TTIC 31190: Natural Language Processing

Kevin Gimpel

Spring 2018

Lecture 13: Brown Clustering; Syntax

Project Proposal

- project proposal due Wednesday

- if you’re still scrambling for project ideas, let me know and I can suggest a “default” project

Midterm

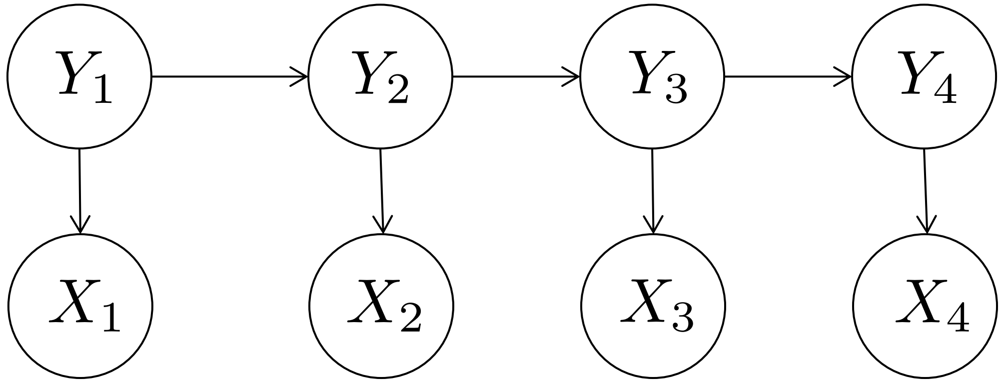
- midterm on Wednesday, May 16th
- you can bring notes
- we'll try to give you all the formulas/definitions you will need

Roadmap

- words, morphology, lexical semantics
- text classification
- language modeling
- word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- semantics, compositionality, semantic parsing
- machine translation and other NLP tasks

Graphical Model for an HMM

(for a sequence of length 4)

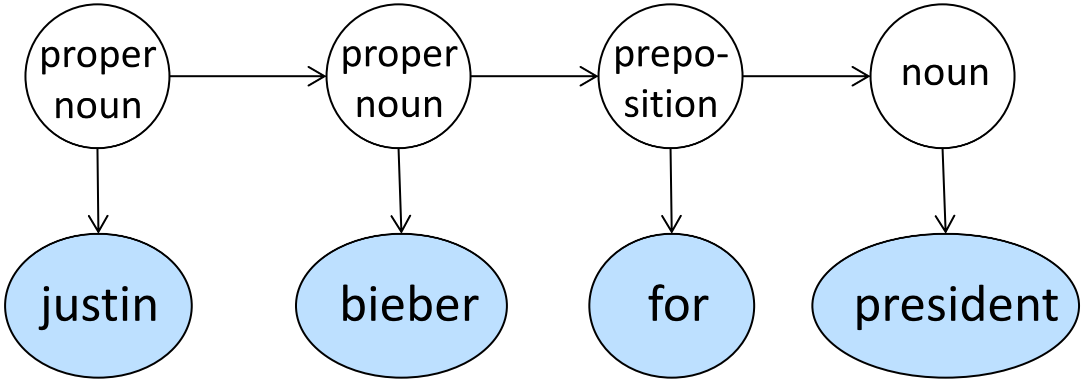


$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\tau}}(</\mathrm{s}> \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

transition parameters: $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

emission parameters: $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

HMMs for Part-of-Speech Tagging



each $y_i \in \mathcal{L}$ is a part-of-speech tag
so, label space is $\mathcal{L} = \{\text{noun}, \text{verb}, \ldots\}$

transition parameters: $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

emission parameters: $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

$$p_{\boldsymbol{\tau}}(y \mid y') \leftarrow \frac{\text{count}(y'\ y)}{\text{count}(y')}$$

Inference in HMMs

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y}} \; p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$$

$$= \operatorname*{argmax}_{\boldsymbol{y}} \; p_{\boldsymbol{\tau}}(</\text{s}> \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

- output is a sequence; argmax requires iterating over an exponentially-large set
- **dynamic programming (DP)** can be used to solve such problems exactly
- for HMMs (& other sequence models): **Viterbi algorithm**

Viterbi Algorithm for HMMs

- recursive equations + memoization:

base case:
returns probability of sequence starting with label *y* for first word

$$V(1, y) = p_{\boldsymbol{\eta}}(x_1 \mid y)\, p_{\boldsymbol{\tau}}(y \mid <\mathrm{s}>)$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

recursive case:
computes probability of max-probability label sequence that ends with label *y* at position *m*

final value is in: $goal(\boldsymbol{x}) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\tau}}(</\mathrm{s}> \mid y')\, V(|\boldsymbol{x}|, y') \right)$

Viterbi Algorithm for Sequence Models

(with tag bigram features)

$$V(1, y) = \text{score}(\boldsymbol{x}, \langle\text{<s>}, y\rangle, 1, \boldsymbol{w})$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, \langle y', y\rangle, m, \boldsymbol{w}) + V(m-1, y')\right)$$

score function for label bigram <*y'*, *y*>
ending at position *m* in ***x***

could be anything!
linear model, feed-forward network,
LSTM, etc.

Viterbi Algorithm for Sequence Models

(with tag bigram features)

$$V(1, y) = \mathrm{score}(\boldsymbol{x}, \langle <\mathrm{s}>, y \rangle, 1, \boldsymbol{w})$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(\mathrm{score}(\boldsymbol{x}, \langle y', y \rangle, m, \boldsymbol{w}) + V(m-1, y')\right)$$

score function for label bigram <*y*’, *y*>
ending at position *m* in ***x***

score for entire sequence:

$$\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \sum_i \mathrm{score}(\boldsymbol{x}, \langle y_{i-1}, y_i \rangle, i, \boldsymbol{w})$$

(with appropriate handling of $<\mathrm{s}>$ and $</\mathrm{s}>$)

Viterbi Algorithm for Sequence Models
(with tag bigram features)

base case:
returns score of sequence starting with label *y* for word 1

$$V(1, y) = \text{score}(\boldsymbol{x}, \langle \text{<s>}, y\rangle, 1, \boldsymbol{w})$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, \langle y', y\rangle, m, \boldsymbol{w}) + V(m-1, y')\right)$$

recursive case:
computes score of max-scoring label sequence that ends with label *y* at position *m*

Sequence Models with Tag Bigram Features

$$\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \sum_i \mathrm{score}(\boldsymbol{x}, \langle y_{i-1}, y_i \rangle, i, \boldsymbol{w})$$

to recover HMM from this generalized score function, use the linear model features and weights we discussed last time

Structured Models with Parts

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \sum_i \text{score}(\boldsymbol{x}, \langle y_{i-1}, y_i \rangle, i, \boldsymbol{w})$$

$$\Downarrow$$

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \sum_{\langle \boldsymbol{x}_r, \boldsymbol{y}_r \rangle \in \text{parts}(\boldsymbol{x}, \boldsymbol{y})} \text{score}(\boldsymbol{x}_r, \boldsymbol{y}_r, \boldsymbol{w})$$

in general: structured prediction relies on decomposition of input/output into "parts"

score function defined on individual parts

size of each part (in terms of ***y***) affects complexity of inference

"Brown Clustering"

Class-Based *n*-gram Models of Natural Language

Peter F. Brown*
Peter V. deSouza*
Robert L. Mercer*
Vincent J. Della Pietra*
Jenifer C. Lai*

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays

June March July April January December October November September August

people guys folks fellows CEOs chaps doubters commies unfortunates blokes

down backwards ashore sideways southward northward overboard aloft downwards adrift

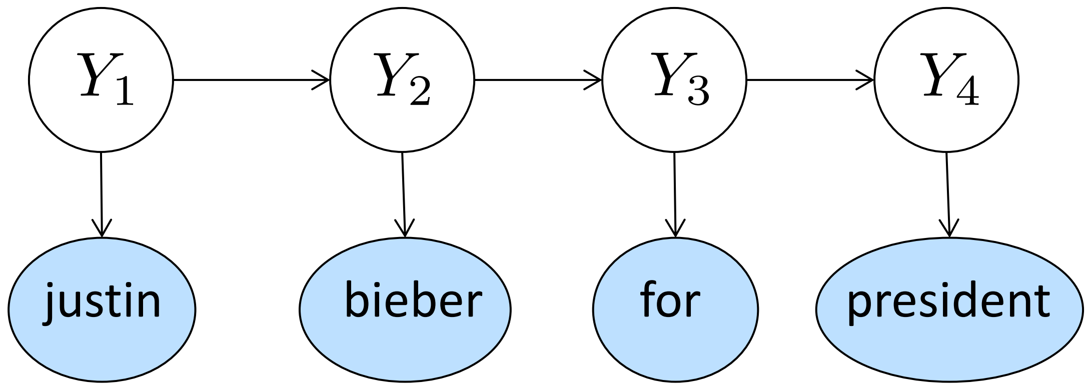
water gas coal liquid acid sand carbon steam shale iron

great big vast sudden mere sheer gigantic lifelong scant colossal

Computational Linguistics, 1992

HMMs for Word Clustering

(Brown et al., 1992)



each $y_i \in \mathcal{L}$ is a cluster ID

so, label space is $\mathcal{L} = \{1, 2, ..., 100\}$

HMMs for POS Tagging

each $y_i \in \mathcal{L}$ is a POS tag
so, label space is

$$\mathcal{L} = \{\text{noun}, \text{verb}, ...\}$$

transition parameters:

$$p_{\boldsymbol{\tau}}(\text{verb} \mid \text{noun})$$

$$p_{\boldsymbol{\tau}}(\text{noun} \mid \text{noun})$$

$$\ldots$$

emission parameters:

$$p_{\boldsymbol{\eta}}(\mathit{for} \mid \text{noun})$$

$$p_{\boldsymbol{\eta}}(\mathit{walk} \mid \text{noun})$$

$$\ldots$$

HMMs for Word Clustering

each $y_i \in \mathcal{L}$ is a cluster ID
so, label space is

$$\mathcal{L} = \{1, 2, ..., 100\}$$

transition parameters:

$$p_{\boldsymbol{\tau}}(1 \mid 17)$$

$$p_{\boldsymbol{\tau}}(2 \mid 17)$$

$$\ldots$$

emission parameters:

$$p_{\boldsymbol{\eta}}(\mathit{for} \mid 17)$$

$$p_{\boldsymbol{\eta}}(\mathit{walk} \mid 17)$$

$$\ldots$$

HMMs for Word Clustering

(Brown et al., 1992)

- given a set of sentences, how should we learn the parameters of our model?

- how about we use maximum likelihood estimation, e.g.:

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$$

- problem: we don't have any $\boldsymbol{y}^{(i)}$'s!
- we only have a set of **unlabeled** sentences: $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$

- we want to maximize likelihood, but:
  - our HMM defines $p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$
  - our data only contains $\boldsymbol{x}$
- solution: marginalize out $\boldsymbol{y}$
- this idea underlies much of unsupervised learning

$$\operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{N} \log p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$$

$$\Downarrow$$

$$\operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})$$

- we want to maximize likelihood, but:
  - our HMM defines $p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$
  - our data only contains $\boldsymbol{x}$
- solution: marginalize out $\boldsymbol{y}$
- this idea underlies much of unsupervised learning

a sum over an exponentially-large set

$$\operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})$$

- learning requires sum over exponentially-large set
  – all possible clusterings of the words

$$\operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})$$

- for HMMs, we can compute this sum in polynomial time using dynamic programming
- however, Brown clustering uses simplifying assumption: **each word is in exactly one cluster**
- the summation above becomes trivial to compute!

One Cluster Per Word Assumption

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})$$

- summation becomes trivial, so we can easily compute the log-likelihood of any given model
- but how should we do learning?
  - gradient-based optimization is tricky due to the constraints
  - and it’s too expensive to iterate through all possible clusterings and compute their likelihoods

Algorithm for Brown Clustering

greedy algorithm:

- initialize each word as its own cluster
- greedily merge clusters to improve data likelihood

we induced 1000 Brown clusters from 56 million English tweets (1 billion words)

only words that appeared at least 40 times

(Owoputi, O'Connor, Dyer, Gimpel, Schneider, and Smith, 2013)

Example Cluster

missed loved hated misread admired underestimated resisted adored disliked regretted missd fancied luved prefered luvd overdid mistyped misd misssed looooved misjudged lovedd loooved loathed lurves lovd

Example Cluster

missed loved hated misread admired
underestimated resisted adored disliked
regretted missd fancied luved prefered luvd
overdid mistyped misd misssed looooved
misjudged lovedd loooved loathed lurves lovd

spelling
variation

“really”

really rly realy genuinely rlly reallly realllly reallyy rele realli relly reallllly reli reali sholl rily reallyyy reeeeally realllllly reaally reeeally rili reaaally reaaaally reallyyyy rilly reallllllly reeeeeally reeally shol realllyyy reely relle reaaaaally shole really2 reallyyyyy _really_ realllllllly reaaly realllyy reallii reallt genuinly relli realllyyyy reeeeeeally weally reaaallly reallllyyy reallllllllly reaallly realyy /really/ reaaaaaally reallu reaaaallly reeaally rreally reallyreally eally reeeaaally reeeaaally reaallyy reallyyyyyy –really- reallyreallyreally rilli reallllyyyy relaly reallllyy really-really r3ally reeli reallie realllllyyy rli reallllllllllly reaaaly reeeeeeeally

“so”

soo sooo soooo sooooo soooooo sooooooo

soooooooo sooooooooo soooooooooo

sooooooooooo soooooooooooo

sooooooooooooo soso soooooooooooooo

sooooooooooooooo soooooooooooooooo

sososo superrr sooooooooooooooooo ssooo

so0o superrrr so0 soooooooooooooooooo

sosososo sooooooooooooooooooo ssoo sssooo

soooooooooooooooooooo #too s0o ssoooo s00

Food-Related Adjectives

hot fried peanut homemade grilled spicy soy cheesy coconut veggie roasted leftover blueberry icy dunkin mashed rotten mellow boiling crispy peppermint fruity toasted crunchy scrambled creamy boiled chunky funnel soggy clam steamed cajun steaming chewy steamy nacho mince reese's shredded salted glazed spiced venti pickled powdered butternut miso beet sizzling

“going to”

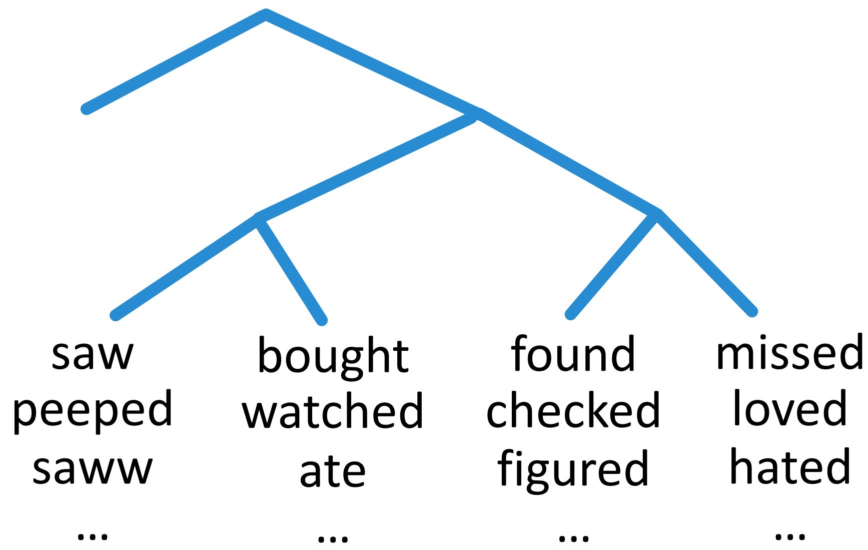
gonna gunna gona gna guna gnna ganna qonna gonnna gana qunna gonne goona gonnaa g0nna goina gonnah goingto gunnah gonaa gonan gunnna going2 gonnnnagunnaa gonny gunaa quna goonna qona gonns goinna gonnae qnna gonnaaa gnaa

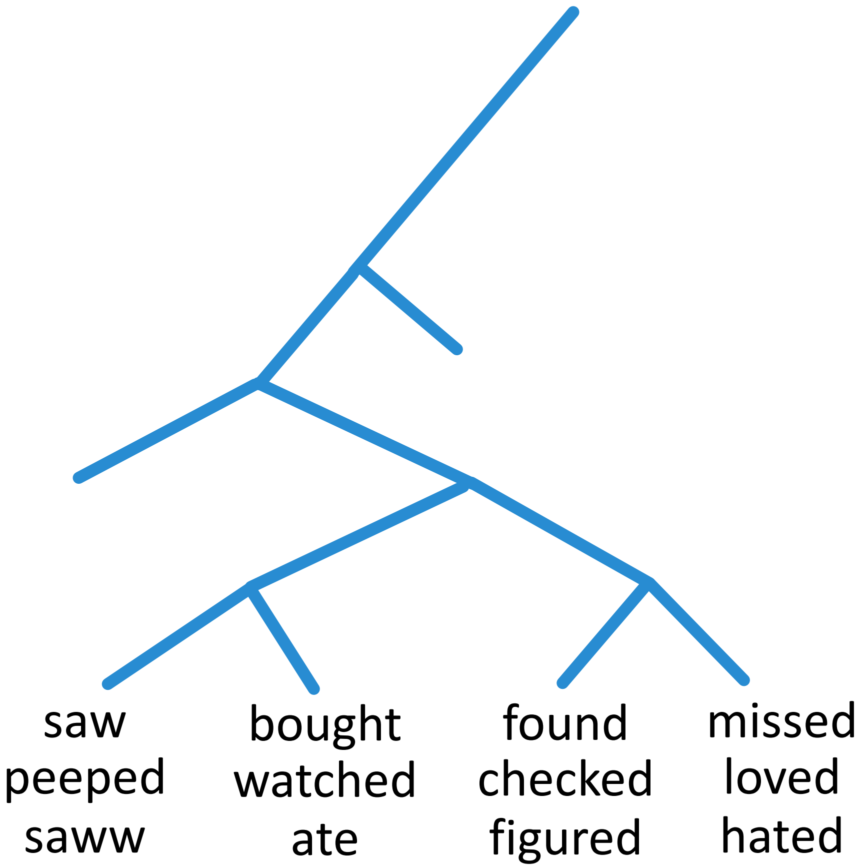
Adjective Intensifiers/Qualifiers

kinda hella sorta hecka kindof kindaa kinna hellla propa helluh kindda justa #slick helllla hela jii sortof hellaa kida wiggity hellllla hekka hellah kindaaa hellaaa kindah knda kind-of slicc wiggidy helllllla jih jye kinnda odhee kiinda heka sorda ohde kind've kidna baree rle hellaaaa jussa

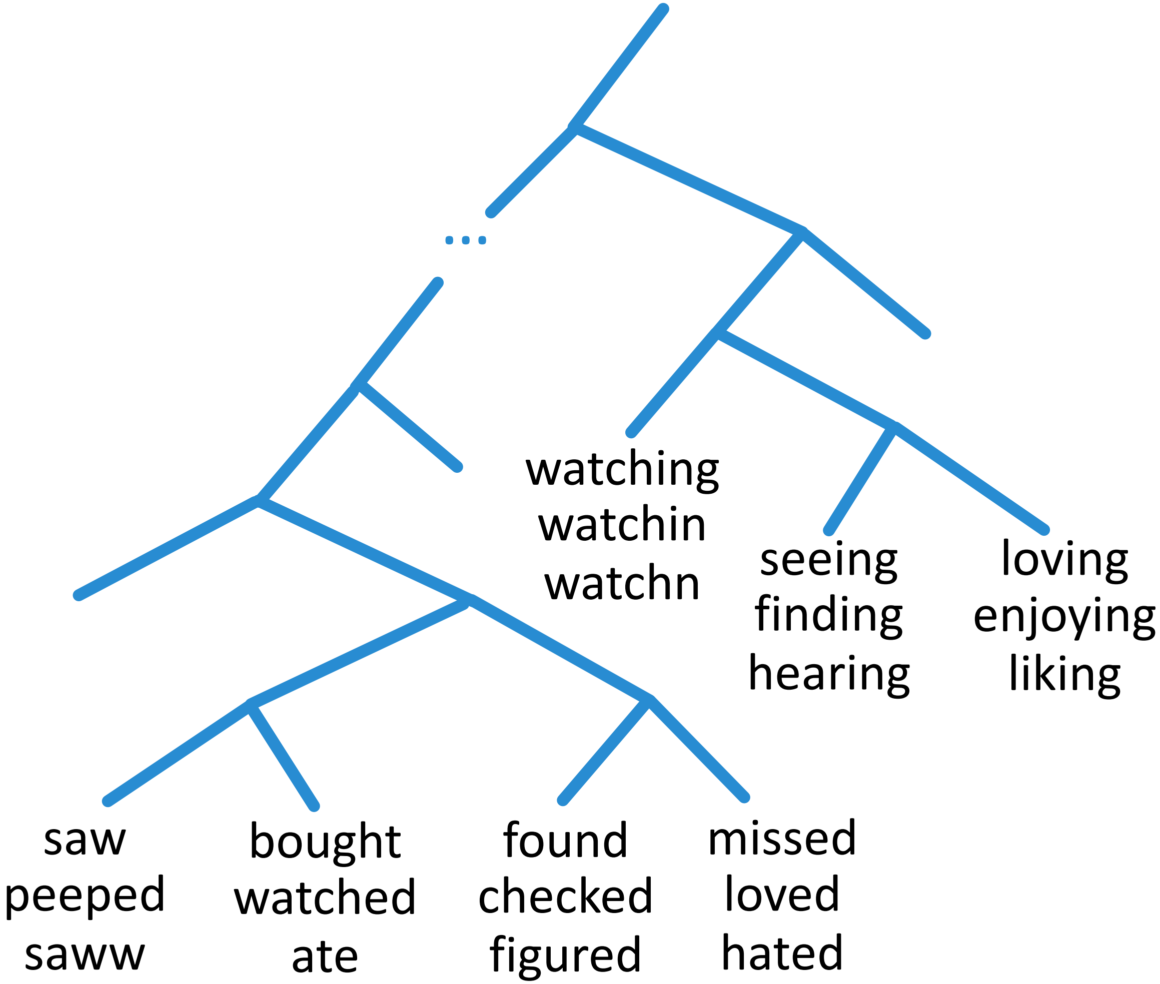
Hierarchical Clustering from Brown Clusters

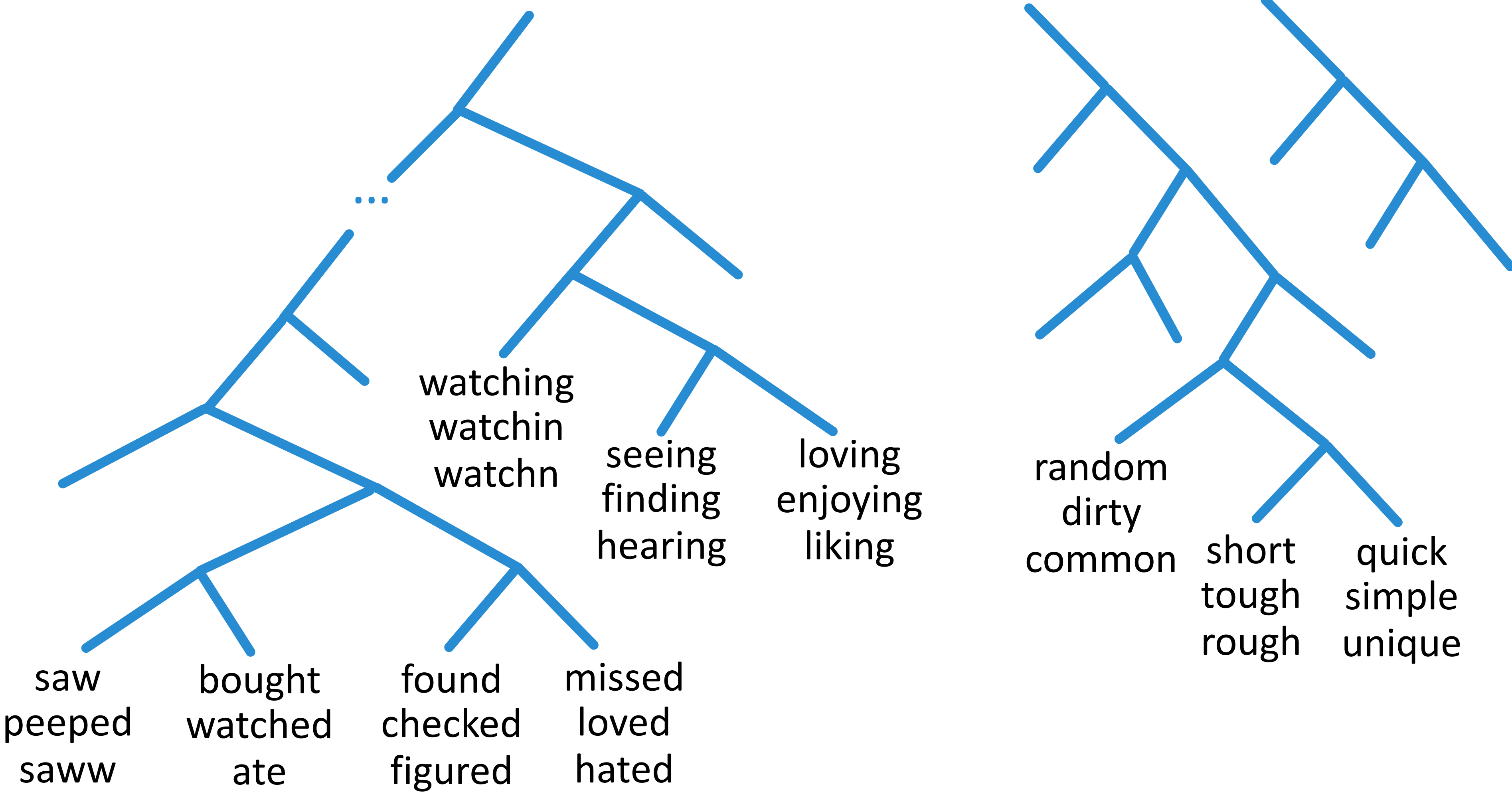
- we have a sequence of clustering decisions
- we can use the order in which clusters are merged to create a **hierarchical clustering**
  - build a binary tree to represent the merges
  - word similarity is captured by distance in the tree

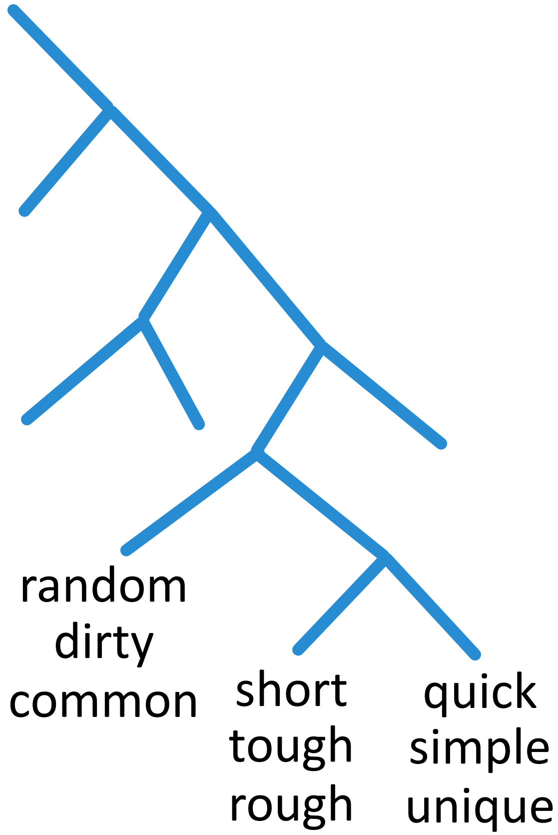


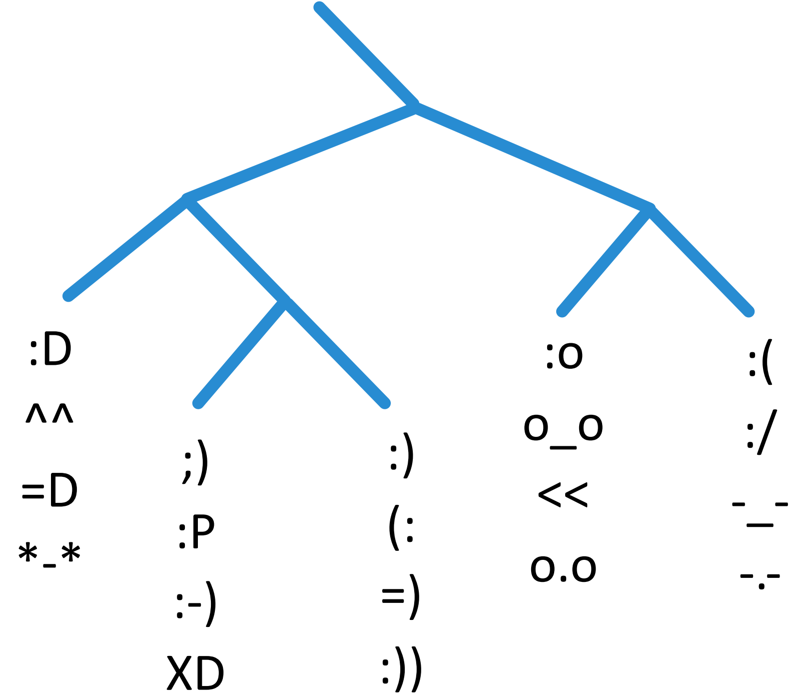


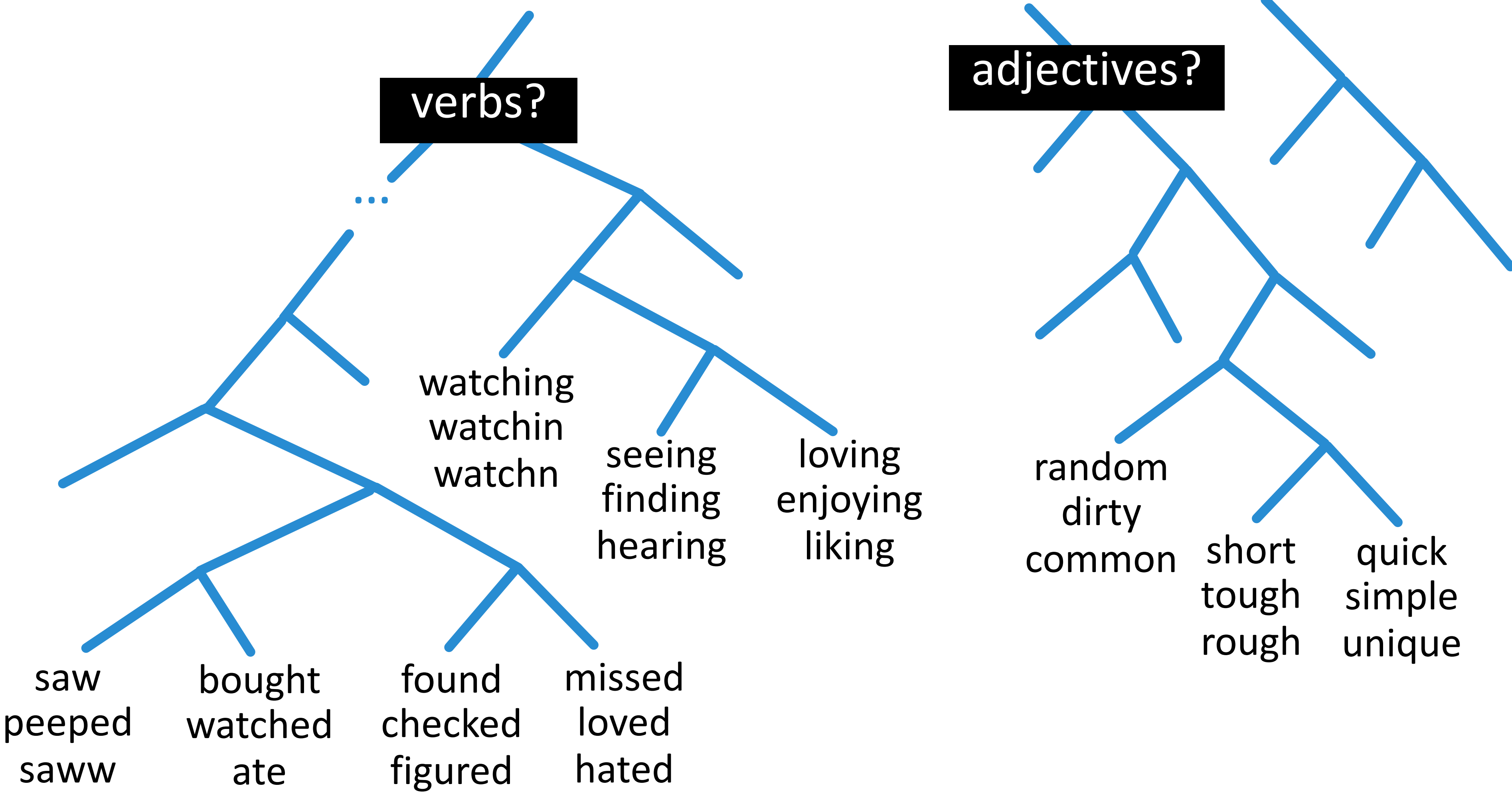
saw
peeped
saww
bought
watched
ate
found
checked
figured
missed
loved
hated

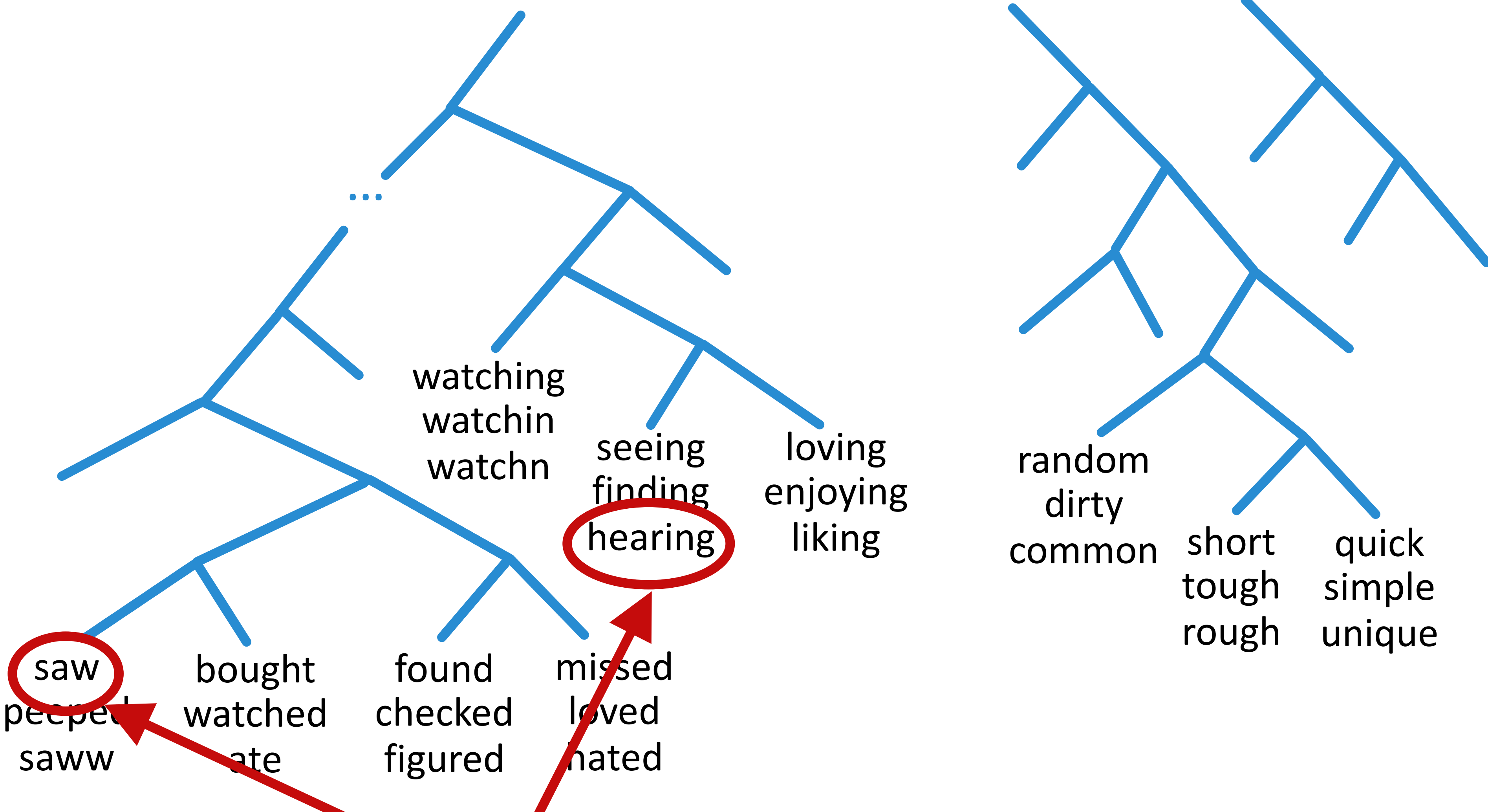

...
watching
watchin
watchn
seeing
finding
hearing
loving
enjoying
liking
saw
peeped
saww
bought
watched
ate
found
checked
figured
missed
loved
hated


...
watching
watchin
watchn
seeing
finding
hearing
loving
enjoying
liking
saw
peeped
saww
bought
watched
ate
found
checked
figured
missed
loved
hated
random
dirty
common
short
tough
rough
quick
simple
unique

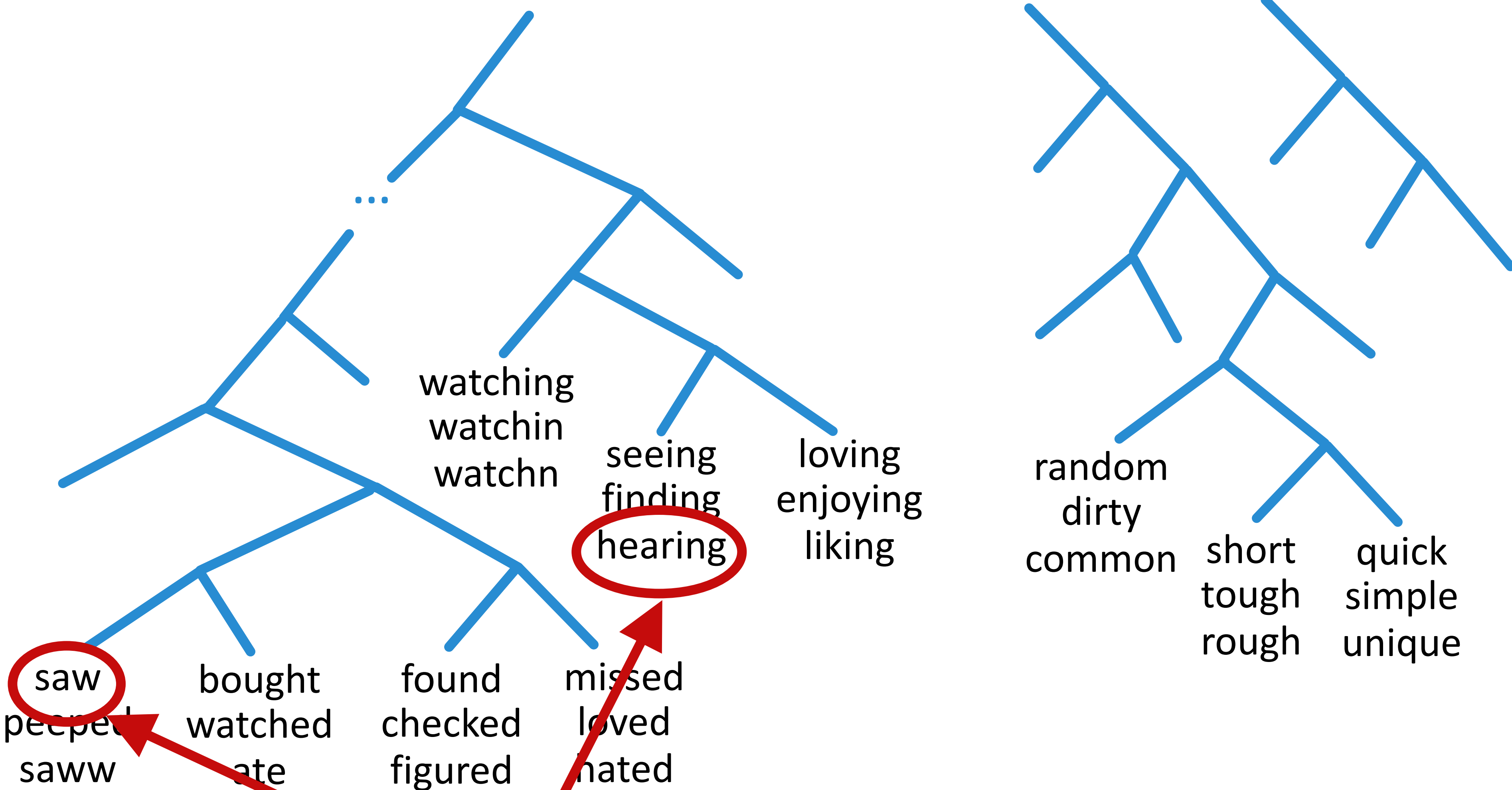

random
dirty
common
short
tough
rough
quick
simple
unique


:D
^^
=D
_
;)
:P
:-)
XD
:)
(:
=)
:))
:o
o_o
<<
o.o
:(
:/
-_-
-.-


verbs?
adjectives?
...
watching
watchin
watchn
seeing
finding
hearing
loving
enjoying
liking
random
dirty
common
short
tough
rough
quick
simple
unique
saw
peeped
saww
bought
watched
ate
found
checked
figured
missed
loved
hated



could be verbs or nouns, but
Brown clustering uses one-cluster-per-word constraint

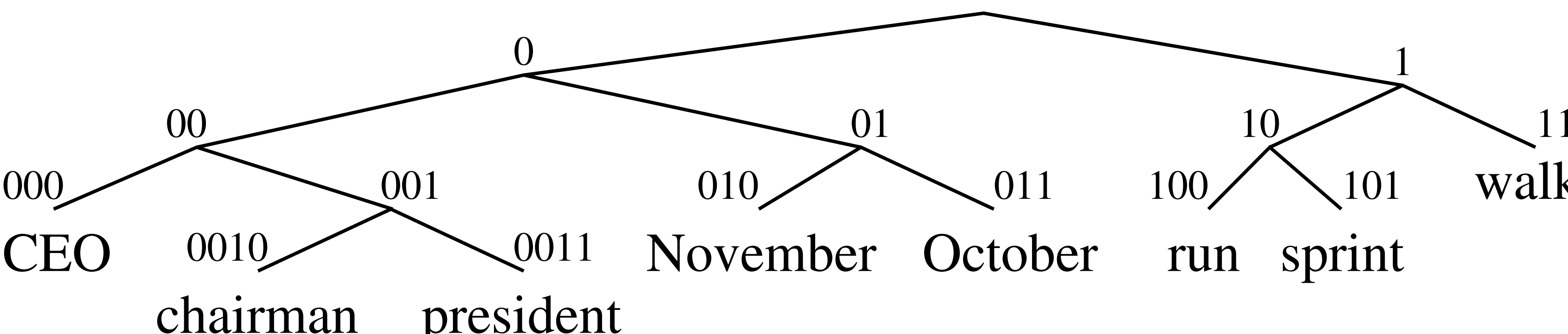


could be verbs or nouns, but one-cluster-per-word

even so, Brown clusters are good unsupervised POS tags!

Brown Clusters as Vectors

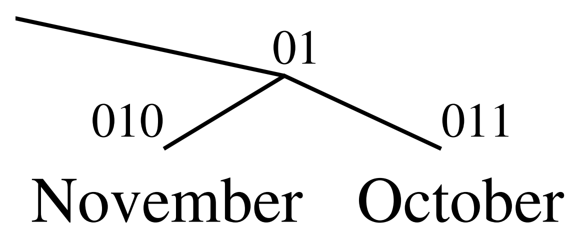
- cluster merging represented by a binary tree
- each word can be represented by its binary string = path from root to leaf
- each intermediate node is a cluster
- *chairman* is 0010, “months” = 01, verbs = 1:



Brown Clusters for Downstream Tasks

- even with one-cluster-per-word constraint, Brown clusters are really useful for syntactic tasks (tagging, parsing, etc.)

- also helps to use different granularities of the hierarchical clustering:
  – define features based on prefixes of binary strings:



  – November and October will share feature that looks at "length-2 prefix of binary string"

Roadmap

- words, morphology, lexical semantics
- text classification
- language modeling
- word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- semantics, compositionality, semantic parsing
- machine translation and other NLP tasks

What is Syntax?

- rules, principles, processes that govern sentence structure of a language
- can differ widely among languages
- but every language has systematic structural principles

Subject, Verb, Object

- syntax determines the ordering of these three objects in a sentence

Word order	English equivalent	Proportion of languages		Example languages
SOV	"She him loves."	45%		Hindi, Latin, Japanese, Marathi
SVO	"She loves him."	42%		English, Hausa, Mandarin, Russian
VSO	"Loves she him."	9%		Biblical Hebrew, Irish, Filipino, Tuareg
VOS	"Loves him she."	3%		Malagasy, Baure
OVS	"Him loves she."	1%		Apalaí, Hixkaryana
OSV	"Him she loves."	0%		Warao

Frequency distribution of word order in languages surveyed by Russell S. Tomlin in 1980s[1][2] (V • T • E)

Yodish

- often (though certainly not always) Yoda uses object-subject-verb order

“Powerful you have become. The dark side I sense in you.”

Grammars

- we will use **grammar** to denote a formal object that represents the rules/principles/processes that determine sentence structure

phrase structure / constituent grammar

- focuses on the **constituent** relation
- informally: “sentences have hierarchical structure”
- a sentence is made up of two pieces:
  - subject, typically a **noun phrase (NP)**
  - predicate, typically a **verb phrase (VP)**
- NPs and VPs are in turn made of up of pieces:
  - old books = (old + books)
  - the old books = (the + (old + books))
  - walked to the park = (walked + (to + (the + park)))
- each parenthesized phrase is a **constituent** in the **constituent parse**

Bracketing

- constituent parse = **bracketing** (that represents the hierarchical structure)
- e.g., sentence:

the man walked to the park

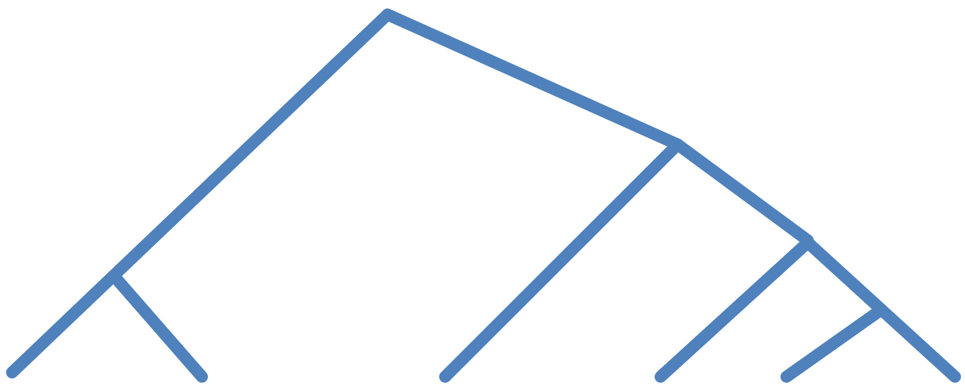
- bracketing:

((the man) (walked (to (the park))))

Bracketing → Tree

((the man) (walked (to (the park))))

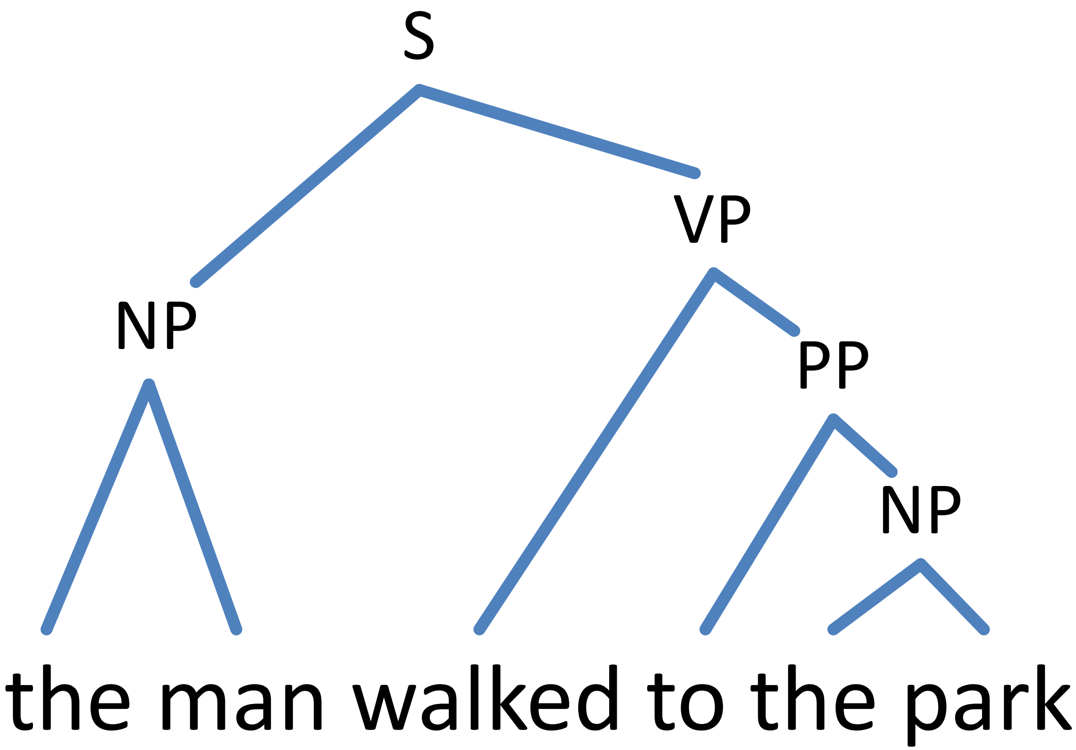
we often draw the bracketing as a tree:



the man walked to the park

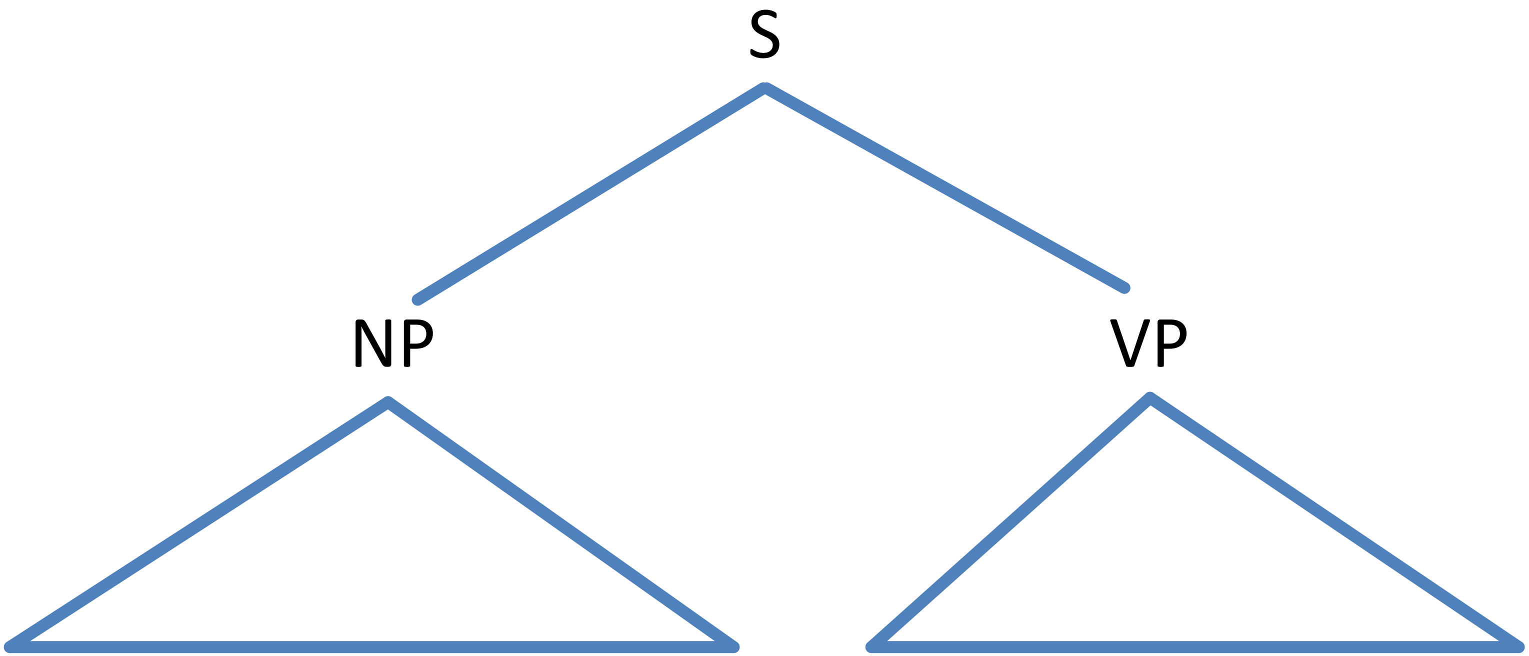
Labeled Bracketings/Trees

(S (NP the man) (VP walked (PP to (NP the park))))



Key:
S = sentence
NP = noun phrase
VP = verb phrase
PP = prepositional phrase

Labels → Substitutability

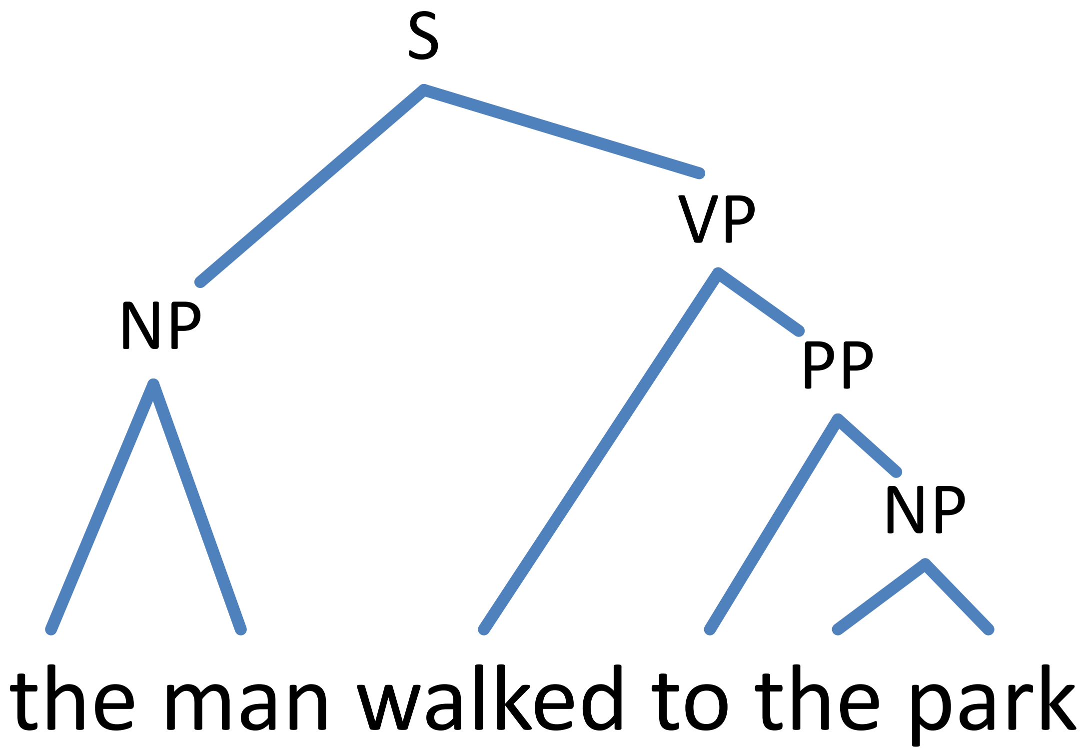


the man
John
the first boss I ever had

walked to the park
fell asleep
is here

Labeled Bracketings/Trees

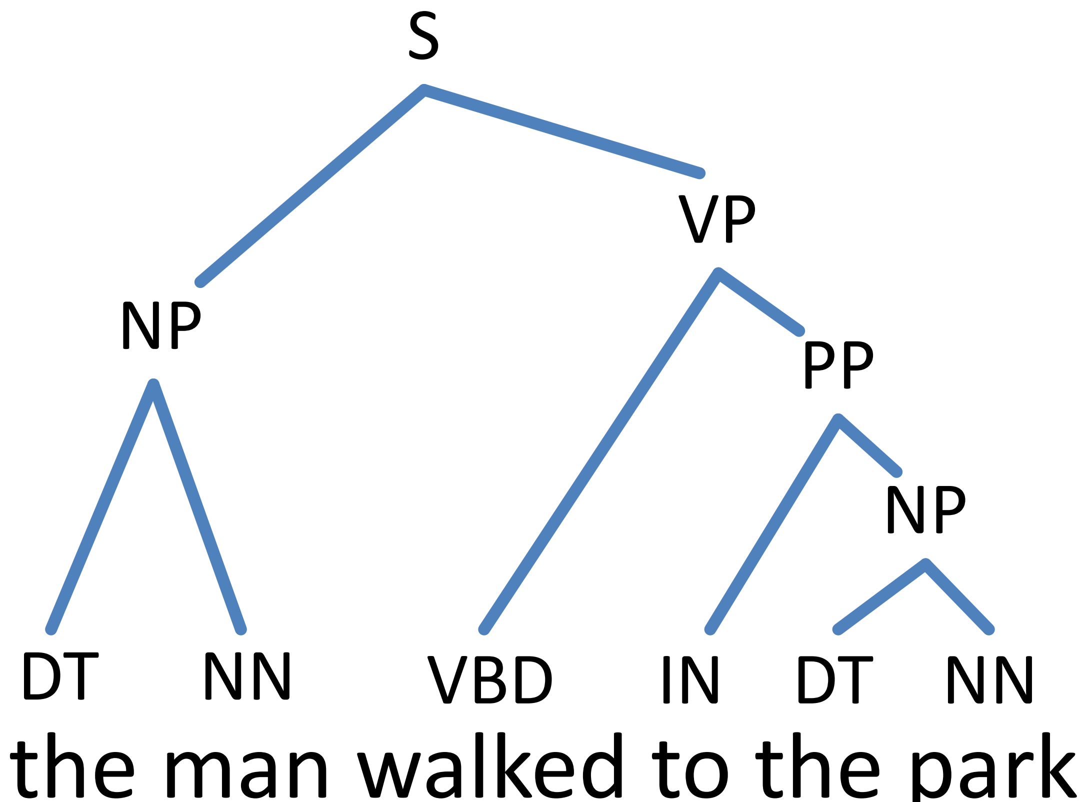
(S (NP the man) (VP walked (PP to (NP the park))))



Key:
S = sentence
NP = noun phrase
VP = verb phrase
PP = prepositional phrase

Labeled Bracketings/Trees

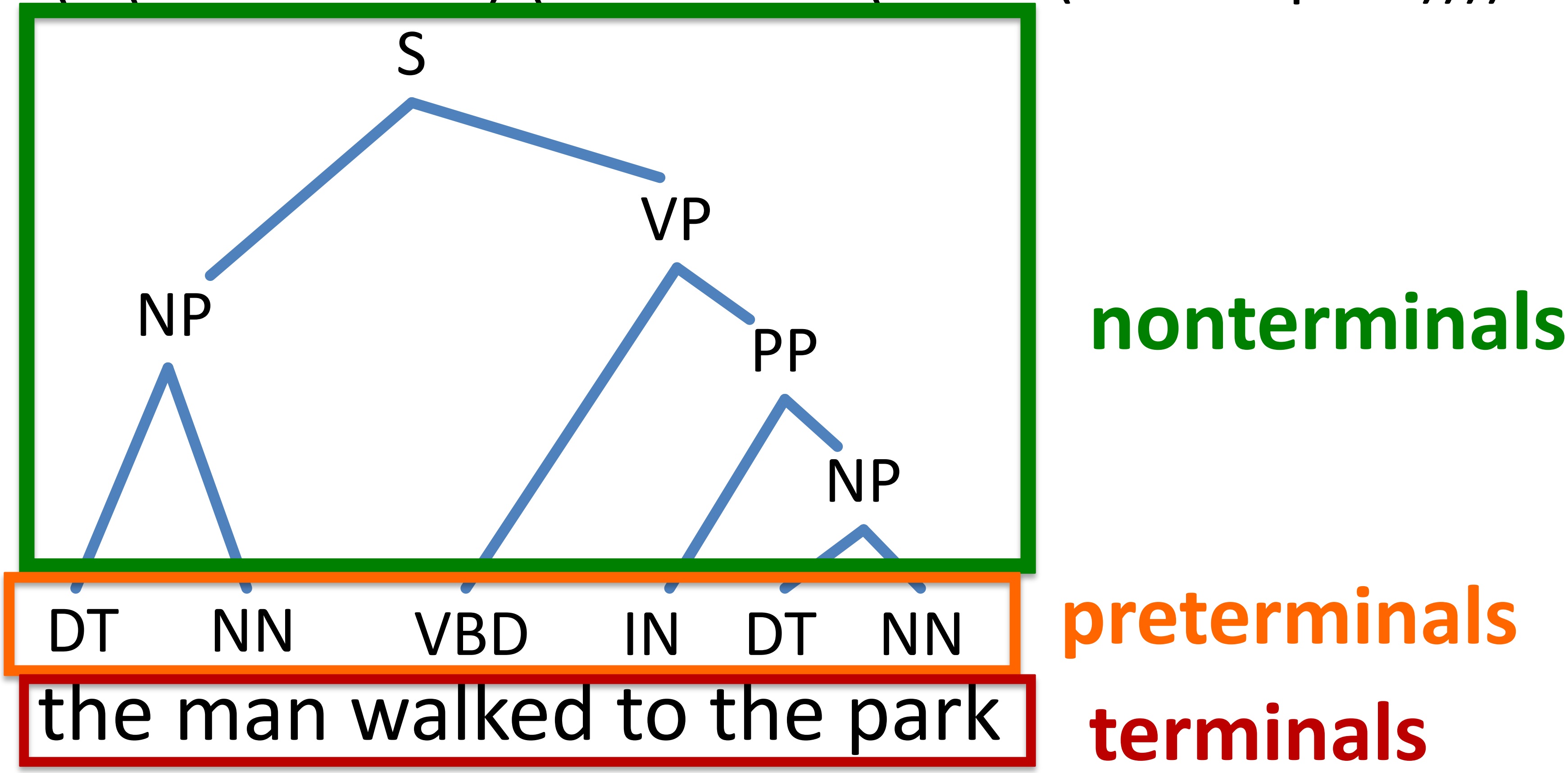
(S (NP the man) (VP walked (PP to (NP the park))))



Key:
S = sentence
NP = noun phrase
VP = verb phrase
PP = prepositional phrase
DT = determiner
NN = noun
VBD = verb (past tense)
IN = preposition

Labeled Bracketings/Trees

(S (NP the man) (VP walked (PP to (NP the park))))



Penn Treebank tag set

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Penn Treebank Nonterminals

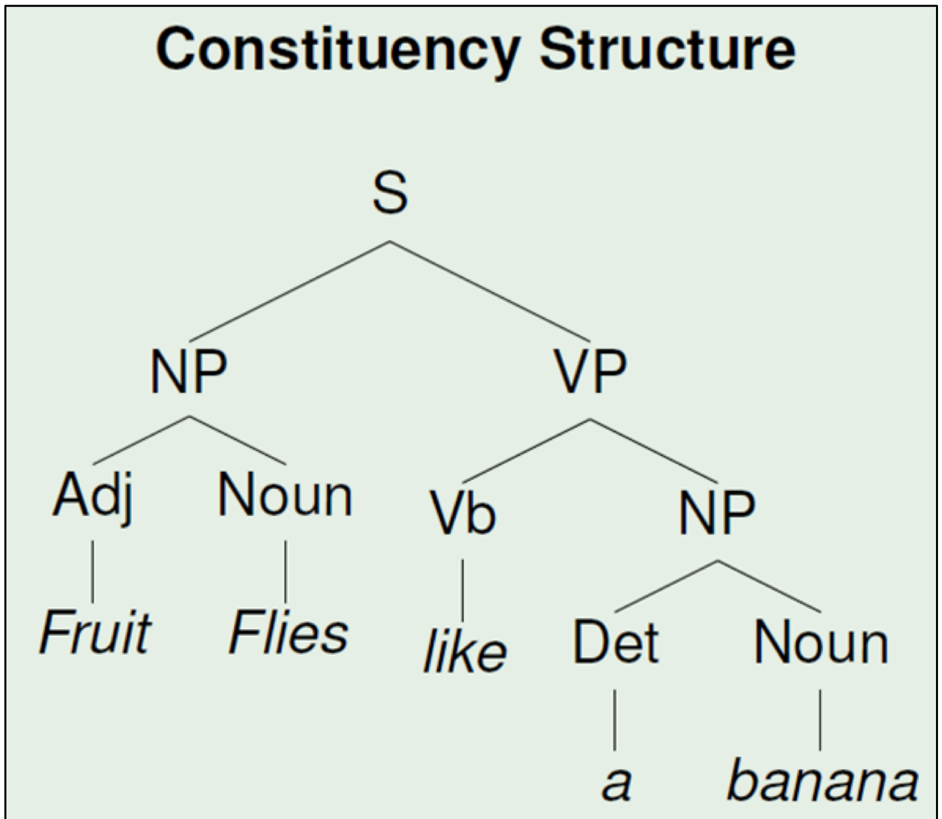
S	Sentence or clause.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a *wh*-word or *wh*-phrase.
SINV	Inverted declarative sentence.
SQ	Inverted yes/no question, or main clause of a *wh*-question.
ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection.
LST	List marker. Includes surrounding punctuation.
NAC	Not A Constituent; used within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head.

PP	Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle.
QP	Quantity Phrase (i.e., complex measure/amount) within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Verb Phrase.
WHADJP	*Wh*-adjective Phrase, as in *how hot*.
WHADVP	*Wh*-adverb Phrase.
WHNP	*Wh*-noun Phrase, e.g. *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.
WHPP	*Wh*-prepositional Phrase, e.g., *of which* or *by whose authority*.
X	Unknown, uncertain, or unbracketable.

Syntactic Ambiguities

Time flies like an arrow.

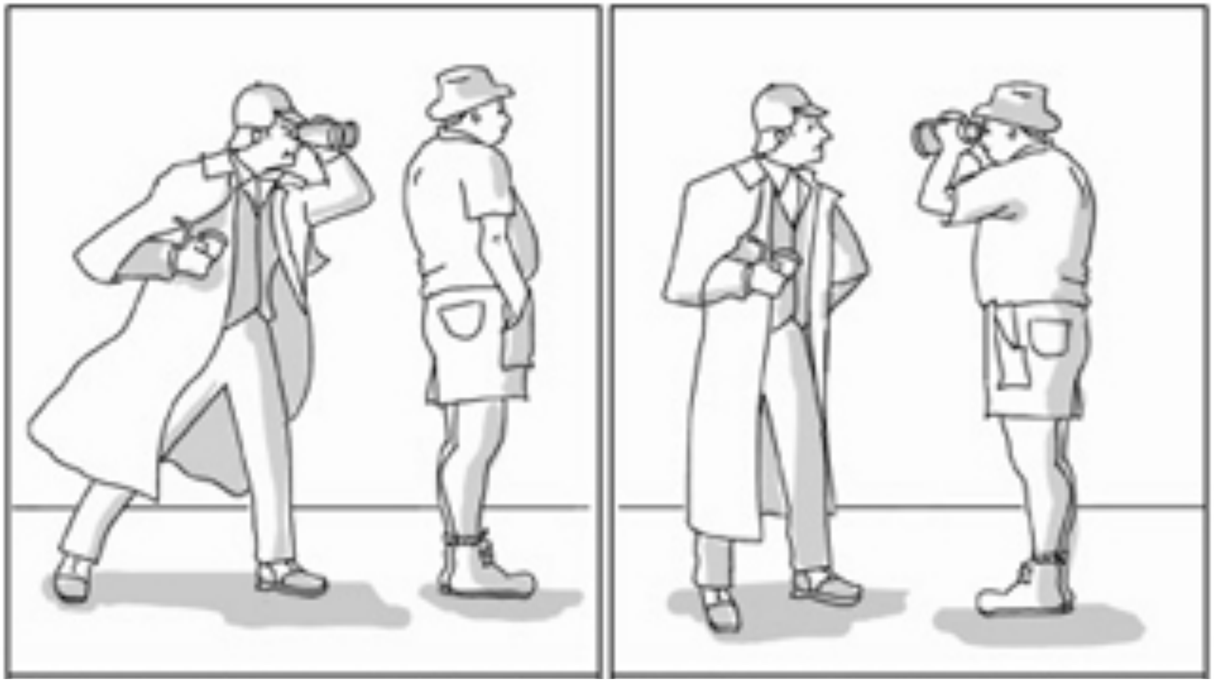
Fruit flies like a banana.



Syntactic Ambiguities

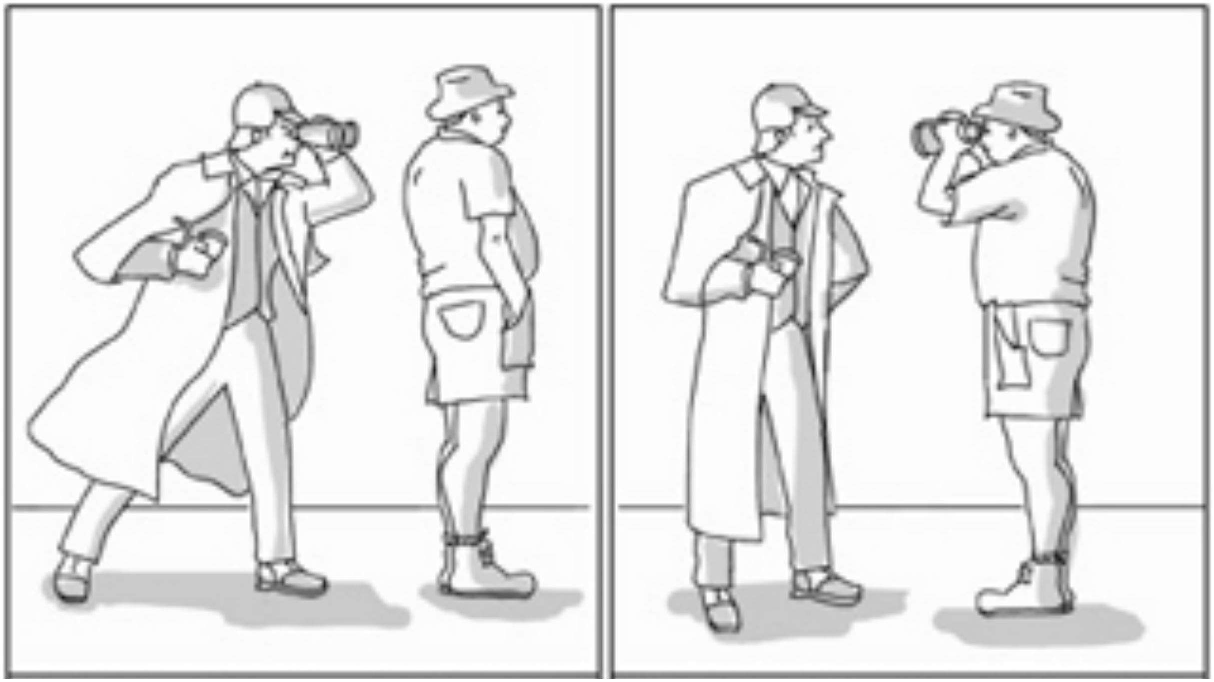
- attachment ambiguity
- coordination ambiguity
- noun compound ambiguity

Attachment Ambiguity



Sherlock saw the man using binoculars

Attachment Ambiguity



Sherlock (saw (the man) (using binoculars))

Sherlock (saw (the man (using binoculars)))

other attachment ambiguities

Infant pulled from car involved in short police pursuit

Somali tied to militants held on U.S. ship for months

other attachment ambiguities

(Infant pulled from car) involved in short police pursuit

Infant pulled from (car involved in short police pursuit)

(Somali tied to militants) held on U.S. ship for months

Somali tied to (militants held on U.S. ship for months)

coordination ambiguities

- often found when modifiers are used with conjunctions:

  keyboard and monitor with the Apple logo

  old men and women

coordination ambiguities

- often found when modifiers are used with conjunctions:

  (keyboard and monitor) with the Apple logo

  keyboard and (monitor with the Apple logo)

  old (men and women)

  (old men) and women

noun compound ambiguity

- California history teacher
- World Trade Center
- student film society

Szymanek;
Fromkin (2000);
Spencer (1991)

adjective/noun modifier ambiguity

- ancient history teacher

- moral philosophy professor

- indoor garden party

NLP Task: Constituent Parsing

- given a sentence, output its constituent parse
- widely-studied task with a rich history
- most based on the Penn Treebank (Marcus et al.), developed at Penn in early 1990s

- Treebank = “corpus of annotated parse trees”

How are constituent parses used?

- language modeling
  - predict the next word better by using syntactic structure
- machine translation
  - there are many syntactic translation models that require parsers for one or both languages
- text classification
  - for certain kinds of classification, features on syntactic fragments can help
- question answering, coreference resolution, etc.