Part-of-Speech Tagging, Syntactic Parsing

Human Language from a Computational Perspective
April 5, 2016

Part-of-Speech Tagging

Language Models Reminder

Use an *n*-gram model to predict the next token:

* ~~My only **wish I**~~

My only **wish is**

Bigram counts (starting with wish):

wish I	8
wish is	6
wish they	4
wish was	4
wish that	2
wish you	1

Lexical Ambiguity

The word WISH is ambiguous

WISH (verb): **לבקש, לאחל**

WISH (noun): **משאלה**

Some Context Helps

Verb:

- How I **wish** you were here
- Careful what you **wish** for
- **Wish** you a happy birthday

Noun:

- Your **wish** is my command
- If you could have one **wish**
- Make a **wish**

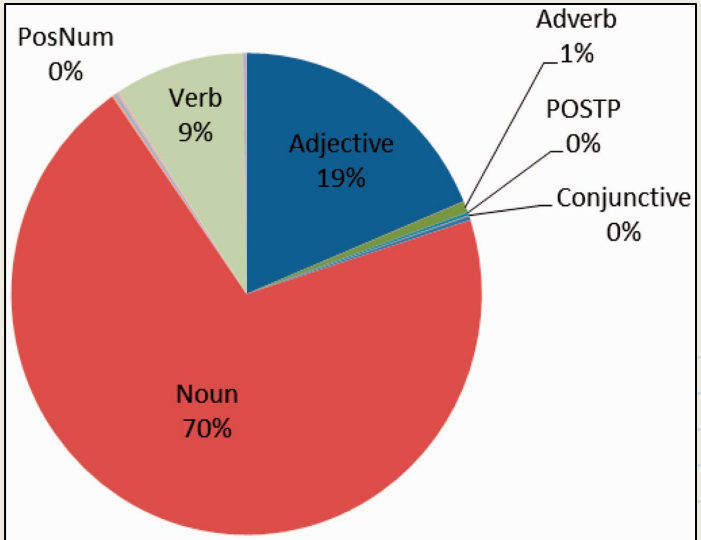
But Sometimes It Doesn’t

Squad helps dog bite victim

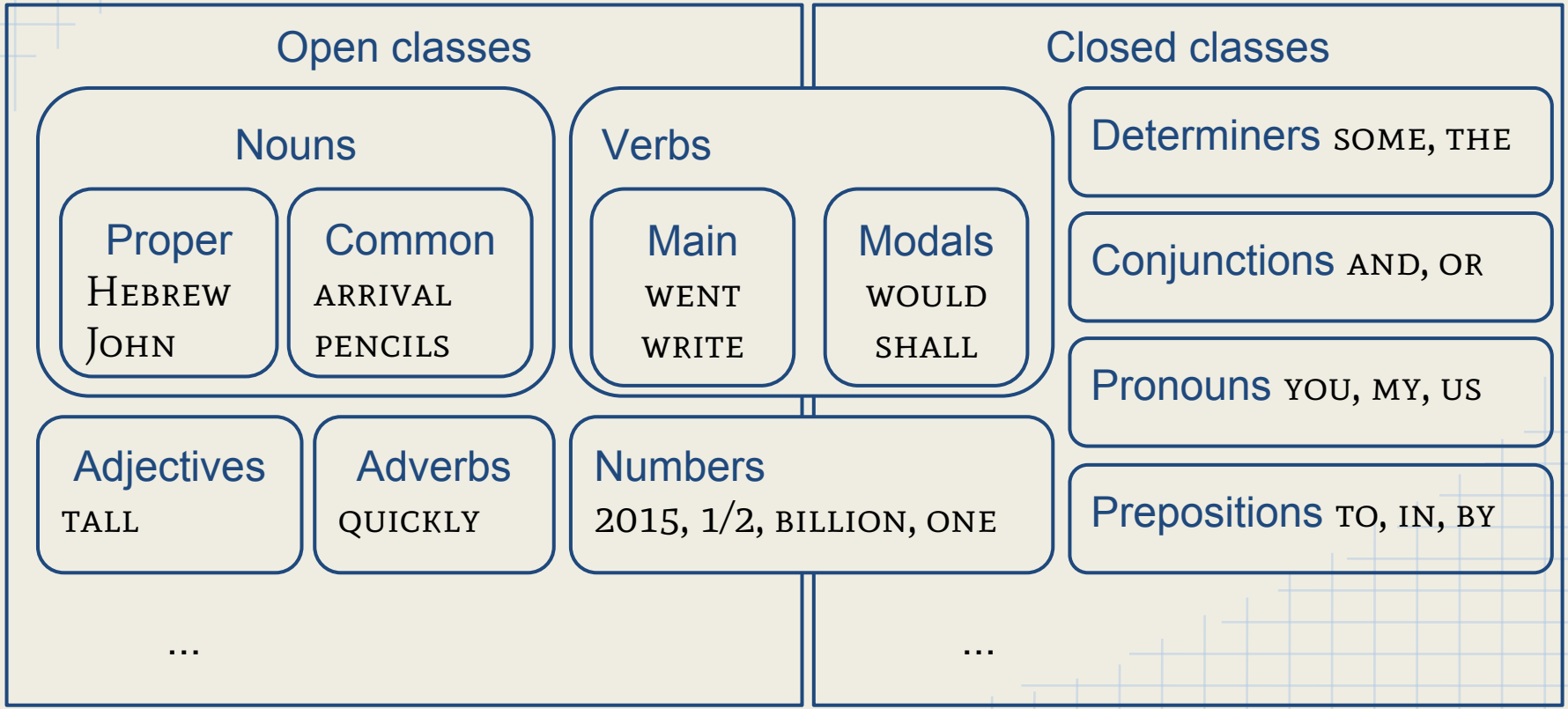
בסוף המרוץ הוא נפל מהסוס. הוא לא פרש.

Parts of Speech

Words can roughly be divided into **distributional categories** based on their **syntactic roles**.



Part-of-Speech Hierarchy



Part-of-Speech Tags

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	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, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	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*			

Tag guide:
https://catalog.ldc.upenn.edu/docs/LDC99T42/tagguid1.pdf

Penn Treebank Part-of-Speech Tags for English, Jurafsky & Martin 2009

Language Variations

AD	adverb	还
AS	aspect marker	着
BA	把 in ba-construction	把, 将
CC	coordinating conjunction	和
CD	cardinal number	一百
CS	subordinating conjunction	虽然
DEC	的 in a relative-clause	的
DEG	associative 的	的
DER	得 in V-de const. and V-de-R	得
DEV	地 before VP	地
DT	determiner	这
ETC	for words 等, 等等	等, 等等
FW	foreign words	I S O
IJ	interjection	啊
JJ	other noun-modifier	男, 共同
LB	被 in long bei-const	被, 给
LC	localizer	里
M	measure word	个
MSP	other particle	所

NN	common noun	书
NR	proper noun	美国
NT	temporal noun	今天
OD	ordinal number	第一
ON	onomatopoeia	哈哈, 哗哗
P	preposition excl. 被 and 把	从
PN	pronoun	他
PU	punctuation	、？。
SB	被 in short bei-const	被, 给
SP	sentence-final particle	吗
VA	predicative adjective	红
VC	是	是
VE	有 as the main verb	有
VV	other verb	走

Penn Treebank Part-of-Speech Tags for Mandarin Chinese, Xia 2000

Part-of-Speech Tagging

Tag the following text for POS:

Alice	was	beginning	to	get	very	tired
NNP	VBD	VBG	TO	VB	RB	JJ

Statistical POS Tagging

We can use counts from the corpus to tag text for POS,

but it requires **annotation**:

just the text is not enough.

Annotated Corpus Example

The/AT grand/JJ jury/NN commented/VBD on/IN a/AT number/NN of/IN other/AP topics/NNS ,/, AMONG/IN them/PPO the/AT Atlanta/NP and/CC Fulton/NP-tl County/NN-tl purchasing/VBG departments/NNS which/WDT it/PPS said/VBD ``/`` ARE/BER well/QL operated/VBN and/CC follow/VB generally/RB accepted/VBN practices/NNS which/WDT inure/VB to/IN the/AT best/JJT interest/NN of/IN both/ABX governments/NNS ''/'' ./.

Lexical POS Counts

Simple method: count the times each word occurred with each POS in the corpus

THE	DT	→	1527
WELL	RB	→	37
WELL	NN	→	3
SLEEP	NN	→	4
SLEEP	VBP	→	2
THAT	IN	→	197
THAT	DT	→	50

POS Tagging Algorithm

Find POS sequence of token sequence

Given: [“WHAT”, ”IS”, ”THE”, ”ANSWER”, “?”]

Return: [“WP”, “VBZ”, “DT”, “NN”, “.”]

POS Tagging Algorithm 1

```
Tag_POS_Simple(Tokens, CountsL):
for Index ← 1 to length(Tokens):
    Tags_Index ← Max2(Tokens_Index, CountsL)
return Tags
```

Tokens is a sequence of strings
CountsL is a table of [string, string]→number
Returns a sequence of strings

Auxiliary Algorithm

Max2(Token, CountsL):

Max ← 0

for each [T_1, T_2] in CountsL:

    if (T_1 = Token) and (CountsL[T_1, T_2] > Max):

        Max ← CountsL[T_1, T_2]

        Best ← T_2

return Best

Token is a string
CountsL is a table of [string, string]→number
Returns a string

Auxiliary Algorithm

Max2 is very similar to the **Bigram** algorithm for LM text generation.

It just returns the T_2 with the highest count among entries with T_1 = Token.

Surprising Accuracy

This simple approach actually gets about 90% of the POS tags correctly!

Most words almost always appear with the same POS.

Problem: Variability

Use the most common POS for each word

THE	FISH	SLEEP	IN	THAT	WELL
DT	~~NN~~	~~NN~~	IN	~~IN~~	~~RB~~

But the correct tags are:

DT	NNS	VBP	IN	DT	NN

State of the Art

The best methods today get slightly more than 97% accuracy, so 90% is not so bad.

Problem: Unknown Words

'T WAS BRILLIG , AND THE SLITHY TOVES
? VBD ? , CC DT ? ?
DID GYRE AND GIMBLE IN THE WABE ;
VBD ? CC ? IN DT ? :
ALL MIMSY WERE THE BOROGOVES ,
DT ? VBD DT ? ,
AND THE MOME RATHS OUTGRABE .
CC DT ? ? ? .

First stanza of *Jabberwocky* from *Through the Looking-Glass, and What Alice Found There* (1871) by Lewis Carroll

Solutions

- Context (above the word level)
- Morphology (below the word level)

Transition Counts

Count the times each tag follows another tag.

These are **tag bigram** counts (transition counts).

NN	NN	→	312
NN	IN	→	690
NN	DT	→	113
IN	NN	→	262
DT	NN	→	1256
PRP	VBD	→	847
VBD	DT	→	464

POS Tagging Algorithm 2

```
Tag_POS(Tokens, CountsL, CountsT):
Tags_1 ← Max2(Tokens_1, CountsL)
for Index ← 2 to length(Tokens):
    Tags_Index ← Max3(Tokens_Index, Tags_Index - 1, CountsL, CountsT)
return Tags
```

Tokens is a sequence of strings
CountsL and CountsT are tables of [string, string]→number
Returns a sequence of strings

Combining the Counts

How to implement **Max3**?

Multiply lexical count (from CountsL)

with transition count (from CountsT)

Auxiliary Algorithm

```
Max3(Token, Tag, CountsL, CountsT):
Max ← 0
for each [T1, T2] in CountsL:
    if (T1 = Token):
        Score ← CountsL[T1, T2] × CountsT[Tag, T2]
        if (Score > Max):
            Max ← Score
            Best ← T2
return Best
```

Token and Tag are strings
CountsL and CountsT are tables of [string, string]→number
Returns a string

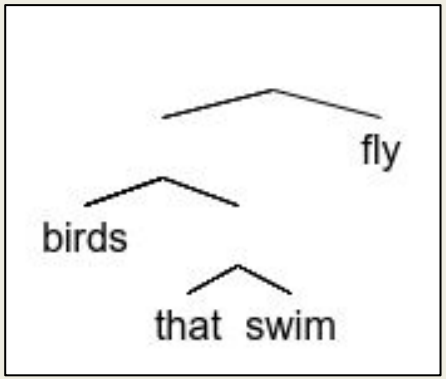
Syntactic Parsing

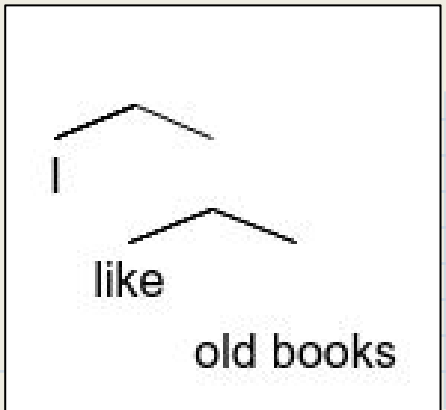
Unlabeled Bracketing

A hierarchy of **constituents**

[[BIRDS [THAT SWIM]] FLY]

[I [LIKE [OLD BOOKS]]]





Labeled Bracketing

Annotates each constituent with a **label**,

and each token with a **part-of-speech tag**

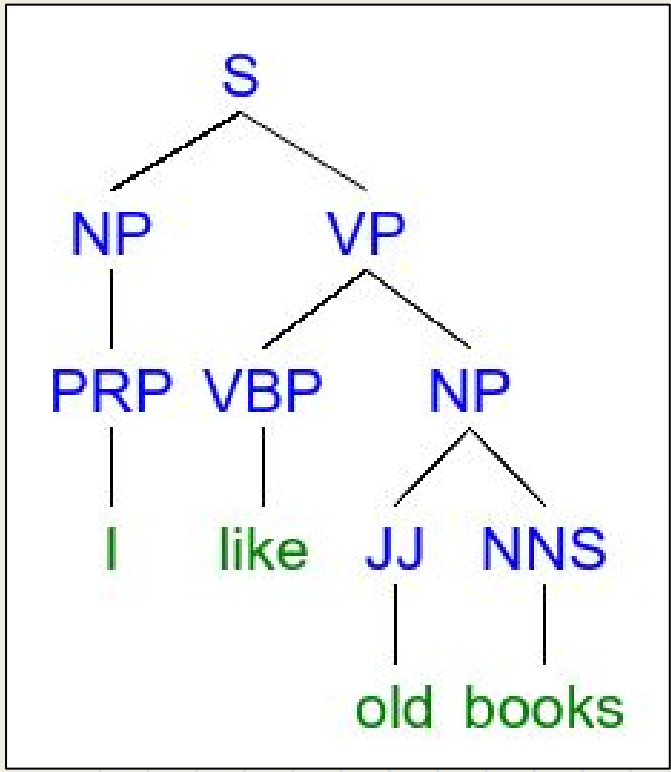
[**S** [**NP** [**PRP** I]] [**VP** [**VBP** LIKE] [**NP** [**JJ** OLD] [**NNS** BOOKS]]]]

Phrase Structure (Constituency Parsing)

Represents text structure as a **tree**: tokens are leaves

(Equivalent to labeled bracketing)

[S [NP [PRP I]] [VP [VBP LIKE] [NP [JJ OLD] [NNS BOOKS]]]]



mshang.ca/syntree

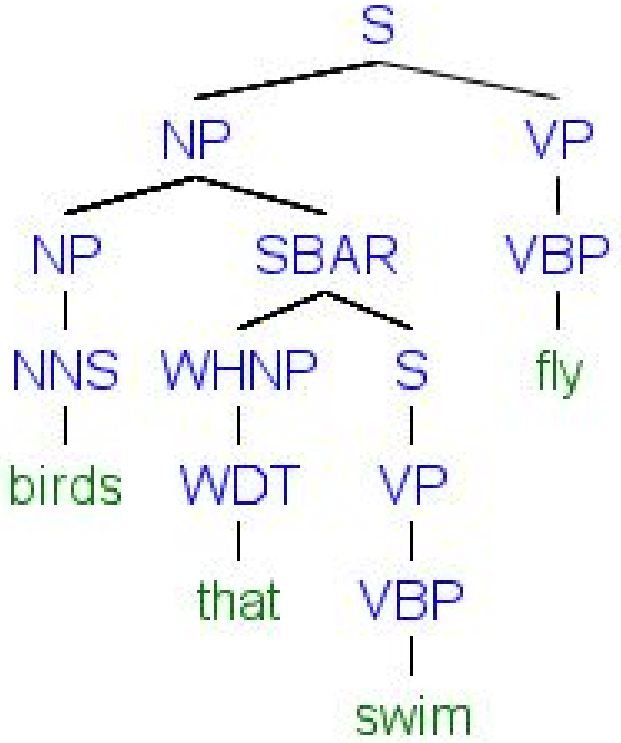
More Complicated Example

Relative clauses introduce

SBAR nodes and are parsed too

http://cs.jhu.edu/~jason/465/hw-parse/treebank-manual.pdf

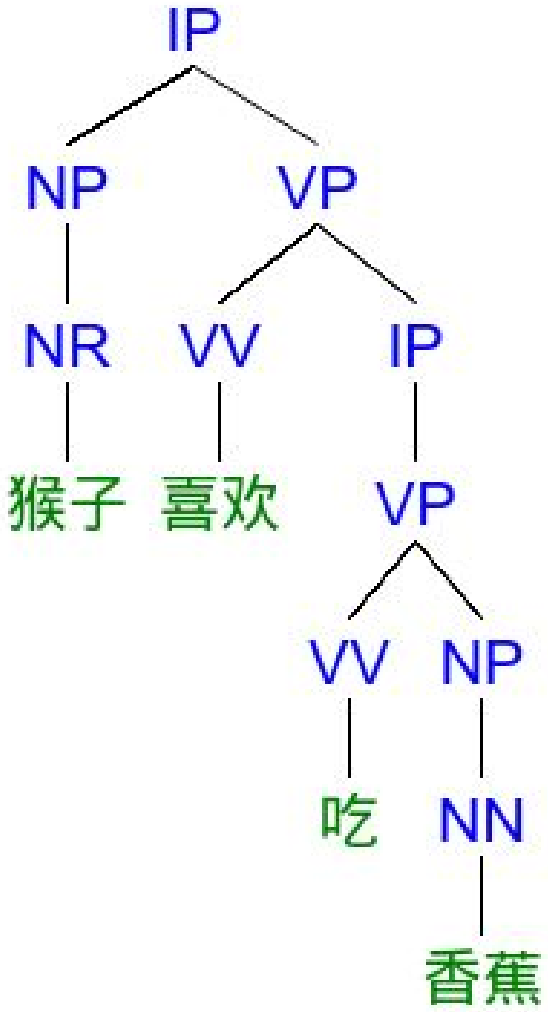
[S [NP [NP [NNS BIRDS]] [SBAR [WHNP [WDT THAT]] [S [VP [VBP SWIM]]]]] [VP [VBP FLY]]]



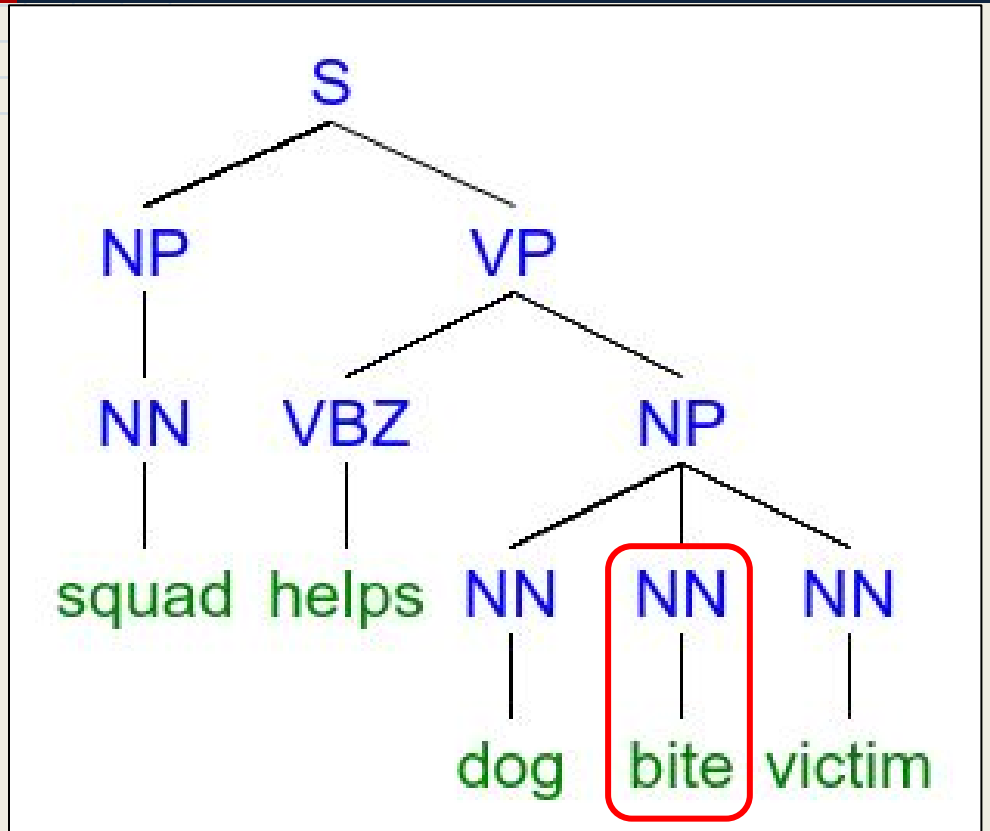
Chinese Example

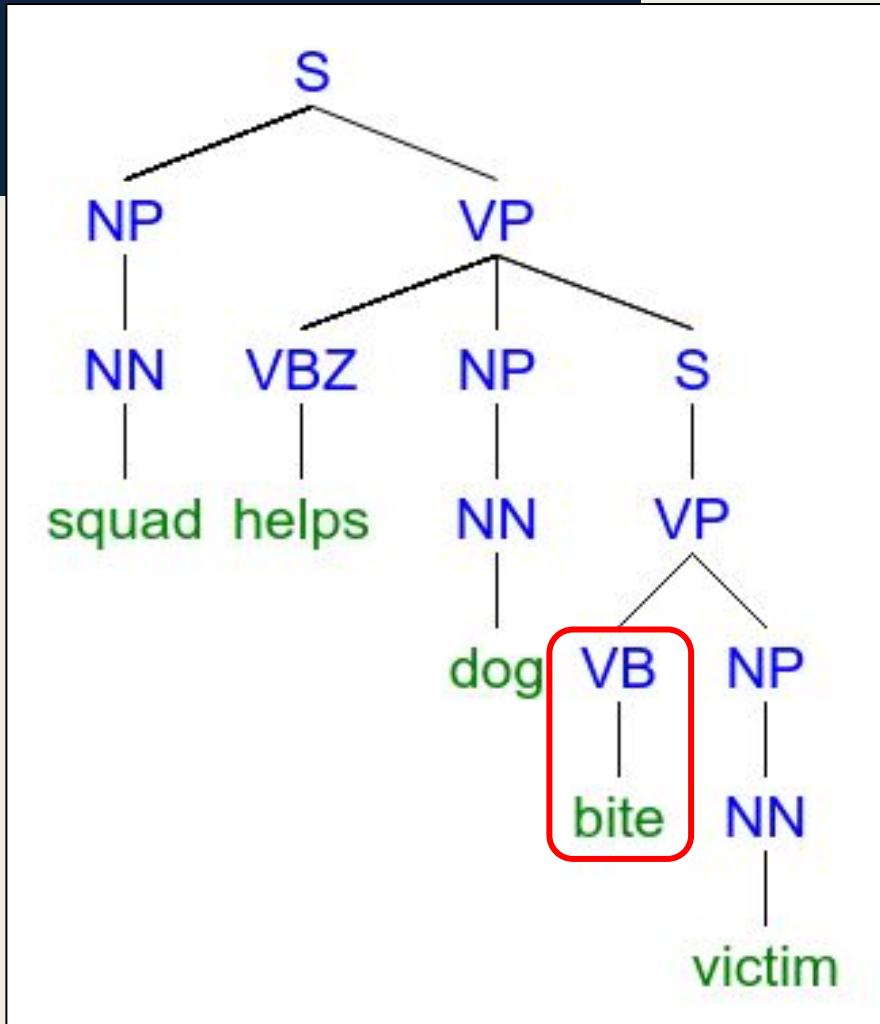
Different rules/labels are used for different languages

[IP [NP [NR 猴子]] [VP [VV 喜欢] [IP [VP [VV 吃] [NP [NN 香蕉]]]]]]

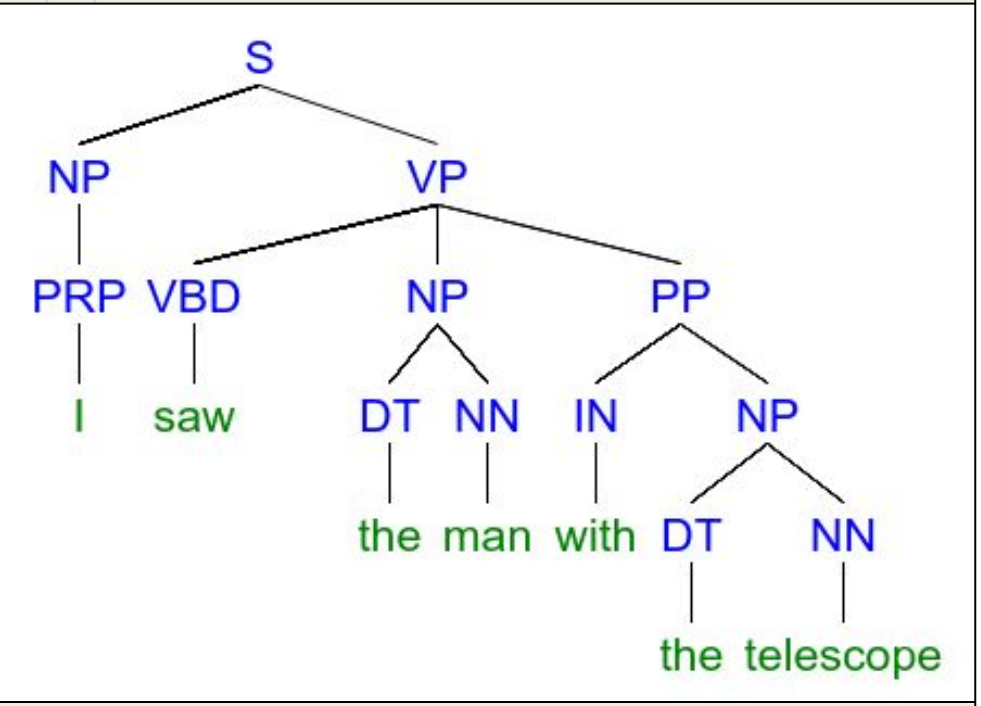


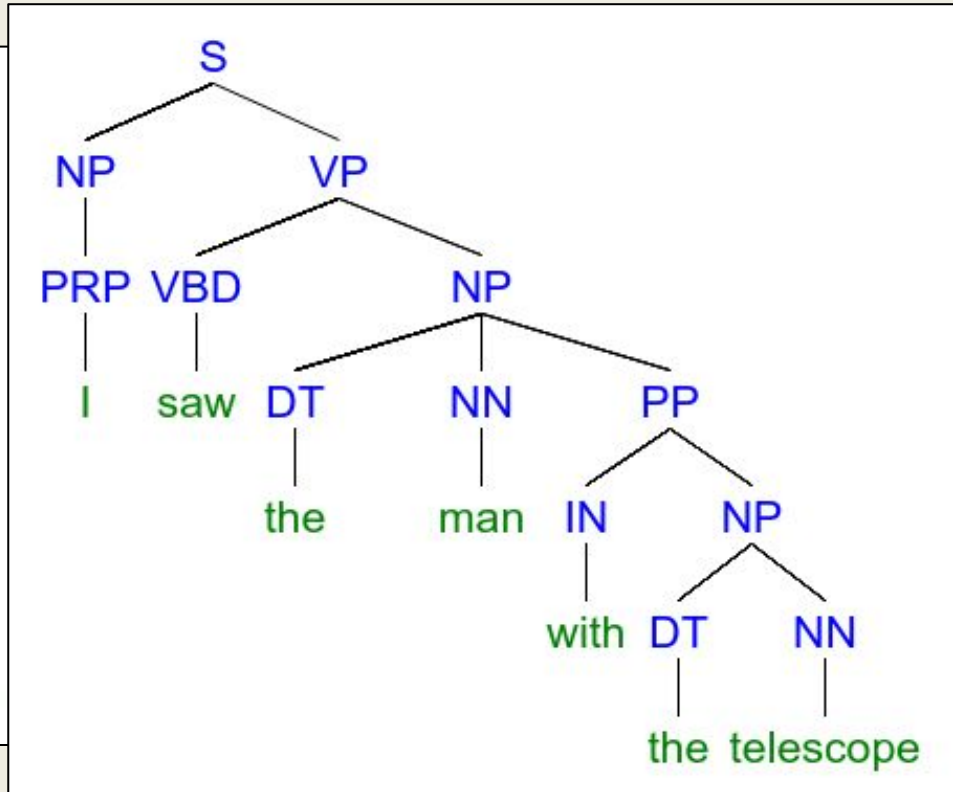
Lexical Ambiguity





Syntactic Ambiguity





Penn Treebank

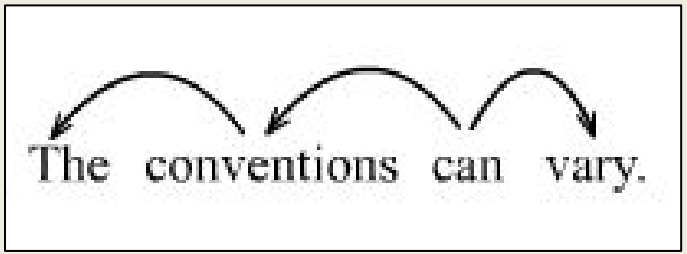
- Started in 1989
- More than 4.5 million tokens
- Mostly *Wall Street Journal*
- Constituency-parsed by humans
- Used to train/test parsers

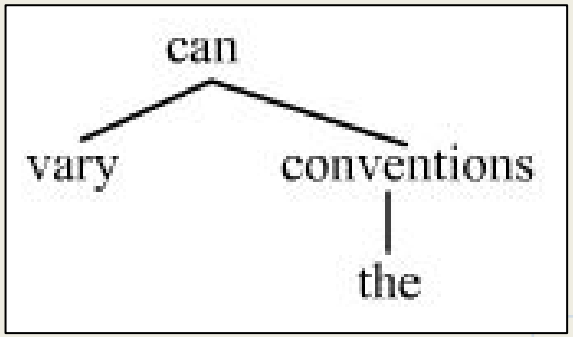
www.cis.upenn.edu/~treebank

Dependency Parsing

Represents text structure as a **tree**:

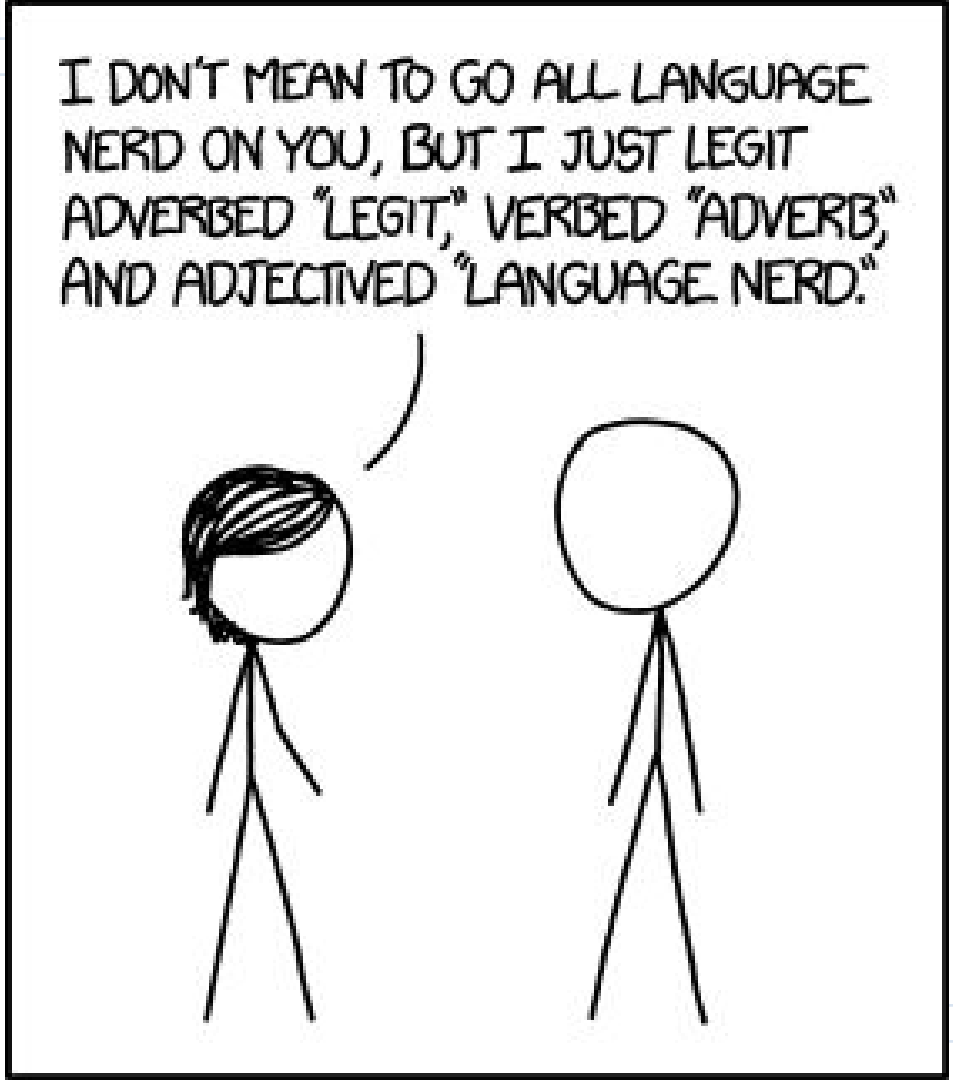
tokens are all the nodes (not just leaves)





References

- NLP class on Coursera: class.coursera.org/nlp
- Parts of speech: en.wikipedia.org/wiki/Part_of_speech
- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition. Prentice-Hall. Pg. 295.



xkcd.com/1443