TTIC 31190: Natural Language Processing

Kevin Gimpel

Winter 2016

Lecture 12: Syntax and Parsing

Announcement

- project proposal due Tuesday

Announcement

- midterm is one week from today, room #530
- it’ll be closed-book
- if you want, you can bring an 8.5x11 sheet, but I don’t think you’ll need to
- on Tuesday we will review all the course material and go through some example questions

Office Hours Next Week

- unfortunately, my office hour on Monday must be canceled (EAC visit)
- I will instead have it on Tuesday 9:30-10:30 am (right before class)
- feel free to email me and make an appointment if that time does not work for you

Roadmap

- classification
- words
- lexical semantics
- language modeling
- sequence labeling
- neural network methods in NLP
- syntax and syntactic parsing
- semantic compositionality
- semantic parsing
- unsupervised learning
- machine translation and other applications

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

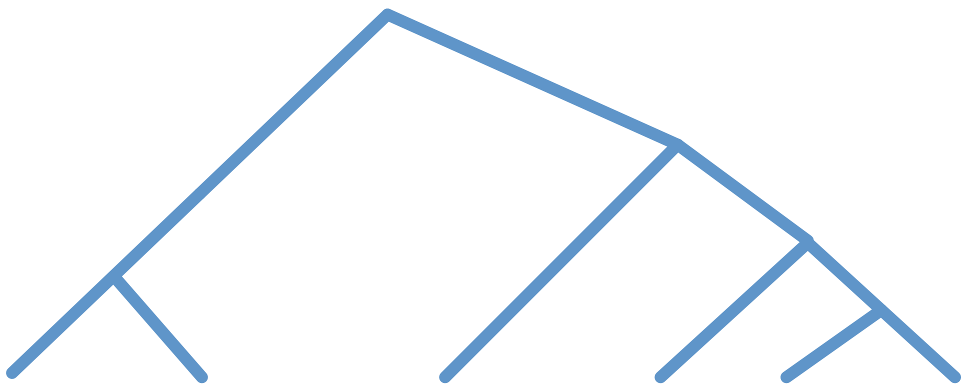
- bracketing:

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

Bracketing → Tree

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

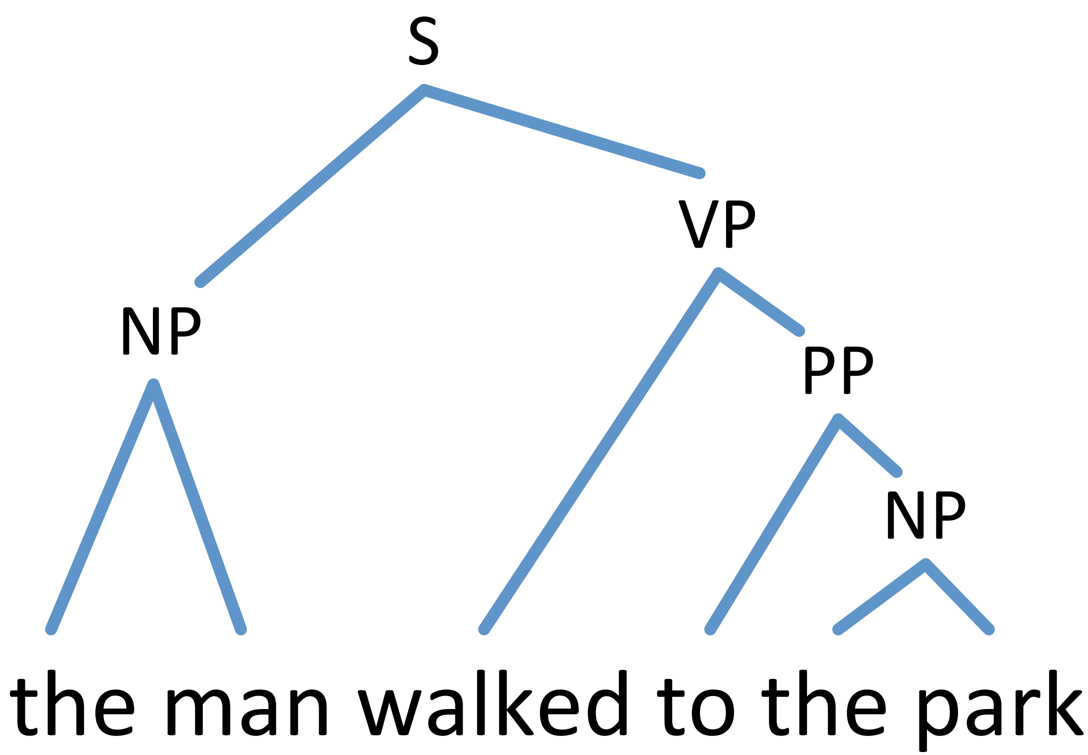
we often draw the bracketing as a tree:



the man walked to the park

Labeled Bracketings/Trees

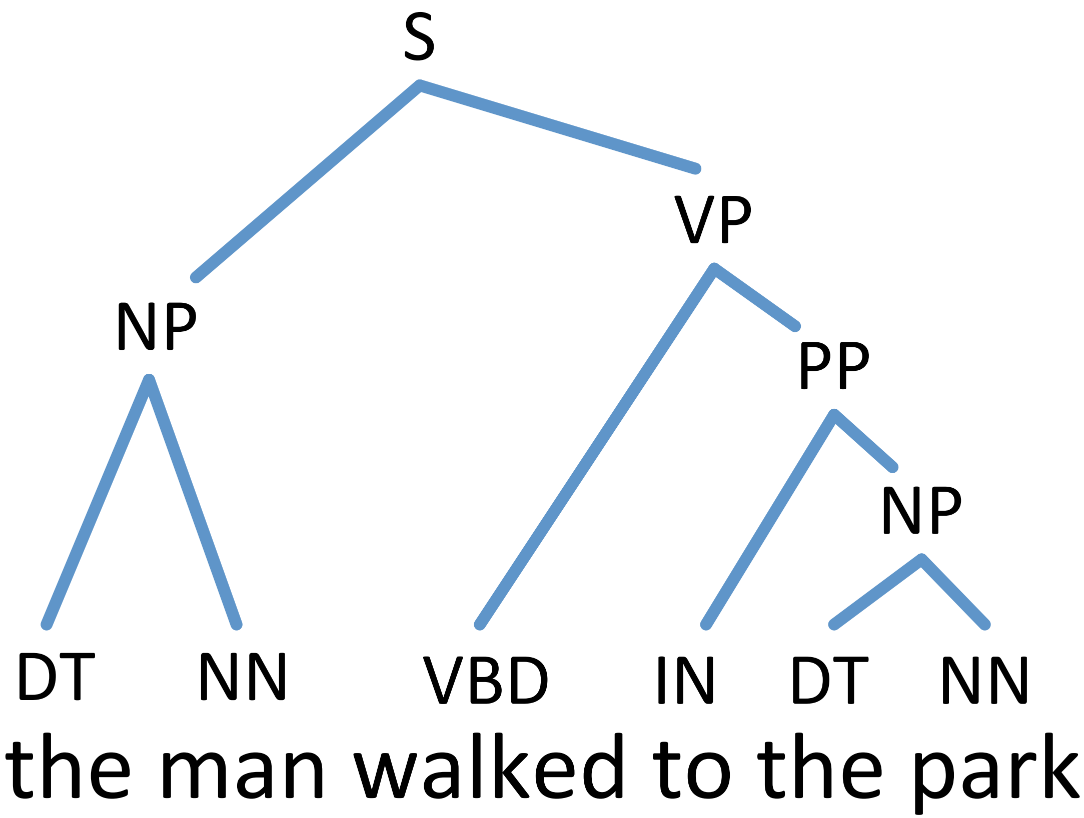
(S (NP the man) (VP walked (PP to (NP the park))))



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

Labeled Bracketings/Trees

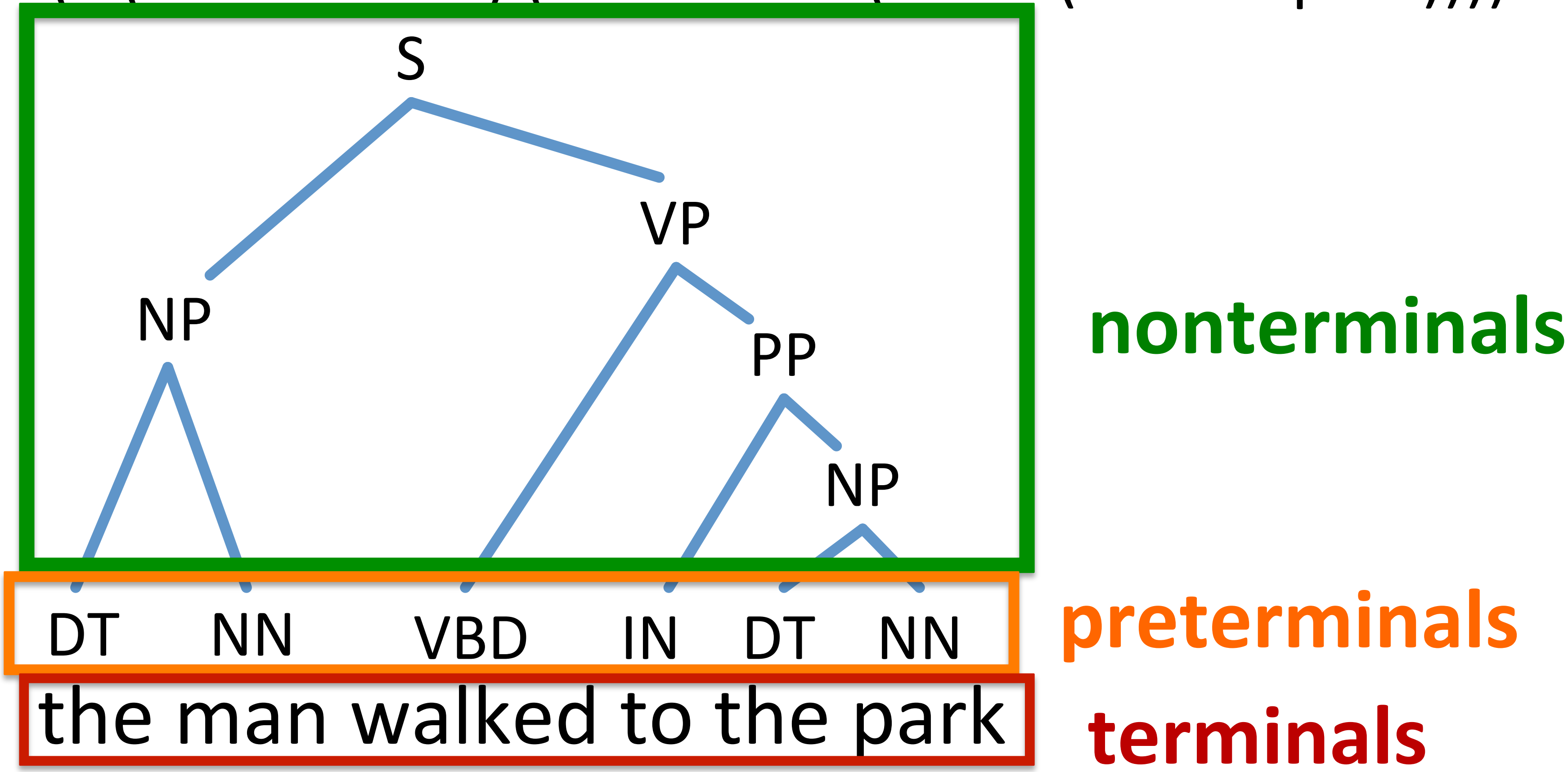
(S (NP the man) (VP walked (PP to (NP the park))))



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

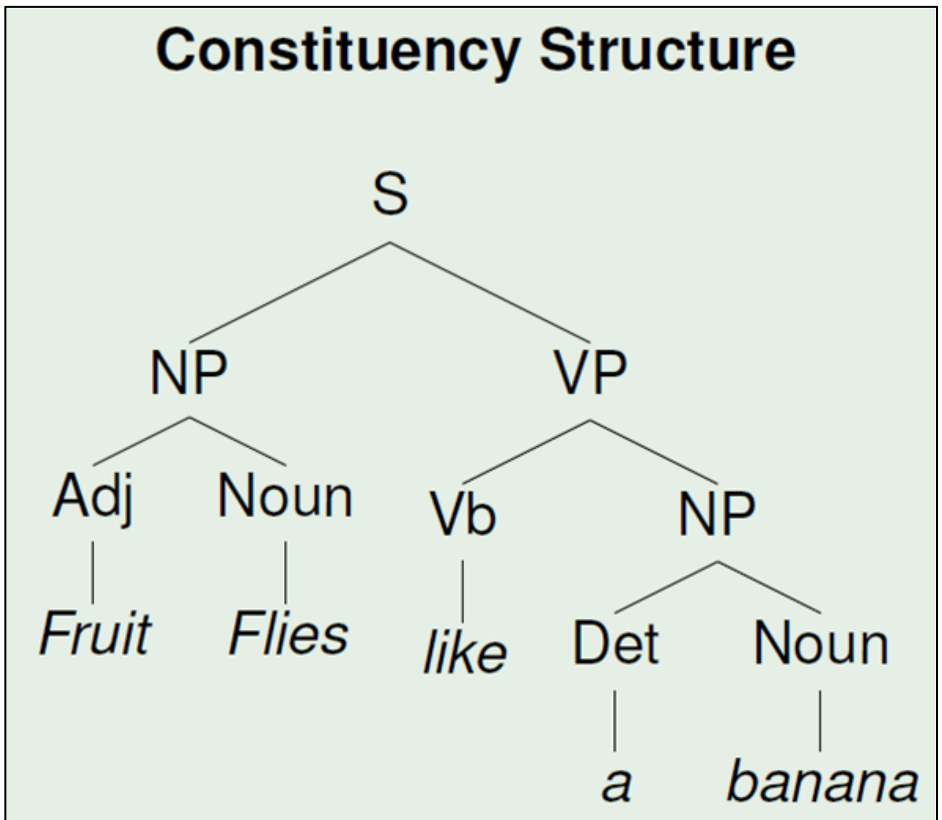
S	Sentence or clause.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a *wh*-word or *wh*-phrase.
SINV	Inverted declarative sentence.
SQ	Inverted yes/no question, or main clause of a *wh*-question.
ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
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.

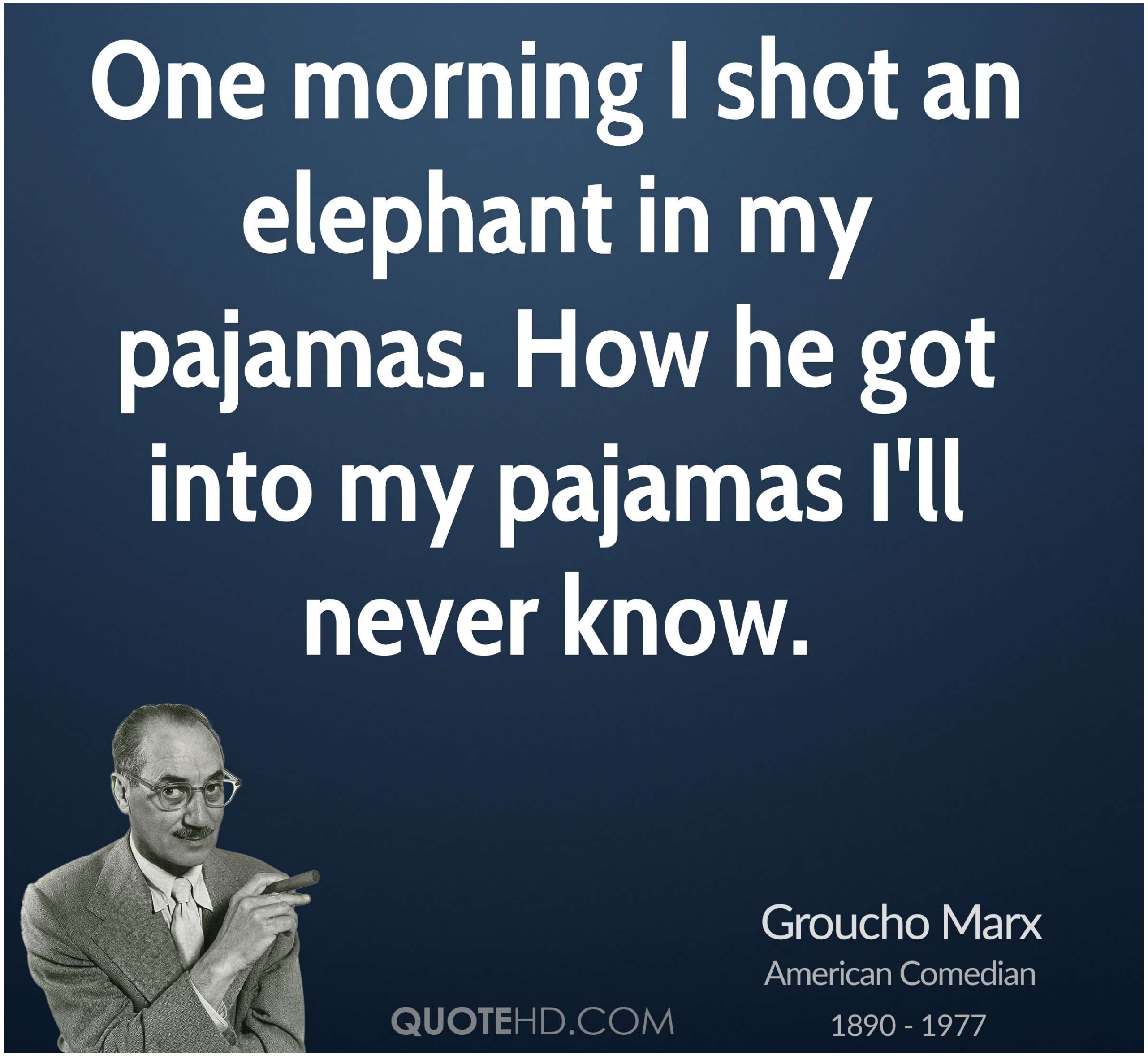
Syntactic Ambiguities

Time flies like an arrow.

Fruit flies like a banana.



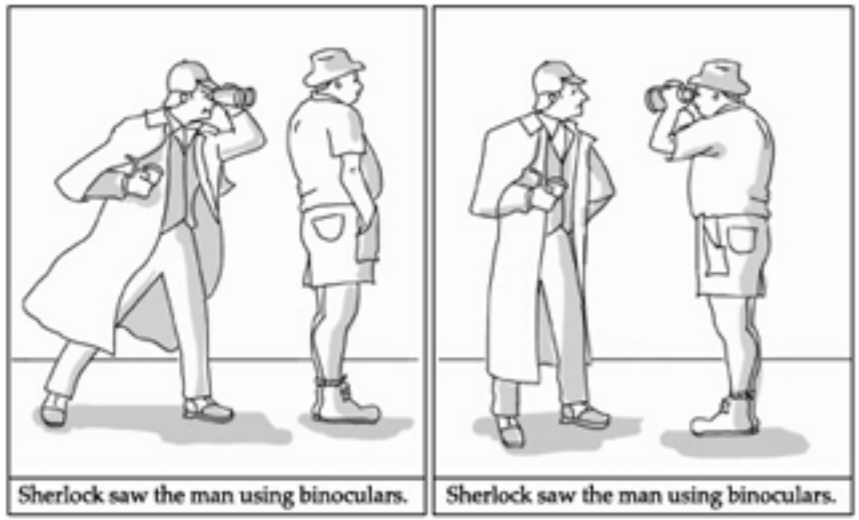
Attachment Ambiguity



Syntactic Ambiguities

- PP attachment ambiguity
- coordination ambiguity
- noun compound ambiguity

Attachment Ambiguity



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

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)

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"

Context-Free Grammar (CFG)

- has "rewrite rules" to rewrite nonterminals as terminals or other nonterminals

  S → NP VP

  "S goes to NP VP"

  NP → DT NN

  VP → VBD PP

  PP → IN NP

  NN → man

  DT → the

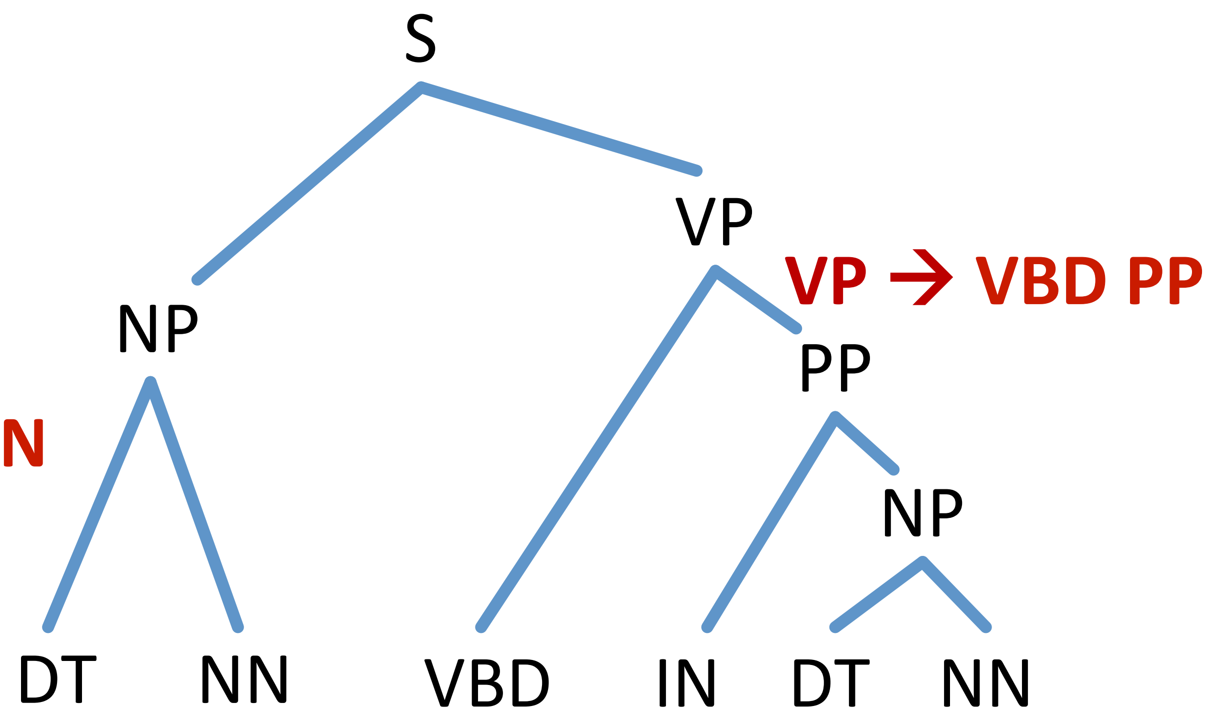
Context-Free Grammar (CFG)

- sequence of rewrites corresponds to a bracketing (induces a hierarchical tree structure)

S → NP VP

NP → DT NN

DT → the



the man walked to the park

Why “context-free”?

- a rule to rewrite NP does not depend on the context of NP
- that is, the left-hand side of a rule is only a single non-terminal (without any other context)

Probabilistic Context-Free Grammar (PCFG)

- assign probabilities to rewrite rules:

NP →	DT NN	0.5
NP →	NNS	0.3
NP →	NP PP	0.2

same nonterminal can be on both left and right sides

Probabilistic Context-Free Grammar (PCFG)

- assign probabilities to rewrite rules:

NP → DT NN	0.5
NP → NNS	0.3
NP → NP PP	0.2

probabilities must sum to one for each left-hand side nonterminal

Probabilistic Context-Free Grammar (PCFG)

- assign probabilities to rewrite rules:

NP → DT NN	0.5
NP → NNS	0.3
NP → NP PP	0.2
NN → man	0.01
NN → park	0.0004
NN → walk	0.002
NN → ….	

given a treebank, we can estimate these probabilities using maximum likelihood estimation ("relative frequency estimates"; "count and normalize"),
just like we did with n-gram language models and HMMs for POS tagging

Probabilistic Context-Free Grammar (PCFG)

- for each nonterminal, a PCFG has a probability distribution over possible right-hand side sequences
- so, a PCFG assigns probabilities to:
  - bracketings of sentences
  - sequences of rewrite operations (**derivations**) that eventually terminate in terminals
  - hierarchical tree structures that ground out in sequences of terminals
- these are different ways of saying the same thing

Constituent Parsing

- evaluation: `evalb` score
  - first compute precision and recall (at the level of constituents)
  - then compute F1 (harmonic mean of precision and recall)

How well does a PCFG work?

- a PCFG learned from the Penn Treebank with maximum likelihood estimation (count & normalize) gets about 73% F1 score
- state-of-the-art parsers are around 92%

How well does a PCFG work?

- a PCFG learned from the Penn Treebank with maximum likelihood estimation (count & normalize) gets about 73% F1 score
- state-of-the-art parsers are around 92%
- but, simple modifications can improve the PCFG a lot!
  – smoothing
  – tree transformations (selective flattening)
  – "parent annotation"

Parent Annotation

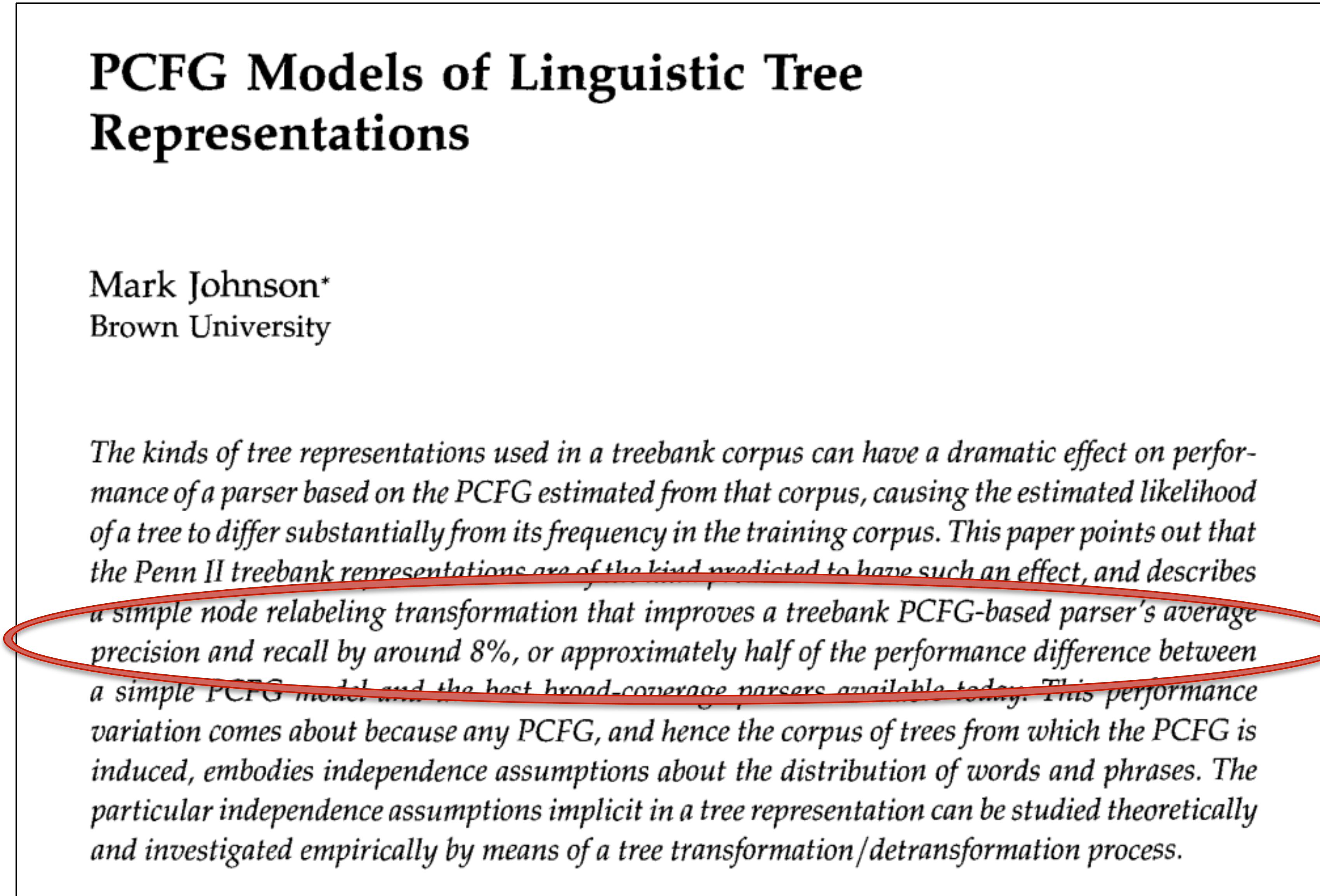
$VP \rightarrow V \;\; NP \;\; PP$

$\Downarrow$

$VP^{S} \rightarrow V \;\; NP^{VP} \;\; PP^{VP}$

adds more information, but also fragments counts, making parameter estimates noisier (since we're just using MLE)

Johnson (1998)



PCFG Models of Linguistic Tree Representations

Mark Johnson*
Brown University

The kinds of tree representations used in a treebank corpus can have a dramatic effect on performance of a parser based on the PCFG estimated from that corpus, causing the estimated likelihood of a tree to differ substantially from its frequency in the training corpus. This paper points out that the Penn II treebank representations are of the kind predicted to have such an effect, and describes a simple node relabeling transformation that improves a treebank PCFG-based parser's average precision and recall by around 8%, or approximately half of the performance difference between a simple PCFG model and the best broad-coverage parsers available today. This performance variation comes about because any PCFG, and hence the corpus of trees from which the PCFG is induced, embodies independence assumptions about the distribution of words and phrases. The particular independence assumptions implicit in a tree representation can be studied theoretically and investigated empirically by means of a tree transformation/detransformation process.

Johnson (1998)

	22	22 Id	Id	NP-VP	N′-V′	Flatten	Parent
Number of rules		2,269	14,962	14,297	14,697	22,652	22,773
Precision	1	0.772	0.735	0.730	0.735	0.745	0.800
Recall	1	0.728	0.697	0.705	0.701	0.723	0.792
NP attachments	279	0	67	330	69	154	217
VP attachments	299	424	384	0	503	392	351
NP* attachments	339	3	67	399	69	161	223
VP* attachments	412	668	662	150	643	509	462

Classification Framework for Constituent Parsing

inference: solve argmax

modeling: define score function

$$\mathrm{classify}(x, \boldsymbol{\theta}) = \underset{y}{\mathrm{argmax}} \;\; \mathrm{score}(x, y, \boldsymbol{\theta})$$

learning: choose $\boldsymbol{\theta}$

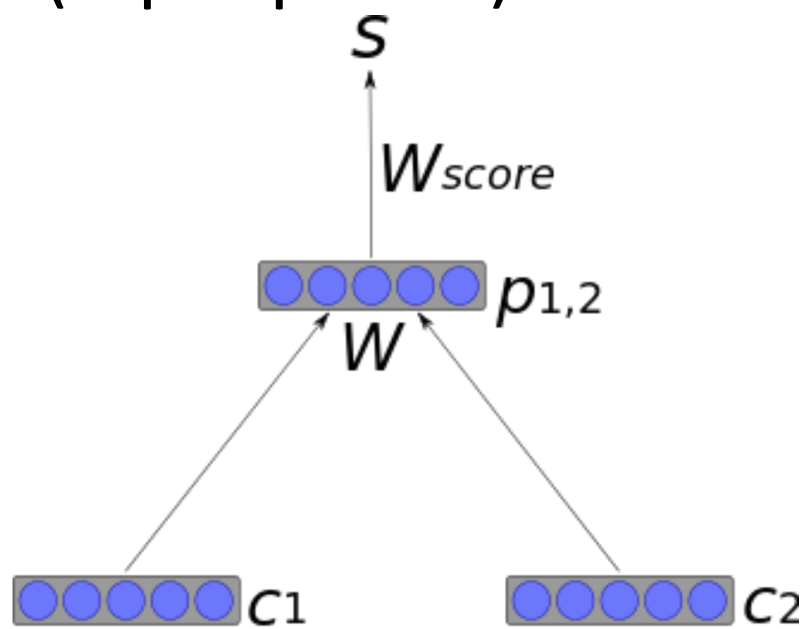
- *x* = a sentence
- *y* = a constituent parse
- inference requires searching all possible constituent parses!
- this is very expensive due to large training sets

How are constituent parses used?

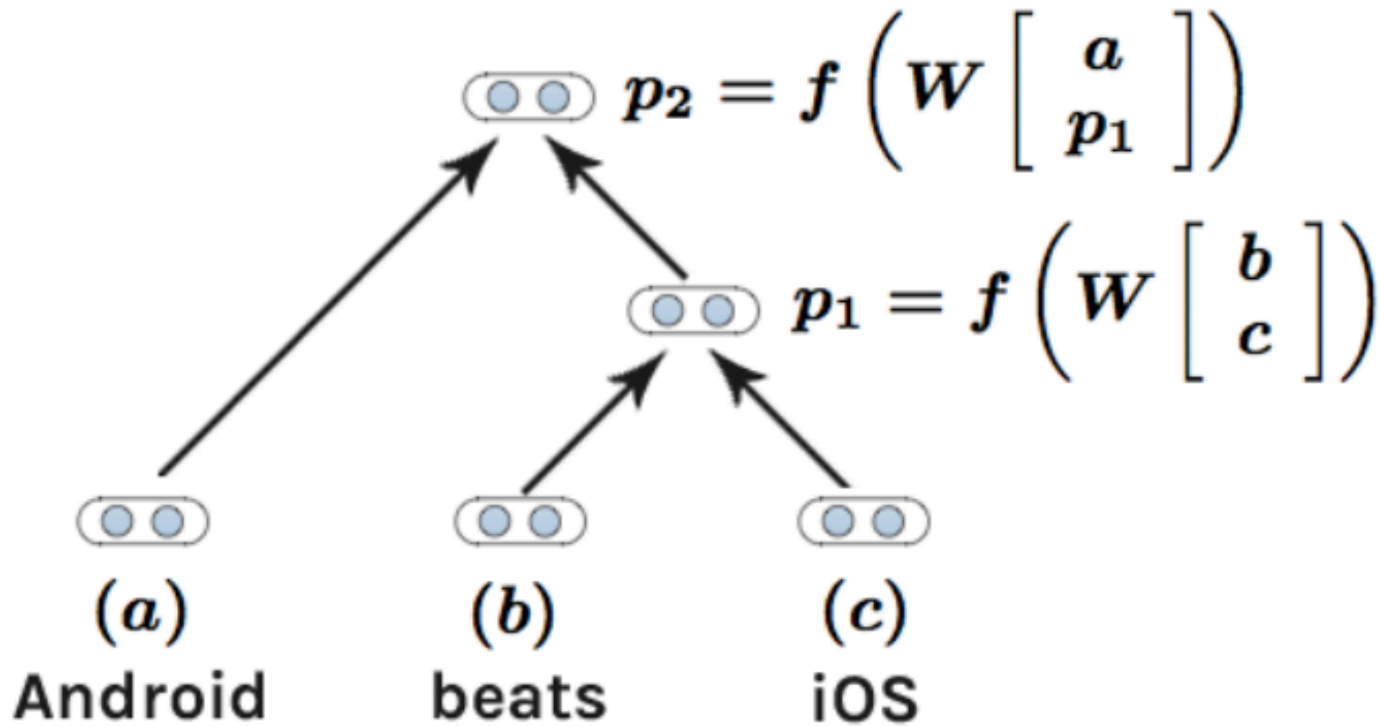
- language modeling
  - predict the next word better by using syntactic structure
- machine translation
  - there are many syntactic translation models that require parsers for one or both languages
- text classification
  - for certain kinds of classification, features on syntactic fragments can help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

Recursive Neural Networks for NLP

- first, run a constituent parser on the sentence
- convert the constituent tree to a binary tree (each rewrite has exactly two children)
- construct vector for sentence recursively at each rewrite ("split point"):



Recursive Neural Networks for NLP



Recursive Neural Networks for NLP