Ambiguity

Find at least 5 meanings of this sentence:

I made her duck

- I cooked waterfowl for her benefit (to eat)
- I cooked waterfowl belonging to her
- I created the (plaster?) duck she owns
- I caused her to quickly lower her head or body
- I waved my magic wand and turned her into undifferentiated waterfowl

Ambiguity is Everywhere

- Lexical category: part of speech
  - Duck can be a Noun or Verb
    - V: *Duck!* I caused her to quickly lower her head or body.
    - N: I cooked waterfowl for her benefit
  - Her can be possessive (of her) or dative (for her)
    - Possessive: I cooked waterfowl belonging to her.
    - Dative: I cooked waterfowl for her benefit
- Lexical Semantics:
  - Make can mean create or cook
    - create: I made the (plaster) duck statue she owns
    - cook: I cooked waterfowl for her benefit

Examples (Challenges)

1. **"Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo"**

1. **Will, will Will will Will Will's will?**

1. **Police police Police police police police Police police.**

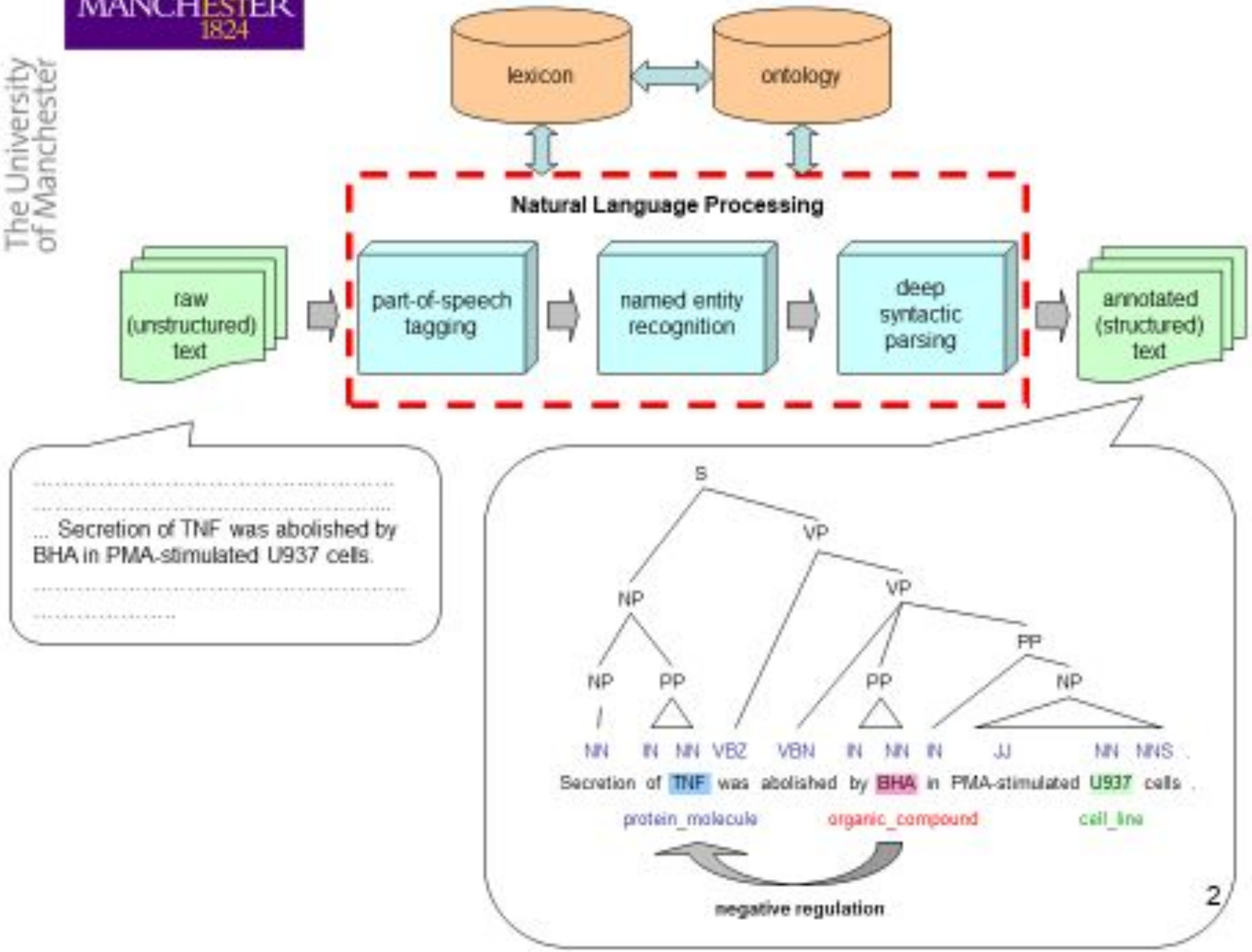
1. **Rose rose to put rose roes on her rows of roses**

1. **Can can can can can can can can can can**

other example

- **Time flies like an arrow; fruit flies like a banana.**
- **Actions speak louder than words.**

NLP fundamentals


lexicon
ontology
Natural Language Processing
raw (unstructured) text
part-of-speech tagging
named entity recognition
deep syntactic parsing
annotated (structured) text
... Secretion of TNF was abolished by BHA in PMA-stimulated U937 cells.
S
VP
NP
VP
PP
NP
PP
PP
NP
NN IN NN VBZ VBN IN NN IN JJ NN NNS .
Secretion of TNF was abolished by BHA in PMA-stimulated U937 cells .
protein_molecule
organic_compound
cell_line
negative regulation

MANCHESTER 1824

The University of Manchester

Basic Steps of Natural Language Processing

- Sentence splitting
- Tokenization
- Part-of-speech tagging
- Shallow parsing
- Named entity recognition
- Syntactic parsing
- (Semantic Role Labeling)

```
User:  I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:  I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:  Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
User:  My mother takes care of me.
ELIZA: WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:  My father.
ELIZA: YOUR FATHER
User:  You are like my father in some ways.
```

Weizenbaum (1966)

Regular Expressions

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- Negations `[^Ss]`
  - Carat means negation only when first in []

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	Oyfn pripetchik
`[^Ss]`	Neither 'S' nor 's'	I have no exquisite reason"
`[^e^]`	Neither e nor ^	Look here
`a^b`	The pattern a carat b	Look up a^b now

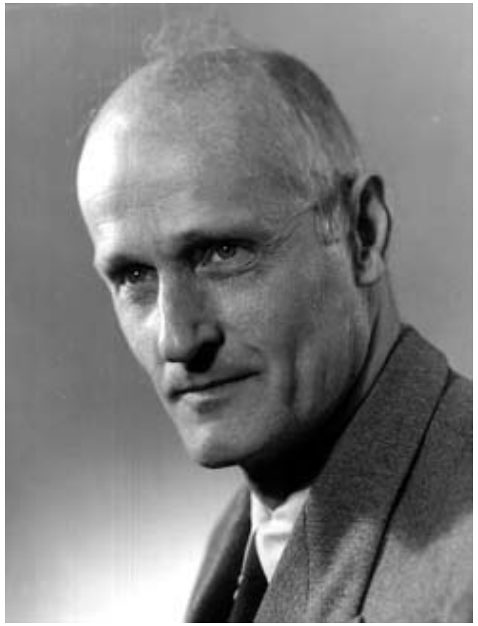
Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

Pattern	Matches
`^[A-Z]`	Palo Alto
`^[^A-Za-z]`	1 "Hello"
`\.$`	The end.
`.$`	The end? The end!

- ^ – The caret anchor matches the beginning of the text.
- $ – The dollar anchor matches the end of the text.

Example

- Find me all instances of the word “the” in a text.

  `the`

  Misses capitalized examples

  `[tT]he`

  Incorrectly returns `other` or `theology`

  `[^a-zA-Z][tT]he[^a-zA-Z]`

Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

Summary

- Regular expressions play a surprisingly large role
    - Sophisticated sequences of regular expressions are often the first model for any text processing
- For many hard tasks, we use machine learning classifiers
    - But regular expressions can be used as features in the classifiers
    - Can be very useful in capturing generalizations

Word tokenization

Text Normalization

- Most NLP tasks need to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens …………(or 14)?
  - 13 types ……………(or 12)?

How many words?

N = number of tokens

V = vocabulary = set of types

|*V*| is the size of the vocabulary

Switchboard is a collection of about 2,400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States.

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

https://books.google.com/ngrams/

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc "A-Za-z" "\n" < sample.txt
      | sort
      | uniq -c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... …
```

The first step: tokenizing

```
tr -sc "A-Za-z" "\n" < sample.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc "A-Za-z" "\n" < sample.txt | sort | head

A
A
A
A
A
A
A
A
A
...
```

More counting

- Merging upper and lower case

```
tr "A-Z" "a-z" < sample.txt | tr -sc "A-Za-z" "\n" | sort | uniq -c
```

- Sorting the counts

```
tr "A-Z" "a-z" < sample.txt | tr -sc "A-Za-z" "\n" | sort | uniq -c | sort -n -r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

A heuristic rule for sentence splitting

sentence boundary

= period + space(s) + capital letter

Regular expression in Perl

```
s/\. +([A-Z])/\.\n\1/g;
```

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L’*** ? ***Le*** ?
    - Want ***l’ensemble*** to match with ***un ensemble***
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - ‘life insurance company employee’
  - German information retrieval needs **compound splitter**

Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Word Normalization and Stemming

Normalization

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero (‘I want’), quieres (‘you want’) same lemma as querer ‘want’

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

➡

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
               sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize   digitizer → digitize
ator→ ate   operator  → operate
…
```

Step 3 (for longer stems)

```
al    → ø  revival    → reviv
able  → ø  adjustable → adjust
ate   → ø  activate   → activ
…
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

```
(*v*)ing → ø  walking    → walk
               sing       → sing
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

```
(*v*)ing → ø  walking   → walk
               sing      → sing
```

```
tr -sc "A-Za-z" "\n" < sample.txt | grep "ing$" | sort | uniq -c | sort -nr
```

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	102 going

```
tr -sc "A-Za-z" "\n" < sample.txt | grep "[aeiou].*ing$" | sort | uniq -c | sort -nr
```

Dealing with complex morphology is sometimes necessary

- Some languages require complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - \`(behaving) as if you are among those whom we could not civilize’
  - Uygar \`civilized’ + las \`become’

    + tir \`cause’ + ama \`not able’

    + dik \`past’ + lar ‘plural’

    + imiz ‘p1pl’ + dan ‘abl’

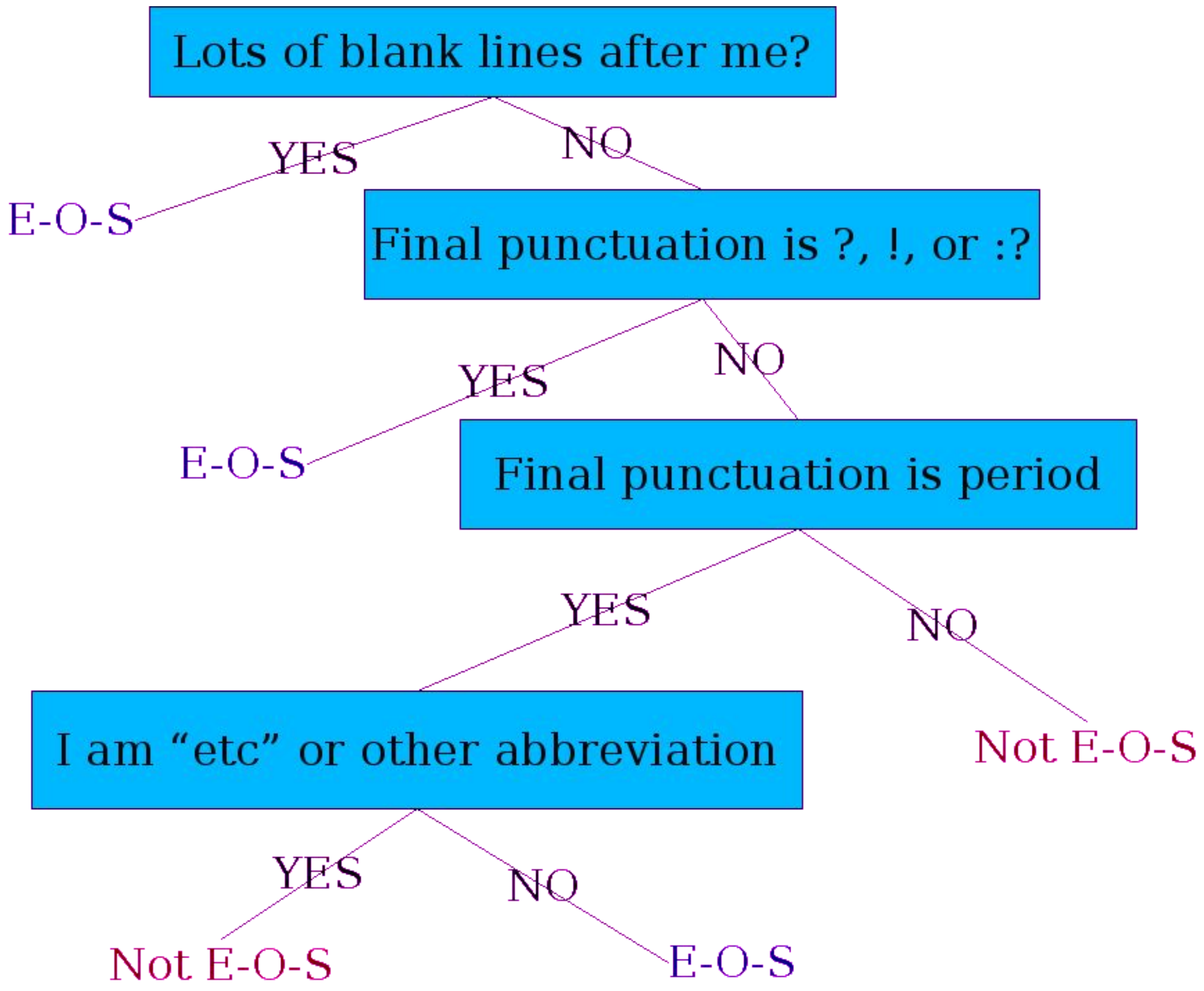
    + mis ‘past’ + siniz ‘2pl’ + casina ‘as if’

Sentence Segmentation and Decision Trees

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number

- Numeric features
  - Length of word with “.”
  - Probability(word with “.” occurs at end-of-s)
  - Probability(word after “.” occurs at beginning-of-s)

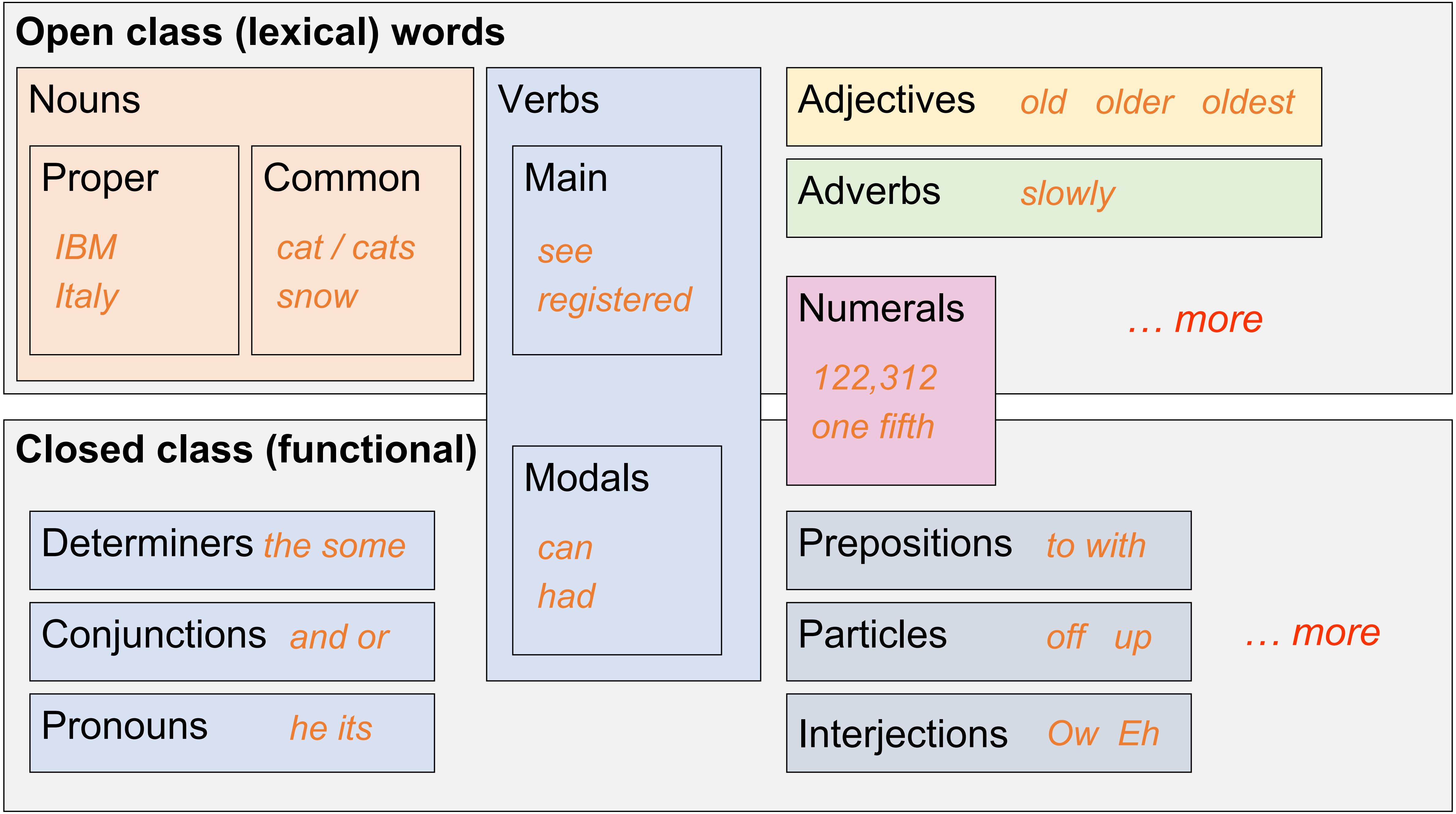
Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, it’s too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus

Decision Trees and other classifiers

- We can think of the questions in a decision tree as features that could be exploited by any kind of classifier:
  - Logistic regression
  - SVM
  - Neural Nets
  - etc.

Part-of-speech (POS) tagging


Open class (lexical) words
Nouns
Proper
IBM
Italy
Common
cat / cats
snow
Verbs
Main
see
registered
Modals
can
had
Adjectives old older oldest
Adverbs slowly
Numerals
122,312
one fifth
… more
Closed class (functional)
Determiners the some
Conjunctions and or
Pronouns he its
Prepositions to with
Particles off up
Interjections Ow Eh
… more

POS Tagging

- Words often have more than one POS: *back*
  - The ***back*** door = JJ
  - On my ***back*** = NN
  - Win the voters ***back*** = RB
  - Promised to ***back*** the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS Tagging

- Input: Plays well with others
- Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
- Output: Plays/VBZ well/RB with/IN others/NNS
- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - As input to or to speed up a full parser
  - If you know the tag, you can back off to it in other tasks

Penn Treebank POS tags

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!

Deciding on the correct part of speech 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 250/CD

How difficult is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can’t go *that* far = RB
- 40% of the word tokens are ambiguous

Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

More and Better Features Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word the: the → DT
  - Lowercased word Importantly: importantly → RB
  - Prefixes unfathomable: un- → JJ
  - Suffixes Importantly: -ly → RB
  - Capitalization Meridian: CAP → NNP
  - Word shapes 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
  - Maxent P(t|w): 93.7% overall / 82.6% unknown

Use of POS tags in downstream NLP tasks

- Features in text classifiers (e.g. spam / not spam)
- Noun-phrase chunking

United Nations
NNP NNP

POS taggers

- Stanford POS tagger
  - https://nlp.stanford.edu/software/tagger.shtml
- Natural Language Tool Kit (NLTK)
  - https://www.nltk.org
- Illinois POS tagger
  - http://cogcomp.org/page/software_view/POS

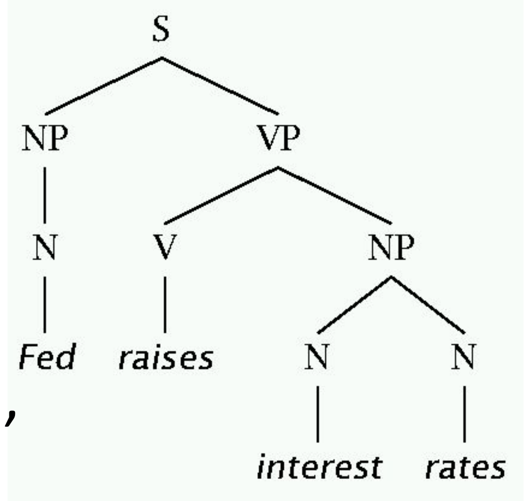
Syntactic parsing

Two views of linguistic structure

1. Constituency (phrase structure)
2. Dependency structure

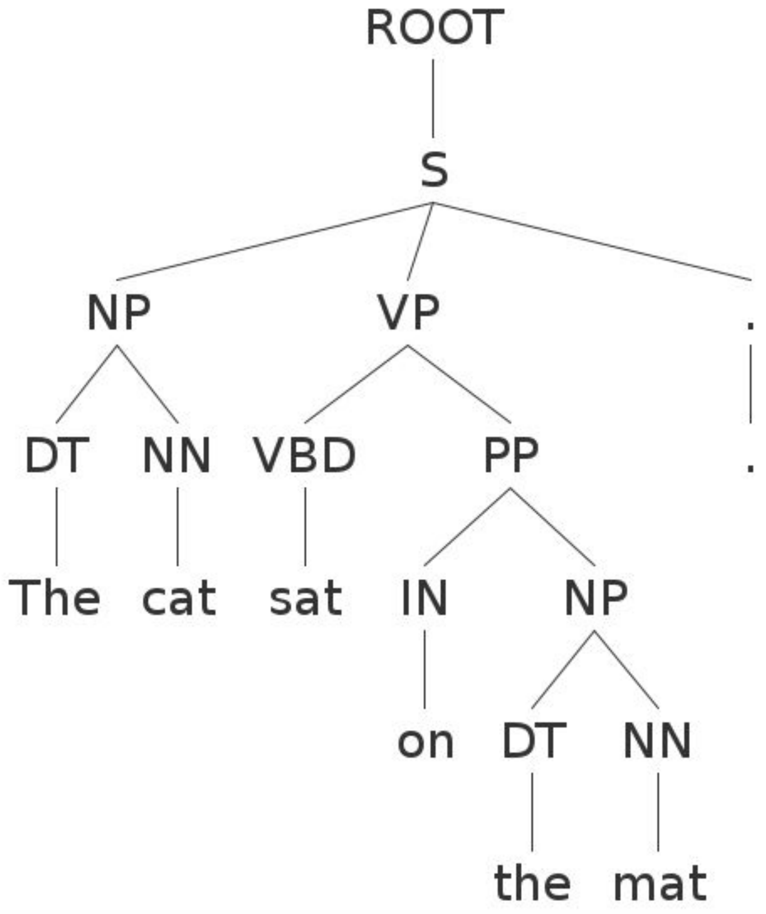
Two views of linguistic structure:
1. Constituency (phrase structure)

- Phrase structure organizes words into nested constituents.
- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
  - Distribution: a constituent behaves as a unit that can appear in different places:
    - John talked [to the children] [about drugs].
    - John talked [about drugs] [to the children].
    - *John talked drugs to the children about
  - Substitution/expansion/pro-forms:
    - I sat [on the box/right on top of the box/there].
  - Coordination, regular internal structure, no intrusion, fragments, semantics, …



Parse Trees

“The cat sat on the mat”



Parse Trees

In bracket notation:

```
(ROOT
  (S
    (NP (DT the) (NN cat))
    (VP (VBD sat)
      (PP (IN on)
        (NP (DT the) (NN mat))))))
```

Grammars

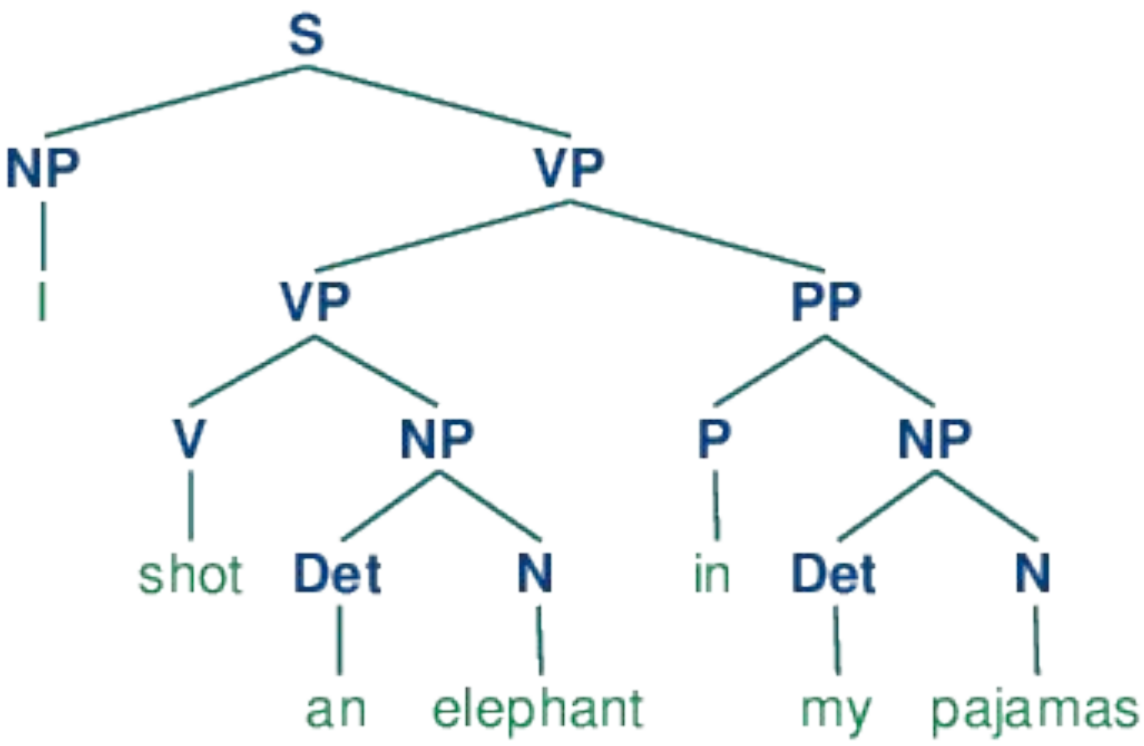
There are typically multiple ways to produce the same sentence.
Consider the statement by Groucho Marx:

“While I was in Africa, I shot an elephant in my pajamas”

“How he got into my pajamas, I don’t know”

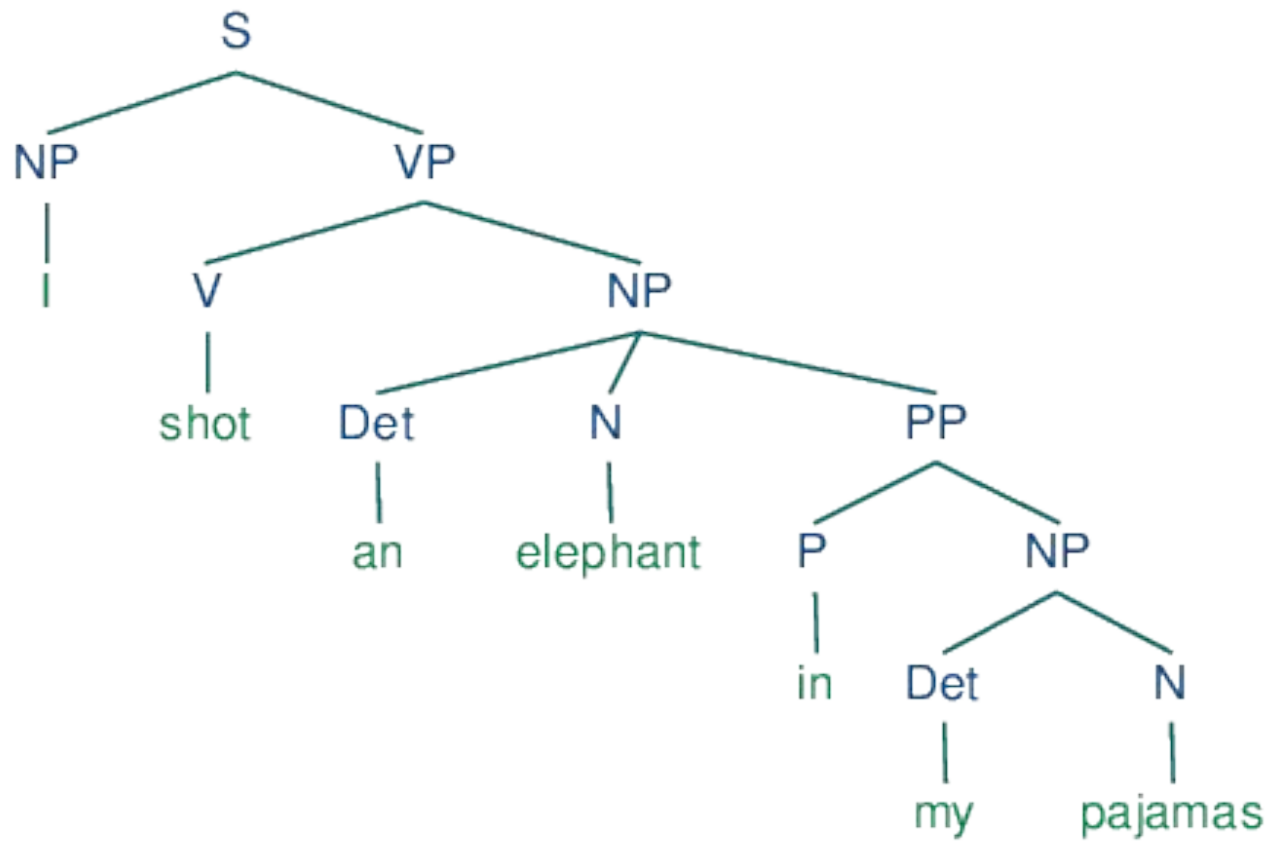
Parse Trees

"…,I shot an elephant in my pajamas" -what people hear first



Parse Trees

Groucho's version



Grammars

Its also possible to have “sentences” inside other sentences…

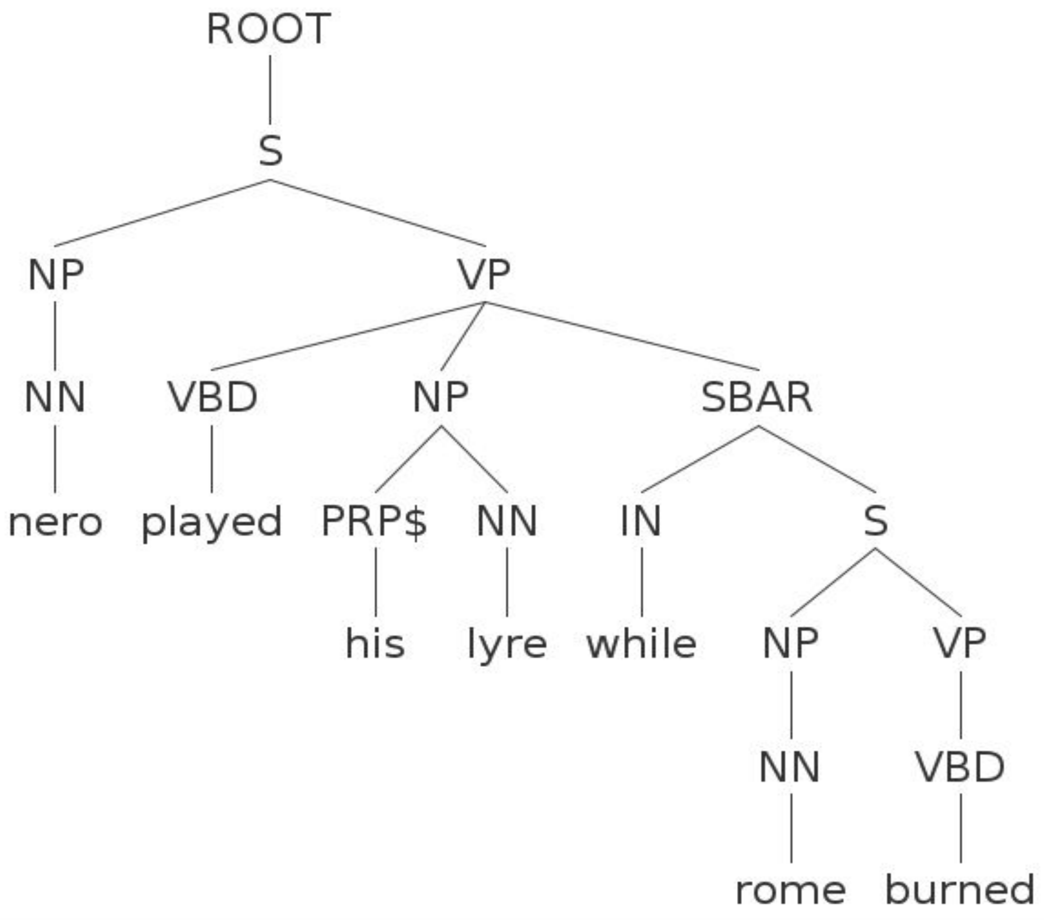
S → NP VP

VP → VB NP SBAR

SBAR → IN S

Recursion in Grammars

“Nero played his lyre while Rome burned”.



Headed phrase structure

- VP → … VB* …
- NP → … NN* …
- ADJP → … JJ* …
- ADVP → … RB* …

- SBAR(Q) → S|SINV|SQ → … NP VP …

- Plus minor phrase types:
  - QP (quantifier phrase in NP), CONJP (multi word constructions: *as well as*), INTJ (interjections), etc.

PCFGs

Complex sentences can be parsed in many ways, most of which make no sense or are extremely improbable (like Groucho's example).

Probabilistic Context-Free Grammars (PCFGs) associate and learn probabilities for each rule:

S → NP VP 0.8

S → NP VP PP 0.2

The parser then tries to find the most likely sequence of productions that generate the given sentence. This adds more realistic "world knowledge" and generally gives much better results.

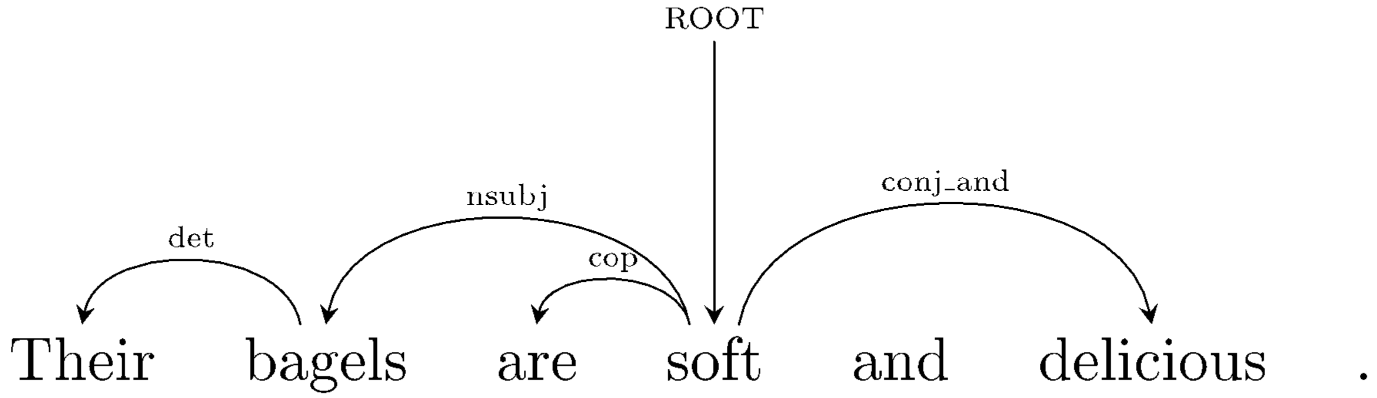
Most state-of-the-art parsers these days use PCFGs.

Systems

- **NLTK:** Python-based NLP system. Many modules, good visualization tools, but not quite state-of-the-art performance.
- **Stanford Parser:** Another comprehensive suite of tools (also POS tagger), and state-of-the-art accuracy. Has the definitive dependency module.
- **Berkeley Parser:** Slightly higher parsing accuracy (than Stanford) but not as many modules.
- Note: high-quality dependency parsing is usually very slow, but see: https://github.com/dlwh/puck

Two views of linguistic structure:
2. Dependency structure

Dependency structure shows which words depend on (modify or are arguments of) which other words.



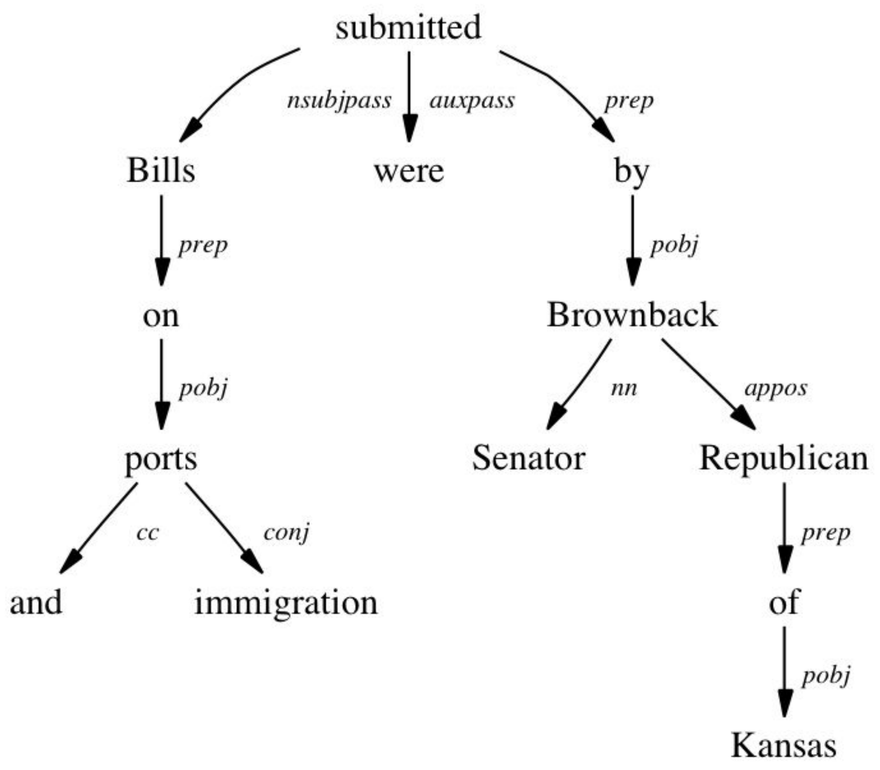
- “Their bagels are soft and delicious.”

    root (ROOT-0 , soft-4)
    nmod:poss (bagels-2 , Their-1)
    nsubj (soft-4 , bagels-2)
    nsubj (delicious-6 , bagels-2)
    cop (soft-4 , are-3)
    cc (soft-4 , and-5)
    conj:and (soft-4 , delicious-6)

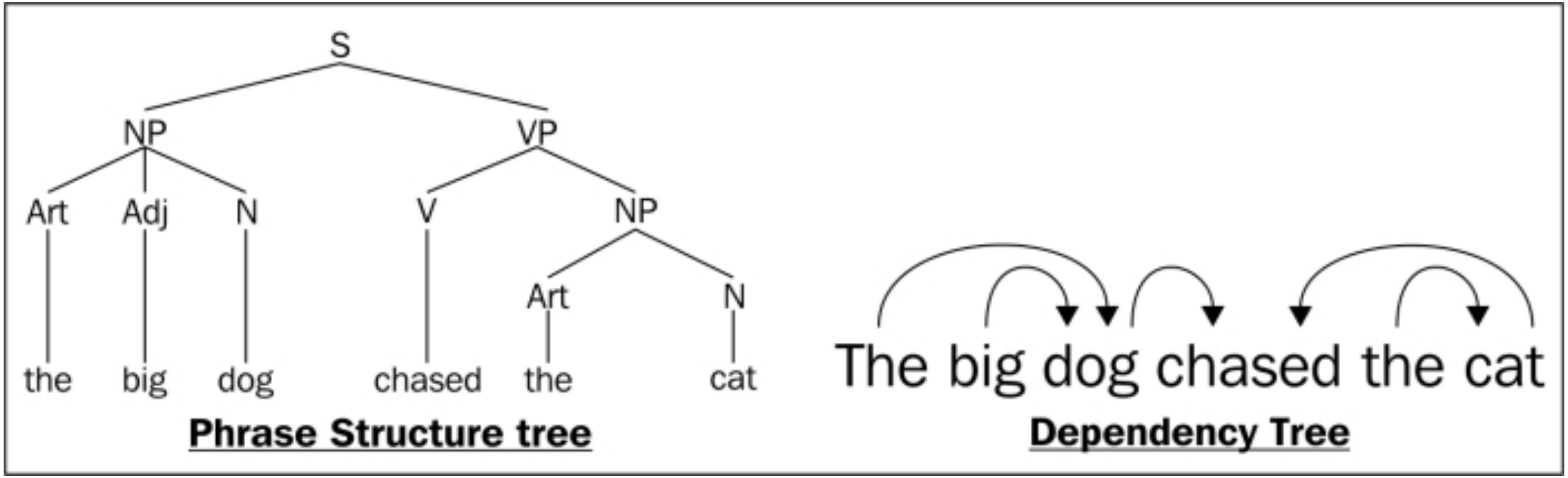
- Typed dependency (relationship) between two words: **head** and **modifier**. Also known as **governor** and **dependent.**

Dependency Parsing

Dependency parses may be non-binary, and structure type is encoded in links rather than nodes:



Comparison between constituent (phrase) and dependency trees



Dependency parser

- Stanford dependency parser

https://nlp.stanford.edu/software/stanford-dependencies.shtml

Statistical parsing applications

Statistical parsers are now robust and widely used in larger NLP applications:

- High precision question answering [Pasca and Harabagiu SIGIR 2001]
- Improving biological named entity finding [Finkel et al. JNLPBA 2004]
- Syntactically based sentence compression [Lin and Wilbur 2007]
- Extracting opinions about products [Bloom et al. NAACL 2007]
- Improved interaction in computer games [Gorniak and Roy 2005]
- Helping linguists find data [Resnik et al. BLS 2005]
- Source sentence analysis for machine translation [Xu et al. 2009]
- Relation extraction systems [Fundel et al. Bioinformatics 2006]

Credits

- Some slides have been adapted from:

https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html