# Syntactic Parsing

Human Language from a Computational Perspective
April 25, 2018

# 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

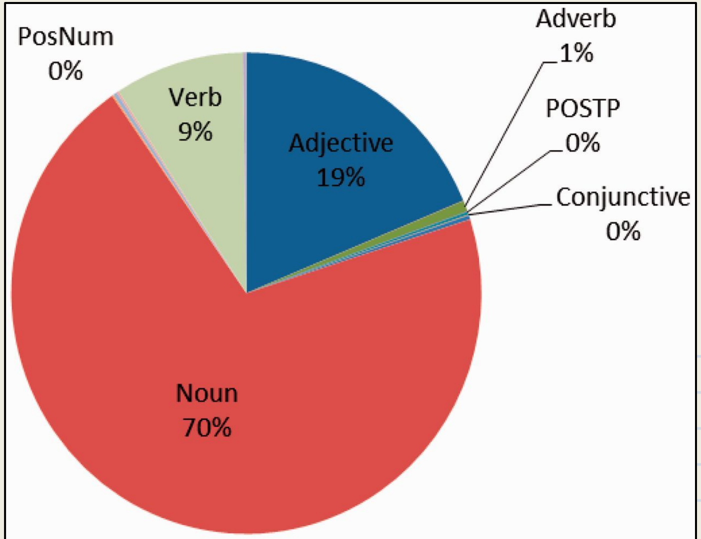
Squad helps dog **bite** victim

Eye **drops** off shelf

Reagan wins on budget, but more **lies** ahead

# Parts of speech (POS)

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



# Part-of-speech hierarchy

**Open classes**

- Nouns
  - Proper: HEBREW, JOHN
  - Common: ARRIVAL, PENCILS
- Verbs
  - Main: WENT, WRITE
  - Modals: WOULD, SHALL
- Adjectives: TALL
- Adverbs: QUICKLY
- Numbers: 2015, 1/2, BILLION, ONE

…

**Closed classes**

- Determiners SOME, THE
- Conjunctions AND, OR
- Pronouns YOU, MY, US
- Prepositions TO, IN, BY

…

# 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

# 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 | ISO |
| 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

# 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

Alice/NNP was/VBD beginning/VBG to/TO get/VB very/RB tired/JJ of/IN sitting/VBG by/IN her/PRP$ sister/NN on/IN the/DT bank/NN ,/, and/CC of/IN having/VBG nothing/NN to/TO do/VB ./. Once/RB or/CC twice/RB she/PRP had/VBD peeped/VBN into/IN the/DT book/NN her/PRP$ sister/NN was/VBD reading/VBG ,/, but/CC it/PRP had/VBD no/DT pictures/NNS or/CC conversations/NNS in/IN it/PRP ,/, "/" and/CC what/WDT is/VBZ the/DT use/NN of/IN a/DT book/NN ,/, "/" thought/VBD Alice/NNP ,/, "/" without/IN pictures/NNS or/CC conversations/NNS ?/. "/" So/CC she/PRP

# Word/tag 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 |

# Algorithm to count word/tag

```
count(L, T):                          ▷ L: list of tokens, T: list of correct tags
    Cwt ← [0]                         ▷ create a table of zeros
    i ← 1                             ▷ assign 1 to i
    while i ≤ len(L):                 ▷ repeat while i is at most len(L)=len(T)
        Cwt[L[i], T[i]] ← Cwt[L[i], T[i]] + 1      ▷ increment count
        i ← i + 1                     ▷ increment i
    return Cwt                        ▷ output is the counts table
```

# POS tagging algorithm

## Find POS sequence of token sequence

$$[\textsc{Bob went out for a swim .}] \rightarrow [\textsc{NNP VBD IN IN DT NN .}]$$

This is the wanted result, but no algorithm is perfect

# POS tagging algorithm 1

```
tag1(L, Cwt):                          ▷ L: tokens, Cwt: word-tag counts
    T ← []                             ▷ create empty list of tags
    i ← 1                              ▷ initialize i to 1
    while i ≤ len(L):                  ▷ repeat for all tokens
        T[i] ← argmax(Cwt[L[i], ·])    ▷ most common tag for L[i]
        i ← i + 1                      ▷ increment i
    return T                           ▷ output is list of tags
```

# POS tagging algorithm 1

```
tag1(L, Cwt):                          ▷ L: tokens, Cwt: word-tag counts
    T ← []                             ▷ create empty list of tags
    i ← 1                              ▷ initialize i to 1
    while i ≤ len(L):                  ▷ repeat for all tokens
        T[i] ← argmax(Cwt[L[i], ·])    ▷ most common tag for L[i]
        i ← i + 1                      ▷ increment i
    return T                           ▷ output is list of tags
```

The same algorithm as prediction with bigram

# Surprising accuracy

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

Most words almost always appear with the same POS.

# Problem 1: 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 2: 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 |

# Algorithm to count tag pairs

```
count(T):                                    ▹ T: list of correct tags from corpus
    Ct2 ← [0]                                ▹ create a table of zeros
    i ← 2                                    ▹ assign 2 to i
    while i ≤ len(L):                        ▹ repeat while i is at most len(L)
        Ct2[T[i − 1], T[i]] ← Ct2[T[i − 1], T[i]] + 1    ▹ increment
        i ← i + 1                            ▹ increment i
    return Ct2                               ▹ output is the counts table
```

The same algorithm as for counting word bigrams

# POS tagging algorithm 2

```
tag2(L, Cwt, Ct2):                          ▷ L: tokens, Cwt: word-tag counts,
                                                    ▷ Ct2: tag bigram counts
    T ← []
    T[1] ← argmax(Cwt[L[1], ·])             ▷ most common tag for first token
    i ← 2                                   ▷ initialize i to 2
    while i ≤ len(L):                       ▷ repeat for all tokens
        T[i] ← argmax(Cwt[L[i], ·] × Ct2[T[i − 1], ·])  ▷ multiply counts
        i ← i + 1                           ▷ increment i
    return T                                ▷ output is list of tags
```

# POS tagging algorithm 2

```
tag2(L, Cwt, Ct2):                      ▷ L: tokens, Cwt: word-tag counts,
                                                ▷ Ct2: tag bigram counts
    T ← []
    T[1] ← argmax(Cwt[L[1], ·])         ▷ most common tag for first token
    i ← 2                               ▷ initialize i to 2
    while i ≤ len(L):                   ▷ repeat for all tokens
        T[i] ← argmax(Cwt[L[i], ·] × Ct2[T[i − 1], ·])   ▷ multiply counts
        i ← i + 1                       ▷ increment i
    return T                            ▷ output is list of tags
```

Multiply corresponding elements in the two tables

# Combining counts

Cwt

| | | |
|---|---|---|
| THE | DT | 1527 |
| WELL | RB | 37 |
| WELL | NN | 3 |
| SLEEP | NN | 4 |
| SLEEP | VBP | 2 |

Ct2

| | | |
|---|---|---|
| NN | NN | 312 |
| NN | IN | 690 |
| NN | RB | 113 |
| IN | NN | 262 |
| DT | NN | 1256 |

Tag the short sentence: SLEEP WELL

# Combining counts

Cwt

| | | |
|---|---|---|
| THE | DT | 1527 |
| WELL | RB | 37 |
| WELL | NN | 3 |
| SLEEP | NN | 4 |
| SLEEP | VBP | 2 |

Ct2

| | | |
|---|---|---|
| NN | NN | 312 |
| NN | IN | 690 |
| NN | RB | 113 |
| IN | NN | 262 |
| DT | NN | 1256 |

Tag the short sentence: SLEEP WELL

NN

# Combining counts

Cwt

| | | |
|---|---|---|
| THE | DT | 1527 |
| WELL | RB | 37 |
| WELL | NN | 3 |
| SLEEP | NN | 4 |
| SLEEP | VBP | 2 |

Ct2

| | | |
|---|---|---|
| NN | NN | 312 |
| NN | IN | 690 |
| NN | RB | 113 |
| IN | NN | 262 |
| DT | NN | 1256 |

NN: 3×312=936

RB: 37×113=4181

Tag the short sentence: SLEEP WELL

NN RB

# Phrases

Parts of speech are for single words, but multi-word phrases may have similar syntactic roles.

# Noun phrases (NP)

I saw a [dog]

[small dog]

[small dog with a black tail]

# Verb phrases (VP)

I [walk]

I [walk home]

I [walk home quickly but surely]

# Adjective phrases (AP)

This car is [fast]

This car is [really very fast]

This car is [faster than my old one]

# Prepositional phrases (PP)

Cats fall [on their feet]

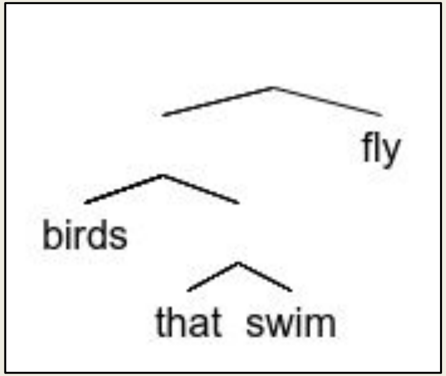
I'm wearing the shirt [from Italy]

I'm taking the bus [from Tel Aviv]

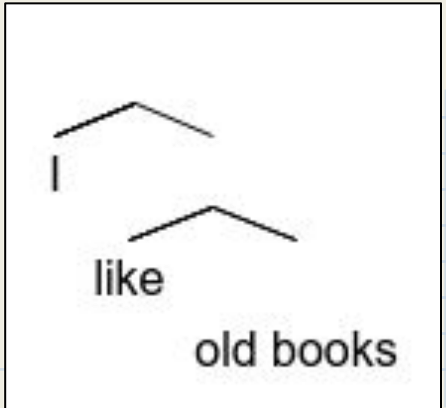
# Bracketing

## A hierarchy of **constituents**

[[BIRDS [THAT SWIM]] FLY]

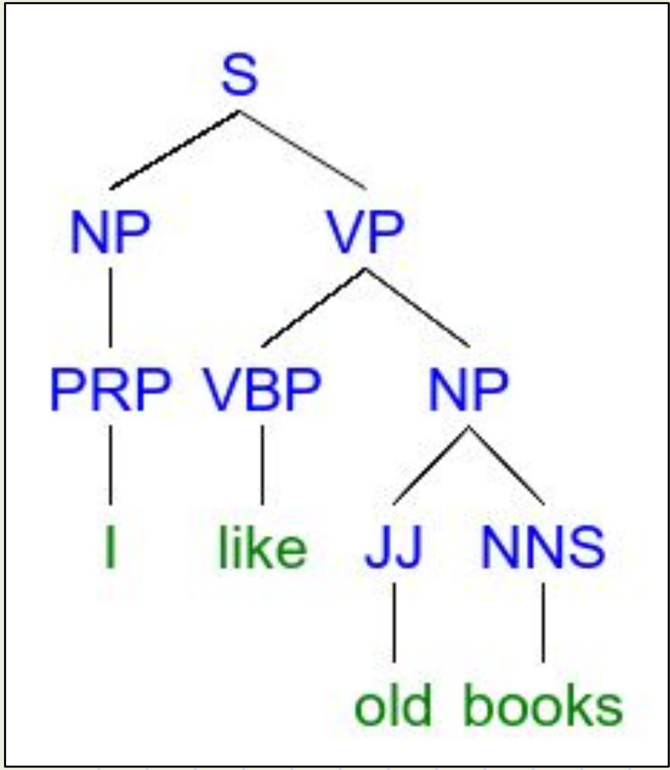



[I [LIKE [OLD BOOKS]]]



# Phrase structure

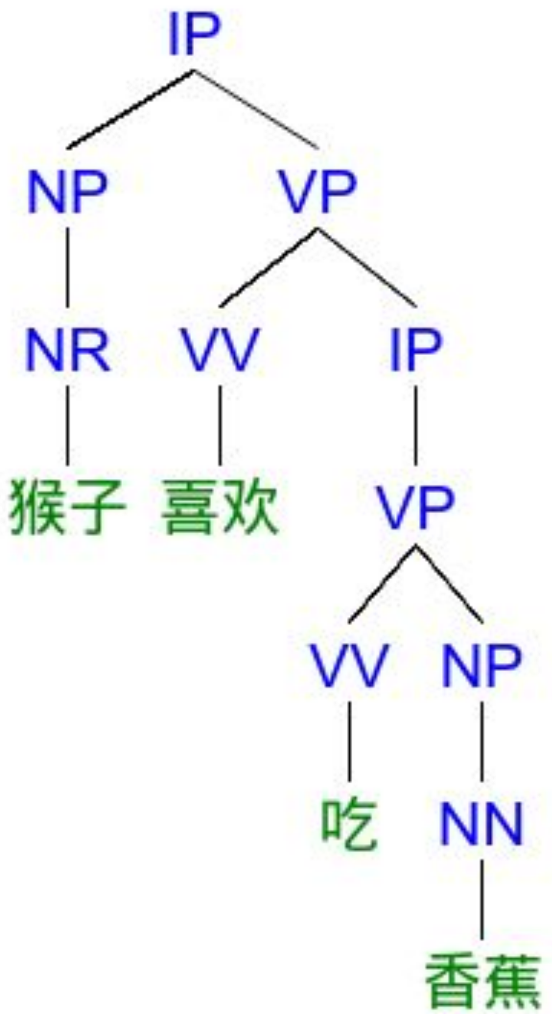
Represents text structure as a **tree**: tokens are leaves



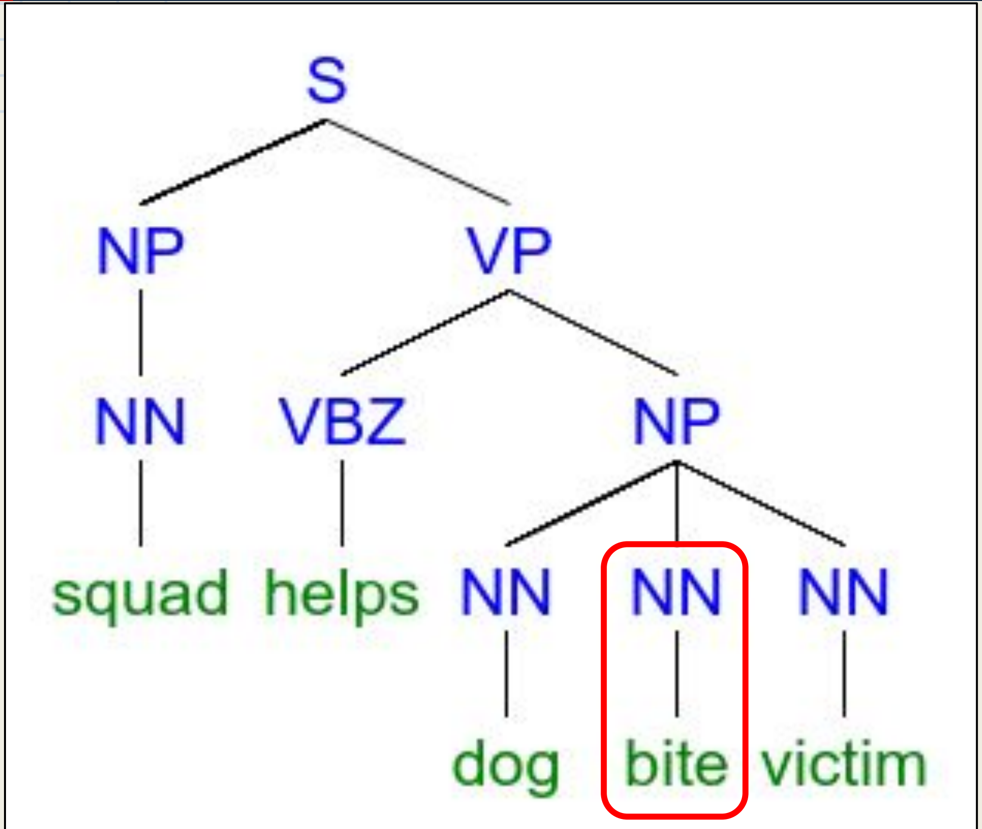

mshang.ca/syntree

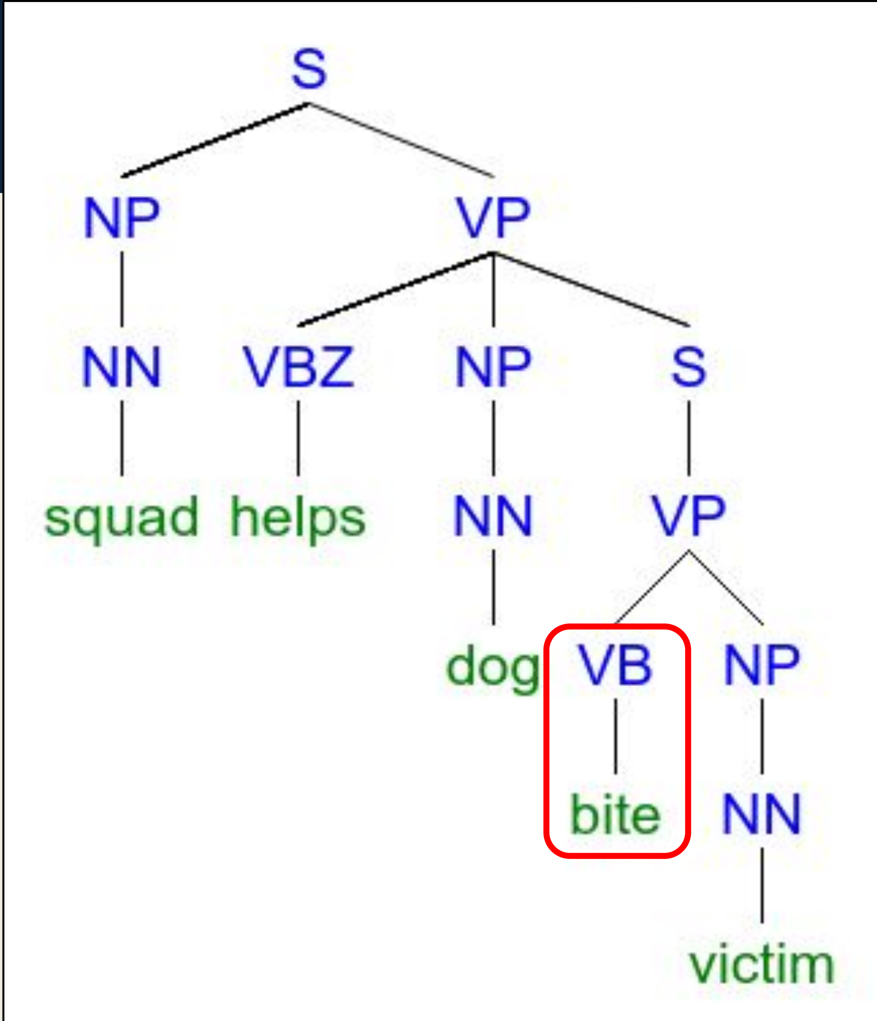
# Chinese example

Different rules/labels are used for different languages

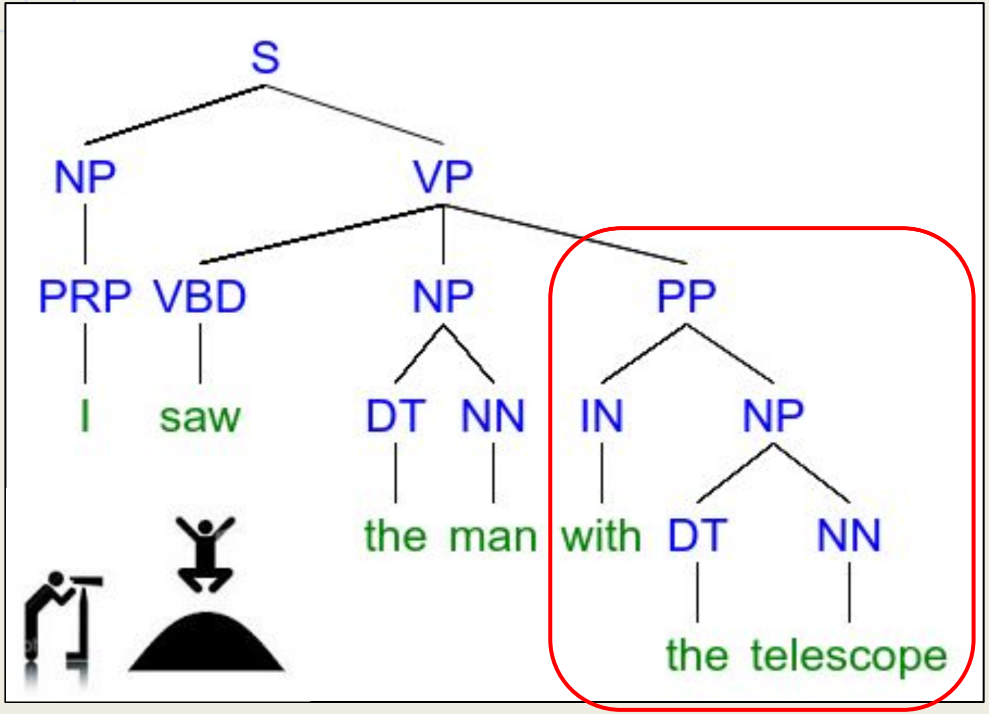


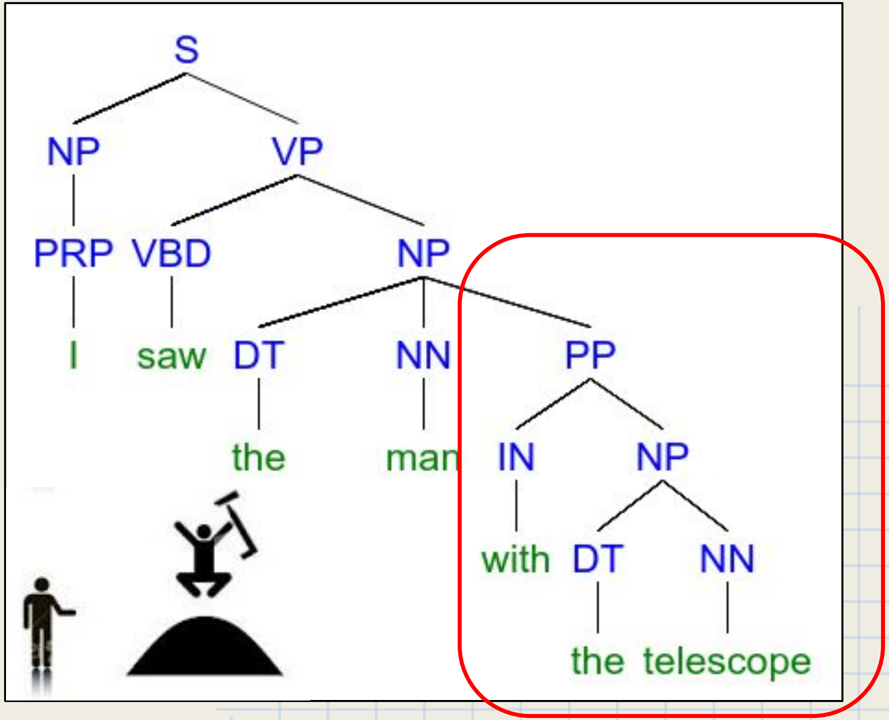
# Lexical ambiguity

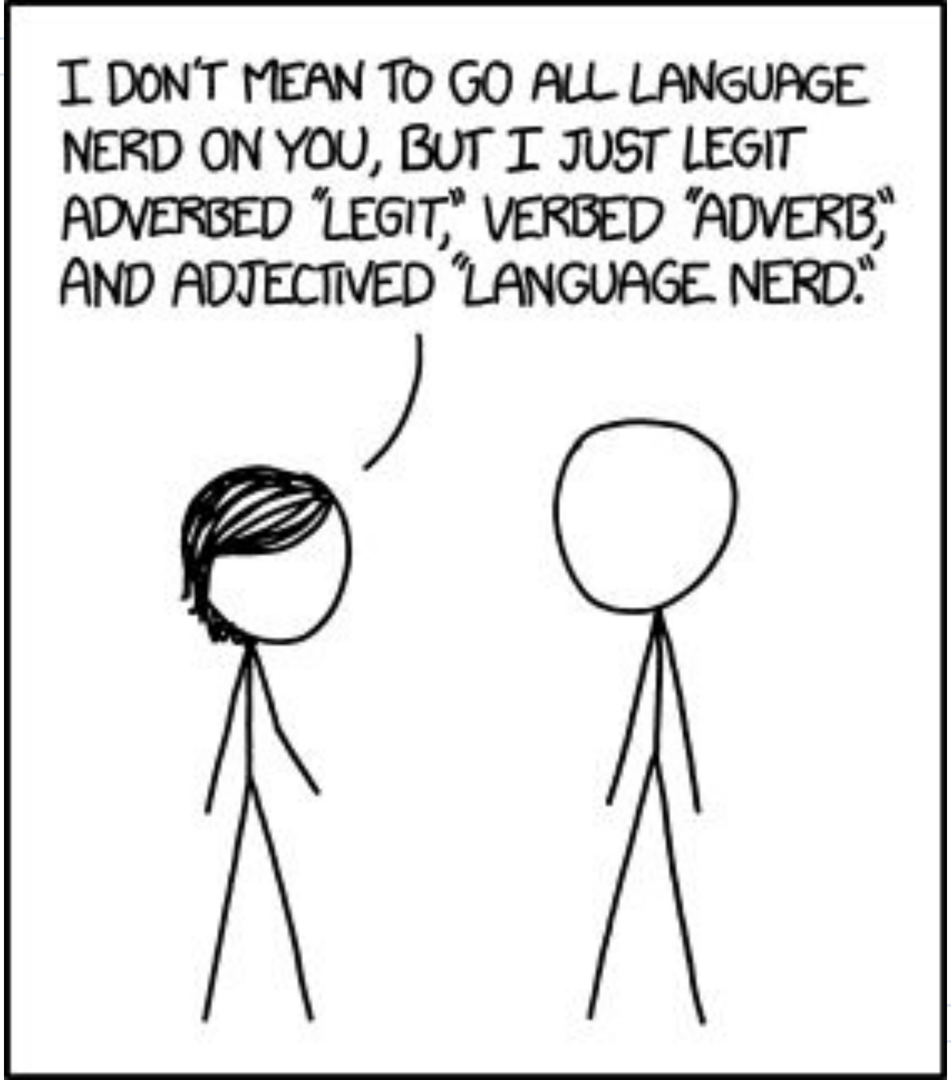


# Syntactic ambiguity



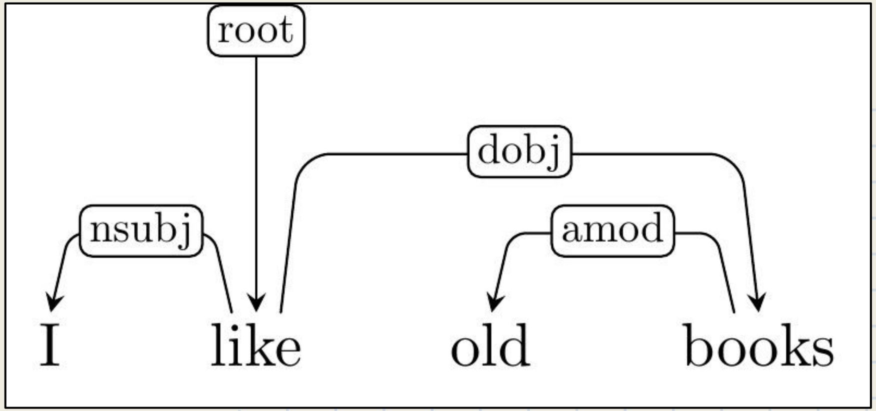



xkcd.com/1443

# Dependency parsing

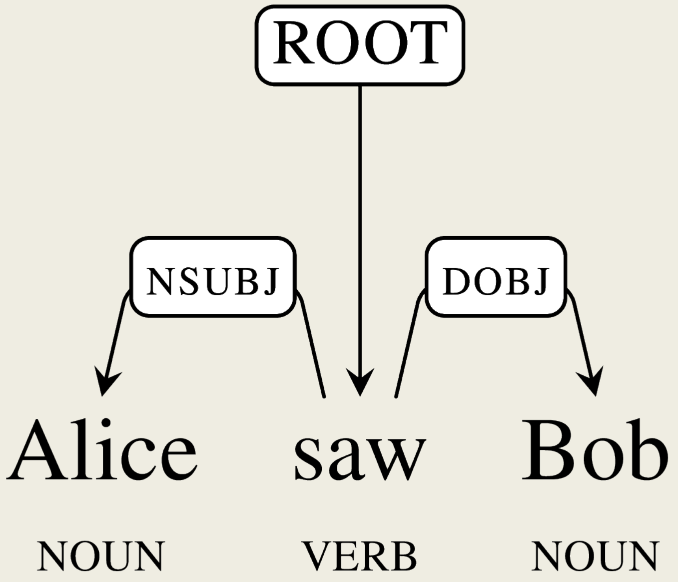
Represents text structure as a **tree**:

tokens are all the nodes (not just leaves)

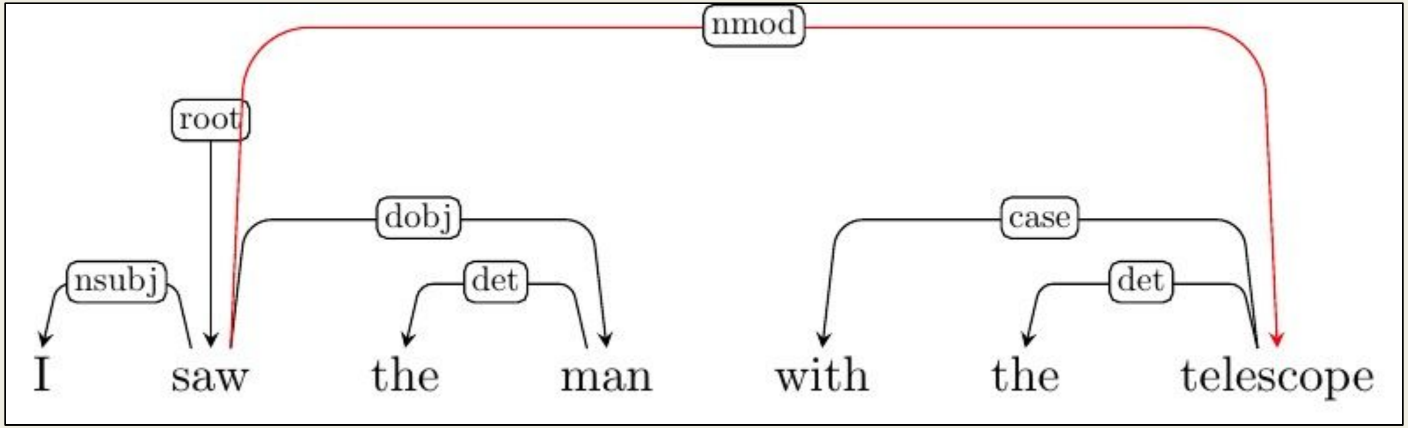


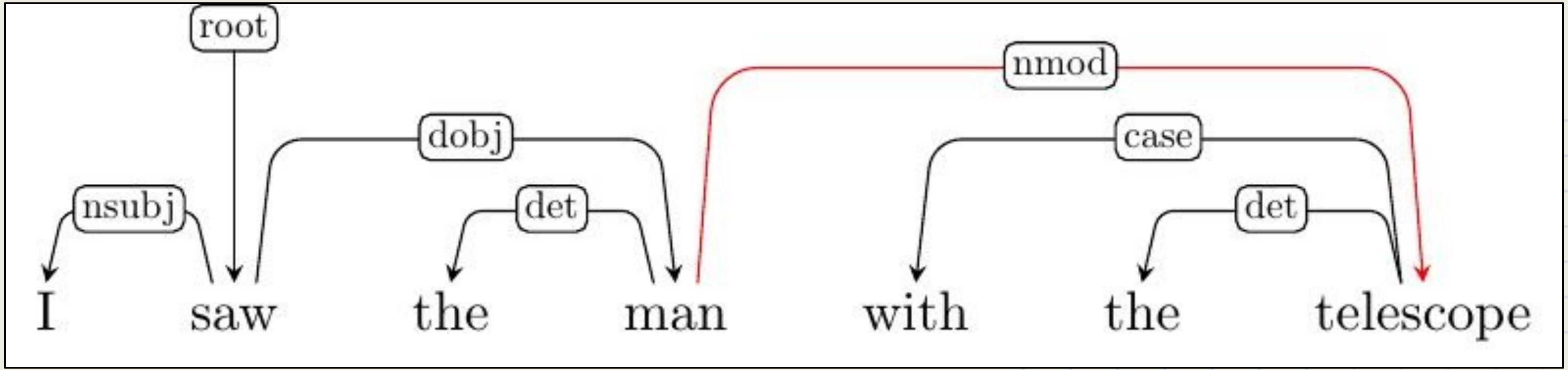
googleresearch.blogspot.co.il/2016/05/announcing-syntaxnet-worlds-most.html

# Dependency Parsing

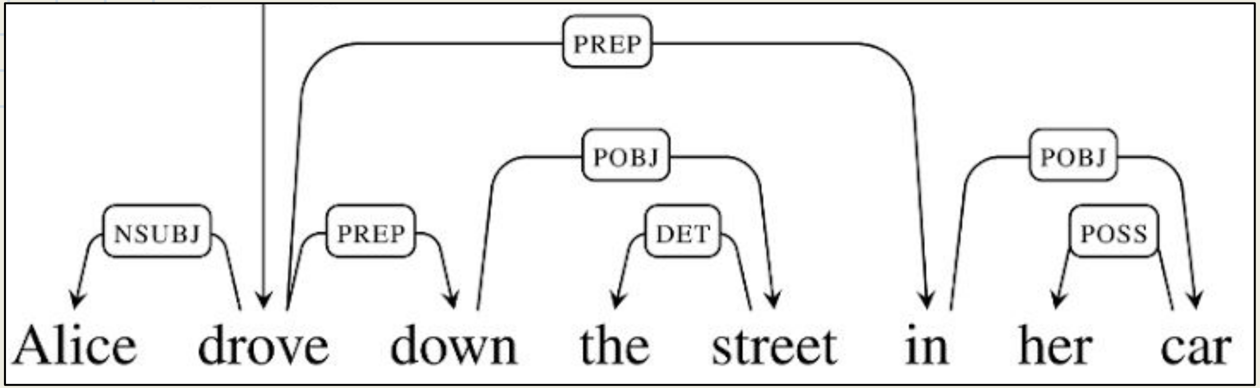


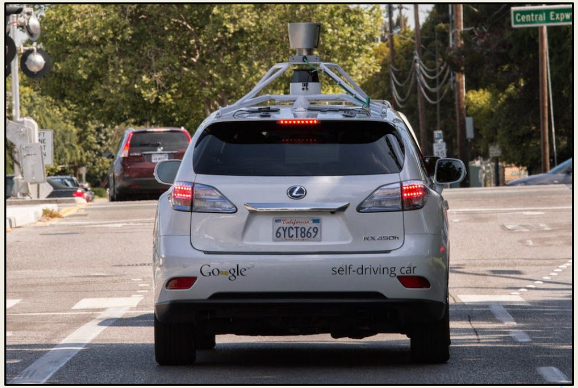
# Syntactic ambiguity

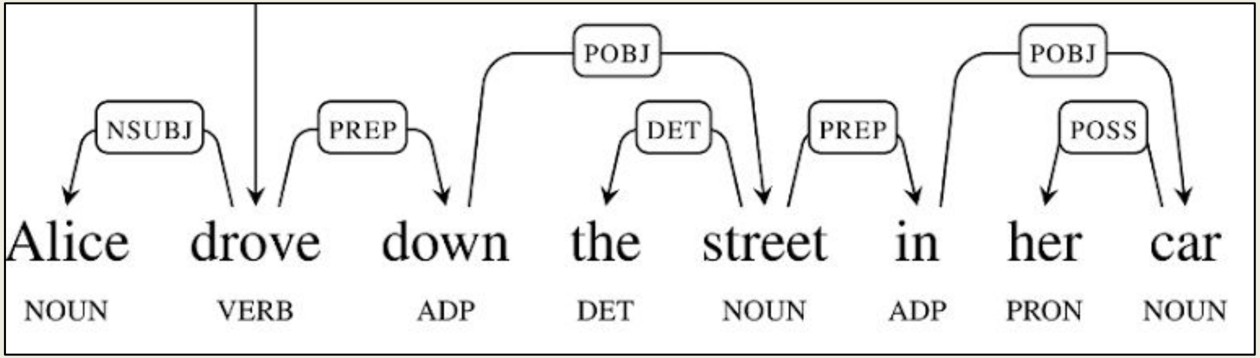


# Syntactic Ambiguity

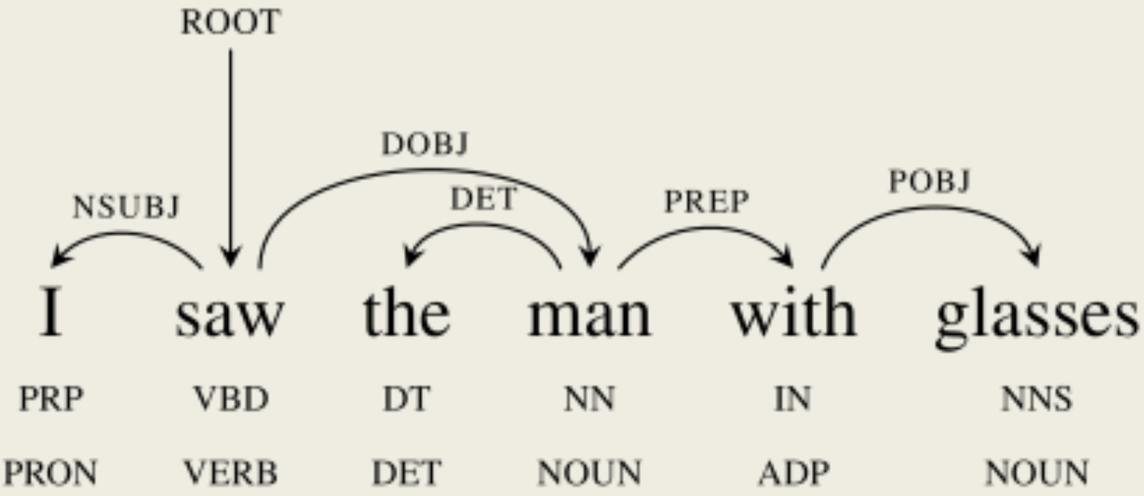


github.com/tensorflow/models/tree/master/syntaxnet

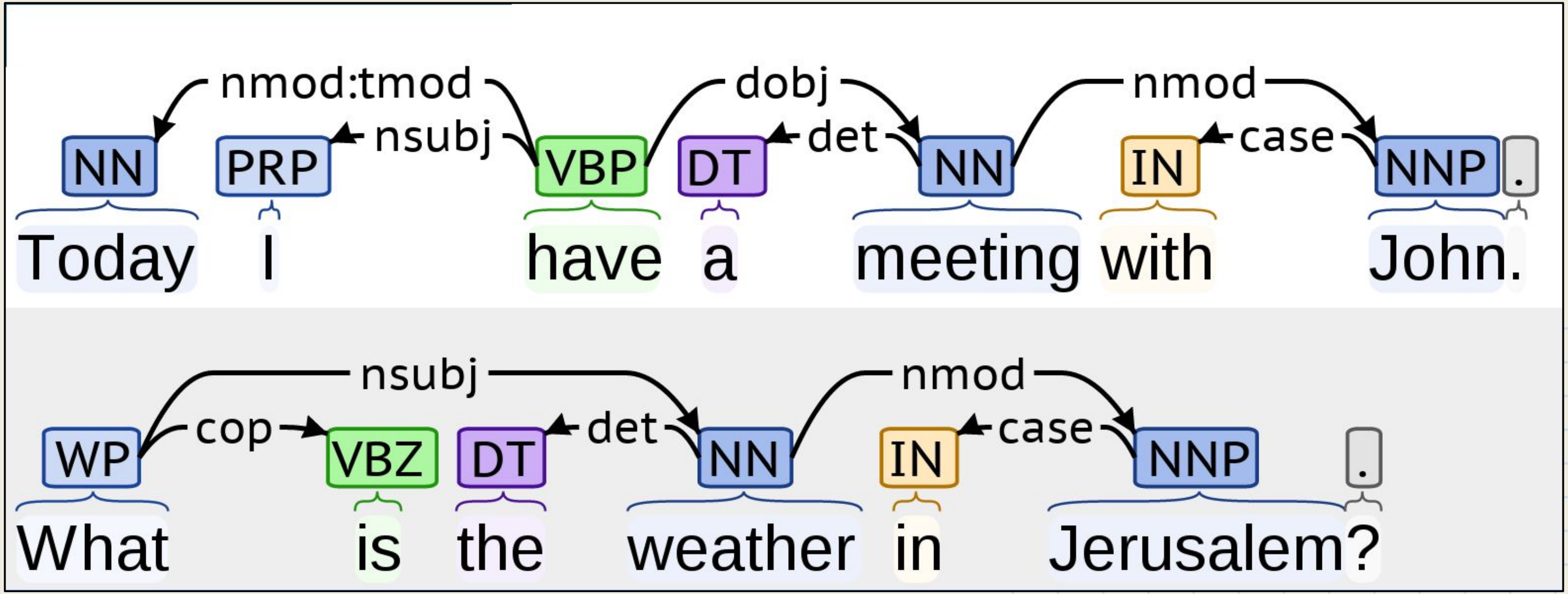
# Syntactic Ambiguity




## Who had the glasses?

# Representation

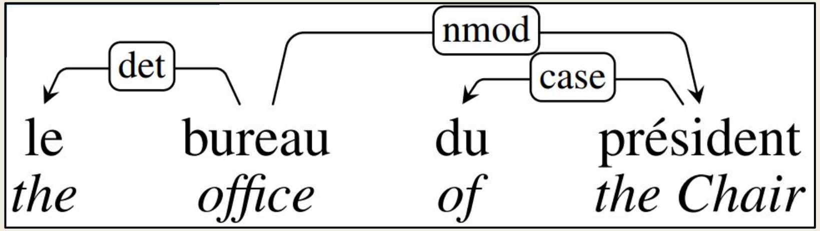
**Natural Language Understanding**: who did what to whom and where and when and how and why?

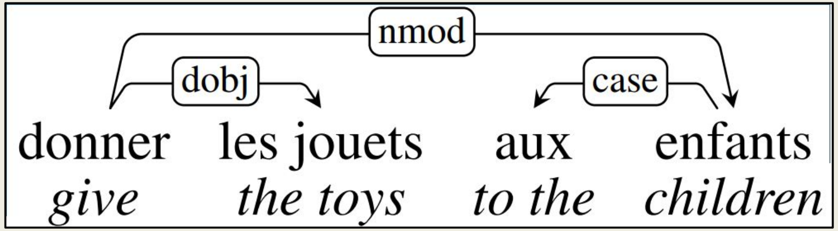


universaldependencies.org

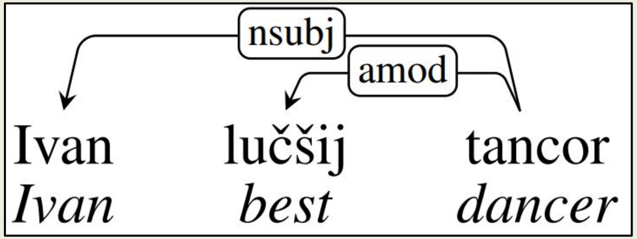
# Cross-Linguistic Examples

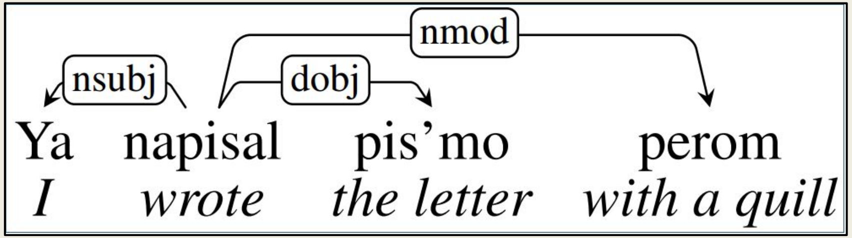
French



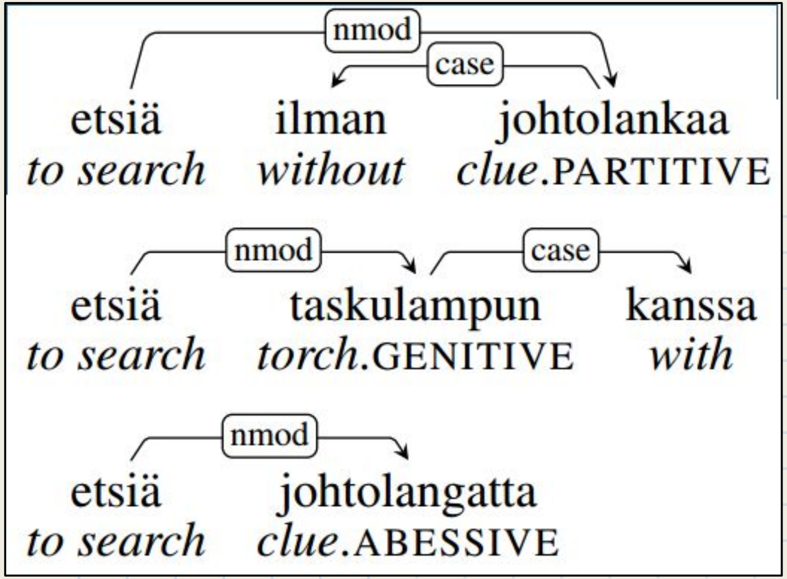




Russian




Finnish



lindat.mff.cuni.cz/services/pmltq/#!/treebanks

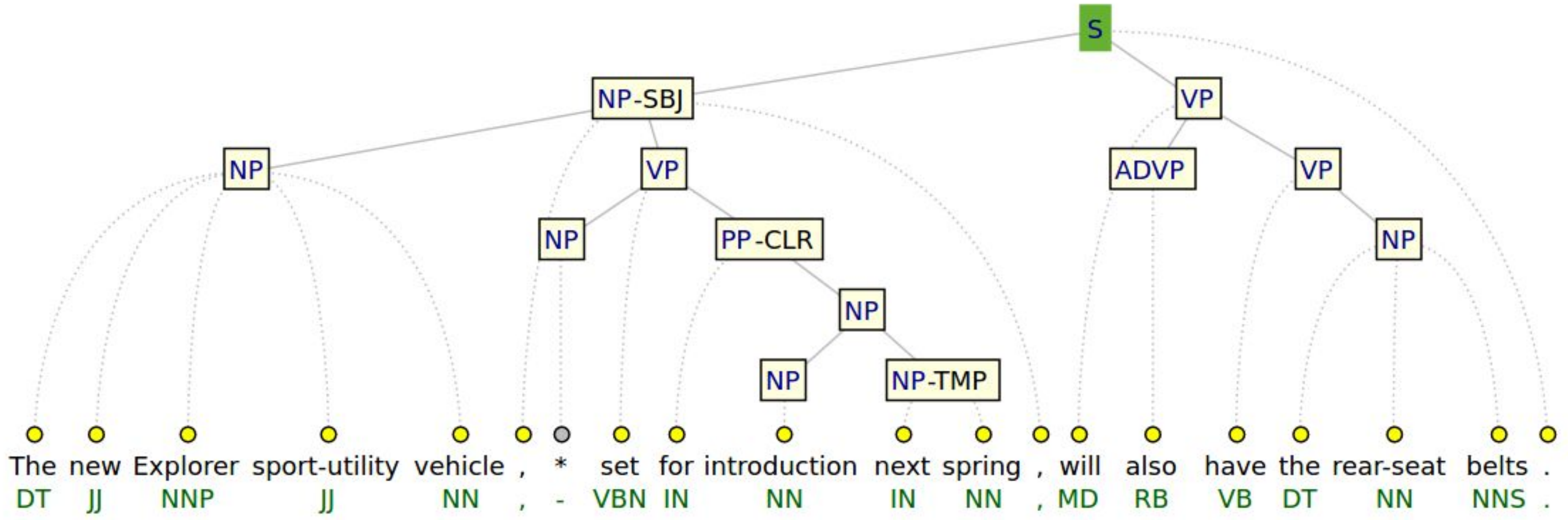
# Resources: Treebanks

- Many text corpora parsed by humans
- Used for training automatic parsers

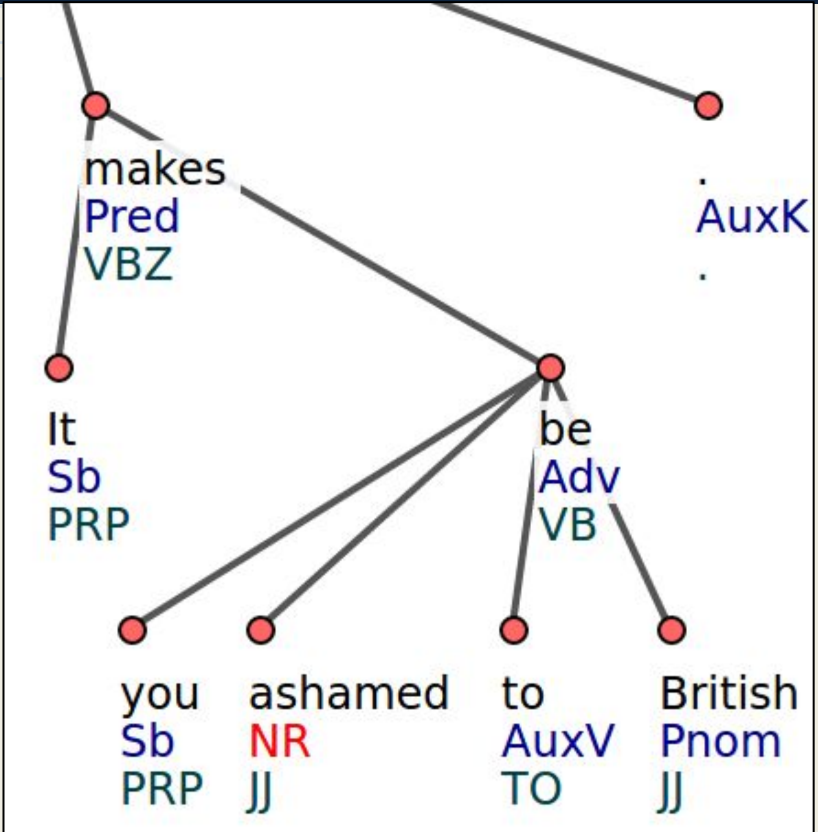
**LANGUAGES:** Ancient Greek 3 · Arabic 2 · Basque 1 · Bengali 1 · Bulgarian 2 · Catalan 2 · Chinese 3 · Croatian 1 · Czech 11 · Danish 1 · Dutch 2 · English 11 · Estonian 2 · Finnish 2 · French 1 · German 4 · Gothic 1 · Greek 1 · Hebrew 1 · Hindi 3 · Hungarian 1 · Indonesian 1 · Irish 1 · Italian 1 · Japanese 2 · Latin 7 · Norwegian 1 · Old Church Slavonic 1 · Persian 2 · Polish 2 · Portuguese 2 · Romanian 1 · Russian 1 · Slovak 1 · Slovenian 3 · Spanish 3 · Swedish 2 · Tamil 2 · Telugu 2 · Turkish 1

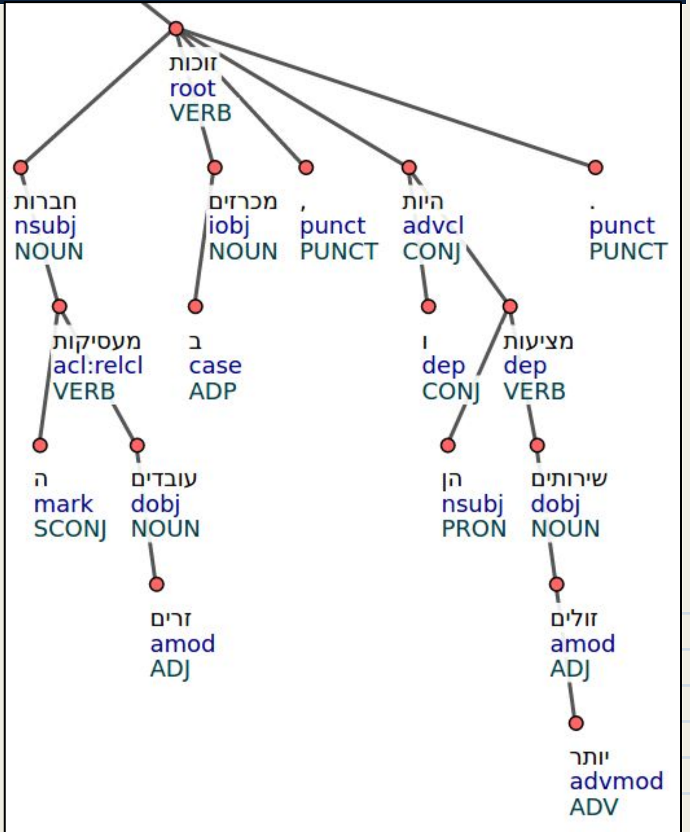
**TAGS:** CoNLL 8 · HamleDT 24 · PDT 3 · Penn Treebank 6 · Treex 1 · Universal Dependencies 35 www.cis.upenn.edu/~treebank

# Penn Treebank (Constituency)



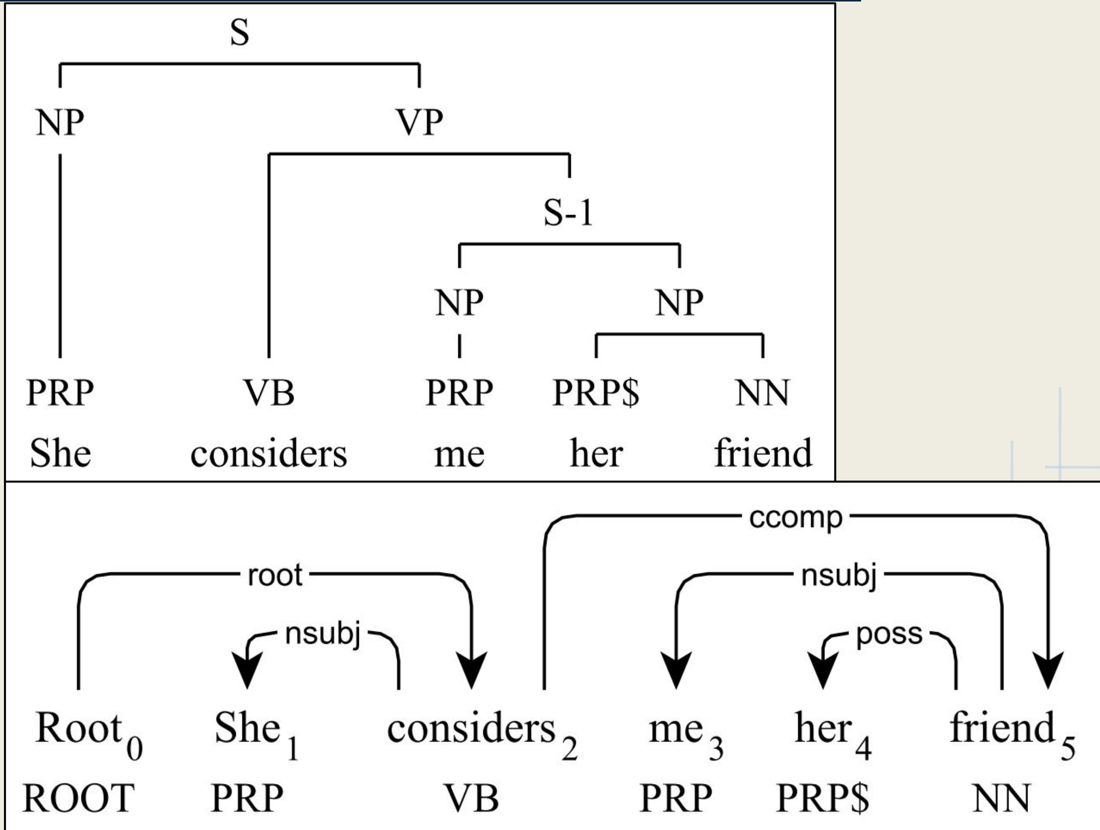
# Dependency Treebanks



# Converted Treebanks

Trees can be automatically converted to save manual work



# Evaluation

Labeled Attachment Score (LAS):

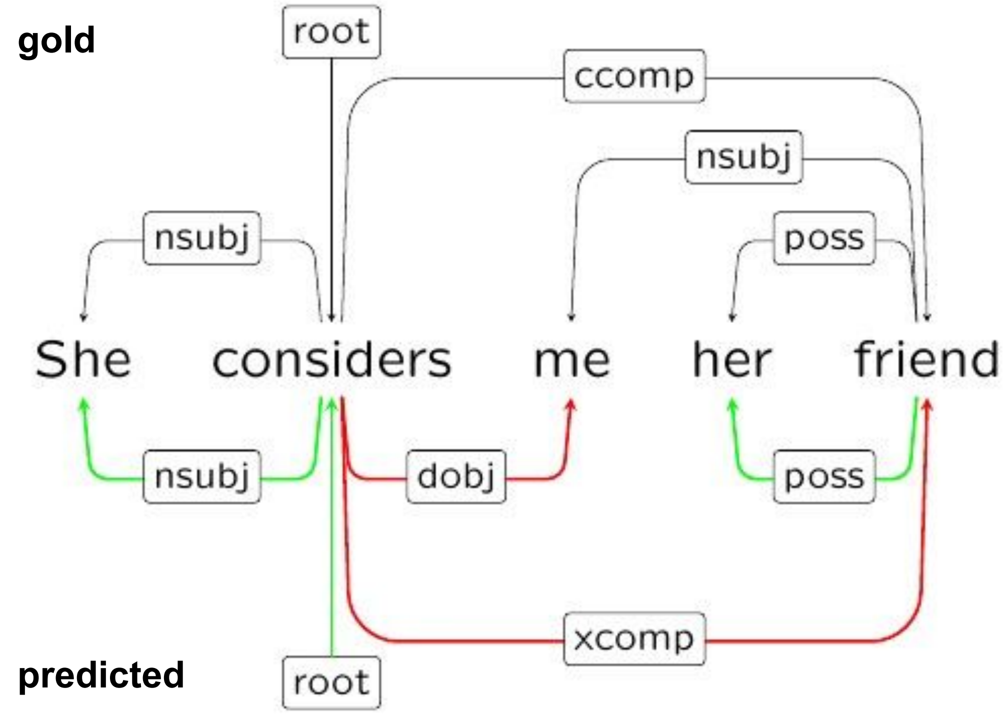
% of edges both in predicted tree and in gold tree

Unlabeled Attachment Score (UAS):

same as LAS, but ignoring edge label

$0 \leq \text{LAS} \leq \text{UAS} \leq 100\%$

# Evaluation Example




$$\mathbf{LAS} = \frac{3}{5} = 60\%$$

$$\mathbf{UAS} = \frac{4}{5} = 80\%$$

# Parser Scores

| Parser | UAS | LAS |
|---|---|---|
| MaltParser | 90.93 | 88.95 |
| MSTParser | 92.17 | 89.86 |
| ZPar | 92.93 | 91.28 |
| TurboParser | 93.80 | 92.00 |
| Parsey McParseface | 94.41 | 92.55 |

# Summary

| | Dependency | Constituency |
|---|---|---|
| **Structure** | tree | tree |
| **Tokens are** | all nodes | only leaves |
| **Labels on** | edges | nodes |

# 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.