, IIT Kharagpur

Week 4, Lecture 5

Practice Question

Suppose you want to use a MaxEnt tagger to tag the sentence, “the light book”. We know that the top 2 POS tags for the words *the*, *light* and *book* are $\{Det, Noun\}$, $\{Verb, Adj\}$ and $\{Verb, Noun\}$, respectively. Assume that the MaxEnt model uses the following history h_i (context) for a word w_i :

$$h_i = \{w_{i-1}, w_i, w_{i+1}, t_{i-1}\}$$

where w_{i-1} and w_{i+1} correspond to the previous and next words and t_{i-1} corresponds to the tag of the previous word. Accordingly, the following features are being used by the MaxEnt model:

- $f_1: t_{i-1} = Det$ and $t_i = Adj$
- $f_2: t_{i-1} = Noun$ and $t_i = Verb$
- $f_3: t_{i-1} = Adj$ and $t_i = Noun$
- $f_4: w_{i-1} = the$ and $t_i = Adj$
- $f_5: w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- $f_6: w_{i-1} = light$ and $t_i = Noun$
- $f_7: w_{i+1} = light$ and $t_i = Det$
- $f_8: w_{i-1} = NULL$ and $t_i = Noun$

Assume that each feature has a uniform weight of 1.0.

Use Beam search algorithm with a beam-size of 2 to identify the highest probability tag sequence for the sentence.

Problem with Maximum Entropy Models

Per-state normalization

All the mass that arrives at a state must be distributed among the possible successor states

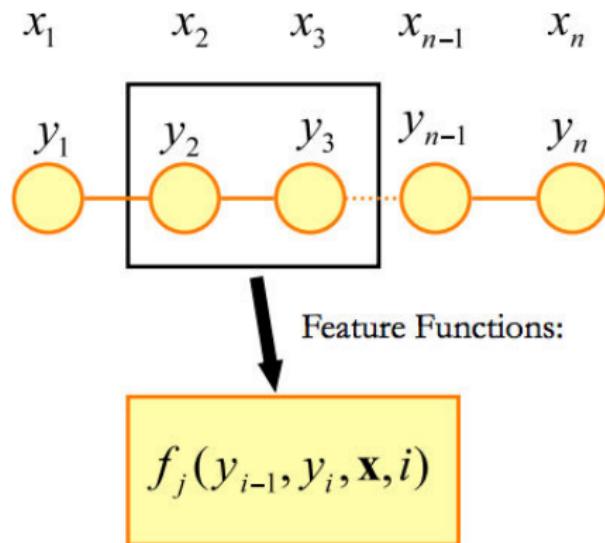
This gives a 'label bias' problem

Let's see the intuition (on paper)

Conditional Random Fields

- CRFs are conditionally trained, undirected graphical models.
- Let's look at the linear chain structure

Conditional Random Fields: Feature Functions



Feature Functions

Express some characteristic of the empirical distribution
that we wish to hold in the model distribution

$$f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

1 if $y_{i-1} = IN$ and
 $y_i = NNP$ and
 $x_i = September$

0 otherwise

Conditional Random Fields: Distribution

Label sequence modelled as a normalized product of feature functions:

$$P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{i=1}^n \sum_j \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \sum_{i=1}^n \sum_j \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

- Have the advantages of MEMM but avoid the label bias problem
- CRFs are globally normalized, whereas MEMMs are locally normalized.
- Widely used and applied. CRFs have been (close to) state-of-the-art in many sequence labeling tasks.

Syntax - Introduction

Pawan Goyal

CSE, IIT Kharagpur

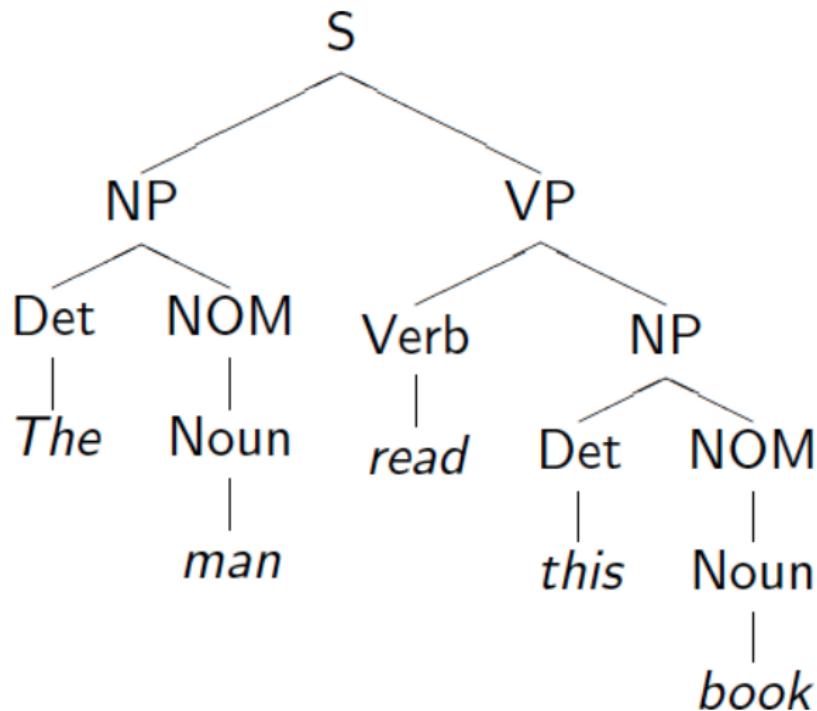
Week 5: Lecture 1

What is Syntax?

- Refers to the way words are arranged together, and the relationship between them.
- **Language Models:** Importance of modeling word order
- **POS categories:** An equivalence class for words
- More complex notions: constituency, grammatical relations, subcategorization etc.



Syntax Tree: Example



Defining the notions: Constituency

Constituent

A group of words acts as a single unit - phrases, clauses etc.

Part of Speech - "Substitution Test"

The {sad, intelligent, green, fat, ...} one is in the corner.

Constituency: Noun Phrase

- *Kermit the frog*
- *they*
- *December twenty-sixth*
- *the reason he is running for president*

Constituent Phrases

Usually named based on the word that heads the constituent:

<i>the man from Amherst</i>	is a Noun Phrase (NP) because the head man is a noun
<i>extremely clever</i>	is an Adjective Phrase (AP) because the head clever is an adjective
<i>down the river</i>	is a Prepositional Phrase (PP) because the head down is a preposition
<i>killed the rabbit</i>	is a Verb Phrase (VP) because the head killed is a verb

Words can also act as phrases

Joe grew potatoes

Joe and *potatoes* are both nouns and noun phrases

Compare with: *The man from Amherst grew beautiful russet potatoes.*

Joe appears in a place that a larger noun phrase could have been.

Evidence that constituency exists

They appear in similar environments

Kermit the frog comes on stage

They come to Massachusetts every summer

December twenty-sixth comes after Christmas

The reason he is running for president comes out only now.

But not each individual word in the constituent

*The comes our... *is comes out... *for comes out...

Can be placed in a number of different locations

Constituent = Prepositional phrase: On December twenty-sixth

On December twenty-sixth I'd like to fly to Florida.

I'd like to fly on December twenty-sixth to Florida.

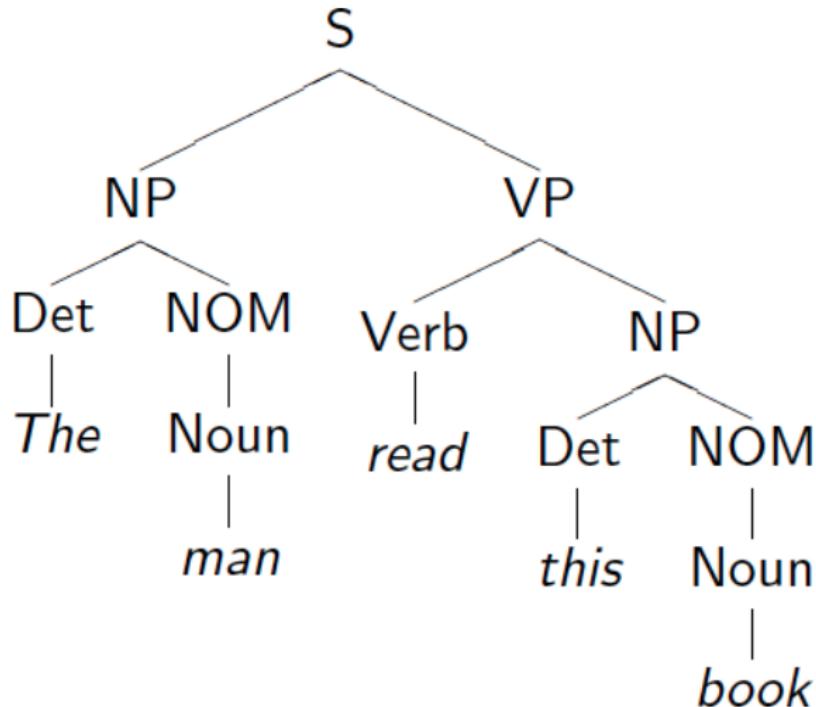
I'd like to fly to Florida on December twenty-sixth.

But not split apart

*On December I'd like to fly twenty-sixth to Florida.

*On I'd like to fly December twenty-sixth to Florida.

Modeling Constituency: what tool do we need?



Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Consists of production Rules

These rules express the ways in which the symbols of the language can be grouped and ordered together

Example

Noun phrase can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal; a Nominal can be more than one nouns

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

$Nominal \rightarrow Noun\ | Noun\ Nominal$

CFG for Languages

CFG: $G = (T, N, S, R)$

- T : set of terminals
- N : set of non-terminals
 - ▶ For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R : Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)^*$

Terminals and pre-terminals

Terminals mainly correspond to words in the language while pre-terminals mainly correspond to POS categories

CFG for Languages

Example

$NP \rightarrow \text{Det Nominal}$

$NP \rightarrow \text{ProperNoun}$

$\text{Nominal} \rightarrow \text{Noun} \mid \text{Noun Nominal}$

Now, these can be combined with other rules, that express facts about a lexicon.

$\text{Det} \rightarrow \text{a}$

$\text{Det} \rightarrow \text{the}$

$\text{Noun} \rightarrow \text{flight}$

Can you identify the terminal, non-terminals and preterminals?

CFG as a generator

NP → Det Nominal

NP → ProperNoun

Nominal → Noun | Noun Nominal

Det → a

Det → the

Noun → flight

Generating 'a flight':

NP → Det Nominal

→ Det Noun → a Noun → a flight

- Thus a CFG can be used to randomly generate a series of strings
- This sequence of rule expansions is called a derivation of the string of words, usually represented as a tree

CFGs and Grammaticality

A CFG defines a formal language = set of all sentences (string of words) that can be derived by the grammar

- Sentences in this set are said to be **grammatical**
- Sentences outside this set are said to be **ungrammatical**

Recursive Definition

- $PP \rightarrow \text{Prep } NP$
- $NP \rightarrow \text{Noun } PP$

Example Sentence

[_sThe mailman ate his [_{NP} lunch [_{PP} with his friend [_{PP} from the cleaning staff [_{PP} of the building [_{PP} at the intersection [_{PP} on the north end [_{PP} of town]]]]]]].

What does Context stand for in CFG?

- The notion of *context* has nothing to do with the ordinary meaning of word context in language
- All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)

$A \rightarrow BC$

- I can rewrite A as B followed by C regardless of the context in which A is found
- Or when I see a B followed by a C , I can infer an A regardless of the surrounding context

Syntax -Parsing I

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 2

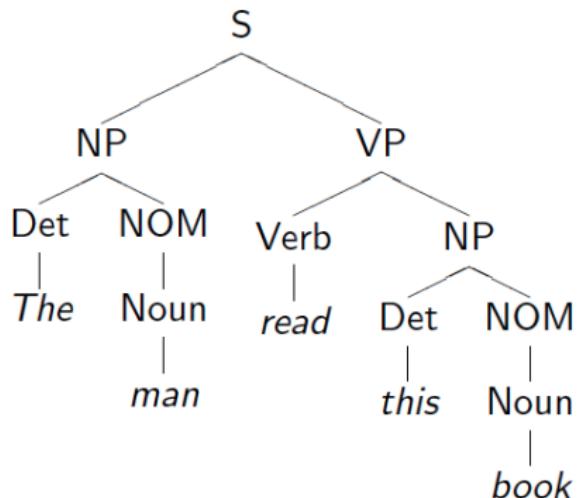
Grammar Rewrite Rules

$S \rightarrow NP\ VP$ Det \rightarrow *that* | *this* | *a* | *the*
 $S \rightarrow Aux\ NP\ VP$ Noun \rightarrow *book* | *flight* | *meal* | *man*
 $S \rightarrow VP$ Verb \rightarrow *book* | *include* | *read*
 $NP \rightarrow Det\ NOM$ Aux \rightarrow *does*
NOM \rightarrow Noun
NOM \rightarrow Noun NOM
VP \rightarrow Verb
VP \rightarrow Verb NP

$S \rightarrow NP\ VP$
 $\rightarrow Det\ NOM\ VP$
 $\rightarrow The\ NOM\ VP$
 $\rightarrow The\ Noun\ VP$
 $\rightarrow The\ man\ VP$
 $\rightarrow The\ man\ Verb\ NP$
 $\rightarrow The\ man\ read\ NP$
 $\rightarrow The\ man\ read\ Det\ NOM$
 $\rightarrow The\ man\ read\ this\ NOM$
 $\rightarrow The\ man\ read\ this\ Noun$
 $\rightarrow The\ man\ read\ this\ book$

Parse Tree

$S \rightarrow NP\ VP$
 $\rightarrow Det\ NOM\ VP$
 $\rightarrow The\ NOM\ VP$
 $\rightarrow The\ Noun\ VP$
 $\rightarrow The\ man\ VP$
 $\rightarrow The\ man\ Verb\ NP$
 $\rightarrow The\ man\ read\ NP$
 $\rightarrow The\ man\ read\ Det\ NOM$
 $\rightarrow The\ man\ read\ this\ NOM$
 $\rightarrow The\ man\ read\ this\ Noun$
 $\rightarrow The\ man\ read\ this\ book$



What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S , which cover exactly the words in the input

What are the constraints? "book that flight"

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S
- Give rise to two search strategies: *top-down* (goal-oriented) and *bottom-up* (data-directed)

Grammar

$S \rightarrow NP\ VP$

$S \rightarrow Aux\ NP\ VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal\ Noun$

$Nominal \rightarrow Nominal\ PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Prep\ NP$

Lexicon

$Det \rightarrow the | a | that | this$

$Noun \rightarrow book | flight | meal | money$

$Verb \rightarrow book | include | prefer$

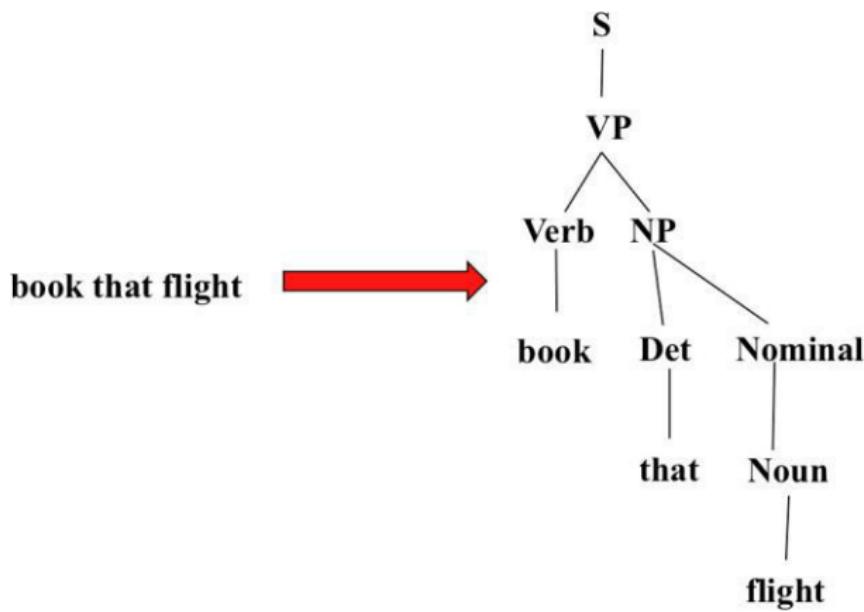
$Pronoun \rightarrow I | he | she | me$

$Proper-Noun \rightarrow Houston | NWA$

$Aux \rightarrow does$

$Prep \rightarrow from | to | on | near | through$

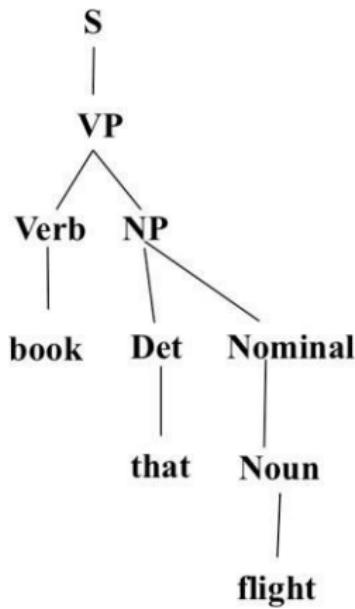
Parsing



Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S , by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom
- Trees whose leaves fail to match the words in the input can be rejected

Top-Down Parsing



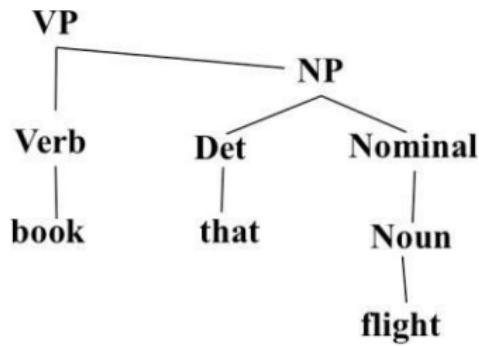
Bottom-Up Parsing

- The parser starts with the words of the input, and tries to build trees from the words up, by applying rules from the grammar one at a time
- Parser looks for the places in the parse-in-progress where the right-hand-side of some rule might fit.

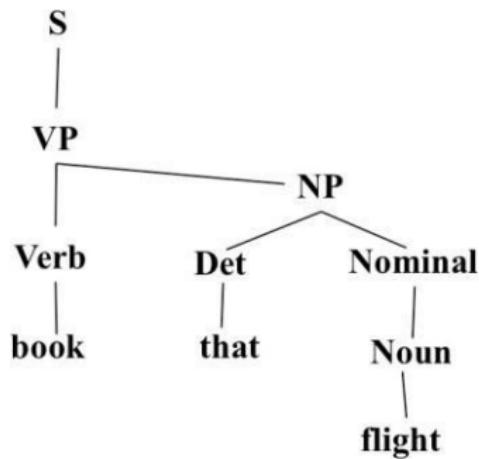
Bottom-Up Parsing

book that flight

Bottom-Up Parsing



Bottom-Up Parsing



Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where n is the length of the input string.

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar
- Earley Parser - top-down, does not require normalizing grammar, more complex
- More generally, *chart parsers* retain completed phrases in a chart and can combine top-down and bottom-up searches.

CKY Algorithm

- Grammar must be converted to Chomsky normal form (CNF) in which all productions must have
 - ▶ Either, exactly two non-terminals on the RHS
 - ▶ Or, 1 terminal symbol on the RHS
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart)

Converting to CNF

Original Grammar

$S \rightarrow NP\ VP$

$S \rightarrow Aux\ NP\ VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal\ Noun$

$Nominal \rightarrow Nominal\ PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Prep\ NP$

$Pronoun \rightarrow I\ | he\ | she\ | me$

$Noun \rightarrow book\ | flight\ | meal\ | money$

$Verb \rightarrow book\ | include\ | prefer$

$Proper-Noun \rightarrow Houston\ | NWA$

Chomsky Normal Form

$S \rightarrow NP\ VP$

$S \rightarrow X1\ VP$

$X1 \rightarrow Aux\ NP$

$S \rightarrow book\ | include\ | prefer$

$S \rightarrow Verb\ NP$

$S \rightarrow VP\ PP$

$NP \rightarrow I\ | he\ | she\ | me$

$NP \rightarrow Houston\ | NWA$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow book\ | flight\ | meal\ | money$

$Nominal \rightarrow Nominal\ Noun$

$Nominal \rightarrow Nominal\ PP$

$VP \rightarrow book\ | include\ | prefer$

$VP \rightarrow Verb\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Prep\ NP$

$Pronoun \rightarrow I\ | he\ | she\ | me$

$Noun \rightarrow book\ | flight\ | meal\ | money$

$Verb \rightarrow book\ | include\ | prefer$

$Proper-Noun \rightarrow Houston\ | NWA$

Syntax -CKY, PCFGs

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 3

CKY Algorithm

- Let n be the number of words in the input. Think about $n + 1$ lines separating them, numbered 0 to n .
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j .
- We build the Table bottom-up.

Home Exercise

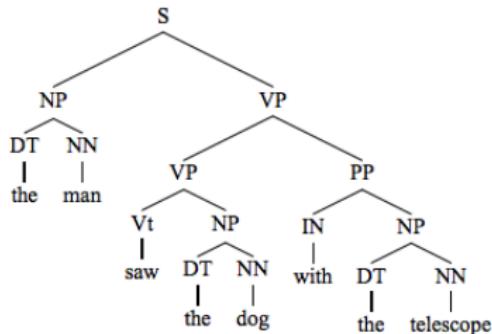
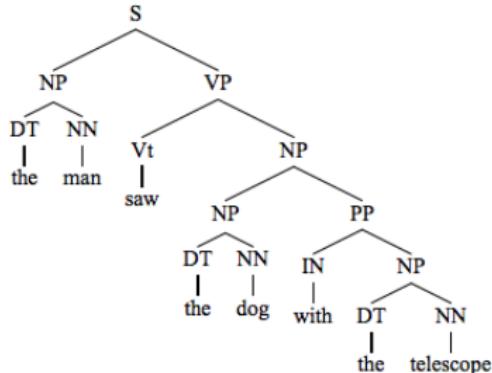
Use CKY algorithm to find the parse tree for “Book the flight through Houston” using the CNF form shown in the previous slide.

CKY for CFG

a 1	pilot 2	likes 3	flying 4	planes 5
DT	NP	-	-	S S
	NN	-	-	-
		VBZ	-	VP VP
			JJ VBG	NP VP
				NNS

$S \rightarrow NP\ VP$
 $VP \rightarrow VBG\ NNS$
 $VP \rightarrow VBZ\ VP$
 $VP \rightarrow VBZ\ NP$
 $NP \rightarrow DT\ NN$
 $NP \rightarrow JJ\ NNS$
 $DT \rightarrow a$
 $NN \rightarrow pilot$
 $VBZ \rightarrow likes$
 $VBG \rightarrow flying$
 $JJ \rightarrow flying$
 $NNS \rightarrow planes$

What about Ambiguities?



Probabilistic Context-free grammars (PCFGs)

PCFG: $G = (T, N, S, R, P)$

- T : set of terminals
- N : set of non-terminals
 - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R : Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)^*$
- $P(R)$ gives the probability of each rule.

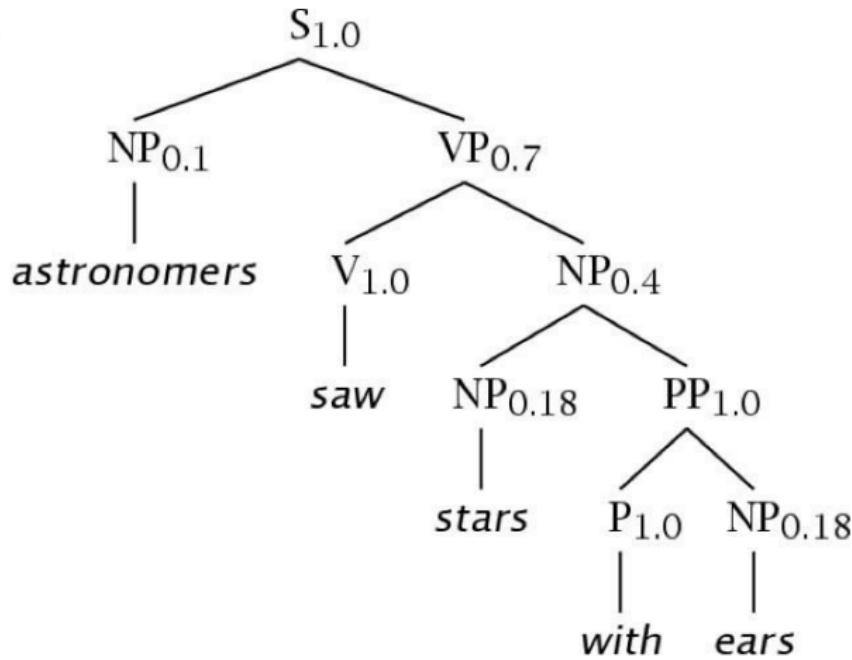
$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

A Simple PCFG (in CNF)

S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	<i>astronomers</i>	0.1
VP	→	VP PP	0.3	NP	→	<i>ears</i>	0.18
PP	→	P NP	1.0	NP	→	<i>saw</i>	0.04
P	→	<i>with</i>	1.0	NP	→	<i>stars</i>	0.18
V	→	<i>saw</i>	1.0	NP	→	<i>telescope</i>	0.1

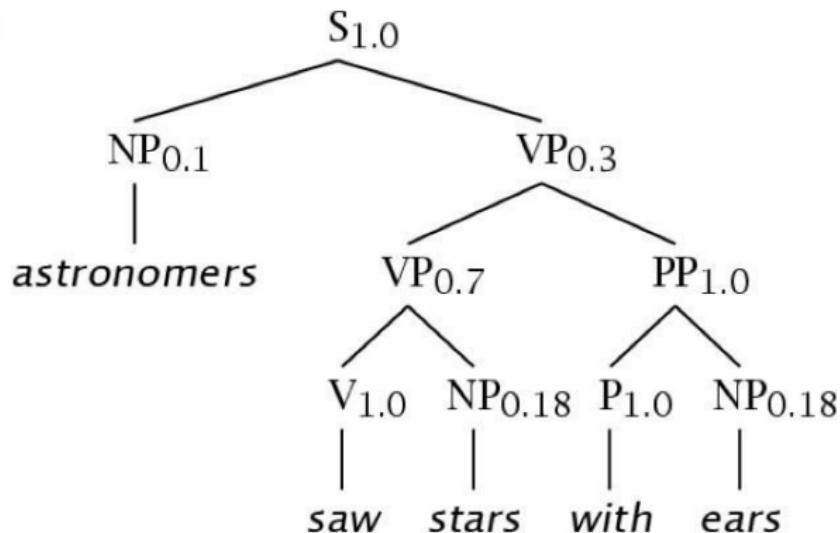
Example Trees

$t_1:$



Example Trees

t_2 :



Probability of trees and strings

- $P(t)$: The probability of tree is the product of the probabilities of the rules used to generate it
- $P(w_{1n})$: The probability of the string is the sum of the probabilities of the trees which have that string as their yield

Tree and String probabilities

w_{15} = *astronomers saw stars with ears*

$$\begin{aligned} P(t_1) &= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 \\ &\quad * 1.0 * 1.0 * 0.18 \end{aligned}$$

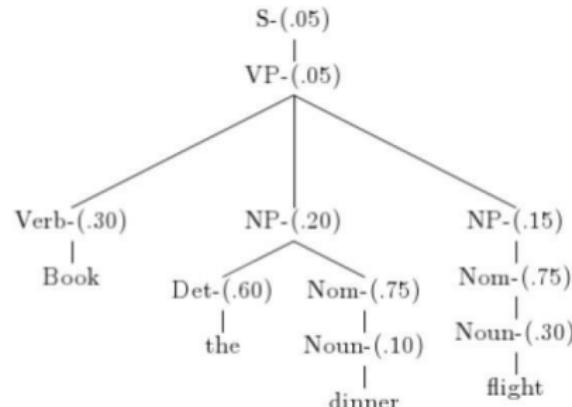
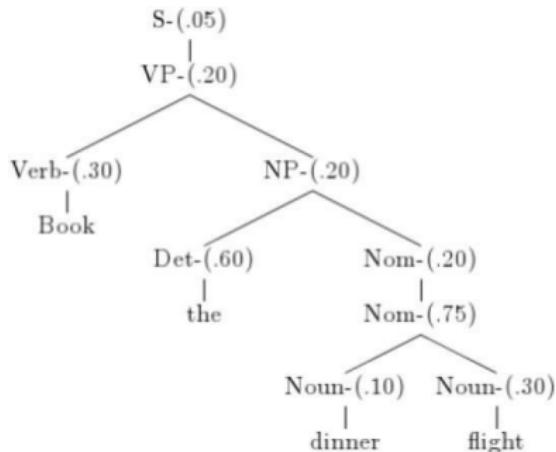
$$= 0.0009072$$

$$\begin{aligned} P(t_2) &= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 \\ &\quad * 1.0 * 1.0 * 0.18 \end{aligned}$$

$$= 0.0006804$$

$$\begin{aligned} P(w_{15}) &= P(t_1) + P(t_2) \\ &= 0.0009072 + 0.0006804 \\ &= 0.0015876 \end{aligned}$$

"Book the dinner flight"



Probabilities

- Parse tree 1: $.05 \times .20 \times .30 \times .20 \times .60 \times .20 \times .75 \times .10 \times .30 = 1.62 \times 10^{-6}$
- Parse tree 2: $.05 \times .05 \times .30 \times .20 \times .60 \times .75 \times .10 \times .15 \times .75 \times .30 = 2.28 \times 10^{-7}$

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- In practice, a PCFG is a worse language model for English than an n-gram model
- All else being equal, the probability of a smaller tree is greater than a larger tree

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$\operatorname{argmax}_t P(t|w_{1m}, G)$$

- What is the probability of a sentence?

$$P(w_{1m}|G)$$

- How to learn the rule probabilities in the grammar G ?

PCFGs - Inside-outside probabilities

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 4

How to find the most likely parse?: CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5

$S \rightarrow NP\ VP$ [1.0]
 $VP \rightarrow VBG\ NNS$ [0.1]
 $VP \rightarrow VBZ\ VP$ [0.1]
 $VP \rightarrow VBZ\ NP$ [0.3]
 $NP \rightarrow DT\ NN$ [0.3]
 $NP \rightarrow JJ\ NNS$ [0.4]
 $DT \rightarrow a$ [0.3]
 $NN \rightarrow pilot$ [0.1]
 $VBZ \rightarrow likes$ [0.4]
 $VBG \rightarrow flying$ [0.5]
 $JJ \rightarrow flying$ [0.1]
 $NNS \rightarrow planes$ [.34]

CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5	
DT [0.3]	NP [.009]	-	-	S $[1.4688 \times 10^{-5}]$ S $[6.12 \times 10^{-6}]$	
	NN [0.1]	-	-	-	
		VBZ [0.4]	-	VP [.001632] VP [.00068]	
			JJ [0.1] VBG [0.5]	NP [.0136] VP [.017]	
				NNS [.34]	$0.009 \times 0.00068 \times$ $1.0 = 6.12 \times 10^{-6}$

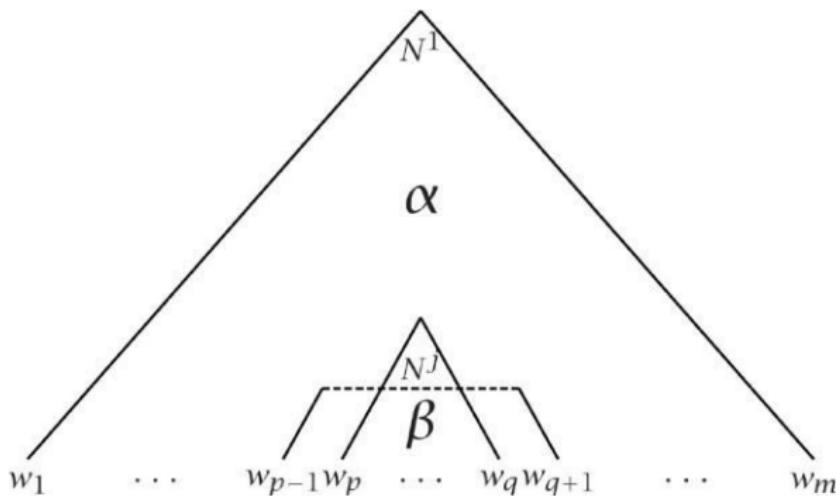
$S \rightarrow NP VP$ [1.0]
 $VP \rightarrow VBG NNS$ [0.1]
 $VP \rightarrow VBZ VP$ [0.1]
 $VP \rightarrow VBZ NP$ [0.3]
 $NP \rightarrow DT NN$ [0.3]
 $NP \rightarrow JJ NNS$ [0.4]
 $DT \rightarrow a$ [0.3]
 $NN \rightarrow pilot$ [0.1]
 $VBZ \rightarrow likes$ [0.4]
 $VBG \rightarrow flying$ [0.5]
 $JJ \rightarrow flying$ [0.1]
 $NNS \rightarrow planes$ [.34]

Probability of a String

$$P(w_{1m}|G)$$

- In general, simply summing the probabilities of all possible parse trees is not an efficient way to calculate the string probability
- We use *inside algorithm*, a dynamic programming algorithm based on inside probabilities.

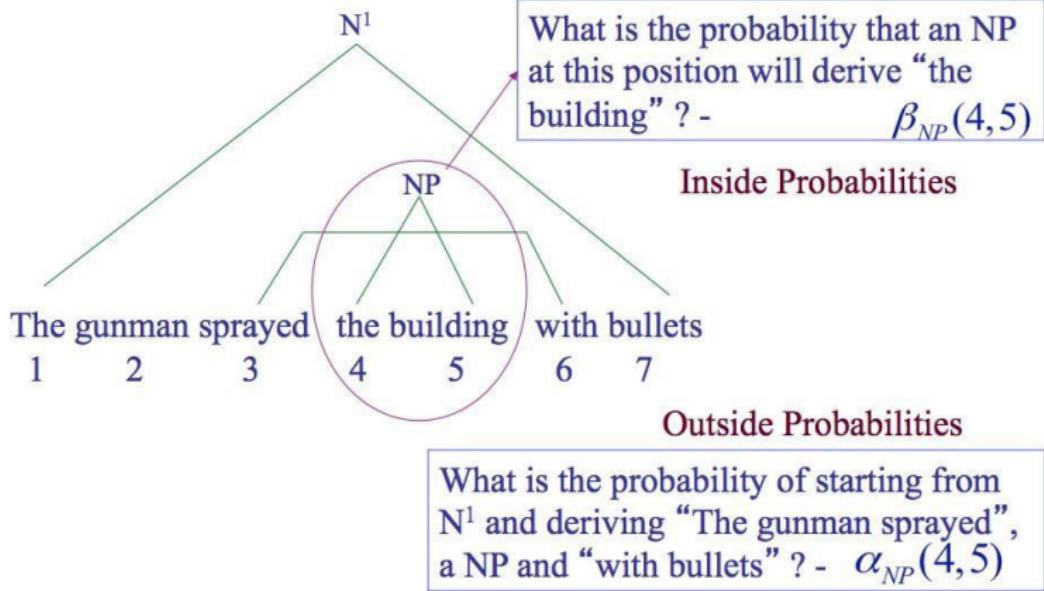
Inside and Outside Probabilities



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

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

Inside-outside probabilities

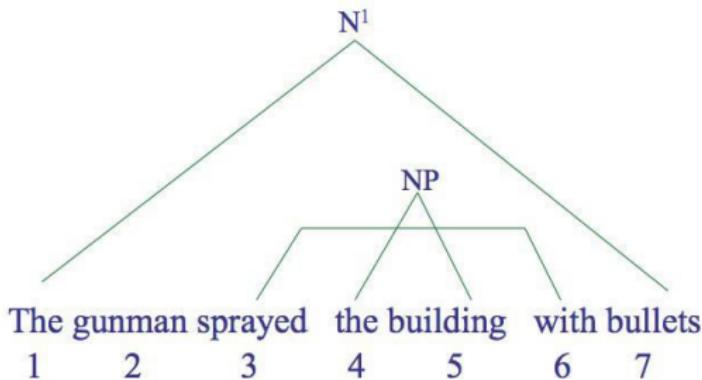


Inside-outside probabilities

$\alpha_{NP}(4,5)$ for "the building"

$= P(\text{The gunman sprayed}, NP_{4,5}, \text{with bullets} | G)$

$\beta_{NP}(4,5)$ for "the building" $= P(\text{the building} | NP_{4,5}, G)$



Inside Probabilities: Base Step

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

Base case

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

Base case for pre-terminals only

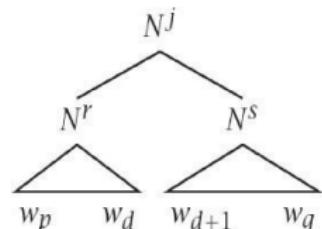
E.g., suppose $N^j = NN$ is being considered and $NN \rightarrow building$ is one of the rules with probability 0.5

$$\beta_{NN}(5, 5) = P(building | NN_{5,5}, G) = P(NN_{5,5} \rightarrow building | G)$$

Inside Probabilities: Induction Step

Assuming Chomsky Normal Form, the first rule must be of the form $N^j \rightarrow N^r N^s$

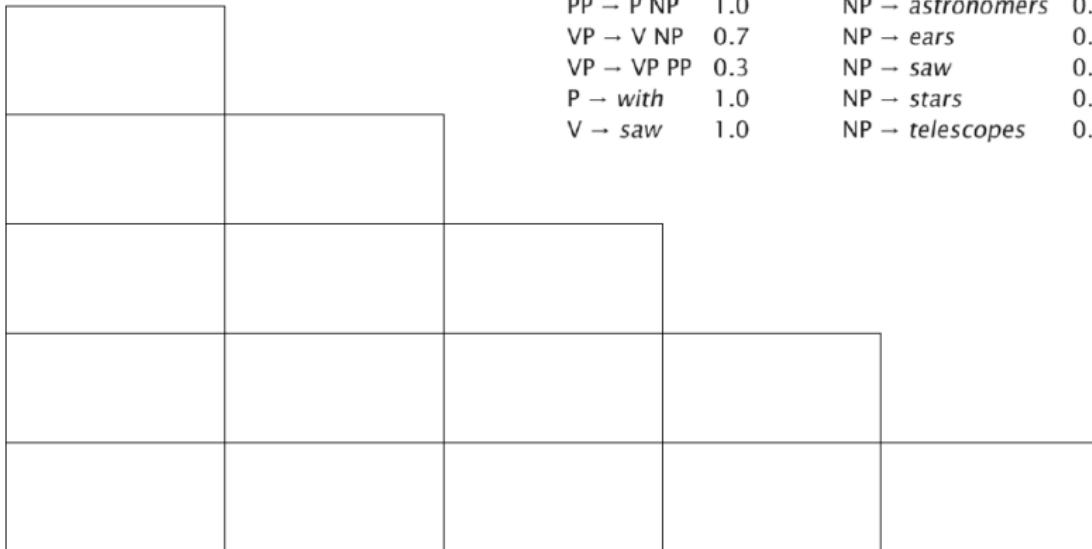
$$\theta_j(p, q) = \sum_{r,s} \sum_{d=p} P(N^j \rightarrow N^r N^s) \theta_r(p, d) \theta_s(d+1, q)$$



- Consider different splits of the words - indicated by d
E.g., *the huge building*
- Consider different non-terminals to be used in the rule:
E.g., $NP \rightarrow DT\ NN$, $NP \rightarrow DT\ NNS$

Calculation of inside probabilities

$S \rightarrow NP VP$	1.0	$NP \rightarrow NP PP$	0.4
$PP \rightarrow P NP$	1.0	$NP \rightarrow \text{astronomers}$	0.1
$VP \rightarrow V NP$	0.7	$NP \rightarrow \text{ears}$	0.18
$VP \rightarrow VP PP$	0.3	$NP \rightarrow \text{saw}$	0.04
$P \rightarrow \text{with}$	1.0	$NP \rightarrow \text{stars}$	0.18
$V \rightarrow \text{saw}$	1.0	$NP \rightarrow \text{telescopes}$	0.1



Calculation of inside probabilities

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.0015876$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	<i>astronomers</i>	<i>saw</i>	<i>stars</i>	<i>with</i>	<i>ears</i>

Outside Probabilities

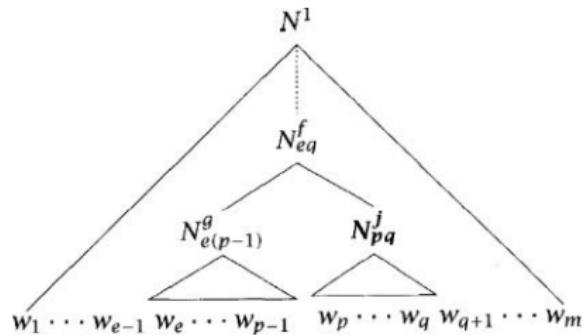
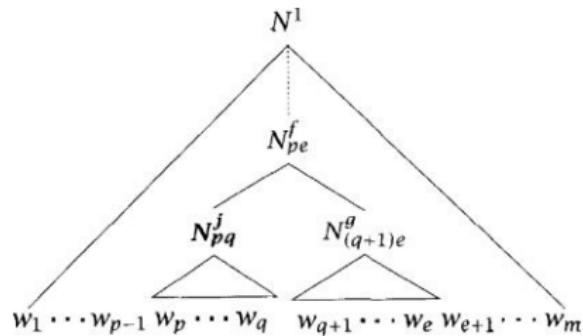
Compute top-down (after inside probabilities)

Base Case

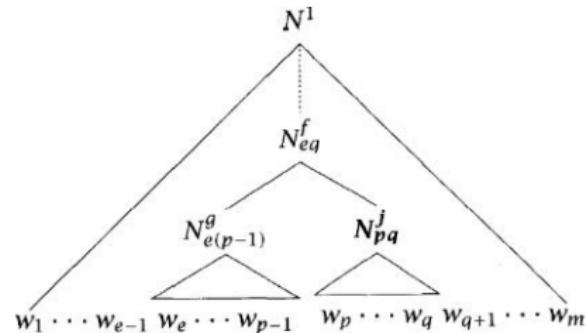
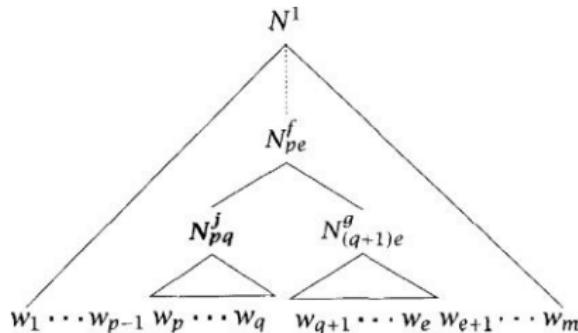
$$\alpha_1(1, m) = 1$$

$$\alpha_j(1, m) = 0, j \neq 1$$

Induction



Outside Probabilities: Induction



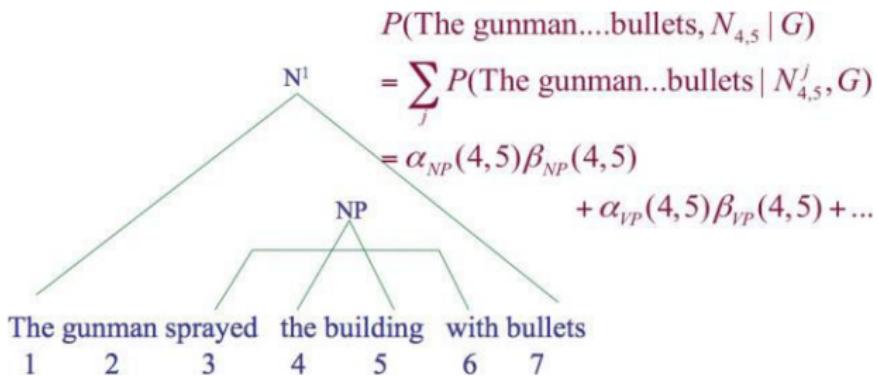
$$\begin{aligned}
 \alpha_j(p, q) &= \sum_{f,g} \alpha_f(p, e) P(N^f \rightarrow N^j N^g) \beta_g(q + 1, e) \\
 &\quad + \sum_{f,g} \alpha_f(e, q) P(N^f \rightarrow N^g N^j) \beta_g(e, p - 1)
 \end{aligned}$$

Product of inside-outside probabilities

$$\alpha_j(p, q)\beta_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} | G)P(w_{pq} | N_{pq}^j, G) = P(w_{1m}, N_{pq}^j | G)$$

The probability of the sentence and that there is some consistent spanning from word p to q is given by:

$$P(w_{1m}, N_{pq} | G) = \sum \alpha_j(p, q)\beta_j(p, q) = P(N_1 \rightarrow w_{1m}, N_{pq} \rightarrow w_{pq} | G)$$



Inside-outside probabilities

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 5

How to get the rule probabilities

Parsed Training Data

You can count!

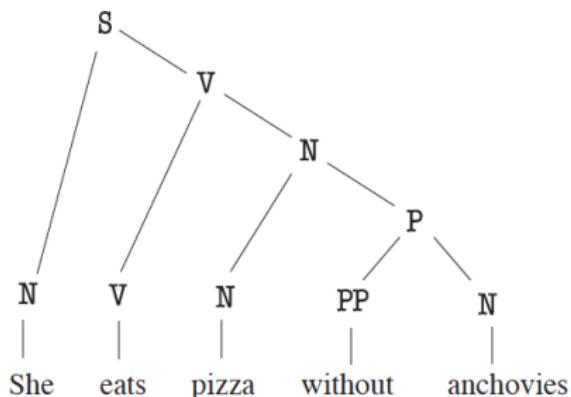
$$\hat{P}(N^j \rightarrow \delta) = \frac{C(N^j \rightarrow \delta)}{\gamma C(N^j \rightarrow \gamma)}$$

But what if the training data is not available?

i.e. gold standard parse is not known.

- Underlying CFG is known and we are given a set of sentences
- For each sentence, we can find out all the possible parses
- *Maximize the likelihood of the sentences in the data under the PCFG constraints*

Example data



Rules of the form $A \rightarrow BC$

$$S \rightarrow NV$$

$$V \rightarrow VN$$

$$N \rightarrow NP$$

$$P \rightarrow PP\ N.$$

Rules of the form $A \rightarrow w$

$$N \rightarrow \text{She}$$

$$V \rightarrow \text{eats}$$

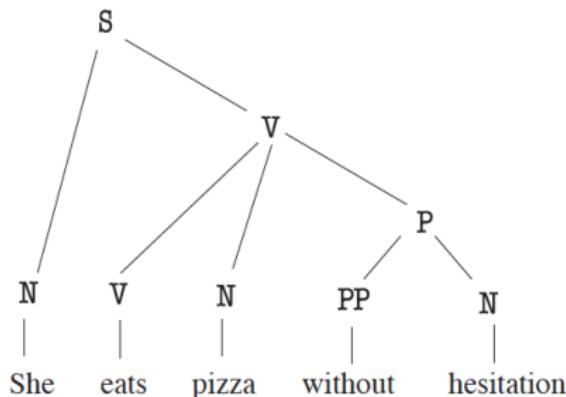
$$N \rightarrow \text{pizza}$$

$$PP \rightarrow \text{without}$$

$$N \rightarrow \text{anchovies}.$$

Example data

Is any other parse possible for *She eats pizza without anchovies* syntactically?
Consider *She eats pizza without hesitation*



New Context-free rules:

$$V \rightarrow V \ N \ P$$

$$N \rightarrow \text{hesitation}.$$

Estimating the model parameters

We need to find probabilities such as

- $\varphi(S \rightarrow N\ V)$
- $\varphi(N \rightarrow \text{pizza})$

Requirements

For each non-terminal A , the derivation probabilities sum up to 1

$$\sum_{\alpha} \varphi(A \rightarrow \alpha) = 1$$

For the example grammar:

$$\begin{aligned}\varphi(N \rightarrow N\ P) + \varphi(N \rightarrow \text{pizza}) + \varphi(N \rightarrow \text{anchovies}) &+ \\ + \varphi(N \rightarrow \text{hesitation}) + \varphi(N \rightarrow \text{She}) &= 1 \\ \varphi(V \rightarrow V\ N) + \varphi(V \rightarrow V\ N\ P) + \varphi(V \rightarrow \text{eats}) &= 1\end{aligned}$$

$$\begin{aligned}\varphi(S \rightarrow N\ V) &= 1 \\ \varphi(P \rightarrow PP\ N) &= 1 \\ \varphi(PP \rightarrow \text{without}) &= 1\end{aligned}$$

Likelihood computation

W_1 = “She eats pizza without anchovies”

W_2 = “She eats pizza without hesitation”.

$$\begin{aligned} P_\phi(W_1, T_1) &= \phi(S \rightarrow N\ V)\ \phi(V \rightarrow V\ N)\ \phi(N \rightarrow N\ P) \times \\ &\quad \times\ \phi(P \rightarrow PP\ N)\ \phi(N \rightarrow She)\ \phi(V \rightarrow eats) \times \\ &\quad \times\ \phi(N \rightarrow pizza)\ \phi(PP \rightarrow without)\ \phi(N \rightarrow anchovies) \end{aligned}$$

$$\begin{aligned} P_\phi(W_2, T_1) &= \phi(S \rightarrow N\ V)\ \phi(V \rightarrow V\ N\ P)\ \phi(P \rightarrow P\ PP) \times \\ &\quad \times\ \phi(N \rightarrow She)\ \phi(V \rightarrow eats)\ \phi(N \rightarrow pizza) \times \\ &\quad \times\ \phi(PP \rightarrow without)\ \phi(N \rightarrow hesitation) \end{aligned}$$

Likelihood computation

$$\begin{aligned} P_\phi(W_1, T_2) = & \phi(S \rightarrow N\ V)\ \phi(V \rightarrow V\ N\ P)\ \phi(P \rightarrow P\ PP) \times \\ & \times \phi(N \rightarrow \text{She})\ \phi(V \rightarrow \text{eats})\ \phi(N \rightarrow \text{pizza}) \times \\ & \times \phi(PP \rightarrow \text{without})\ \phi(N \rightarrow \text{anchovies}) \end{aligned}$$

$$\begin{aligned} P_\phi(W_2, T_1) = & \phi(S \rightarrow N\ V)\ \phi(V \rightarrow V\ N)\ \phi(N \rightarrow N\ P) \times \\ & \times \phi(P \rightarrow PP\ N)\ \phi(N \rightarrow \text{She})\ \phi(V \rightarrow \text{eats}) \times \\ & \times \phi(N \rightarrow \text{pizza})\ \phi(PP \rightarrow \text{without})\ \phi(N \rightarrow \text{hesitation}) \end{aligned}$$

Likelihood of the corpus

Probability of a sentence W : $P_\phi(W) = \prod_{t=1}^T P_\phi(W_t, T_t)$

If the training data comprises of sentences W_1, W_2, \dots, W_N , then the likelihood is

$$L(\varphi) = P_\varphi(W_1)P_\varphi(W_2) \cdots P_\varphi(W_N)$$

Likelihood maximization

Approach

Starting at some initial parameters φ , re-estimate to obtain new parameters φ^j for which $L(\varphi^j) \geq L(\varphi)$. Repeat until convergence

Parameter Estimation

Given some rule probabilities φ and training corpus $W_1, W_2 \dots W_n$, the new parameters are obtained as:

$$\phi'(\text{A} \rightarrow \text{B C}) = \frac{\text{count}(\text{A} \rightarrow \text{B C})}{\sum_{\alpha} \text{count}(\text{A} \rightarrow \alpha)}$$

$$\phi'(\text{A} \rightarrow \text{w}) = \frac{\text{count}(\text{A} \rightarrow \text{w})}{\sum_{\alpha} \text{count}(\text{A} \rightarrow \alpha)}$$

What is $\text{count}(.)$?

$$\text{count}(\text{A} \rightarrow \text{B C}) = \sum_{i=1}^N c_{\varphi}(\text{A} \rightarrow \text{B C}, W_i)$$

$$\text{count}(\text{A} \rightarrow \text{w}) = \sum_{i=1}^N c_{\varphi}(\text{A} \rightarrow \text{w}, W_i)$$

$c_{\varphi}(A \rightarrow \alpha, W_i)$ is the expected number of times $(A \rightarrow \alpha)$ is used in generating the sentence W_i , when the rule probabilities are given by φ

Computing Expected counts

Inside probabilities

The nonterminal A derives the string of words $w_i, \dots w_j$ in the sentence :

$$\beta_{ij}(A) = P_\varphi(A \Rightarrow^* w_i \dots w_j)$$

Outside probabilities

Beginning with the start symbol S we can derive the string

$$w_1 \dots w_{i-1} A w_{j+1} \dots w_n : \alpha_{ij}(A) = P_\varphi(S \Rightarrow^* w_1 \dots w_{i-1} A w_{j+1} \dots w_n)$$

Expected count

$$c_\varphi(A \rightarrow BC, W) = \frac{\varphi(A \rightarrow BC)}{P_\varphi(W)} \sum_{1 \leq i \leq j \leq k \leq n} \alpha_{ik}(A) \beta_{ij}(B) \beta_{j+1,k}(C)$$
$$c_\varphi(A \rightarrow w, W) = \frac{\varphi(A \rightarrow w)}{P_\varphi(W)} \sum_{1 \leq i \leq n} \alpha_{ii}(A)$$

And how to compute inside-outside probabilities

Inductively, as discussed earlier

$$\beta_{ii}(A) = \varphi(A \rightarrow w_i)$$

$$\alpha_{1n}(S) = 1$$

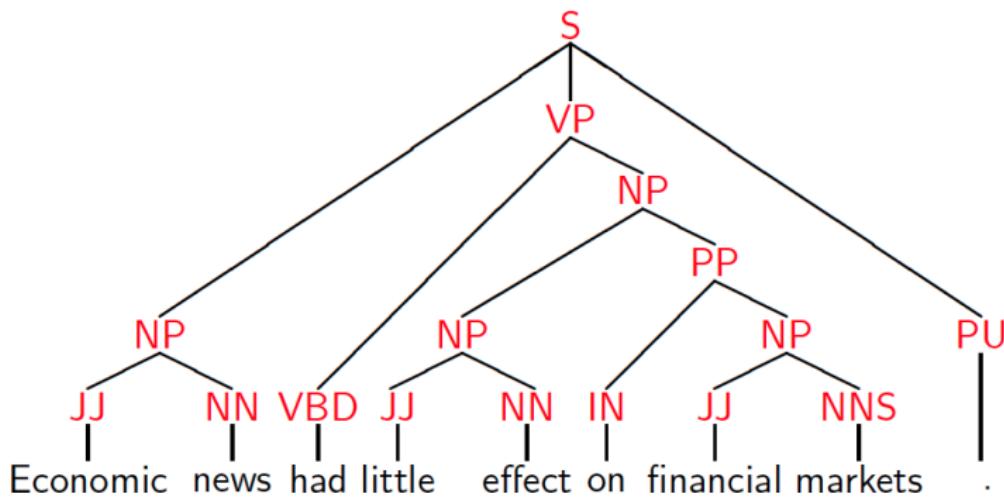
Dependency Grammars and Parsing - Introduction

Pawan Goyal

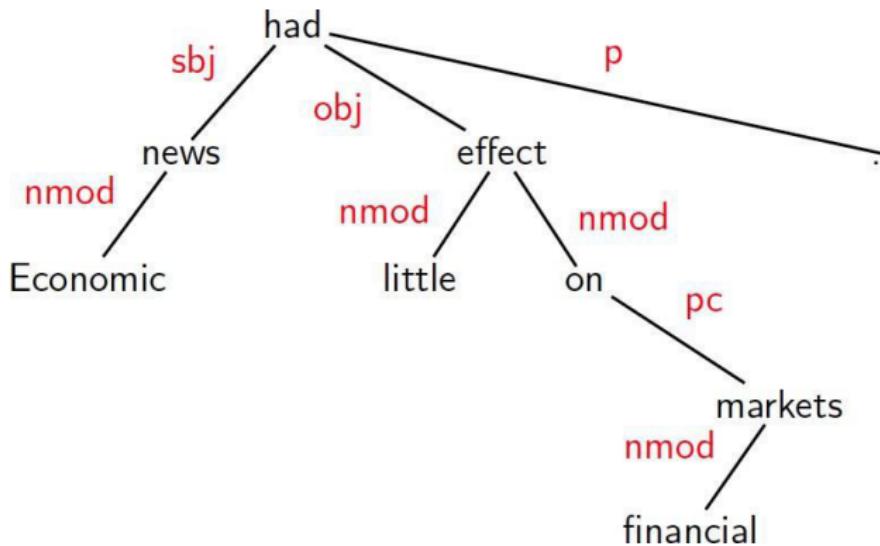
CSE, IIT Kharagpur

Week 6, Lecture 1

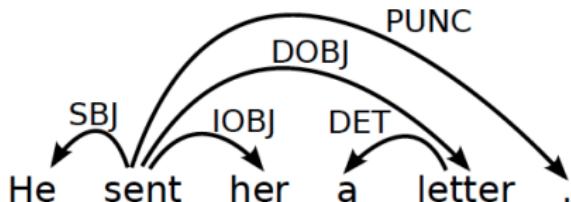
Phrase Structure



Dependency Structure Representation

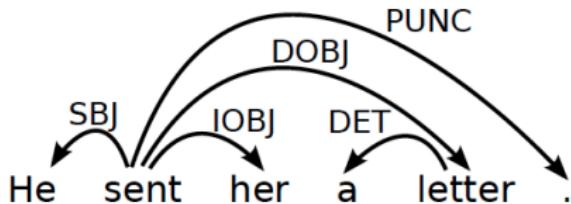


Dependency Structure



- Connects the words in a sentence by putting arrows between the words.
- Arrows show relations between the words and are typed by some grammatical relations.
- Arrows connect a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate).
- Usually dependencies form a tree.

Criteria for Heads and Dependents

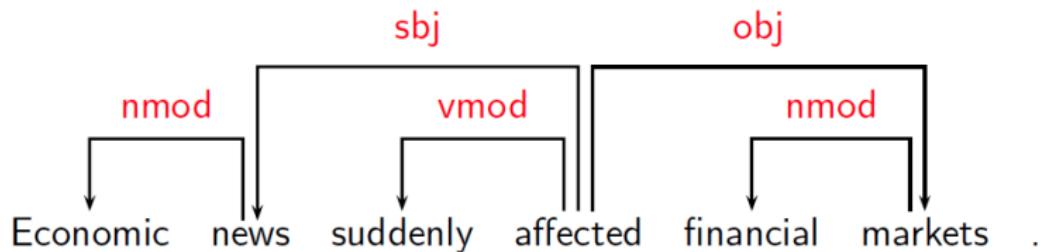


Criteria for a syntactic relation between a head H and a dependent D in a construction C

- H determines the syntactic category of C ; H can replace C .
- D specifies H .
- H is obligatory; D may be optional.
- H selects D and determines whether D is obligatory.
- The form of D depends on H (agreement or government).
- The linear position of D is specified with reference to H .

Some Clear Cases

Construction	Head	Dependent
Exocentric	Verb	Subject (sbj)
	Verb	Object (obj)
Endocentric	Verb	Adverbial (vmod)
	Noun	Attribute (nmod)



Comparison

Phrase structures explicitly represent

- Phrases (nonterminal nodes)
- Structural categories (nonterminal labels)

Dependency structures explicitly represent

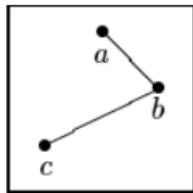
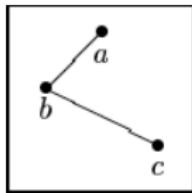
- Head-dependent relations (directed arcs)
- Functional categories (arc labels)

Dependency Graphs

- A dependency structure can be defined as a directed graph G , consisting of
 - a set V of nodes,
 - a set A of arcs (edges),
- Labeled graphs:
 - Nodes in V are labeled with word forms (and annotation).
 - Arcs in A are labeled with dependency types.
- Notational convention:
 - Arc (w_i, d, w_j) links head w_i to dependent w_j with label d
 - $w_i \xrightarrow{d} w_j \Leftrightarrow (w_i, d, w_j) \in A$
 - $i \rightarrow j \equiv (i, j) \in A$
 - $i \rightarrow *j \equiv i = j \vee \exists k : i \rightarrow k, k \rightarrow *j$

Formal conditions on Dependency Graphs

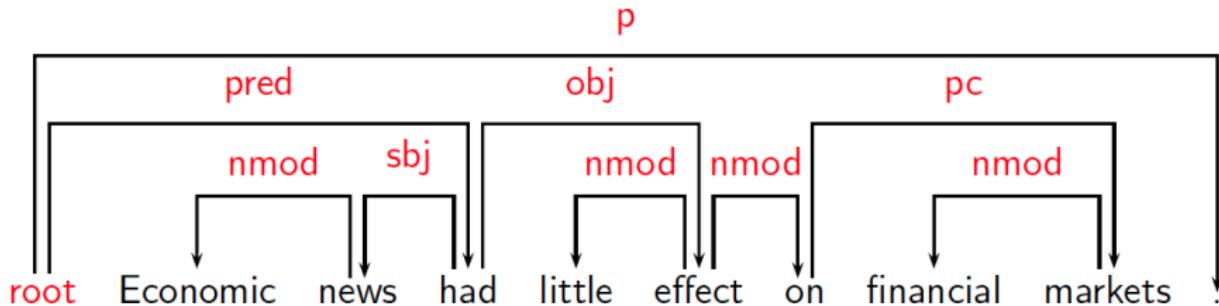
- G is connected:
 - ▶ For every node i there is a node j such that $i \rightarrow j$ or $j \rightarrow i$.
- G is acyclic:
 - ▶ if $i \rightarrow j$ then not $j \rightarrow *i$.
- G obeys the single head constraint:
 - ▶ if $i \rightarrow j$ then not $k \rightarrow j$, for any $k \neq i$.
- G is projective:
 - ▶ if $i \rightarrow j$ then $j \rightarrow *k$, for any k such that both j and k lie on the same side of i .



Formal Conditions: Basic Intuitions

Connectedness, Acyclicity and Single-Head

- **Connectedness:** Syntactic structure is complete.
- **Acyclicity:** Syntactic structure is hierarchical.
- **Single-Head:** Every word has at most one syntactic head.
- **Projectivity:** No crossing of dependencies.



Dependency Parsing

Dependency Parsing

- **Input:** Sentence $x = w_1, \dots, w_n$
- **Output:** Dependency graph G

Parsing Methods

- Deterministic Parsing
- Maximum Spanning Tree Based
- Constraint Propagation Based

Transition Based Parsing: Formulation

Pawan Goyal

CSE, IIT Kharagpur

Week 6, Lecture 2

Deterministic Parsing

Basic idea

Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions

Configurations

A parser configuration is a triple $c = (S, B, A)$, where

- S : a stack $[..., w_i]_S$ of partially processed words,
- B : a buffer $[w_j, ...]_B$ of remaining input words,
- A : a set of labeled arcs (w_i, d, w_j) .

Stack

$[\text{sent, her, a}]_S$

Buffer

$[\text{letter, .}]_B$

Arcs

$\text{He} \xleftarrow{\text{SBJ}} \text{sent}$

Transition System

A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$, where

- C is a set of configurations,
- T is a set of transitions, such that $t: C \rightarrow C$,
- c_s is an initialization function
- $C_t \subseteq C$ is a set of terminal configurations.

A transition sequence for a sentence x is a set of configurations $C_{0,m} = (c_0, c_1, \dots, c_m)$ such that
 $c_0 = c_s(x)$, $c_m \in C_t$, $c_i = t(c_{i-1})$ for some $t \in T$

Initialization: $([], [w_1, \dots, w_n]_B, \{\})$

Termination: $(S, [], A)$

Transitions for Arc-Eager Parsing

$$\text{Left-Arc}(\mathbf{d}) \frac{([\dots, \mathbf{w}_i]_S, [\mathbf{w}_j, \dots]_B, A)}{([\dots]_S, [\mathbf{w}_j, \dots]_B, A \cup \{(\mathbf{w}_j, \mathbf{d}, \mathbf{w}_i)\})} \neg \text{HEAD}(\mathbf{w}_i)$$

$$\text{Right-Arc}(\mathbf{d}) \frac{([\dots, \mathbf{w}_i]_S, [\mathbf{w}_j, \dots]_B, A)}{([\dots, \mathbf{w}_i, \mathbf{w}_j]_S, [\dots]_B, A \cup \{(\mathbf{w}_i, \mathbf{d}, \mathbf{w}_j)\})}$$

$$\text{Reduce} \frac{([\dots, \mathbf{w}_i]_S, B, A)}{([\dots]_S, B, A)} \text{HEAD}(\mathbf{w}_i)$$

$$\text{Shift} \frac{([\dots]_S, [\mathbf{w}_i, \dots]_B, A)}{([\dots, \mathbf{w}_i]_S, [\dots]_B, A)}$$

Example

Parse Example

Transitions:

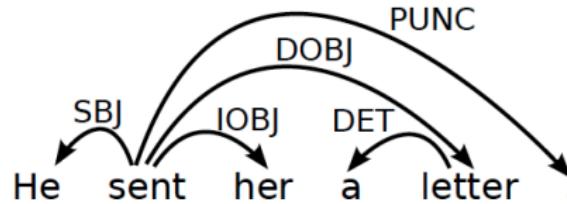
Stack

[]_s

Buffer

[He, sent, her, a, letter, .]__B

Arcs



Example

Parse Example

Transitions: SH

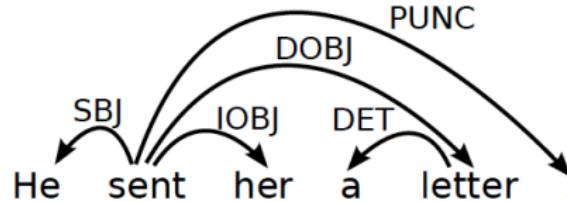
Stack

[He]_S

Buffer

[sent, her, a, letter, .]_B

Arcs



Example

Parse Example

Transitions: SH-LA

Stack

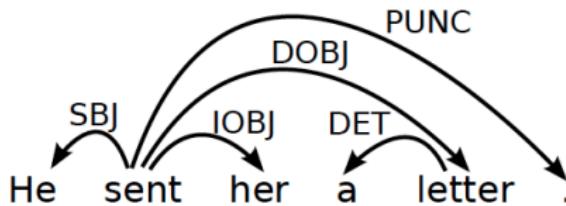
[]s

Buffer

[sent, her, a, letter, .]B

Arcs

He $\xleftarrow{\text{SBJ}}$ sent

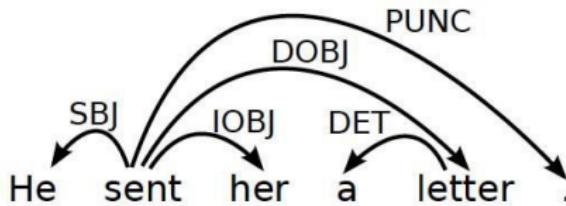


Example

Parse Example

Transitions: SH-LA-SH

Stack	Buffer	Arcs
[sent] _S	[her, a, letter, .] _B	He $\xleftarrow{\text{SBJ}}$ sent

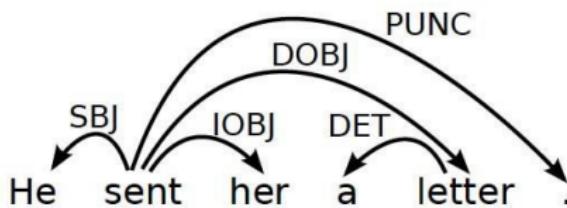


Example

Parse Example

Transitions: SH-LA-SH-RA

Stack	Buffer	Arcs
[sent, her] _S	[a, letter, .] _B	He ← ^{SBJ} sent sent → ^{IOBJ} her



Example

Parse Example

Transitions: SH-LA-SH-RA-SH

Stack

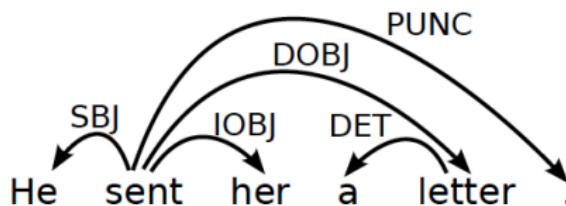
[sent, her, a]_S

Buffer

[letter, .]_B

Arcs

He $\xleftarrow{\text{SBJ}}$ sent
sent $\xrightarrow{\text{IOBJ}}$ her



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA

Stack

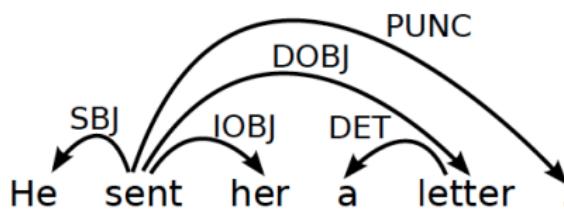
[sent, her]_S

Buffer

[letter, .]_B

Arcs

He $\xleftarrow{\text{SBJ}}$ sent
sent $\xrightarrow{\text{IOBJ}}$ her
a $\xleftarrow{\text{DET}}$ letter



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE

Stack

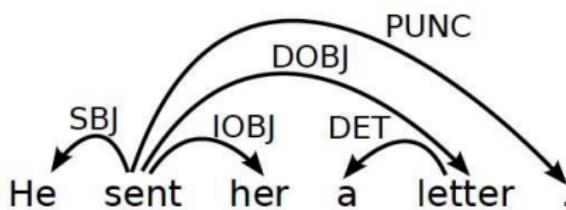
[sent]_S

Buffer

[letter, .]_B

Arcs

He $\xleftarrow{\text{SBJ}}$ sent
sent $\xrightarrow{\text{IOBJ}}$ her
a $\xleftarrow{\text{DET}}$ letter



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA

Stack

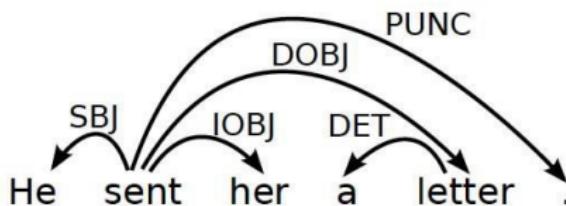
[sent, letter]_S

Buffer

[.]_B

Arcs

He $\xleftarrow{\text{SBJ}}$ sent
sent $\xrightarrow{\text{IOBJ}}$ her
a $\xleftarrow{\text{DET}}$ letter
sent $\xrightarrow{\text{DOBJ}}$ letter



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA-RE

Stack

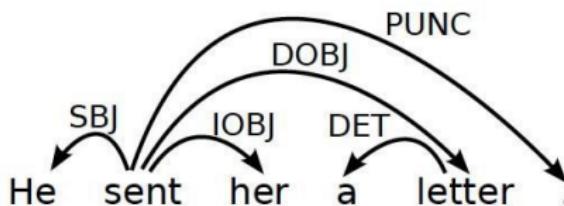
$[\text{sent}]_S$

Buffer

$[.]_B$

Arcs

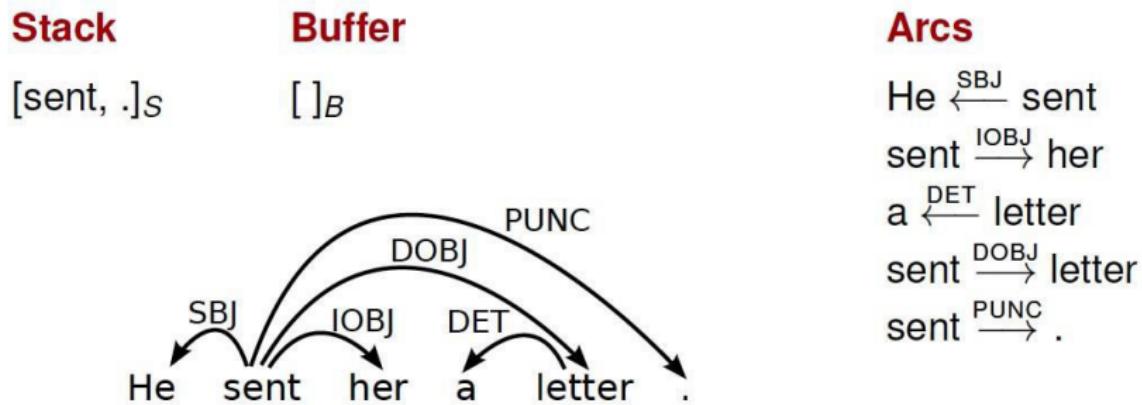
$\text{He} \xleftarrow{\text{SBJ}} \text{sent}$
 $\text{sent} \xrightarrow{\text{IOBJ}} \text{her}$
 $\text{a} \xleftarrow{\text{DET}} \text{letter}$
 $\text{sent} \xrightarrow{\text{DOBJ}} \text{letter}$



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA-RE-RA



Transition Based Parsing: Learning

Pawan Goyal

CSE, IIT Kharagpur

Week 6, Lecture 3

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle**.
- An oracle can be approximated by a **classifier**.
- A classifier can be trained using **treebank** data.

Learning Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.

Three issues

- How to represent configurations by feature vectors?
- How to derive training data from treebanks?
- How to learn classifiers?

Feature Models

A feature representation $f(c)$ of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

■ Nodes:

- ▶ Target nodes (top of S , head of B)
- ▶ Linear context (neighbors in S and B)
- ▶ Structural context (parents, children, siblings in G)

■ Attributes:

- ▶ Word form (and lemma)
- ▶ Part-of-speech (and morpho-syntactic features)
- ▶ Dependency type (if labeled)
- ▶ Distance (between target tokens)

Deterministic Parsing

To guide the parser, a linear classifier can be used:

$$t^* = \arg \max_t w.f(c, t)$$

Weight vector w learned from treebank data.

Using classifier at run-time

```
PARSE( $w_1, \dots, w_n$ )
1    $c \leftarrow ([]_S, [w_1, \dots, w_n]_B, \{\})$ 
2   while  $B_c \neq []$ 
3        $t^* \leftarrow \arg \max_t w.f(c, t)$ 
4        $c \leftarrow t^*(c)$ 
5   return  $T = (\{w_1, \dots, w_n\}, A_c)$ 
```

Training data

- Training instances have the form $(f(c), t)$, where
 - $f(c)$ is a feature representation of a configuration c ,
 - t is the correct transition out of c (i.e., $o(c) = t$).
- Given a dependency treebank, we can sample the oracle function o as follows:
 - For each sentence x with gold standard dependency graph G_x , construct a transition sequence $C_{0,m} = (c_0, c_1, \dots, c_m)$ such that
$$c_0 = c_s(x), \\ G_{c_m} = G_x$$
 - For each configuration $c_i (i < m)$, we construct a training instance $(f(c_i), t_i)$, where $t_i(c_i) = c_{i+1}$.

Standard Oracle for Arc-Eager Parsing

$o(c, T) =$

- **Left-Arc** if $\text{top}(S_c) \leftarrow \text{first}(B_c)$ in T
- **Right-Arc** if $\text{top}(S_c) \rightarrow \text{first}(B_c)$ in T
- **Reduce** if $\exists w < \text{top}(S_c) : w \leftrightarrow \text{first}(B_c)$ in T
- **Shift** otherwise

Online Learning with an Oracle

```
LEARN({ $T_1, \dots, T_N\}$ )
1    $w \leftarrow 0.0$ 
2   for  $i$  in  $1..K$ 
3     for  $j$  in  $1..N$ 
4        $c \leftarrow ([]_S, [w_1, \dots, w_{n_j}]_B, \{\})$ 
5       while  $B_c \neq []$ 
6          $t^* \leftarrow \arg \max_t w.f(c, t)$ 
7          $t_o \leftarrow o(c, T_i)$ 
8         if  $t^* \neq t_o$ 
9            $w \leftarrow w + f(c, t_o) - f(c, t^*)$ 
10         $c \leftarrow t_o(c)$ 
11    return  $w$ 
```

Oracle $o(c, T_i)$ returns the optimal transition of c and T_i

Example

Consider the sentence, ‘John saw Mary’.

- Draw a dependency graph for this sentence.
- Assume that you are learning a classifier for the data-driven deterministic parsing and the above sentence is a gold-standard parse in your training data. You are also given that *John* and *Mary* are ‘Nouns’, while the POS tag of *saw* is ‘Verb’. Assume that your features correspond to the following conditions:
 - ▶ The stack is empty
 - ▶ Top of stack is Noun and Top of buffer is Verb
 - ▶ Top of stack is Verb and Top of buffer is Noun

Initialize the weights of all your features to 5.0, except that in all of the above cases, you give a weight of 5.5 to *Left-Arc*. Define your feature vector and the initial weight vector.

- Use this gold standard parse during online learning and report the weights after completing one full iteration of Arc-Eager parsing over this sentence.

MST-based Dependency Parsing

Pawan Goyal

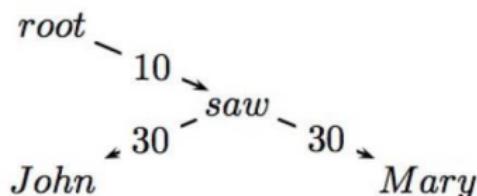
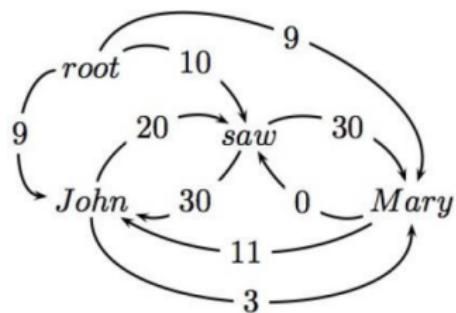
CSE, IIT Kharagpur

Week 6, Lecture 4

Maximum Spanning Tree Based

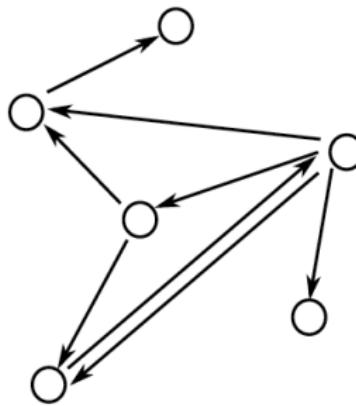
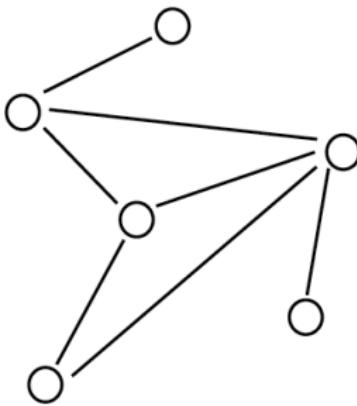
Basic Idea

Starting from all possible connections, find the maximum spanning tree.



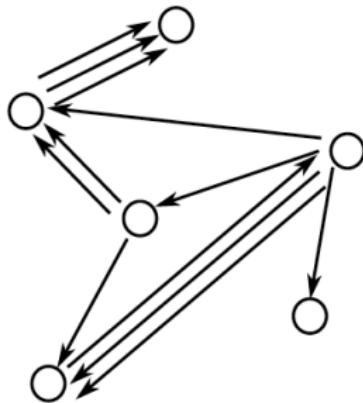
Some Graph Theory Reminders

- A graph $G = (V, A)$ is a set of vertices V and arcs $(i, j) \in A$ where $i, j \in V$.
- Undirected graphs: $(i, j) \in A \Leftrightarrow (j, i) \in A$
- Directed graphs (digraphs): $(i, j) \in A$ & $(j, i) \in A$



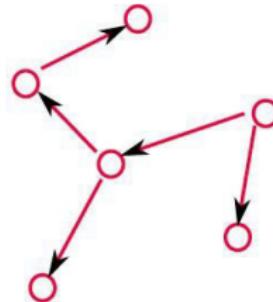
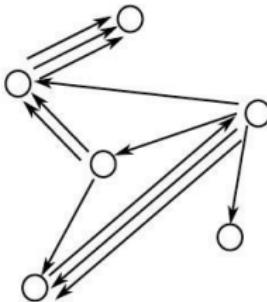
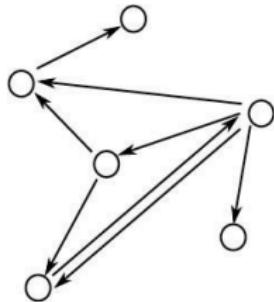
Multi-Digraphs

- A multi-digraph is a digraph where multiple arcs between vertices are possible
- $(i, j, k) \in A$ represents the k^{th} arc from vertex i to vertex j .



Directed Spanning Trees

- A directed spanning tree of a (multi-)digraph $G = (V, A)$ is a subgraph $G^J = (V^J, A^J)$ such that:
 - $V^J = V$
 - $A^J \subseteq A$, and $|A^J| = |V^J| - 1$
 - G^J is a tree (acyclic)
- A spanning tree of the following (multi-)digraphs



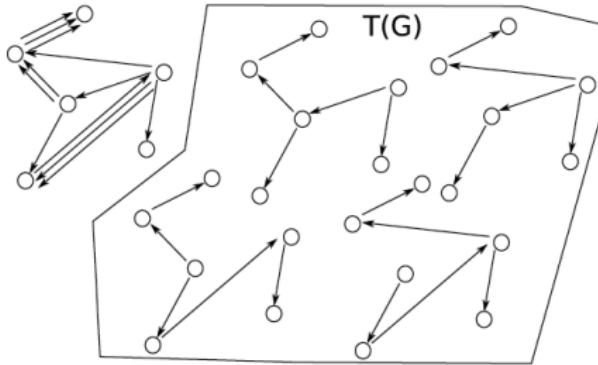
Weighted Directed Spanning Trees

- Assume we have a weight function for each arc in a multi-digraph $G = (V, A)$.
- Define $w_{ij}^k \geq 0$ to be the weight of $(i, j, k) \in A$ for a multi-digraph
- Define the weight of directed spanning tree G^J of graph G as

$$w(G^J) = \prod_{(i,j,k) \in G^J} w_{ij}^k$$

Maximum Spanning Trees (MST)

Let $T(G)$ be the set of all spanning trees for graph G



The MST problem

Find the spanning tree G^J of the graph G that has the highest weight
X

$$G^J = \arg \max_{G^J \in T(G)} w(G^J) = \arg \max_{G^J \in T(G)} \sum_{(i,j,k) \in G^J} w_{ij}^k$$

Finding MST

Directed Graph

For each sentence x , define the directed graph $G_x = (V_x, E_x)$ given by

$$V_x = \{x_0 = \text{root}, x_1, \dots, x_n\}$$

$$E_x = \{(i, j) : i \neq j, (i, j) \in [0 : n] \times [1 : n]\}$$

G_x is a graph with

- the sentence words and the dummy root symbol as vertices and
- a directed edge between every pair of distinct words and
- a directed edge from the root symbol to every word

Chu-Liu-Edmonds Algorithm

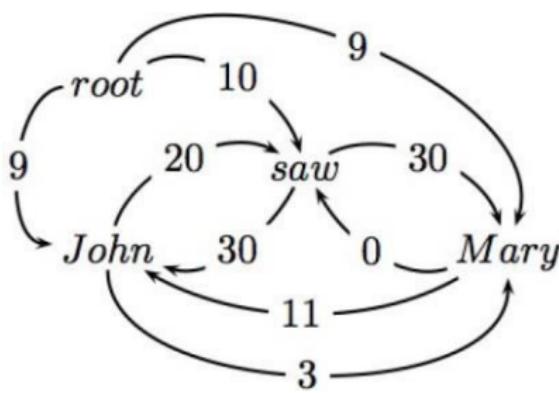
Chu-Liu-Edmonds Algorithm

- Each vertex in the graph greedily selects the incoming edge with the highest weight.
- If a tree results, it must be a maximum spanning tree.
- If not, there must be a cycle.
 - ▶ Identify the cycle and contract it into a single vertex.
 - ▶ Recalculate edge weights going into and out of the cycle.

Chu-Liu-Edmonds Algorithm

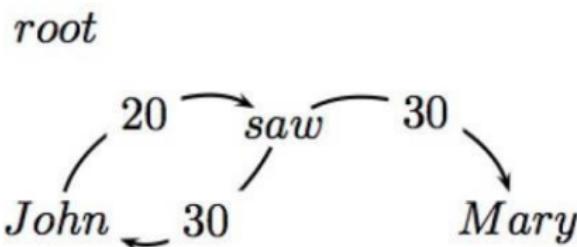
$x = \text{John saw Mary}$

- Build the directed graph



Chu-Liu-Edmonds Algorithm

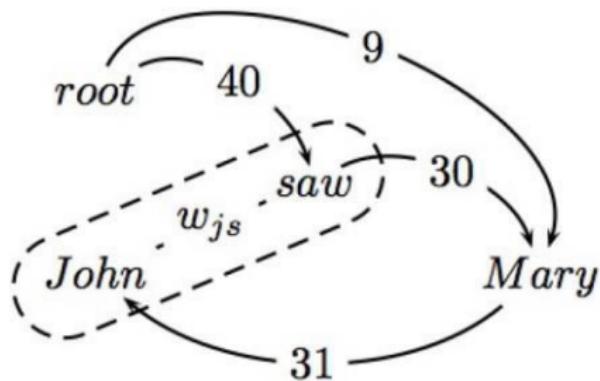
- Find the highest scoring incoming arc for each vertex



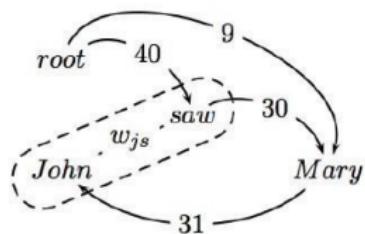
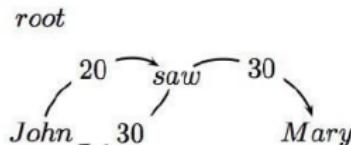
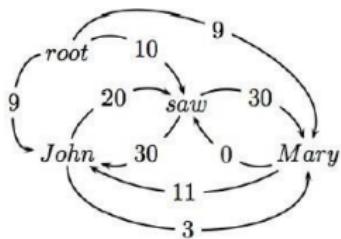
- If this is a tree, then we have found MST.

Chu-Liu-Edmonds Algorithm

- If not a tree, identify cycle and contract
- Recalculate arc weights into and out-of cycle



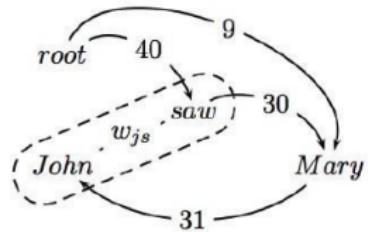
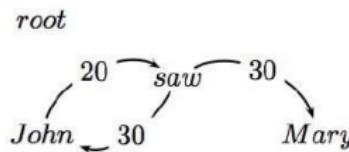
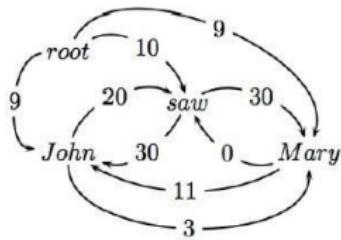
Chu-Liu-Edmonds Algorithm



Outgoing arc weights

- Equal to the max of outgoing arc over all vertices in cycle
- e.g., $\text{John} \rightarrow \text{Mary}$ is 3 and $\text{saw} \rightarrow \text{Mary}$ is 30.

Chu-Liu-Edmonds Algorithm

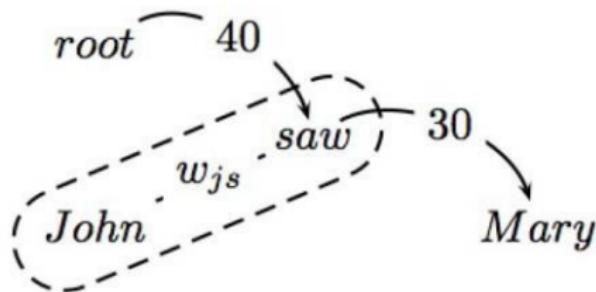


Incoming arc weights

- Equal to the weight of best spanning tree that includes head of incoming arc and all nodes in cycle
- root → saw → John is 40
- root → John → saw is 29

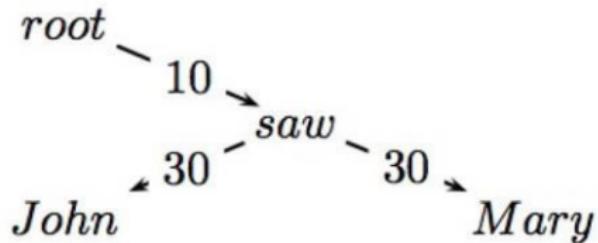
Chu-Liu-Edmonds Algorithm

Calling the algorithm again on the contracted graph:



- This is a tree and the MST for the contracted graph
- Go back up the recursive call and reconstruct final graph

Chu-Liu-Edmonds Algorithm



- The edge from w_{js} to *Mary* was from *saw*
- The edge from *root* to w_{js} represented a tree from *root* to *saw* to *John*.

MST-based Dependency Parsing: Learning

Pawan Goyal

CSE, IIT Kharagpur

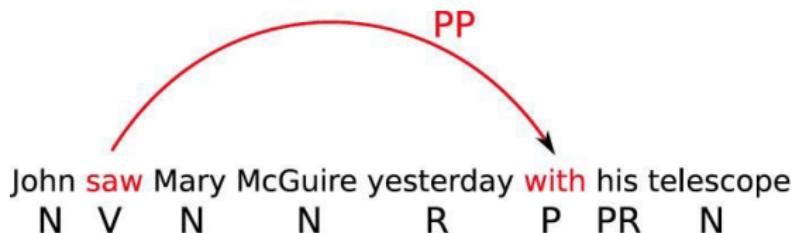
Week 6, Lecture 5

Arc weights as linear classifiers

$$w_{ij}^k = w.f(i,j,k)$$

- Arc weights are a linear combination of features of the arc $f(i,j,k)$ and a corresponding weight vector w
- What arc features?

Arc Features $f(i,j,k)$

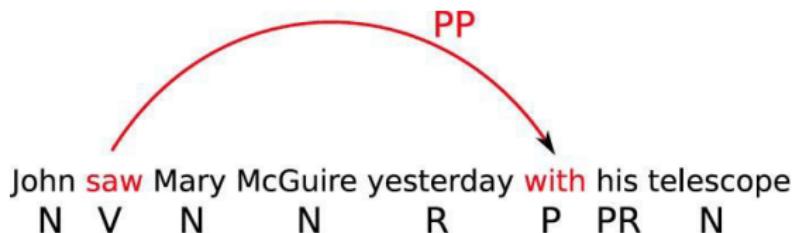


Features

Identities of the words w_i and w_j for a label l_k

head = saw & dependent=with

Arc Features $f(i,j,k)$



Features

Part-of-speech tags of the words w_i and w_j for a label l_k

head-pos = Verb & dependent-pos=Preposition

Arc Features $f(i,j,k)$



Features

Part-of-speech of words surrounding and between w_i and w_j

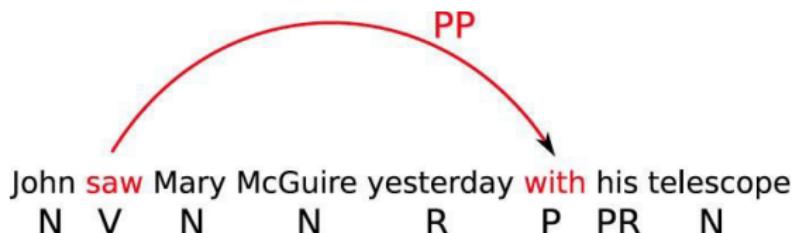
inbetween-pos = Noun

inbetween-pos = Adverb

dependent-pos-right = Pronoun

head-pos-left=Noun

Arc Features $f(i,j,k)$



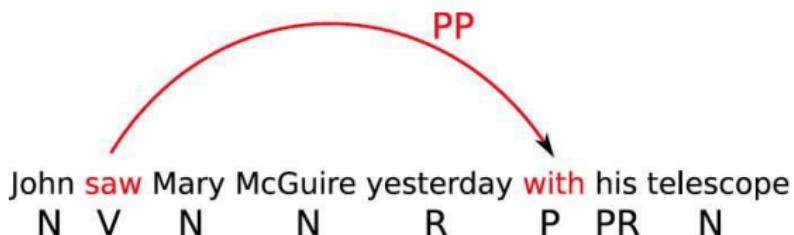
Features

Number of words between w_i and w_j , and their orientation

arc-distance = 3

arc-direction = right

Arc Features $f(i,j,k)$



Features

- Combinations
 - head-pos=Verb & dependent-pos=Preposition & arc-label=PP
 - head-pos=Verb & dependent=with & arc-distance=3
- No limit : any feature over arc (i,j,k) or input x

Learning the parameters

- Re-write the inference problem

$$\begin{aligned} G &= \arg \max_{G \in T(G_x)} w_{ij}^k \\ &= \arg \max_{G \in T(G_x)} w \cdot \sum_{(i,j,k) \in G} f(i,j,k) \\ &= \arg \max_{G \in T(G_x)} w \cdot f(G) \end{aligned}$$

- Which can be plugged into learning algorithms

Inference-based Learning

Training data: $T = \{(x_t, G_t)\}_{t=1}^{|T|}$

1. $w^{(0)} = 0; i = 0$
2. for $n : 1..N$
3. for $t : 1..|T|$
4. Let $G^j = argmax_{G^j} w^{(i)}.f(G^j)$
5. if $G^j \neq G_t$
6. $w^{(i+1)} = w^{(i)} + f(G_t) - f(G^j)$
7. $i = i + 1$
8. return w^i

Example

Suppose you are training MST Parser for dependency and the sentence, “John saw Mary” occurs in the training set. Also, for simplicity, assume that there is only one dependency relation, “rel”. Thus, for every arc from word w_i to w_j , your features may be simplified to depend only on words w_i and w_j and not on the relation label.

Below is the set of features

- $f_1: \text{pos}(w_i) = \text{Noun}$ and $\text{pos}(w_j) = \text{Noun}$
- $f_2: \text{pos}(w_i) = \text{Verb}$ and $\text{pos}(w_j) = \text{Noun}$
- $f_3: w_i = \text{Root}$ and $\text{pos}(w_j) = \text{Verb}$
- $f_4: w_i = \text{Root}$ and $\text{pos}(w_j) = \text{Noun}$
- $f_5: w_i = \text{Root}$ and w_j occurs at the end of sentence
- $f_6: w_i$ occurs before w_j in the sentence
- $f_7: \text{pos}(w_i) = \text{Noun}$ and $\text{pos}(w_j) = \text{Verb}$

The feature weights before the start of the iteration are: {3, 20, 15, 12, 1, 10, 20}. Determine the weights after an iteration over this example.



Distributional Semantics - Introduction

Pawan Goyal

CSE, IIT Kharagpur

Week 7, Lecture 1

Introduction

What is Semantics?

The study of meaning: Relation between symbols and their denotata.
John told Mary that the train moved out of the station at 3 o'clock.

Computational Semantics

The study of how to automate the process of constructing and reasoning with meaning representations of natural language expressions.

Methods in Computational Semantics generally fall in two categories:

- **Formal Semantics:** Construction of precise mathematical models of the relations between expressions in a natural language and the world.
John chases a bat → $\exists x[bat(x) \wedge chase(john, x)]$
- **Distributional Semantics:** The study of statistical patterns of human word usage to extract semantics.

Distributional Hypothesis

Distributional Hypothesis: Basic Intuition

“The meaning of a word is its use in language.” (Wittgenstein, 1953)

“You know a word by the company it keeps.” (Firth, 1957)

→ Word meaning (whatever it might be) is reflected in linguistic distributions.

“Words that occur in the same contexts tend to have similar meanings.” (Zellig Harris, 1968)

→ Semantically similar words tend to have similar distributional patterns.

Distributional Semantics: a linguistic perspective

“If linguistics is to deal with meaning, it can only do so through distributional analysis.” (Zellig Harris)

“If we consider words or morphemes A and B to be more different in meaning than A and C, then we will often find that the distributions of A and B are more different than the distributions of A and C. In other words, difference in meaning correlates with difference of distribution.” (Zellig Harris, “Distributional Structure”)

Differential and not referential

Contextual representation

A word's contextual representation is an abstract cognitive structure that accumulates from encounters with the word in various linguistic contexts.

We learn new words based on contextual cues

He filled the **wampimuk** with the substance, passed it around and we all drunk some.

We found a little **wampimuk** sleeping behind the tree.

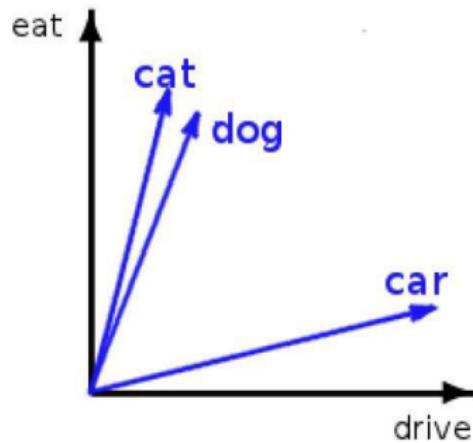
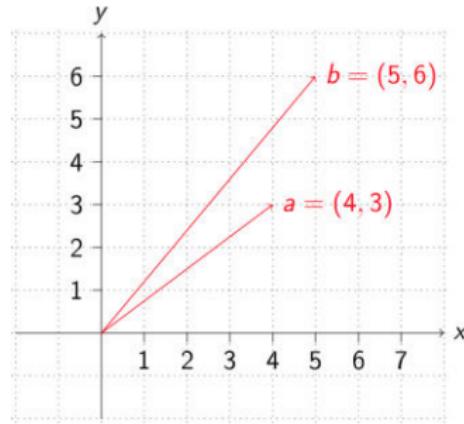
Distributional Semantic Models (DSMs)

- Computational models that build contextual semantic representations from corpus data
- DSMs are models for semantic representations
 -) The semantic content is represented by a vector
 -) Vectors are obtained through the statistical analysis of the linguistic contexts of a word
- Alternative names
 -) corpus-based semantics
 -) statistical semantics
 -) geometrical models of meaning
 -) vector semantics
 -) word space models

Distributional Semantics: The general intuition

- **Distributions** are vectors in a multidimensional semantic space, that is, objects with a magnitude and a direction.
- The **semantic space** has dimensions which correspond to possible contexts, as gathered from a given corpus.

Vector Space



In practice, many more dimensions are used.

$cat = [...\text{dog } 0.8, \text{ eat } 0.7, \text{ joke } 0.01, \text{ mansion } 0.2, ...]$

Word Space

Small Dataset

An automobile is a wheeled motor vehicle used for transporting passengers .

A car is a form of transport , usually with four wheels and the capacity to carry around five passengers .

Transport for the London games is limited , with spectators strongly advised to avoid the use of cars .

The London 2012 soccer tournament began yesterday , with plenty of goals in the opening matches .

Giggs scored the first goal of the football tournament at Wembley , North London .

Bellamy was largely a passenger in the football match , playing no part in either goal .

Target words: {automobile, car, soccer, football}

Term vocabulary: {wheel, transport, passenger, tournament, London, goal, match}

Constructing Word spaces

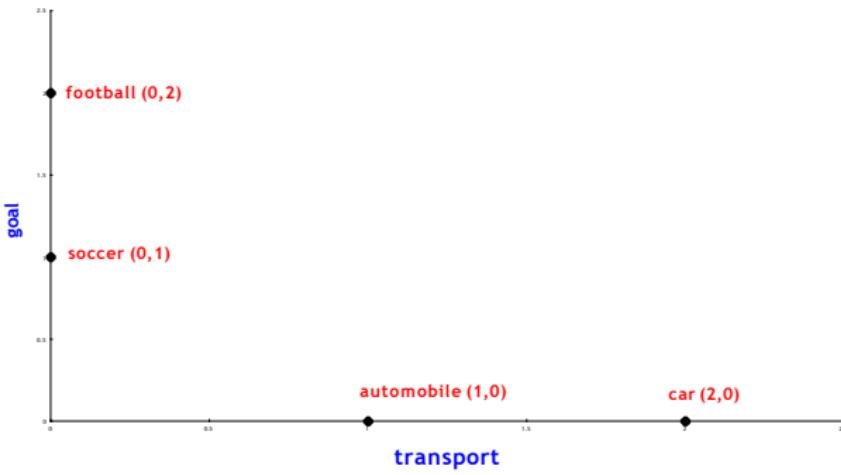
Informal algorithm for constructing word spaces

- Pick the words you are interested in: **target words**
- Define a **context window**, number of words surrounding target word
 -) The context can in general be defined in terms of documents, paragraphs or sentences.
- Count number of times the target word co-occurs with the context words: **co-occurrence matrix**
- Build vectors out of (a function of) these co-occurrence counts

Constructing Word spaces: distributional vectors

distributional matrix = **targets X contexts**

	wheel	transport	passenger	tournament	London	goal	match
automobile	1	1	1	0	0	0	0
car	1	2	1	0	1	0	0
soccer	0	0	0	1	1	1	1
football	0	0	1	1	1	2	1



Computing similarity

	wheel	transport	passenger	tournament	London	goal	match
automobile	1	1	1	0	0	0	0
car	1	2	1	0	1	0	0
soccer	0	0	0	1	1	1	1
football	0	0	1	1	1	2	1

Using simple vector product

$$\text{automobile} \cdot \text{car} = 4$$

$$\text{car} \cdot \text{soccer} = 1$$

$$\text{automobile} \cdot \text{soccer} = 0$$

$$\text{car} \cdot \text{football} = 2$$

$$\text{automobile} \cdot \text{football} = 1$$

$$\text{soccer} \cdot \text{football} = 5$$

Distributional Models of Semantics

Pawan Goyal

CSE, IIT Kharagpur

Week 7, Lecture 2

Vector Space Model without distributional similarity

Words are treated as atomic symbols

One-hot representation

motel [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] = 0

Distributional Similarity Based Representations

You know a word by the company it keeps

government debt problems turning into banking crises as has happened in

saying that Europe needs unified banking regulation to replace the hodgepodge

These words will represent banking

Building a DSM step-by-step

The “linguistic” steps

Pre-process a corpus (to define targets and contexts)



Select the targets and the contexts

The “mathematical” steps

Count the target-context co-occurrences



Weight the contexts (optional)



Build the distributional matrix



Reduce the matrix dimensions (optional)



Compute the vector distances on the (reduced) matrix

Many design choices

Matrix type	Weighting	Dimensionality reduction	Vector comparison
word × document	probabilities	LSA	Euclidean
word × word	length normalization	PLSA	Cosine
word × search proximity	TF-IDF	X LDA	X Dice
adj. × modified noun	PMI	PCA	Jaccard
word × dependency rel.	Positive PMI	IS	KL
verb × arguments	PPMI with discounting	DCA	KL with skew
⋮	⋮	⋮	⋮

General Questions

- How do the rows (words, ...) relate to each other?
- How do the columns (contexts, documents, ...) relate to each other?

The parameter space

A number of parameters to be fixed

- Which type of context?
- Which weighting scheme?
- Which similarity measure?
- ...

A specific parameter setting determines a particular type of DSM (e.g. LSA, HAL, etc.)

Documents as context: Word × document

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
against	0	0	0	1	0	0	3	2	3	0
age	0	0	0	1	0	3	1	0	4	0
agent	0	0	0	0	0	0	0	0	0	0
ages	0	0	0	0	0	2	0	0	0	0
ago	0	0	0	2	0	0	0	0	3	0
agree	0	1	0	0	0	0	0	0	0	0
ahead	0	0	0	1	0	0	0	0	0	0
ain't	0	0	0	0	0	0	0	0	0	0
air	0	0	0	0	0	0	0	0	0	0
aka	0	0	0	1	0	0	0	0	0	0

Words as context: Word × Word

	against	age	agent	ages	ago	agree	ahead	ain.t	air	aka	al
against	2003	90	39	20	88	57	33	15	58	22	24
age	90	1492	14	39	71	38	12	4	18	4	39
agent	39	14	507	2	21	5	10	3	9	8	25
ages	20	39	2	290	32	5	4	3	6	1	6
ago	88	71	21	32	1164	37	25	11	34	11	38
agree	57	38	5	5	37	627	12	2	16	19	14
ahead	33	12	10	4	25	12	429	4	12	10	7
ain't	15	4	3	3	11	2	4	166	0	3	3
air	58	18	9	6	34	16	12	0	746	5	11
aka	22	4	8	1	11	19	10	3	5	261	9
al	24	39	25	6	38	14	7	3	11	9	861

Words as contexts

Parameters

- Window size
- Window shape - rectangular/triangular/other

5 words window (filtered): 2 words either side of the target word

Suspected communist rebels on 4 July 1989 killed Col. Herminio Taylo, police chief of Makati, the Philippines major financial center, in an escalation of street violence sweeping the Capitol area. The gunmen shouted references to the rebel New People's Army. They fled in a commandeered passenger jeep. The military says communist rebels have killed up to 65 soldiers and police in the Capitol region since January.

Context weighting: documents as context

Indexing function F : Essential factors

- **Word frequency (f_{ij}):** How many times a word appears in the document?

$$F \propto f_{ij}$$

- **Document length ($|D_i|$):** How many words appear in the document?

$$F \propto \frac{1}{|D_i|}$$

- **Document frequency (N_j):** Number of documents in which a word appears.

$$F \propto \frac{1}{N_j}$$

Indexing Weight: $tf\text{-}Idf$

- $f_{ij} * \log(\frac{N}{N_j})$ for each term, normalize the weight in a document with respect to L_2 -norm.

Context weighting: words as context

basic intuition

word1	word2	freq(1,2)	freq(1)	freq(2)
dog	small	855	33,338	490,580
dog	domesticated	29	33,338	918

Context weighting: words as context

basic intuition

Association measures are used to give more weight to contexts that are more significantly associated with a target word.

- The less frequent the target and context element are, the higher the weight given to their co-occurrence count should be.
⇒ Co-occurrence with frequent context element *small* is less informative than co-occurrence with rarer *domesticated*.
- different measures - e.g., Mutual information, Log-likelihood ratio

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log_2 \frac{P_{corpus}(w_1, w_2)}{P_{ind}(w_1, w_2)}$$

$$PMI(w_1, w_2) = \log_2 \frac{P_{corpus}(w_1, w_2)}{P_{corpus}(w_1)P_{corpus}(w_2)}$$

$$P_{corpus}(w_1, w_2) = \frac{\text{freq}(w_1, w_2)}{N}$$

$$P_{corpus}(w) = \frac{\text{freq}(w)}{N}$$

Positive PMI

All PMI values less than zero are replaced with zero.

Bias towards infrequent events

Consider w_j having the maximum association with w_i ,

$$P_{corpus}(w_i) \approx P_{corpus}(w_j) \approx P_{corpus}(w_i, w_j)$$

PMI increases as the probability of w_i decreases.

Also, consider a word w_j that occurs once in the corpus, also in the context of w_i . A discounting factor proposed by Pantel and Lin:

$$\delta_{ij} = \frac{f_{ij}}{f_{ij} + 1} \frac{\min(f_i, f_j)}{\min(f_i, f_j) + 1}$$

$$PMI_{new}(w_i, w_j) = \delta_{ij} PMI(w_i, w_j)$$

Distributional Vectors: Example

Normalized Distributional Vectors using Pointwise Mutual Information

petroleum	oil:0.032 gas:0.029 crude:0.029 barrels:0.028 exploration:0.027 barrel:0.026 opec:0.026 refining:0.026 gasoline:0.026 fuel:0.025 natural:0.025 exporting:0.025
drug	trafficking:0.029 cocaine:0.028 narcotics:0.027 fda:0.026 police:0.026 abuse:0.026 marijuana:0.025 crime:0.025 colombian:0.025 arrested:0.025 addicts:0.024
insurance	insurers:0.028 premiums:0.028 lloyds:0.026 reinsurance:0.026 underwriting:0.025 pension:0.025 mortgage:0.025 credit:0.025 investors:0.024 claims:0.024 benefits:0.024
forest	timber:0.028 trees:0.027 land:0.027 forestry:0.026 environmental:0.026 species:0.026 wildlife:0.026 habitat:0.025 tree:0.025 mountain:0.025 river:0.025 lake:0.025
robotics	robots:0.032 automation:0.029 technology:0.028 engineering:0.026 systems:0.026 sensors:0.025 welding:0.025 computer:0.025 manufacturing:0.025 automated:0.025

Distributional Semantics: Applications, Structured Models

Pawan Goyal

CSE, IIT Kharagpur

Week 7, Lecture 3

Application to Query Expansion: Addressing Term Mismatch

Term Mismatch Problem in Information Retrieval

- Stems from the word independence assumption during document indexing.
- User query: *insurance cover which pays for long term care.*
- A relevant document may contain terms different from the actual user query.
- Some relevant words concerning this query: *{medicare, premiums, insurers}*

Using DSMs for Query Expansion

Given a user query, reformulate it using related terms to enhance the retrieval performance.

- The distributional vectors for the query terms are computed.
- Expanded query is obtained by a linear combination or a functional combination of these vectors.

Query Expansion using Unstructured DSMs

TREC Topic 104: catastrophic health insurance

Query Representation: surtax:1.0 hcfa:0.97 medicare:0.93 hmos:0.83
medicaid:0.8 hmo:0.78 beneficiaries:0.75 ambulatory:0.72 premiums:0.72
hospitalization:0.71 hhs:0.7 reimbursable:0.7 deductible:0.69

- Broad expansion terms: **medicare, beneficiaries, premiums ...**
- Specific domain terms: **HCFA** (Health Care Financing Administration), **HMO** (Health Maintenance Organization), **HHS** (Health and Human Services)

TREC Topic 355: ocean remote sensing

Query Representation: radiometer:1.0 landsat:0.97 ionosphere:0.94
cnes:0.84 altimeter:0.83 nasda:0.81 meterology:0.81 cartography:0.78
geostationary:0.78 doppler:0.78 oceanographic:0.76

- Broad expansion terms: **radiometer, landsat, ionosphere ...**
- Specific domain terms: **CNES** (Centre National d'Études Spatiales) and **NASDA** (National Space Development Agency of Japan)

Similarity Measures for Binary Vectors

Let X and Y denote the binary distributional vectors for words X and Y .

Similarity Measures

$$\text{Dice coefficient} : \frac{2|X \cap Y|}{|X| + |Y|}$$

$$\text{Jaccard Coefficient} : \frac{|X \cap Y|}{|X \cup Y|}$$

$$\text{Overlap Coefficient} : \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

Jaccard coefficient penalizes small number of shared entries, while Overlap coefficient uses the concept of inclusion.

Similarity Measures for Vector Spaces

Let \vec{X} and \vec{Y} denote the distributional vectors for words X and Y .
 $\vec{X} = [x_1, x_2, \dots, x_n], \vec{Y} = [y_1, y_2, \dots, y_n]$

Similarity Measures

$$\text{Cosine similarity : } \cos(\vec{X}, \vec{Y}) = \frac{\vec{X} \cdot \vec{Y}}{\|\vec{X}\| \|\vec{Y}\|}$$

$$\text{Euclidean distance : } \|\vec{X} - \vec{Y}\| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Similarity Measure for Probability Distributions

Let p and q denote the probability distributions corresponding to two distributional vectors.

Similarity Measures

$$\text{KL-divergence} : D(p||q) = \sum_i p_i \log \frac{p_i}{q_i}$$

$$\text{Information Radius} : D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2})$$

$$L_1\text{-norm} : \sum_i |p_i - q_i|$$

Attributional Similarity vs. Relational Similarity

Attributional Similarity

The attributional similarity between two words a and b depends on the degree of correspondence between the properties of a and b .

Ex: dog and wolf

Relational Similarity

Two pairs (a, b) and (c, d) are relationally similar if they have many similar relations.

Ex: dog: bark and cat: meow

Relational Similarity: Pair-pattern matrix

Pair-pattern matrix

- Row vectors correspond to pairs of words, such as *mason: stone* and *carpenter: wood*
- Column vectors correspond to the patterns in which the pairs occur, e.g. *X cuts Y* and *X works with Y*
- Compute the similarity of rows to find similar pairs

Extended Distributional Hypothesis; Lin and Pantel

Patterns that co-occur with similar pairs tend to have similar meanings.

This matrix can also be used to measure the semantic similarity of patterns.

Given a pattern such as “X solves Y”, you can use this matrix to find similar patterns, such as “Y is solved by X”, “Y is resolved in X”, “X resolves Y”.

Basic Issue

- Words may not be the basic context units anymore
- How to capture and represent syntactic information?
X solves Y and Y is solved by X

An Ideal Formalism

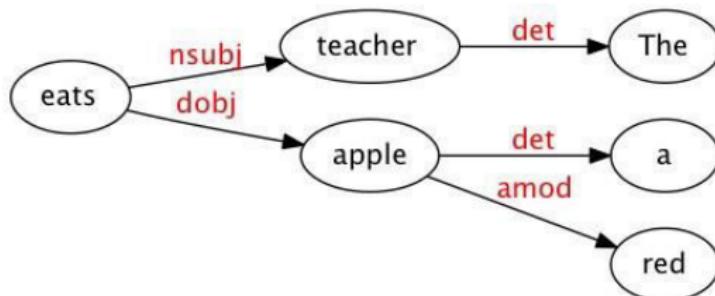
- Should mirror semantic relationships as close as possible
- Incorporate word-based information and syntactic analysis
- Should be applicable to different languages

Use Dependency grammar framework

Structured DSMs

Using Dependency Structure: How does it help?

The teacher eats a red apple.



- ‘eat’ is not a legitimate context for ‘red’.
- The ‘object’ relation connecting ‘eat’ and ‘apple’ is treated as a different type of co-occurrence from the ‘modifier’ relation linking ‘red’ and ‘apple’.

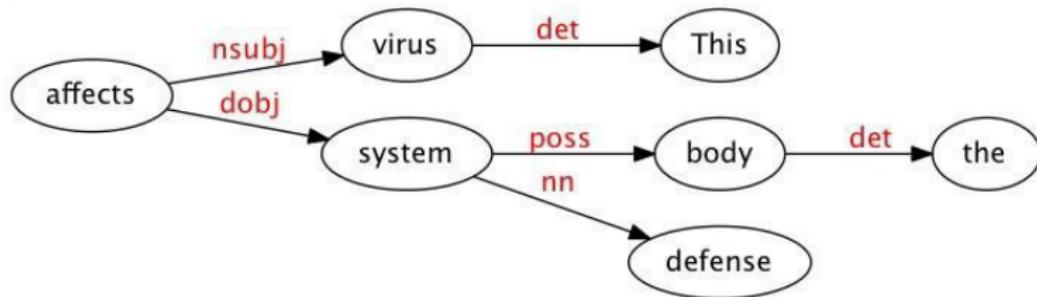
Structured DSMs: Words as 'legitimate' contexts

- Co-occurrence statistics are collected using parser-extracted relations.
- To qualify as context of a target item, a word must be linked to it by some (interesting) lexico-syntactic relation

Structured DSMs

Distributional models, as guided by dependency

Ex: For the sentence '*This virus affects the body's defense system.*', the dependency parse is:



Word vectors

<system, dobj, affects> ...

Corpus-derived ternary data can also be mapped onto a 2-way matrix

2-way matrix

<system, dobj, affects>
<virus, nsubj, affects>

The dependency information can be dropped

- <system, dobj, affects> \Rightarrow <system, affects>
- <virus, nsubj, affects> \Rightarrow <virus, affects>

Link and one word can be concatenated and treated as attributes

- virus={nsubj-affects:0.05,...},
- system={dobj-affects:0.03,...}

Structured DSMs for Selectional Preferences

Selectional Preferences for Verbs

Most verbs prefer arguments of a particular type. This regularity is known as selectional preference.

- From a parsed corpus, noun vectors are calculated as shown for ‘virus’ and ‘system’.

	obj-carry	obj-buy	obj-drive	obj-eat	obj-store	sub-fly	...
car	0.1	0.4	0.8	0.02	0.2	0.05	...
vegetable	0.3	0.5	0	0.6	0.3	0.05	...
biscuit	0.4	0.4	0	0.5	0.4	0.02	...
...

Selectional Preferences

- Suppose we want to compute the selectional preferences of the nouns as object of verb ‘eat’.
- n nouns having highest weight in the dimension ‘obj-eat’ are selected, let $\{\text{vegetable}, \text{biscuit}, \dots\}$ be the set of these n nouns.
- The complete vectors of these n nouns are used to obtain an ‘object prototype’ of the verb.
- ‘object prototype’ will indicate various attributes such as these nouns can be consumed, bought, carried, stored etc.
- Similarity of a noun to this ‘object prototype’ is used to denote the plausibility of that noun being an object of verb ‘eat’.

Word Embeddings - Part I

Pawan Goyal

CSE, IIT Kharagpur

Week 7, Lecture 4

Word Vectors

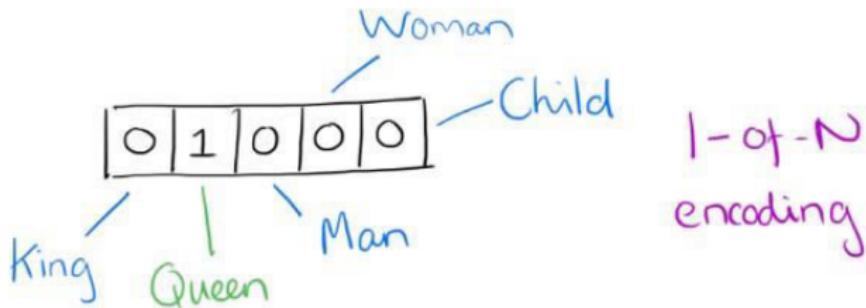
- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or ‘one-hot’) encoding every element in the vector is associated with a word in the vocabulary.
- The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

One-hot representation

motel [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] = 0

Word Vectors - One-hot Encoding

- Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child.
- We could encode the word ‘Queen’ as:



Limitations of One-hot encoding

Word vectors are not comparable

Using such an encoding, there is no meaningful comparison we can make between word vectors other than equality testing.

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a d -dimensional vector, which is mostly learnt!

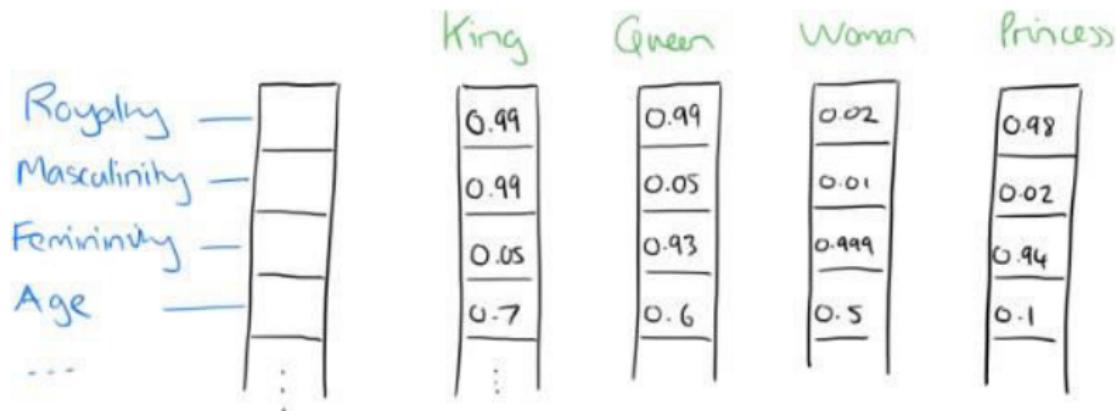
$$\text{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Distributional Representation

- Take a vector with several hundred dimensions (say 1000).
- Each word is represented by a distribution of weights across those elements.
- So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and
- Each element in the vector contributes to the definition of many words.

Distributional Representation: Illustration

If we label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:



Such a vector comes to represent in some abstract way the 'meaning' of a word

Word Embeddings

- d typically in the range 50 to 1000
- Similar words should have similar embeddings

SVD can also be thought of as an embedding method

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Case of Singular-Plural Relations

If we denote the vector for word i as x_i , and focus on the singular/plural relation, we observe that

$$x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families} \approx x_{car} - x_{cars}$$

and so on.

Reasoning with Word Vectors

Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

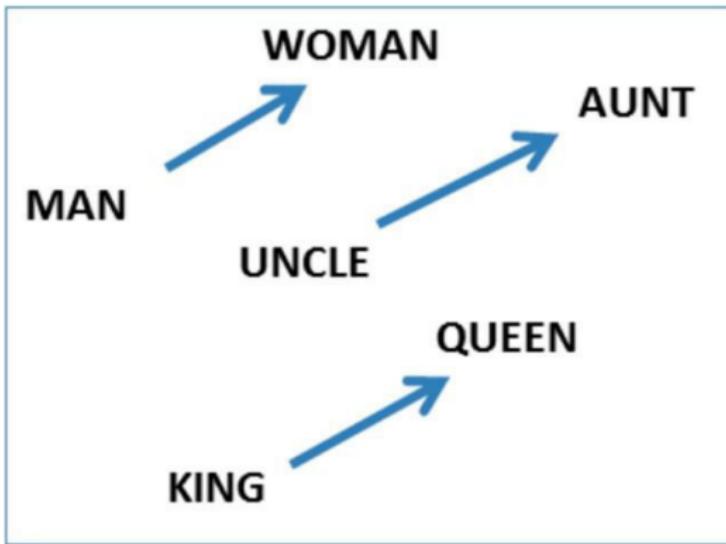
Good at answering analogy questions

a is to b, as c is to ?

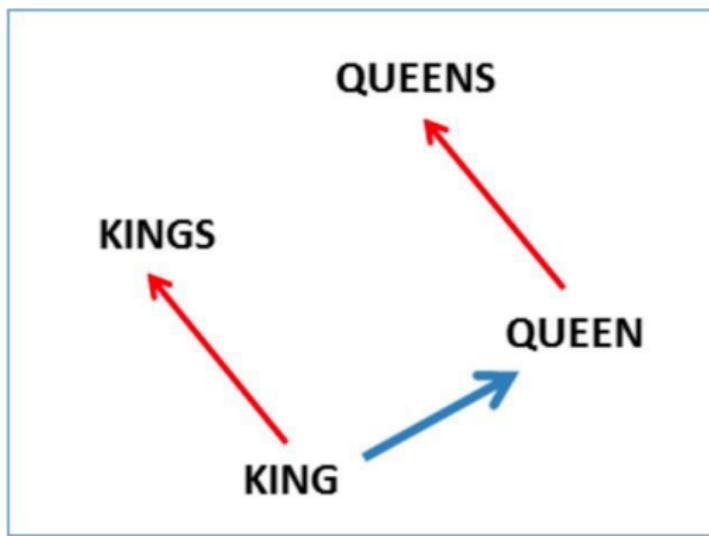
man is to woman as uncle is to ? (aunt)

A simple vector offset method based on cosine distance shows the relation.

Vector Offset for Gender Relation



Vector Offset for Singular-Plural Relation



Encoding Other Dimensions of Similarity

Analogy Testing

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Analogy Testing

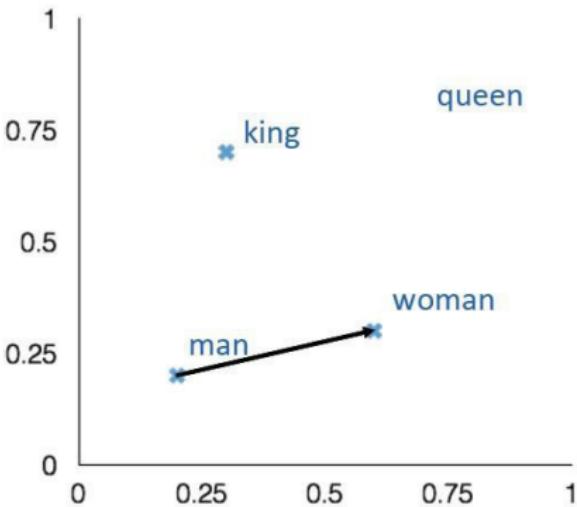
a:b :: c:?



$$d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{\|w_b - w_a + w_c\|}$$

man:woman :: king:?

+ king	[0.30 0.70]
- man	[0.20 0.20]
+ woman	[0.60 0.30]
<hr/>	
queen	[0.70 0.80]



Country-capital city relationships

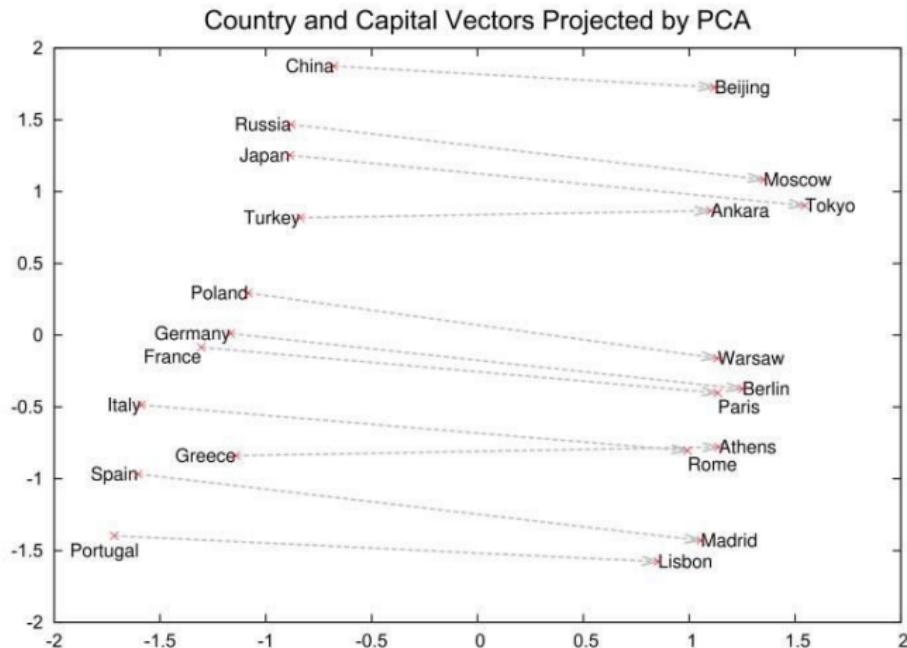


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

More Analogy Questions

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Element Wise Addition

We can also use element-wise addition of vector elements to ask questions such as ‘German + airlines’ and by looking at the closest tokens to the composite vector come up with impressive answers:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Czech crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

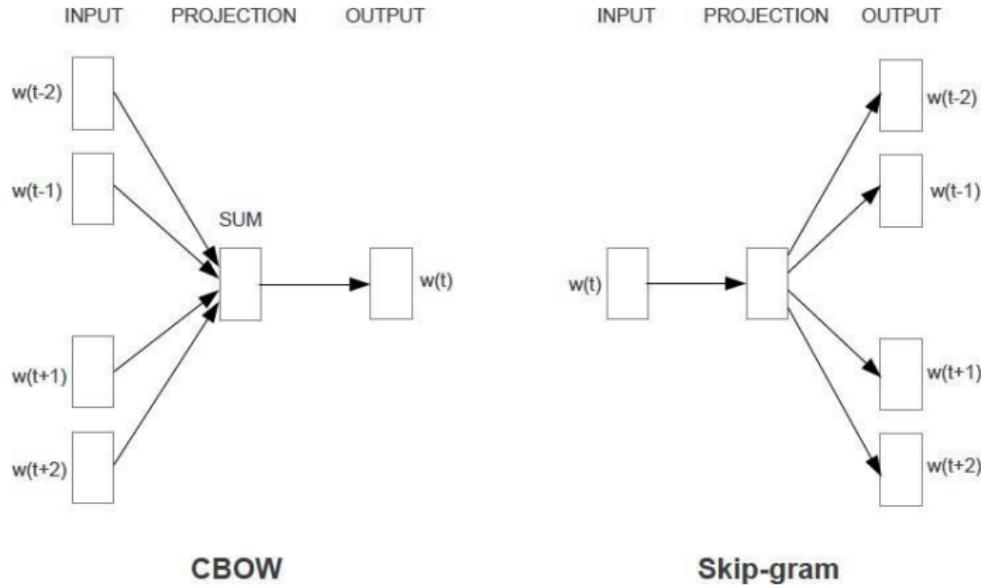
Learning Word Vectors

Basic Idea

Instead of capturing co-occurrence counts directly, predict (using) surrounding words of every word.

Code as well as word-vectors: <https://code.google.com/p/word2vec/>

Two Variations: CBOW and Skip-grams



Word Embeddings - Part II

Pawan Goyal

CSE, IIT Kharagpur

Week 7, Lecture 5

CBOW

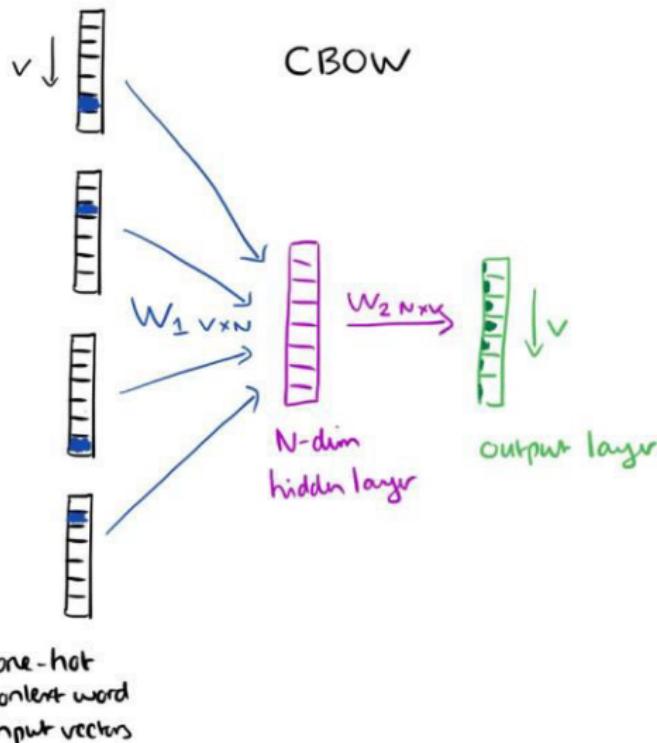
- Consider a piece of prose such as:
“The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of syntactic and semantic word relationships.”
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:

...an efficient method for learning high quality distributed vector ...

The diagram illustrates a sliding window over a piece of text. A blue arrow points upwards from the text to a blue label 'focus word'. Below the text, two green curly braces, each labeled 'Context' in green, encompass the four words preceding the focus word and the four words following it.

CBOW

The context words form the input layer. Each word is encoded in one-hot form. A single hidden and output layer.



CBOW: Training Objective

- The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words, with regard to the weights.
- In our example, given the input (“an”, “efficient”, “method”, “for”, “high”, “quality”, “distributed”, “vector”), we want to maximize the probability of getting “learning” as the output.

CBOW: Input to Hidden Layer

Since our input vectors are one-hot, multiplying an input vector by the weight matrix W_1 amounts to simply selecting a row from W_1 .

$$\begin{array}{c} \text{input} \\ 1 \times V \end{array} \quad \begin{array}{c} W_1 \\ V \times N \end{array} \quad \begin{array}{c} \text{hidden} \\ 1 \times N \end{array}$$
$$[0 \ 1 \ 0] \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}_{W_1} = [e \ f \ g \ h]$$

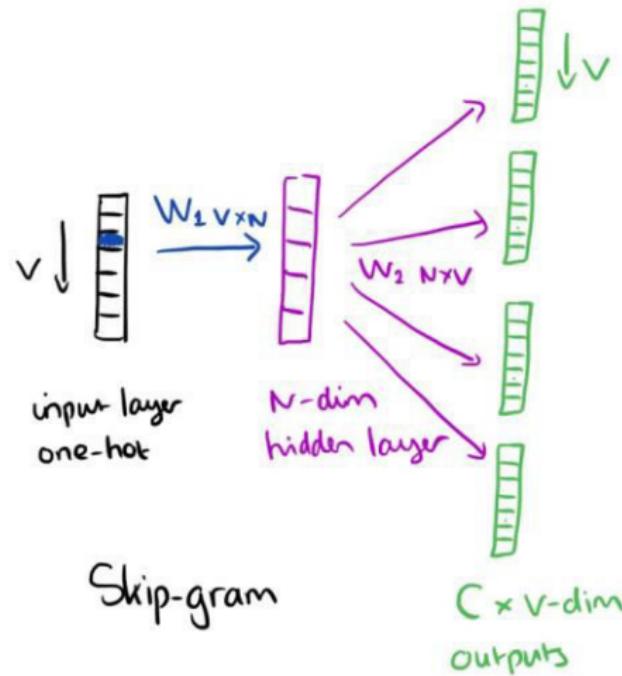
Given C input word vectors, the activation function for the hidden layer h amounts to simply summing the corresponding ‘hot’ rows in W_1 , and dividing by C to take their average.

CBOW: Hidden to Output Layer

From the hidden layer to the output layer, the second weight matrix W_2 can be used to compute a score for each word in the vocabulary, and softmax can be used to obtain the posterior distribution of words.

Skip-gram Model

The skip-gram model is the opposite of the CBOW model. It is constructed with the focus word as the single input vector, and the target context words are now at the output layer:



Skip-gram Model: Training

- The activation function for the hidden layer simply amounts to copying the corresponding row from the weights matrix W_1 (linear) as we saw before.
- At the output layer, we now output C multinomial distributions instead of just one.
- The training objective is to minimize the summed prediction error across all context words in the output layer. In our example, the input would be “learning”, and we hope to see (“an”, “efficient”, “method”, “for”, “high”, “quality”, “distributed”, “vector”) at the output layer.

Skip-gram Model

Details

Predict surrounding words in a window of length c of each word

Objective Function: Maximize the log probability of any context word given the current center word:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T \sum_{\substack{-c \leq j \leq c, j \neq 0}} \log p(w_{t+j}|w_t)$$

Word Vectors

For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(w_O|w_I) = \frac{\exp(v_w^{'} {}^T v_{WI})}{\sum_{w=1}^W \exp(v_w^{'} {}^T v_{WI})}$$

where v and v' are “input” and “output” vector representations of w (so every word has two vectors)

Parameters θ

With d -dimensional words and V many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ v'_{aardvark} \\ v'_a \\ \vdots \\ v'_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Gradient Descent for Parameter Updates

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

Two sets of vectors

Best solution is to sum these up

$$L_{final} = L + L'$$

A good tutorial to understand parameter learning:

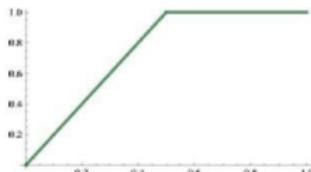
<https://arxiv.org/pdf/1411.2738.pdf>

An interactive Demo

<https://ronxin.github.io/wevi/>

Glove

$$J = \frac{1}{2} \sum_{ij} f(P_{ij}) (w_i \cdot \tilde{w}_j - \log P_{ij})^2 \quad f \sim$$



Combine the best of both worlds - count based methods as well as direct prediction methods

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

Code and vectors: <http://nlp.stanford.edu/projects/glove/>

Lexical Semantics

Pawan Goyal

CSE, IIT Kharagpur

Week 8, Lecture 1

Definition

Lexical semantics is concerned with the systematic meaning related connections among lexical items, and the internal meaning-related structure of individual lexical items.

To identify the semantics of lexical items, we need to focus on the notion of **lexeme**, an individual entry in the lexicon.

What is a lexeme?

Lexeme should be thought of as a pairing of a particular orthographic and phonological form with some sort of symbolic meaning representation.

- Orthographic form, and phonological form refer to the appropriate form part of a lexeme
- Sense refers to a lexeme's meaning counterpart.

Example

verge¹ | vərj |

noun

an edge or border: *they came down to the verge of the lake.*

- an extreme limit beyond which something specified will happen: *I was on the verge of tears.*
- Brit. a grass edging such as that by the side of a road or path.
- Architecture an edge of tiles projecting over a gable.

verb [no obj.] (**verge on**)

approach (something) closely; be close or similar to (something): *despair verging on the suicidal.*

ORIGIN late Middle English: via Old French from Latin *virga 'rod.'* The current verb sense dates from the late 18th cent.

verge² | vərj |

noun

a wand or rod carried before a bishop or dean as an emblem of office.

ORIGIN late Middle English: from Latin *virga 'rod.'*

verge³ | vərj |

verb [no obj.]

incline in a certain direction or toward a particular state: *his style verged into the art nouveau school.*

ORIGIN early 17th cent. (in the sense '*descend (to the horizon)*') : from Latin *vergere 'to bend, incline.'*

Example: meaning related facts?

Definitions from the American Heritage Dictionary (Morris, 1985)

- **right** *adj.* located near the right hand esp. being on the right when facing the same direction as the observer
- **left** *adj.* located near to this side of the body than the right
- **red** *n.* the color of blood or a ruby
- **blood** *n.* the red liquid that circulates in the heart, arteries and veins of animals

- The entries are description of lexemes in terms of other lexemes
- Definitions make it clear that *right* and *left* are similar kind of lexemes that stand in some kind of alternation, or opposition, to one another
- We can glean that *red* is a color, it can be applied to both *blood* and *rubies*, and that *blood* is a liquid.

Relations between word meanings

- Homonymy
- Polysemy
- Synonymy
- Antonymy
- Hypernymy
- Hyponymy
- Meronymy

Homonymy

Definition

Homonymy is defined as a relation that holds between words that have the same form with unrelated meanings.

Examples

- Bat (wooden stick-like thing) vs Bat (flying mammal thing)
- Bank (financial institution) vs Bank (riverside)

homophones and homographs

homophones are the words with the same pronunciation but different spellings.

- write vs right
- piece vs peace

homographs are the lexemes with the same orthographic form but different meaning. Ex: bass

Problems for NLP applications

Text-to-Speech

Same orthographic form but different phonological form

Information Retrieval

Different meaning but same orthographic form

Speech Recognition

to, two, too

Perfect homonyms are also problematic

Polysemy

Multiple related meanings within a single lexeme.

- The *bank* was constructed in 1875 out of local red brick.
- I withdrew the money from the *bank*.

Are those the same sense?

- Sense 1: “The building belonging to a financial institution”
- Sense 2: “A financial institution”

Another example

- Heavy snow caused the roof of the *school* to collapse.
- The *school* hired more teachers this year than ever before.

Polysemy: multiple related meanings

Often, the relationships are systematic

E.g., building vs. organization

school, university, hospital, church, supermarket

More examples:

- Author (Jane Austen wrote Emma) ←→ Works of Author (I really love Jane Austen)
- Animal (The chicken was domesticated in Asia) ←→ Meat (The chicken was overcooked)
- Tree (Plums have beautiful blossoms) ←→ Fruit (I ate a preserved plum yesterday)

Polysemy: multiple related meanings

Zeugma test

- Which of these flights *serve* breakfast?
- Does Midwest Express *serve* Philadelphia?

*Does Midwest Express *serve* breakfast and San Jose?

Combine two separate uses of a lexeme into a single example using conjunction

Since it sounds weird, we say that these are two different senses of *serve*.

Synonymy

Words that have the same meaning in some or all contexts.

- filbert / hazelnut
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- water / H_2O

Two lexemes are synonyms if they can be successfully substituted for each other in all situations.

Synonymy: A relation between senses

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, for instance, became a kind of **big** sister to Benjamin.
- *Miss Nelson, for instance, became a kind of **large** sister to Benjamin.

Why?

- *big* has a sense that means being older, or grown up
- *large* lacks this sense

Synonyms

Shades of meaning

- What is the cheapest first class *fare*?
- *What is the cheapest first class *price*?

Collocational constraints

- We frustate 'em and frustate 'em, and pretty soon they make a *big* mistake.
- *We frustate 'em and frustate 'em, and pretty soon they make a *large* mistake.

Antonyms

- Senses that are opposites with respect to one feature of their meaning
- Otherwise, they are similar!
 - ▶ dark / light
 - ▶ short / long
 - ▶ hot / cold
 - ▶ up / down
 - ▶ in / out

More formally: antonyms can

- define a binary opposition or at opposite ends of a scale (*long/short, fast/slow*)
- Be **reversives**: *rise/fall*

Hyponymy and Hypernymy

Hyponymy

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*
- *dog* is a hyponym of *animal*
- *mango* is a hyponym of *fruit*

Hypernymy

Conversely

- *vehicle* is a hypernym/superordinate of *car*
- *animal* is a hypernym of *dog*
- *fruit* is a hypernym of *mango*

Hyponymy more formally

Entailment

Sense *A* is a hyponym of sense *B* if being an *A* entails being a *B*.

Ex: dog, animal

Transitivity

A hypo *B* and *B* hypo *C* entails *A* hypo *C*

Meronyms and holonyms

Definition

Meronymy: an asymmetric, transitive relation between senses.

X is a **meronym** of *Y* if it denotes a part of *Y*.

The inverse relation is **holonymy**.

meronym	holonym
porch	house
wheel	car
leg	chair
nose	face

Lexical Semantics - WordNet

Pawan Goyal

CSE, IIT Kharagpur

Week 8, Lecture 2

<https://wordnet.princeton.edu/wordnet/>

- A hierarchically organized lexical database
- A machine-readable thesaurus, and aspects of a dictionary
- Versions for other languages are under development

part of speech	no. synsets
noun	82,115
verb	13,767
adjective	18,156
adverb	3,621

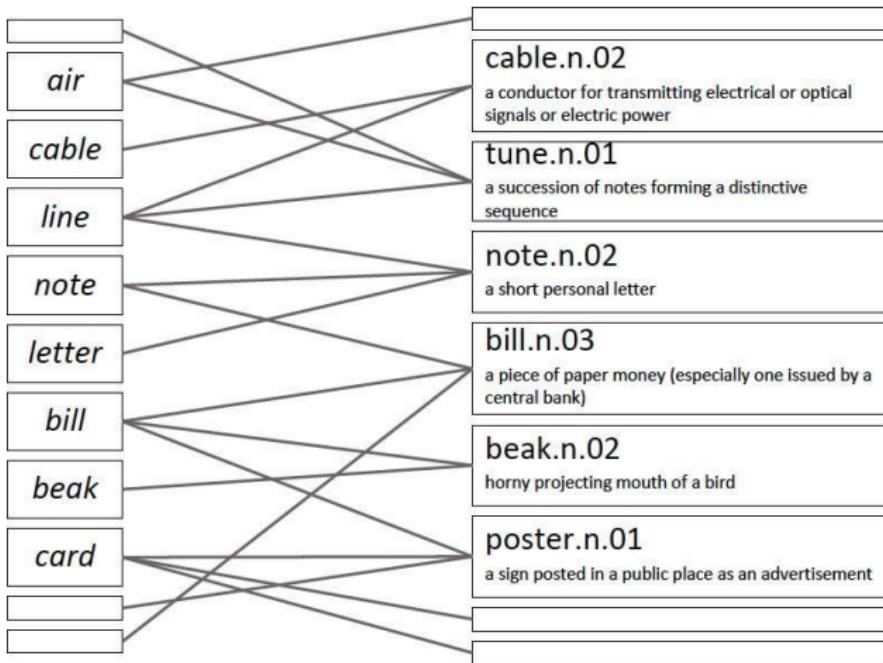
Synsets in WordNet

- A **synset** is a set of synonyms representing a sense
- Example: chump as a noun to mean ‘a person who is gullible and easy to take advantage of’

{chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²}

- Each of these senses share this same gloss.
- For WordNet, the meaning of this sense of chump is this list.

lemma vs. synsets



All relations in WordNet

searchtype is at least one of the following:

-ants{n v a r}	Antonyms
-hype{n v}	Hypernyms
-hypo{n v}, -tree{n v}	Hyponyms & Hyponym Tree
-entav	Verb Entailment
-syns{n v a r}	Synonyms (ordered by estimated frequency)
-smemn	Member of Holonyms
-ssubn	Substance of Holonyms
-sprtn	Part of Holonyms
-membn	Has Member Meronyms
-subsn	Has Substance Meronyms
-partn	Has Part Meronyms
-meron	All Meronyms
-holon	All Holonyms
-causv	Cause to
-pert{a r}	Pertainyms
-attr{n a}	Attributes
-deri{n v}	Derived Forms
-domn{n v a r}	Domain
-domt{n v a r}	Domain Terms
-faml{n v a r}	Familiarity & Polysemy Count
-framv	Verb Frames
-coor{n v}	Coordinate Terms (sisters)
-simsv	Synonyms (grouped by similarity of meaning)
-hmern	Hierarchical Meronyms
-hholn	Hierarchical Holonyms
-grep{n v a r}	List of Compound Words
-over	Overview of Senses
-	-

Wordnet noun and verb 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

Relation	Definition	Example
Hypernym	From events to superordinate events	<i>fly</i> ⁹ → <i>travel</i> ⁵
Troponym	From a verb (event) to a specific manner elaboration of that verb	<i>walk</i> ¹ → <i>stroll</i> ¹
Entails	From verbs (events) to the verbs (events) they entail	<i>snore</i> ¹ → <i>sleep</i> ¹
Antonym	Opposites	<i>increase</i> ¹ ⇔ <i>decrease</i> ¹

WordNet Hierarchies

Synonyms/Hypernyms (Ordered by Estimated Frequency) of noun mouse

4 senses of mouse

Sense 1

mouse

```
=> rodent, gnawer
=> placental, placental mammal, eutherian, eutherian mammal
=> mammal, mammalian
=> vertebrate, craniate
=> chordate
=> animal, animate being, beast, brute, creature, fauna
=> organism, being
=> living thing, animate thing
=> whole, unit
=> object, physical object
=> physical entity
=> entity
```

Sense 4

mouse, computer mouse

```
=> electronic device
=> device
=> instrumentality, instrumentation
=> artifact, artefact
=> whole, unit
=> object, physical object
=> physical entity
=> entity
```

Word Similarity

- Synonymy is a binary relation
 - ▶ Two words are either synonymous or not
- We want a looser metric
 - ▶ Word similarity or
 - ▶ Word distance
- Two words are more similar
 - ▶ If they share more features of meaning
- Actually these are really relations between **senses**:
 - ▶ Instead of saying “bank is like fund”
 - ▶ We say
 - F Bank¹ is similar to fund³
 - F Bank² is similar to slope⁵
- We will compute similarity over both words and senses

Two classes of algorithms

Distributional algorithms

By comparing words based on their distributional context in the corpora

Thesaurus-based algorithms

Based on whether words are “nearby” in WordNet

Thesaurus-based Word Similarity

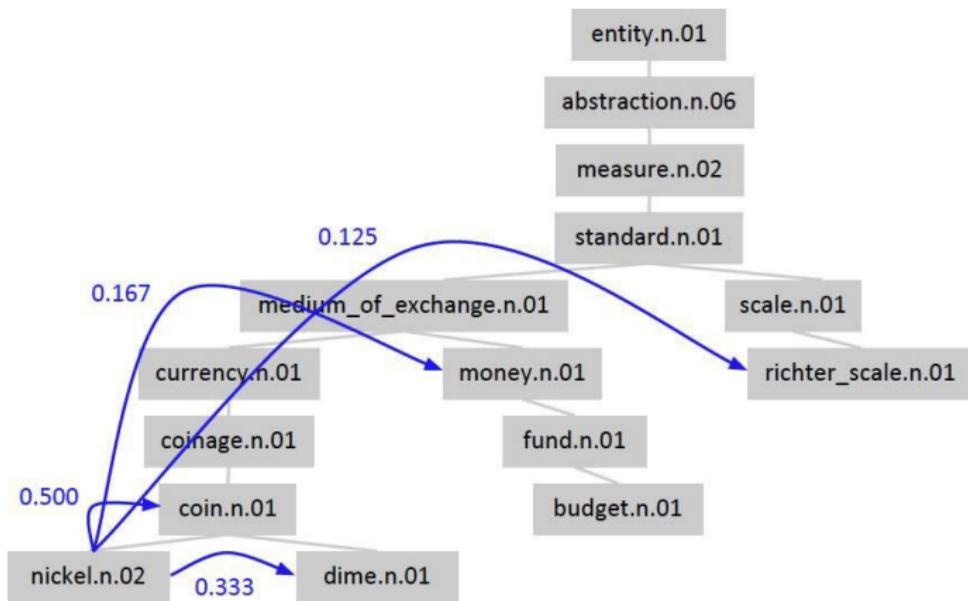
- We could use anything in the thesaurus:
 - ▶ Meronymy, hyponymy, troponymy
 - ▶ Glosses and example sentences
- In practice, “thesaurus-based” methods usually use:
 - ▶ the is-a/subsumption/hypernymy hierarchy
 - ▶ and sometimes the glosses too
- Word similarity vs. word relatedness
 - ▶ Similar words are near-synonyms
 - ▶ Related words could be related any way
 - F car, gasoline : related, but not similar
 - F car, bicycle: similar

Path-based similarity

Basic Idea

- Two words are similar if they are nearby in the hypernym graph
- $\text{pathlen}(c_1, c_2)$ = number of edges in shortest path (in hypernym graph) between senses c_1 and c_2
- $\text{sim}_{\text{path}}(c_1, c_2) = \frac{1}{1 + \text{pathlen}(c_1, c_2)}$
- $\text{sim}(w_1, w_2) = \max_{c_1 \in \text{senses}(w_1), c_2 \in \text{senses}(w_2)} \text{sim}(c_1, c_2)$

Shortest path in the hierarchy



Leacock-Chodorow (L-C) Similarity

L-C similarity

$$sim_{LC}(c_1, c_2) = -\log(pathlen(c_1, c_2)/2d)$$

d: maximum depth of the hierarchy

Problems with L-C similarity

- Assumes each edge represents a uniform distance
- ‘nickel-money’ seems closer than ‘nickel-standard’
- We want a metric which lets us assign different “lengths” to different edges - but how?

Concept probability models

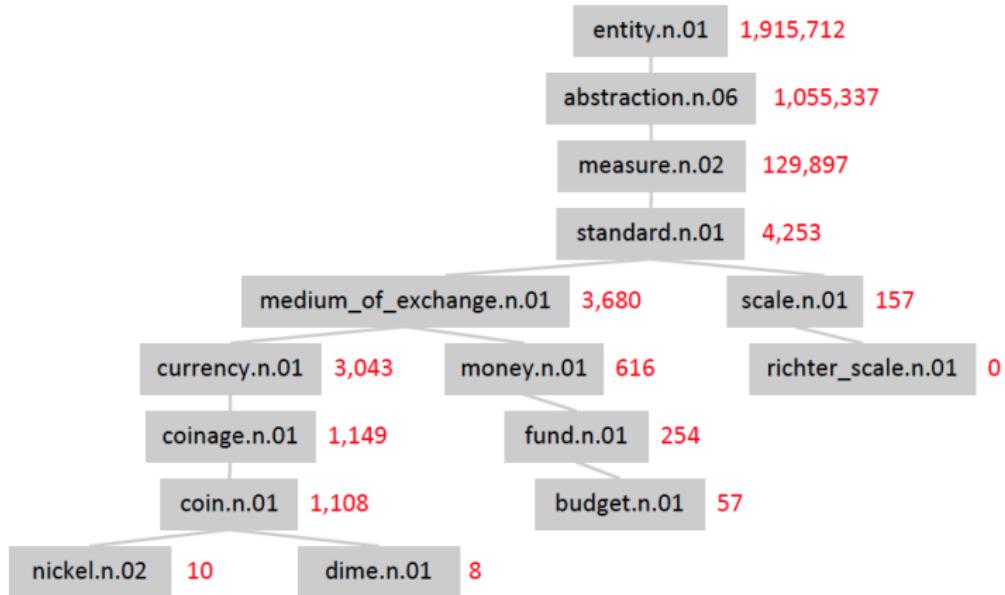
Concept probabilities

- For each concept (synset) c , let $P(c)$ be the probability that a randomly selected word in a corpus is an instance (hyponym) of c
- $P(ROOT) = 1$
- The lower a node in the hierarchy, the lower its probability

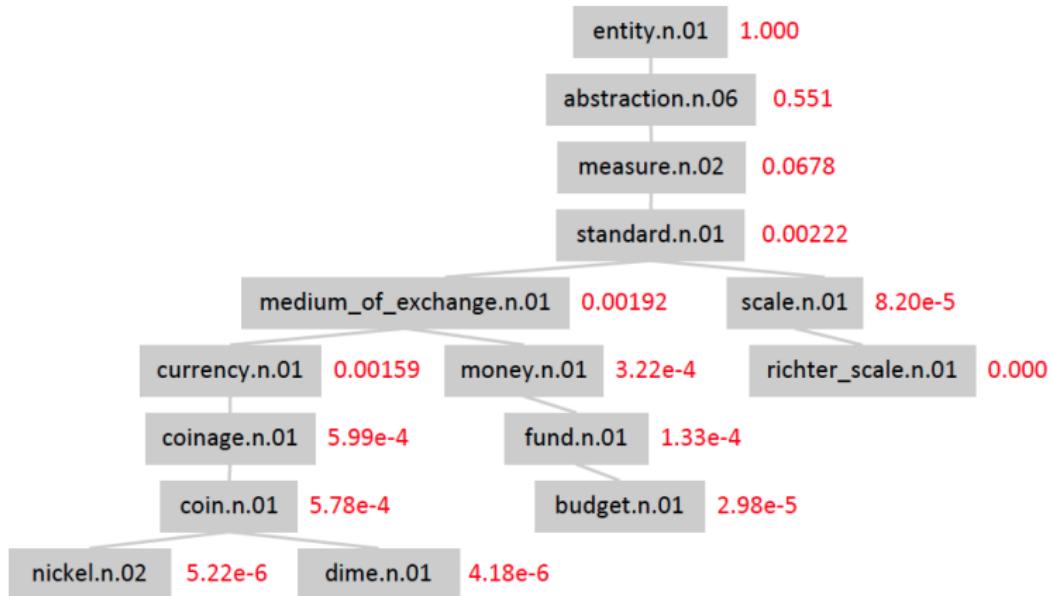
Estimating concept probabilities

- Train by counting “concept activations” in a corpus
- Each occurrence of *dime* also increments counts for *coin*, *currency*, *standard*, etc.

Example : concept count



Example : concept probabilities

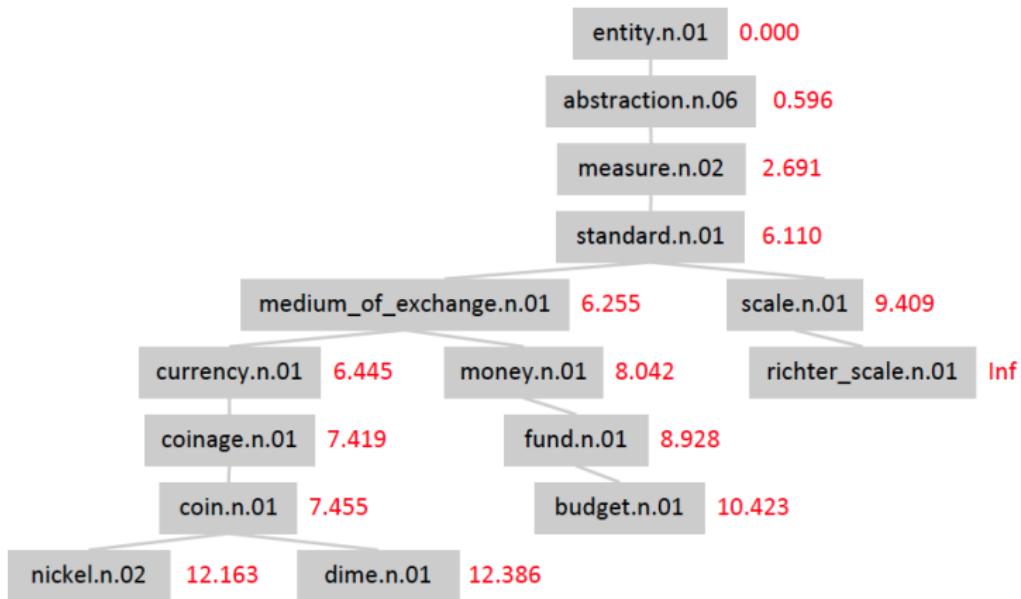


Information content

Information content

- Information content: $IC(c) = -\log P(c)$
- Lowest common subsumer : $LCS(c_1, c_2)$: the lowest node in the hierarchy that subsumes (is a hypernym of) both c_1 and c_2
- We are now ready to see how to use information content (IC) as a similarity metric.

Example : Information content

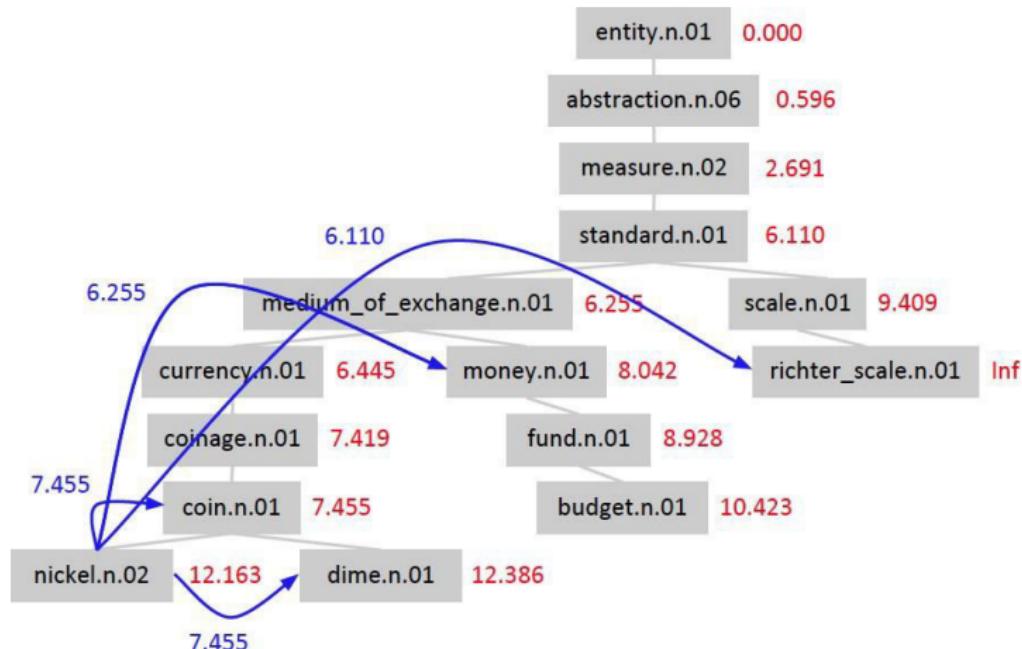


Resnik Similarity

Resnik Similarity

- Intuition: how similar two words are depends on how much they have in common
- It measures the commonality by the information content of the lowest common subsumer
- $sim_{resnik}(c_1, c_2) = IC(LCS(c_1, c_2)) = -\log P(LCS(c_1, c_2))$

Example: Resnik similarity



Lin similarity

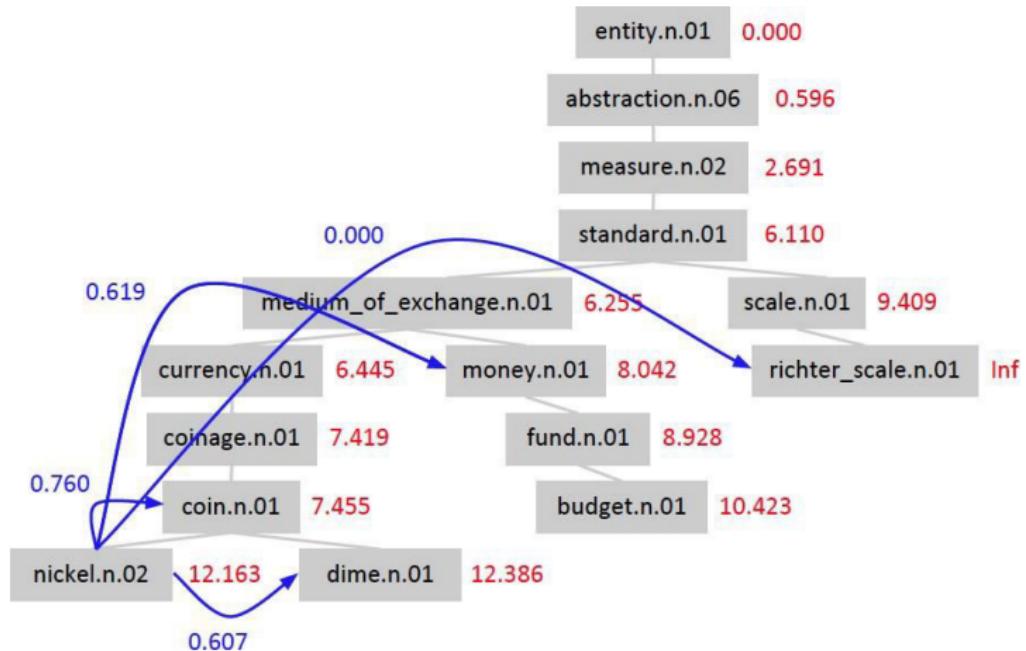
Proportion of shared information

- It's not just about commonalities - it's also about differences!
- **Resnik:** The more information content they share, the more similar they are
- **Lin:** The more information content they don't share, the less similar they are
- Not the *absolute* quantity of shared information but the *proportion* of shared information

$$sim_{Lin}(c_1, c_2) = \frac{2\log P(LCS(c_1, c_2))}{\log P(c_1) + \log P(c_2)}$$

The information content common to c_1 and c_2 , normalized by their average information content.

Example: Lin similarity



Jiang-Conrath distance

JC similarity

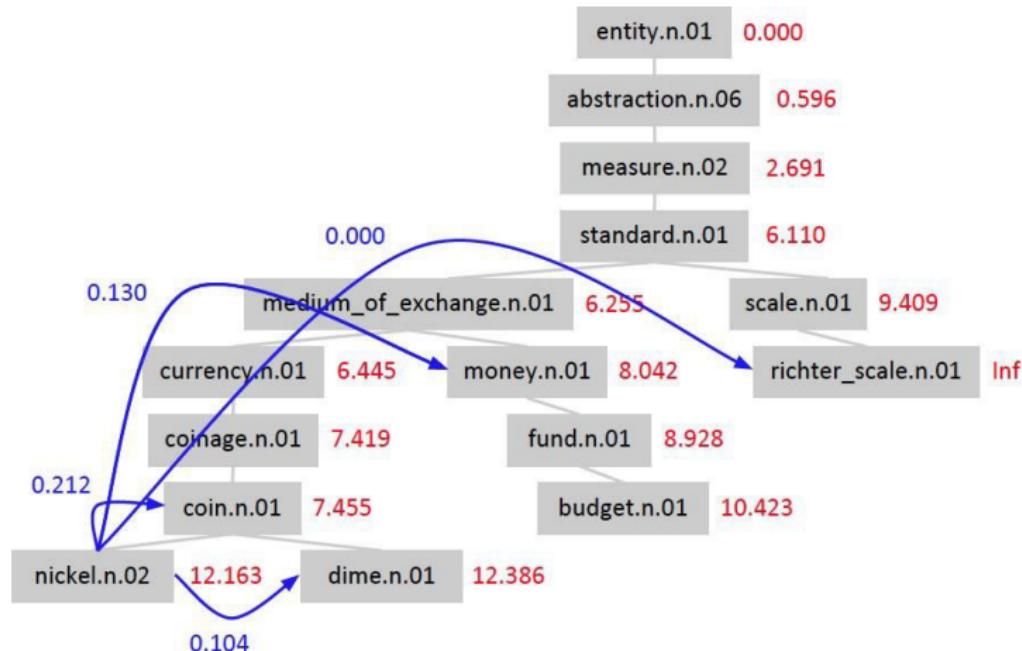
We can use IC to assign lengths to graph edges:

$$dist_{JC}(c, \text{hypernym}(c)) = IC(c) - IC(\text{hypernym}(c))$$

$$\begin{aligned} dist_{JC}(c_1, c_2) &= dist_{JC}(c_1, LCS(c_1, c_2)) + dist_{JC}(c_2, LCS(c_1, c_2)) \\ &= IC(c_1) - IC(LCS(c_1, c_2)) + IC(c_2) - IC(LCS(c_1, c_2)) \\ &= IC(c_1) + IC(c_2) - 2 \times IC(LCS(c_1, c_2)) \end{aligned}$$

$$sim_{JC}(c_1, c_2) = \frac{1}{IC(c_1) + IC(c_2) - 2 \times IC(LCS(c_1, c_2))}$$

Example: Jiang-Conrath distance



The (extended) Lesk Algorithm

- 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 occurs in both glosses, add a score of n^2
- **paper and specially prepared** → $1 + 4 = 5$

Problem in mapping words to wordnet senses

I saw a man who is 98 years old and can still walk and tell jokes

Ambiguity is rampant!

I saw a man who is 98 years old and can still walk and tell jokes



Word Sense Disambiguation - I

Pawan Goyal

CSE, IIT Kharagpur

Week 8, Lecture 3

Word Sense Disambiguation (WSD)

Sense ambiguity

- Many words have several meanings or senses
- The meaning of **bass** depends on the context
- Are we talking about music, or fish?
 - ▶ An electric guitar and **bass** player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.
 - ▶ And it all started when fishermen decided the striped **bass** in Lake Mead were too skinny.

Disambiguation

- The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word.
- This is done by looking at the context of the word's use.

Algorithms

- Knowledge Based Approaches
 - ▶ Overlap Based Approaches
- Machine Learning Based Approaches
 - ▶ Supervised Approaches
 - ▶ Semi-supervised Algorithms
 - ▶ Unsupervised Algorithms
- Hybrid Approaches

Knowledge Based Approaches

Overlap Based Approaches

- Require a **Machine Readable Dictionary (MRD)**.
- Find the overlap between the features of different senses of an ambiguous word (**sense bag**) and the features of the words in its context (**context bag**).
- The features could be sense definitions, example sentences, hypernyms etc.
- The features could also be given weights.
- The sense which has the maximum overlap is selected as the contextually appropriate sense.

Lesk's Algorithm

Sense Bag: contains the words in the definition of a candidate sense of the ambiguous word.

Context Bag: contains the words in the definition of each sense of each context word.

On burning coal we get ash.

Ash

- Sense 1
Trees of the olive family with pinnate leaves, thin furrowed bark and gray branches.
- Sense 2
The **solid** residue left when **combustible** material is thoroughly **burned** or oxidized.
- Sense 3
To convert into ash

Coal

- Sense 1
A piece of glowing carbon or **burnt** wood.
- Sense 2
charcoal.
- Sense 3
A black **solid combustible** substance formed by the partial decomposition of vegetable matter without free access to air and under the influence of moisture and often increased pressure and temperature that is widely used as a fuel for **burning**

In this case Sense 2 of ash would be the winner sense.

Walker's Algorithm

- A Thesaurus Based approach
- **Step 1:** For each sense of the target word find the thesaurus category to which that sense belongs
- **Step 2:** Calculate the score for each sense by using the context words.
A context word will add 1 to the score of the sense if the thesaurus category of the word matches that of the sense.
 - ▶ E.g. The money in this bank fetches an interest of 8% per annum
 - ▶ Target word: *bank*
 - ▶ Clue words from the context: *money, interest, annum, fetch*

	Sense1: Finance	Sense2: Location
Money	+1	0
Interest	+1	0
Fetch	0	0
Annum	+1	0
Total	3	0

Context words add 1 to the sense when the topic of the word matches that of the sense

WSD Using Random Walk Algorithm

The church bells no longer rung on Sundays.

church

- 1: one of the groups of Christians who have their own beliefs and forms of worship
- 2: a place for public (especially Christian) worship
- 3: a service conducted in a church

bell

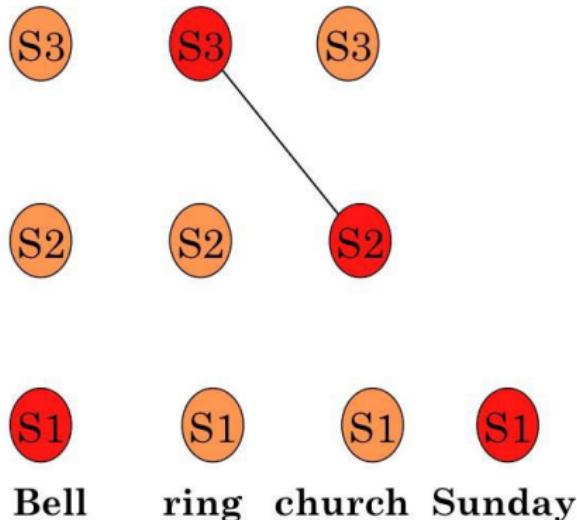
- 1: a hollow device made of metal that makes a ringing sound when struck
- 2: a push button at an outer door that gives a ringing or buzzing signal when pushed
- 3: the sound of a bell

ring

- 1: make a ringing sound
- 2: ring or echo with sound
- 3: make (bells) ring, often for the purposes of musical edification

Sunday

- 1: first day of the week; observed as a day of rest and worship by most Christians



Step 1: Add a vertex for each possible sense of each word in the text.

WSD Using Random Walk Algorithm

The church bells no longer rung on Sundays.

church

- 1: one of the groups of Christians who have their own beliefs and forms of worship
- 2: a place for public (especially Christian) worship
- 3: a service conducted in a church

bell

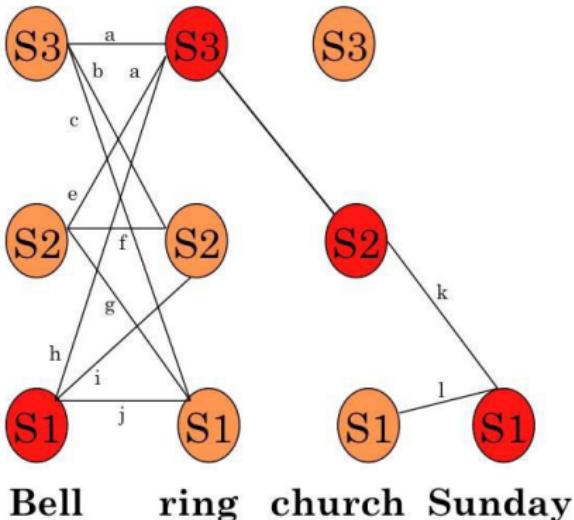
- 1: a hollow device made of metal that makes a ringing sound when struck
- 2: a push button at an outer door that gives a ringing or buzzing signal when pushed
- 3: the sound of a bell

ring

- 1: make a ringing sound
- 2: ring or echo with sound
- 3: make (bells) ring, often for the purposes of musical edification

Sunday

- 1: first day of the week; observed as a day of rest and worship by most Christians



Step 2: Add weighted edges using definition based semantic similarity (Lesk's method).

WSD Using Random Walk Algorithm

The church bells no longer rung on Sundays.

church

- 1: one of the groups of Christians who have their own beliefs and forms of worship
- 2: a place for public (especially Christian) worship
- 3: a service conducted in a church

bell

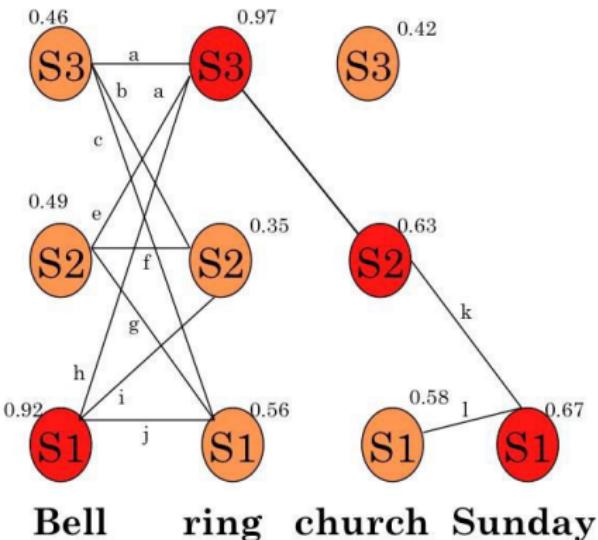
- 1: a hollow device made of metal that makes a ringing sound when struck
- 2: a push button at an outer door that gives a ringing or buzzing signal when pushed
- 3: the sound of a bell

ring

- 1: make a ringing sound
- 2: ring or echo with sound
- 3: make (bells) ring, often for the purposes of musical edification

Sunday

- 1: first day of the week; observed as a day of rest and worship by most Christians



Step 3: Apply graph based ranking algorithm to find score of each vertex (i.e. for each word sense).

WSD Using Random Walk Algorithm

The church bells no longer rung on Sundays.

church

- 1: one of the groups of Christians who have their own beliefs and forms of worship
- 2: a place for public (especially Christian) worship
- 3: a service conducted in a church

bell

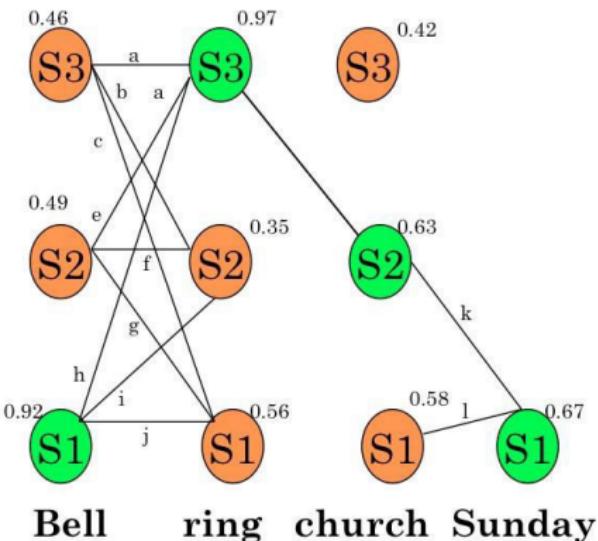
- 1: a hollow device made of metal that makes a ringing sound when struck
- 2: a push button at an outer door that gives a ringing or buzzing signal when pushed
- 3: the sound of a bell

ring

- 1: make a ringing sound
- 2: ring or echo with sound
- 3: make (bells) ring, often for the purposes of musical edification

Sunday

- 1: first day of the week; observed as a day of rest and worship by most Christians



Step 4: Select the vertex (sense) which has the highest score.

Naïve Bayes for WSD

- A Naïve Bayes classifier chooses the most likely sense for a word given the features of the context:

$$\hat{s} = \arg \max_{s \in S} P(s|f)$$

- Using Bayes' law, this can be expressed as:

$$\begin{aligned}\hat{s} &= \arg \max_{s \in S} \frac{P(s)P(f|s)}{P(f)} \\ &= \arg \max_{s \in S} P(s)P(f|s)\end{aligned}$$

- The ‘Naïve’ assumption: all the features are conditionally independent, given the sense’:

$$\hat{s} = \operatorname{argmax}_{s \in S} \prod_{j=1}^n P(f_j|s)$$

Training for Naïve Bayes

- ‘ f ’ is a feature vector consisting of:
 - ▶ POS of w
 - ▶ Semantic and Syntactic features of w
 - ▶ Collocation vector (set of words around it) → next word (+1), +2, -1, -2 and their POS’s
 - ▶ Co-occurrence vector
- Set parameters of Naïve Bayes using maximum likelihood estimation (MLE) from training data

$$P(s_i) = \frac{\text{count}(s_i, w_j)}{\text{count}(w_j)}$$

$$P(f_j | s_i) = \frac{\text{count}(f_j, s_i)}{\text{count}(s_i)}$$

Decision List Algorithm

- Based on ‘One sense per collocation’ property
 - ▶ Nearby words provide strong and consistent clues as to the sense of a target word
- Collect a large set of collocations for the ambiguous word
- Calculate word-sense probability distributions for all such collocations
- Calculate the log-likelihood ratio

$$\log\left(\frac{P(\text{Sense}-A|\text{Collocation}_i)}{P(\text{Sense}-B|\text{Collocation}_i)}\right)$$

- Higher log-likelihood \Rightarrow more predictive evidence
- Collocations are ordered in a decision list, with most predictive collocations ranked highest

Decision List Algorithm

Training Data

Sense	Training Examples (Keyword in Context)
A	used to strain microscopic <i>plant</i> life from the ...
A	... zonal distribution of <i>plant</i> life ...
A	close-up studies of <i>plant</i> life and natural ...
A	too rapid growth of aquatic <i>plant</i> life in water ...
A	... the proliferation of <i>plant</i> and animal life ...
A	establishment phase of the <i>plant</i> virus life cycle ...
B
B	computer manufacturing <i>plant</i> and adjacent ...
B	discovered at a St. Louis <i>plant</i> manufacturing
B	... copper manufacturing <i>plant</i> found that they
B	copper wire manufacturing <i>plant</i> , for example ...
B	's cement manufacturing <i>plant</i> in Alpena ...
B	polystyrene manufacturing <i>plant</i> at its Dow ...
B	company manufacturing <i>plant</i> is in Orlando ...

Resultant Decision List

Final decision list for <i>plant</i> (abbreviated)		
Log L	Collocation	Sense
10.12	<i>plant</i> growth	⇒ A
9.68	car (within $\pm k$ words)	⇒ B
9.64	<i>plant</i> height	⇒ A
9.61	union (within $\pm k$ words)	⇒ B
9.54	equipment (within $\pm k$ words)	⇒ B
9.51	assembly <i>plant</i>	⇒ B
9.50	nuclear <i>plant</i>	⇒ B
9.31	flower (within $\pm k$ words)	⇒ A
9.24	job (within $\pm k$ words)	⇒ B
9.03	fruit (within $\pm k$ words)	⇒ A
9.02	<i>plant</i> species	⇒ A
...	...	

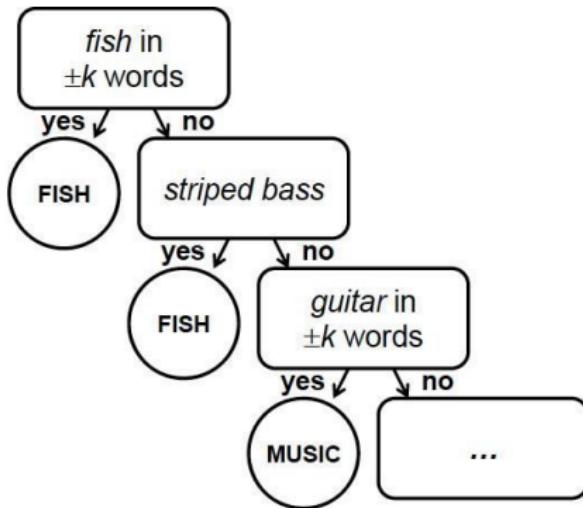
Classification of a test sentence is based on the highest ranking collocation, found in the test sentences.

plucking flowers affects plant growth.

Decision List: Example

Example: discriminating between bass (fish) and bass (music):

Context	Sense
<i>fish in $\pm k$ words</i>	FISH
<i>striped bass</i>	FISH
<i>guitar in $\pm k$ words</i>	MUSIC
<i>bass player</i>	MUSIC
<i>piano in $\pm k$ words</i>	MUSIC
<i>sea bass</i>	FISH
<i>play bass</i>	MUSIC
<i>river in $\pm k$ words</i>	FISH
<i>on bass</i>	MUSIC
<i>bass are</i>	FISH



Word Sense Disambiguation - II

Pawan Goyal

CSE, IIT Kharagpur

Week 8, Lecture 4

Minimally Supervised WSD - Yarowsky

- Annotations are expensive!
- “Bootstrapping” or co-training
 - Start with (small) seed, learn decision list
 - Use decision list to label rest of corpus
 - Retain ‘confident’ labels, treat as annotated data to learn new decision list
 - Repeat ...
- Heuristics (derived from observation):
 - One sense per discourse
 - One sense per collocation

More about heuristics

One Sense per Discourse

- A word tends to preserve its meaning across all its occurrences in a given discourse

One Sense per Collocation

- A word tends to preserve its meaning when used in the same collocation
 - ▶ Strong for adjacent collocations
 - ▶ Weaker as the distance between the words increases

Yarowsky's Method

Example

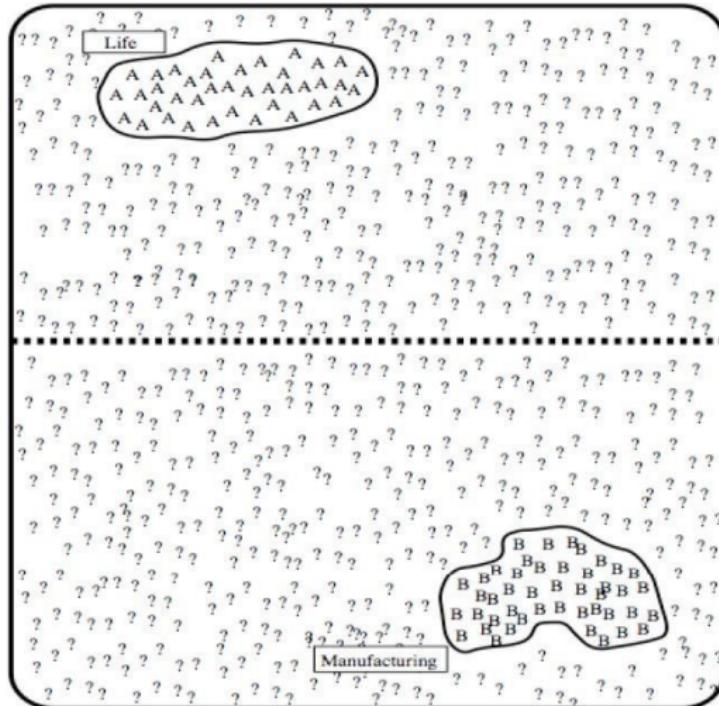
- Disambiguating plant (industrial sense) vs. plant (living thing sense)
- Think of seed features for each sense
 - Industrial sense: co-occurring with ‘manufacturing’
 - Living thing sense: co-occurring with ‘life’
- Use ‘one sense per collocation’ to build initial decision list classifier
- Treat results (having high probability) as annotated data, train new decision list classifier, iterate

Yarowsky's Method: Example

used to strain microscopic plant life from the zonal distribution of plant life .
close-up studies of plant life and natural too rapid growth of aquatic plant life in water
the proliferation of plant and animal life establishment phase of the plant virus life cycle
that divide life into plant and animal kingdom many dangers to plant and animal life
mammals . Animal and plant life are delicately automated manufacturing plant in Fremont
vast manufacturing plant and distribution
chemical manufacturing plant , producing viscose
keep a manufacturing plant profitable without computer manufacturing plant and adjacent discovered at a St. Louis plant manufacturing copper manufacturing plant found that they copper wire manufacturing plant , for example s cement manufacturing plant in Alpena

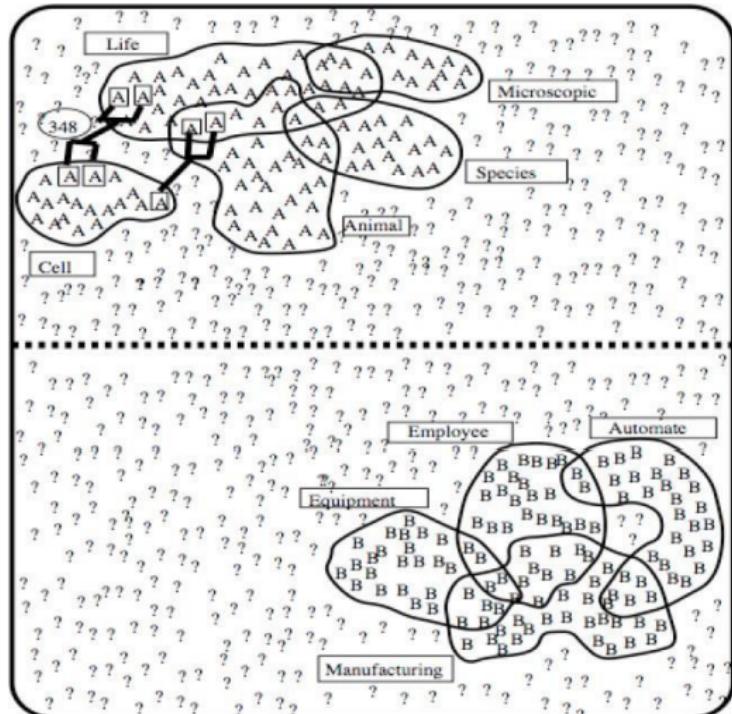
vinyl chloride monomer plant , which is molecules found in plant and animal tissue Nissan car and truck plant in Japan is and Golgi apparatus of plant and animal cells union responses to plant closures .
cell types found in the plant kingdom are company said the plant is still operating Although thousands of plant and animal species animal rather than plant tissues can be

Yarowsky's Method: Example



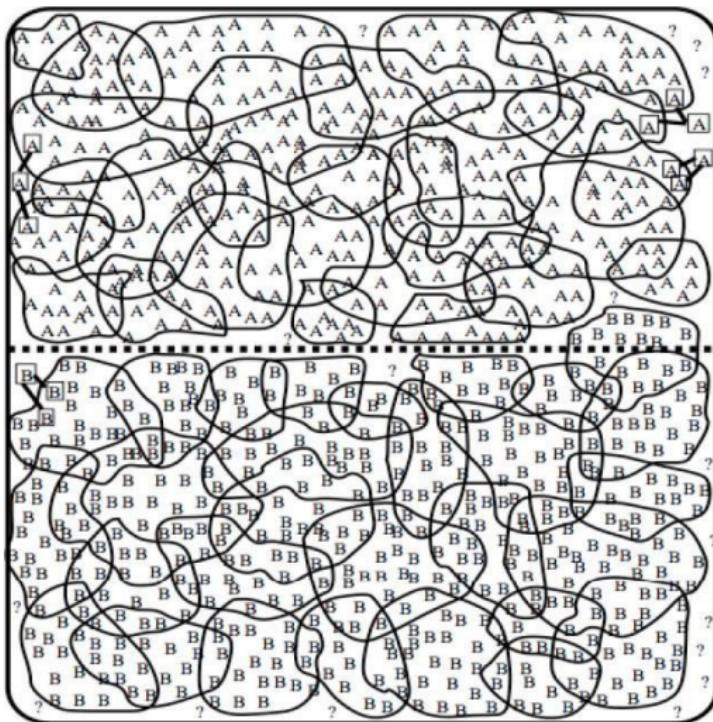
Initial state after use of seed rules

Yarowsky's Method: Example



Intermediate state

Yarowsky's Method: Example



Final state

Yarowsky's Method

Termination

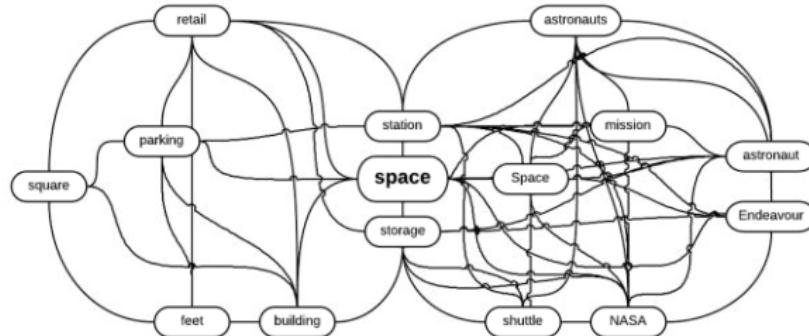
- Stop when
 - ▶ Error on training data is less than a threshold
 - ▶ No more training data is covered
- Use final decision list for WSD

Advantages

- Accuracy is about as good as a supervised algorithm
- Bootstrapping: far less manual effort

Key Idea: Word Sense Induction

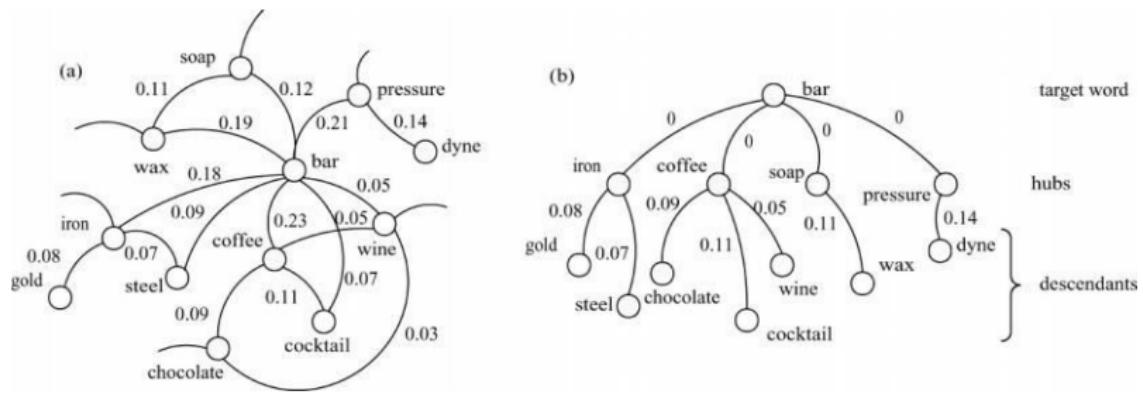
- Instead of using “dictionary defined senses”, extract the “senses from the corpus” itself
- These “corpus senses” or “uses” correspond to clusters of similar contexts for a word.



Detecting Root Hubs

- Different uses of a target word form highly interconnected bundles (or high density components)
- In each high density component one of the nodes (hub) has a higher degree than the others.
- **Step 1:** Construct co-occurrence graph, G .
- **Step 2:** Arrange nodes in G in decreasing order of degree.
- **Step 3:** Select the node from G which has the highest degree. This node will be the hub of the first high density component.
- **Step 4:** Delete this hub and all its neighbors from G .
- **Step 5:** Repeat Step 3 and 4 to detect the hubs of other high density components

HyperLex: Detecting Root Hubs



Delineating Components

- Attach each node to the root hub closest to it.
- The distance between two nodes is measured as the smallest sum of weights of the edges on the paths linking them.

Computing distance between two nodes w_i and w_j

$$w_{ij} = 1 - \max\{P(w_i|w_j), P(w_j|w_i)\}$$

where $P(w_i|w_j) = \frac{\text{freq}_{ij}}{\text{freq}_j}$

Disambiguation

- Let $W = (w_1, w_2, \dots, w_i, \dots, w_n)$ be a context in which w_i is an instance of our target word.
- Let w_i has k hubs in its minimum spanning tree
- A score vector s is associated with each $w_j \in W (j \neq i)$, such that s_k represents the contribution of the k th hub as:

$$s_k = \frac{1}{1 + d(h_k, w_j)} \text{ if } h_k \text{ is an ancestor of } w_j$$

$$s_i = 0 \text{ otherwise.}$$

- All score vectors associated with all $w_j \in W (j \neq i)$ are summed up
- The hub which receives the maximum score is chosen as the most appropriate sense

Novel Word Sense Detection

Pawan Goyal

CSE, IIT Kharagpur

Week 8, Lecture 5

Tracking Sense Changes

Classical sense

sick  adjective \sik\

: affected with a disease or illness

: of or relating to people who are ill

: very annoyed or bored by something because you have had too much of it

1

Novel sense

 Favorited 85,144 times

 Niall Horan @NiallOfficial · Apr 24
Listening to Paulo nutini 's new record! It's sick !

[Collapse](#) [Reply](#) [Retweet](#) [Favorite](#) [More](#)

RETWEETS	FAVORITES
47,293	85,145

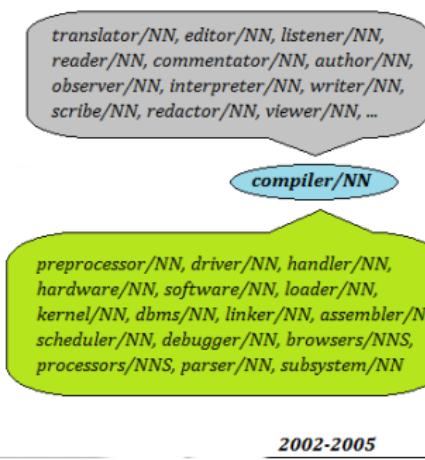
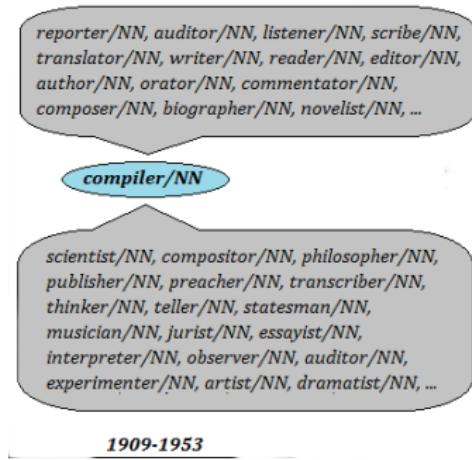


11:50 PM - 24 Apr 2014 · Details

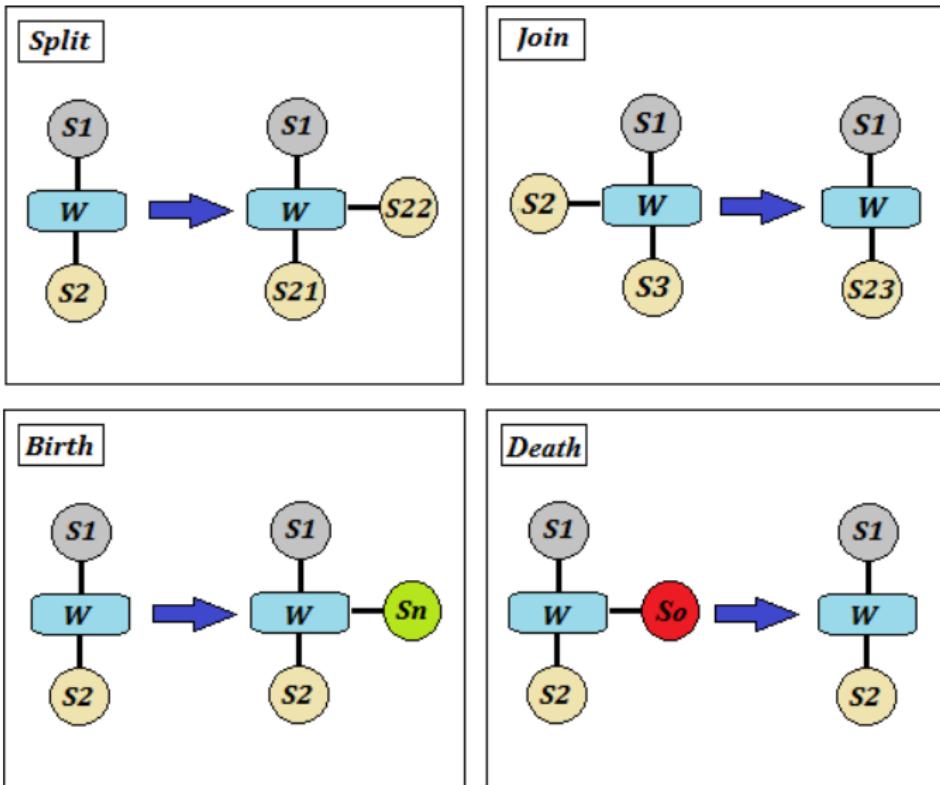
¹<http://www.merriam-webster.com/>

Comparing sense clusters

- If a word undergoes sense change, this can be detected by comparing the sense clusters obtained from two different time periods



Split, join, birth and death



A real example of birth

