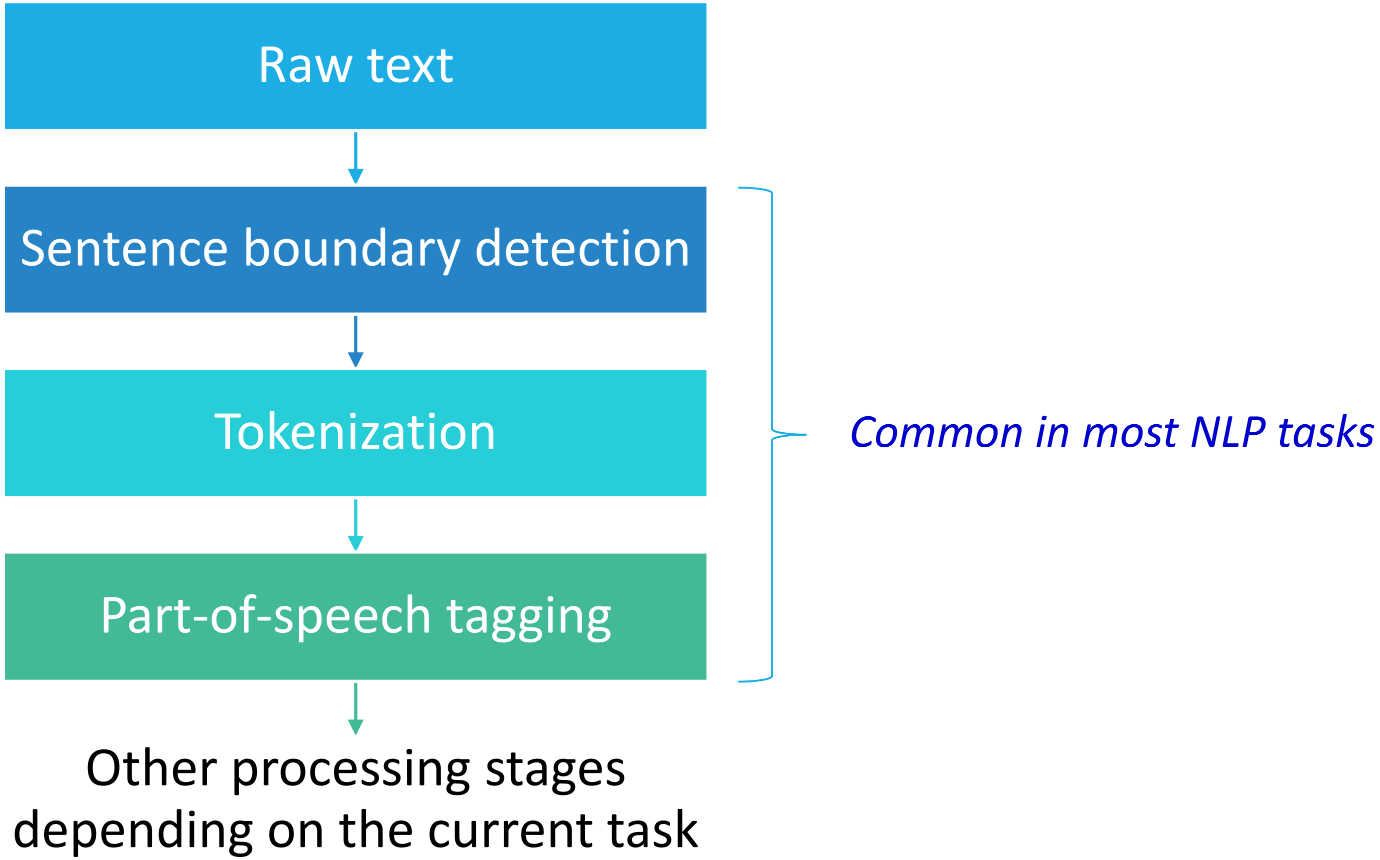
CSEN1076: NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

LECTURE 4 – PART-OF-SPEECH TAGGING

MERVAT ABUELKHEIR

NLP PIPELINE



Raw text

Sentence boundary detection

Tokenization

Part-of-speech tagging

Other processing stages depending on the current task

Common in most NLP tasks

PART-OF-SPEECH (POS) TAGGING

Also known as: lexical **categories**, word **classes**

POS tagging is a process by which **a single POS tag is assigned to each word** (and numerals/symbols/punctuations) in a text

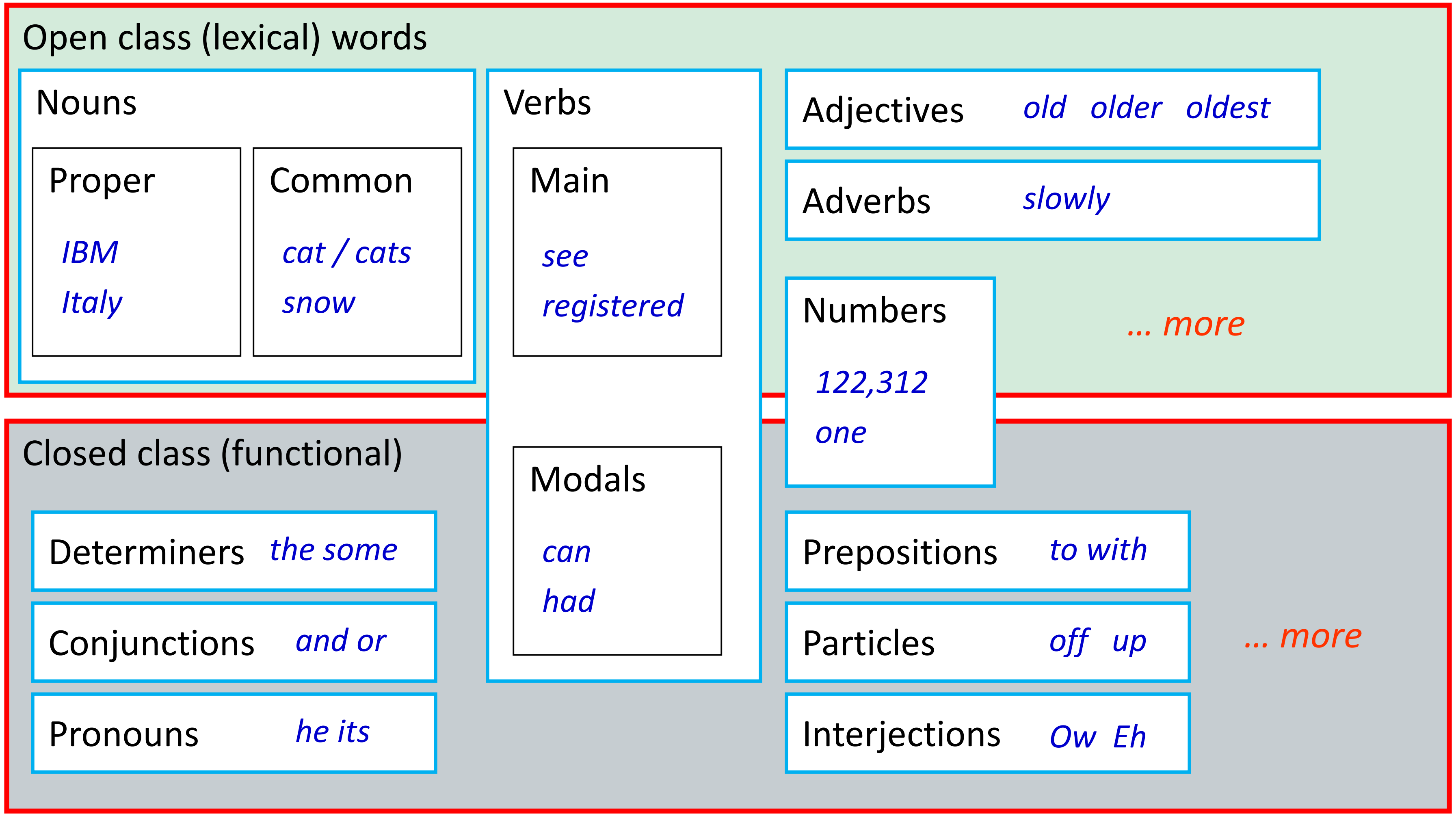
- NOUN, VERB, Determiner, Adjective, Adverb, Pronoun, Particle, Conjunction, etc.

POS tagging is concerned with how words fit together – “shallow” syntax

One of the earlier steps in the NLP pipeline, following tokenization

The	**happy**	**girl**	**eats**	**candy**	**.**
DET	ADJ	NOUN	VERB	NOUN	PUNC

OPEN VS. CLOSED POS CLASSES



OPEN VS. CLOSED CLASSES

Open vs. Closed classes

- **Closed:**
  - determiners: a, an, the
  - pronouns: she, he, I
  - prepositions: on, under, over, near, by, …
  - Why “closed”?
- **Open:**
  - Nouns, Verbs, Adjectives, Adverbs

POS TAGSETS

There are multiple POS tagsets in use

- Some are larger (fine grained), some are smaller (coarse grained)

The Brown Corpus tagset (78 tags)

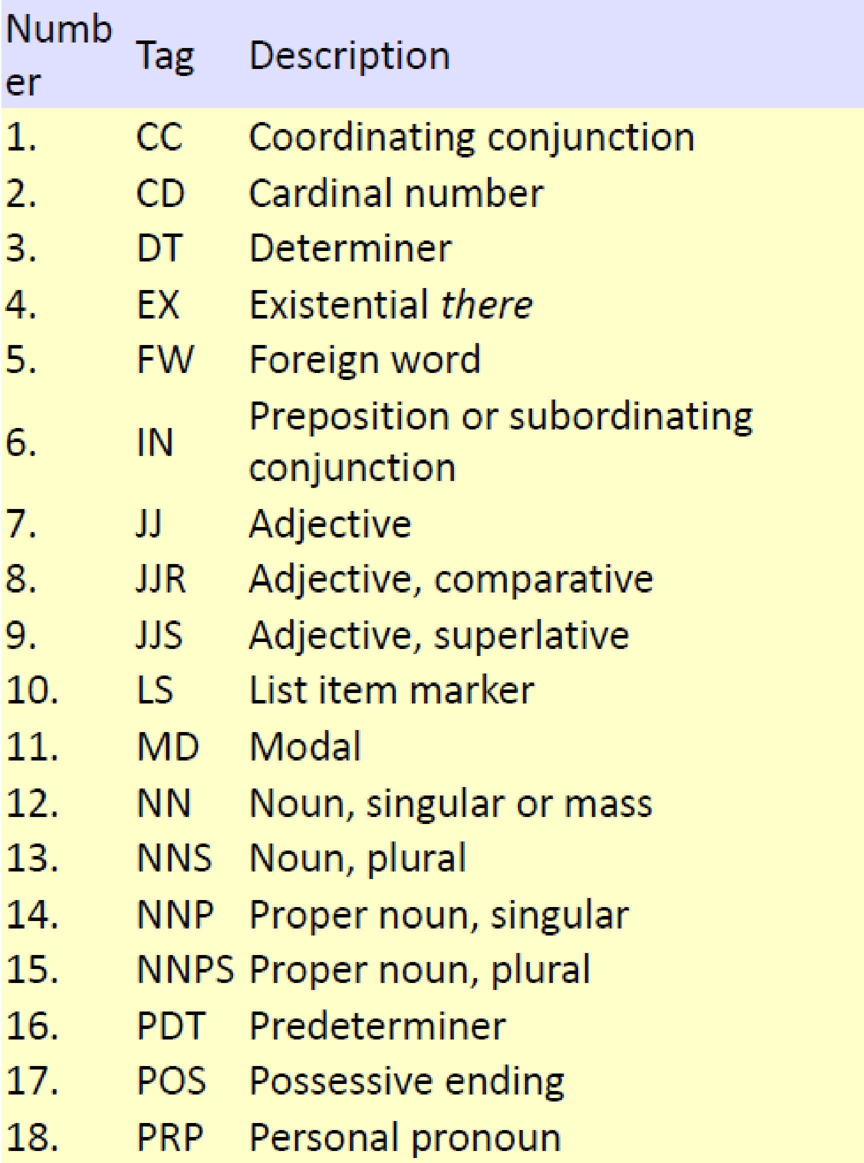
In NLP, the **Penn Treebank tagset** (36 tags) has become **de facto standard**

- Default tagset for nltk.pos_tag()

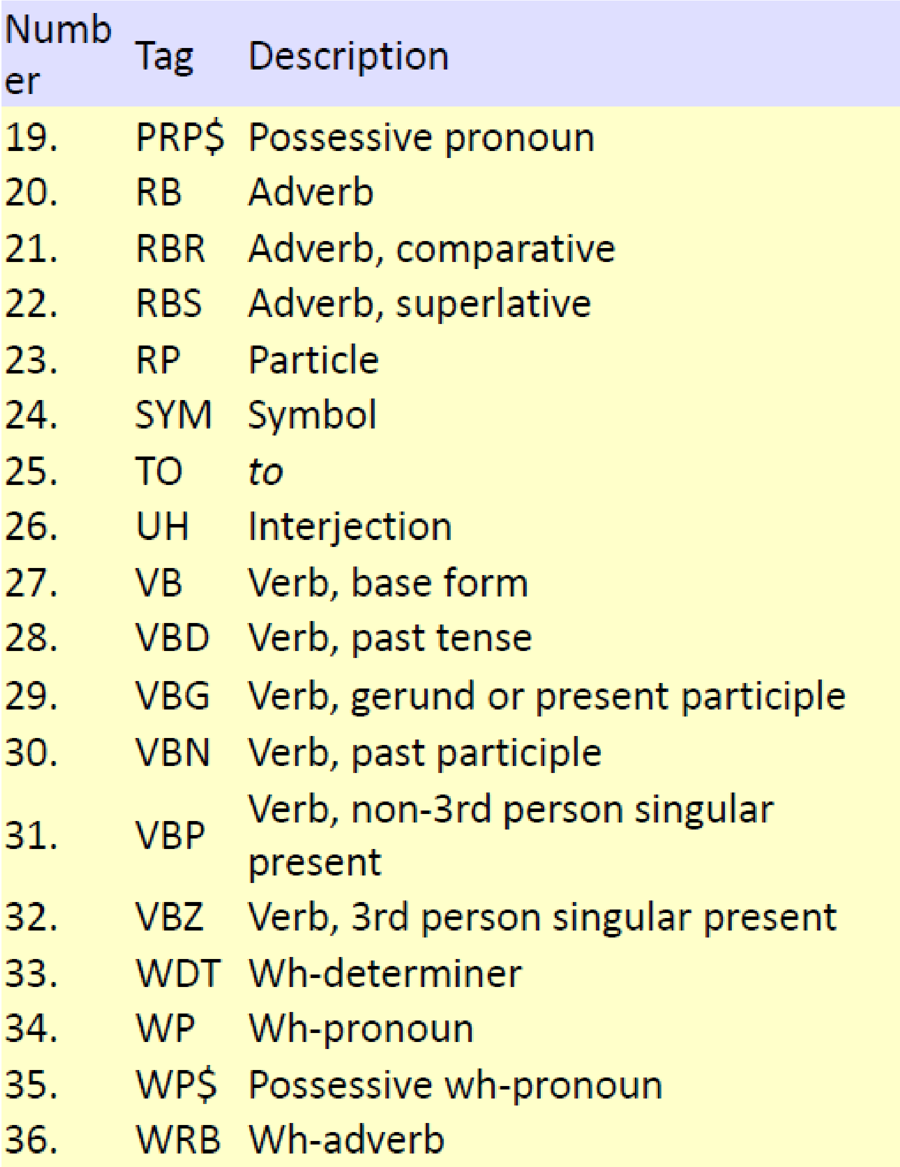
NLTK lets you load a POS-tagged corpus using "Universal" POS tagset (only 17 tags)

http://www.nltk.org/book/ch05.html#a-universal-part-of-speech-tagset

PENN TREEBANK TAGSET



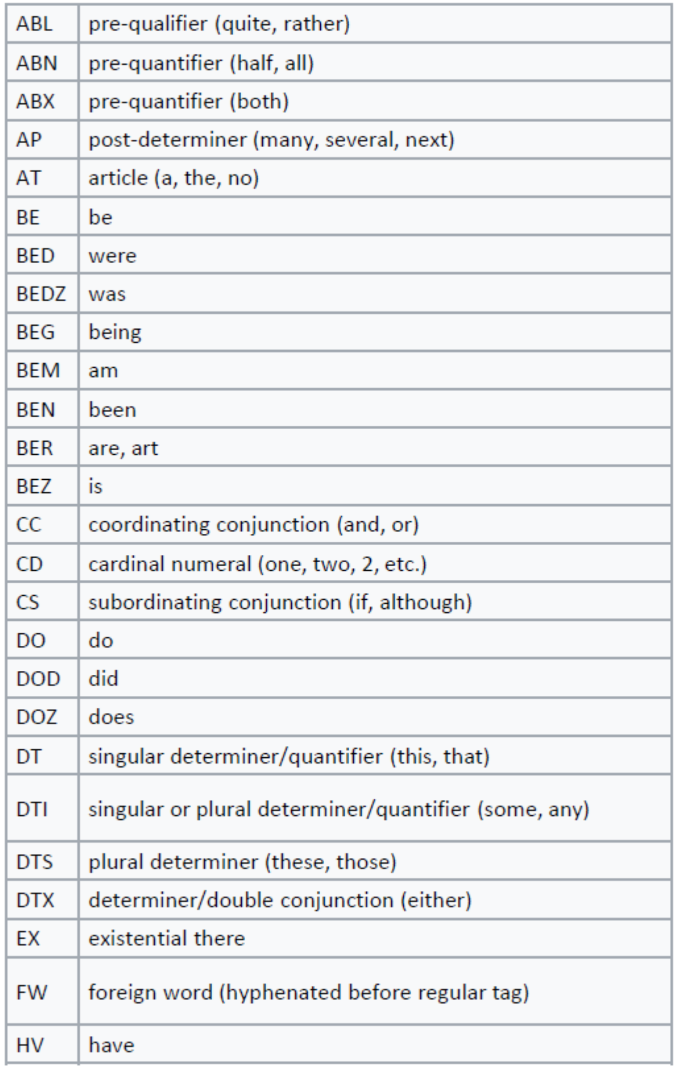
Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun



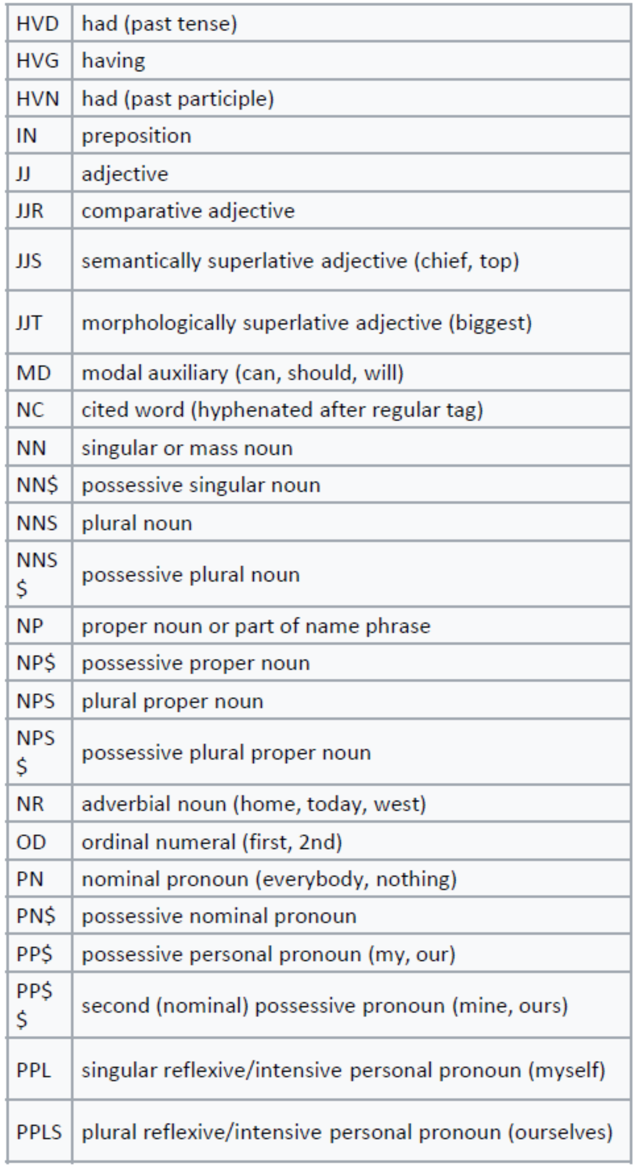
Number	Tag	Description
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

https://sites.google.com/site/partofspeechhelp/home

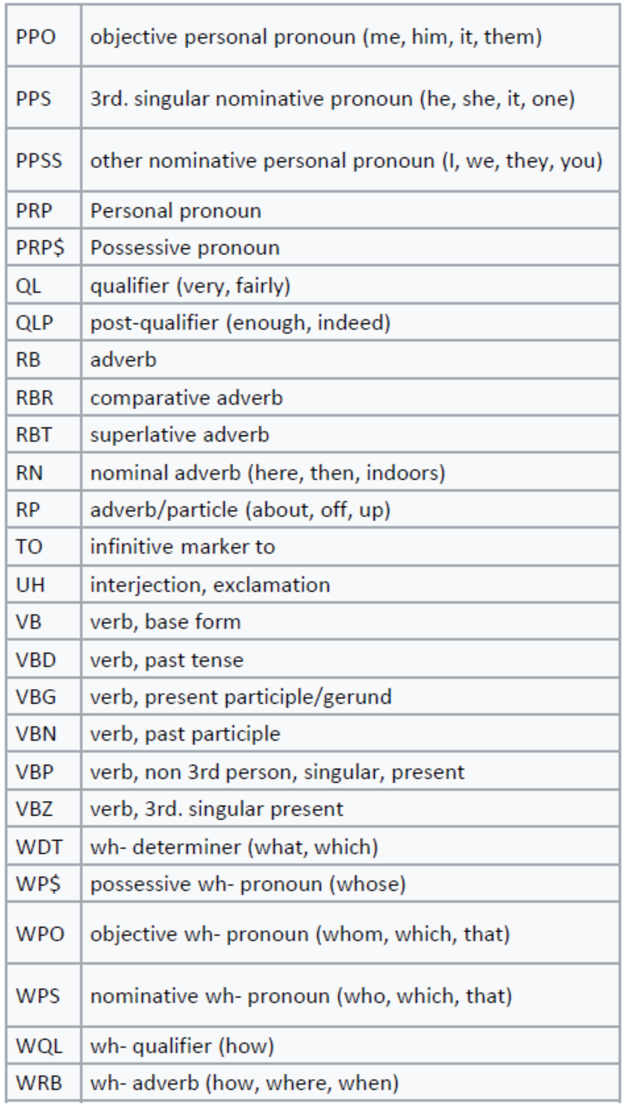
BROWN CORPUS TAGSET



ABL	pre-qualifier (quite, rather)
ABN	pre-quantifier (half, all)
ABX	pre-quantifier (both)
AP	post-determiner (many, several, next)
AT	article (a, the, no)
BE	be
BED	were
BEDZ	was
BEG	being
BEM	am
BEN	been
BER	are, art
BEZ	is
CC	coordinating conjunction (and, or)
CD	cardinal numeral (one, two, 2, etc.)
CS	subordinating conjunction (if, although)
DO	do
DOD	did
DOZ	does
DT	singular determiner/quantifier (this, that)
DTI	singular or plural determiner/quantifier (some, any)
DTS	plural determiner (these, those)
DTX	determiner/double conjunction (either)
EX	existential there
FW	foreign word (hyphenated before regular tag)
HV	have



HVD	had (past tense)
HVG	having
HVN	had (past participle)
IN	preposition
JJ	adjective
JJR	comparative adjective
JJS	semantically superlative adjective (chief, top)
JJT	morphologically superlative adjective (biggest)
MD	modal auxiliary (can, should, will)
NC	cited word (hyphenated after regular tag)
NN	singular or mass noun
NN$	possessive singular noun
NNS	plural noun
NNS$	possessive plural noun
NP	proper noun or part of name phrase
NP$	possessive proper noun
NPS	plural proper noun
NPS$	possessive plural proper noun
NR	adverbial noun (home, today, west)
OD	ordinal numeral (first, 2nd)
PN	nominal pronoun (everybody, nothing)
PN$	possessive nominal pronoun
PP$	possessive personal pronoun (my, our)
PP$$	second (nominal) possessive pronoun (mine, ours)
PPL	singular reflexive/intensive personal pronoun (myself)
PPLS	plural reflexive/intensive personal pronoun (ourselves)



PPO	objective personal pronoun (me, him, it, them)
PPS	3rd. singular nominative pronoun (he, she, it, one)
PPSS	other nominative personal pronoun (I, we, they, you)
PRP	Personal pronoun
PRP$	Possessive pronoun
QL	qualifier (very, fairly)
QLP	post-qualifier (enough, indeed)
RB	adverb
RBR	comparative adverb
RBT	superlative adverb
RN	nominal adverb (here, then, indoors)
RP	adverb/particle (about, off, up)
TO	infinitive marker to
UH	interjection, exclamation
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle/gerund
VBN	verb, past participle
VBP	verb, non 3rd person, singular, present
VBZ	verb, 3rd. singular present
WDT	wh- determiner (what, which)
WP$	possessive wh- pronoun (whose)
WPO	objective wh- pronoun (whom, which, that)
WPS	nominative wh- pronoun (who, which, that)
WQL	wh- qualifier (how)
WRB	wh- adverb (how, where, when)

UNIVERSAL POS TAGSET

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

More details can be found at:
[*http://universaldependencies.org/u/pos/*](http://universaldependencies.org/u/pos/)

ADJ: adjective (new, good, high)
ADV: adverb (really, already, still)
INTJ: interjection (oops, ouch)
NOUN: noun (year, home, costs)
PROPN: proper noun
VERB: verb (is, say, told, given, playing, would)
ADP: adposition (in, to, during, on, of)
AUX: auxiliary (has, is, will, was)
CCONJ: coordinating conjunction (and, or, but)
DET: determiner (the, a, an)
NUM: numeral (twenty-four, fourth, 1991)
PART: particle (‘s, not, at, on, out)
PRON: pronoun (he, their, her, its)
SCONJ: subordinating conjunction (if, while, that)
PUNCT: punctuation (. , ; !)
SYM: symbol
X: other (ersatz, esprit, dunno, gr8)

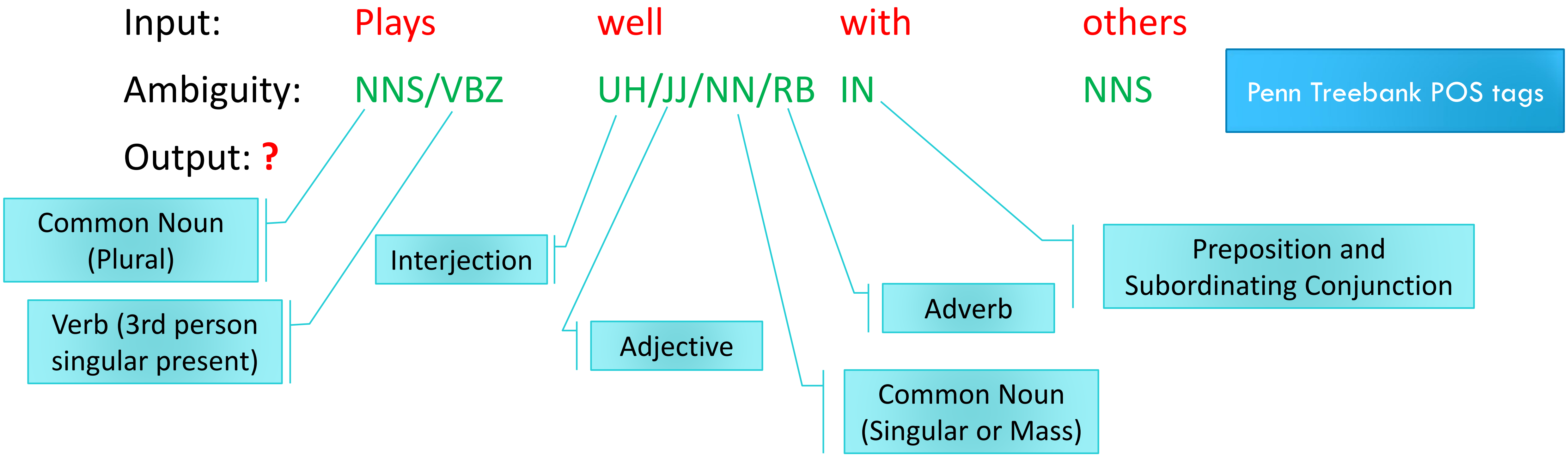
POS TAGGING

Words often have more than one POS tag: 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**

AMBIGUITY IN POS TAGGING



Input: Plays well with others

Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS

Output: ?

Penn Treebank POS tags

Common Noun (Plural)

Verb (3rd person singular present)

Interjection

Adjective

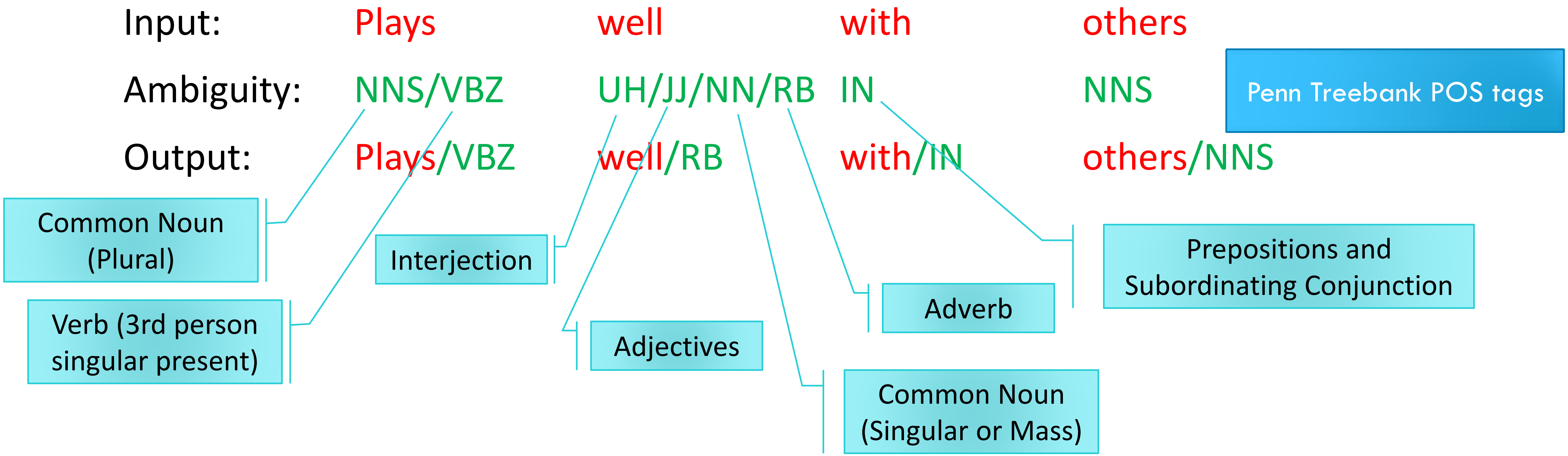
Common Noun (Singular or Mass)

Adverb

Preposition and Subordinating Conjunction

https://sites.google.com/site/partofspeechhelp/home

AMBIGUITY IN 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

Penn Treebank POS tags

Common Noun (Plural)

Verb (3rd person singular present)

Interjection

Adjectives

Common Noun (Singular or Mass)

Adverb

Prepositions and Subordinating Conjunction

NLTK CAN HELP

We don’t even remember what all the tags mean sometimes

How to get help on POS tags:

```
nltk.help.upenn_tagset(‘VBP’)
nltk.help.brown_tagset(‘JJ’)
```

- `nltk.help.upenn_tagset(‘RB’)`

```
RB: adverb
occasionally unabatingly maddeningly adventurously professedly stirringly
prominently technologically magisterially predominately swiftly fiscally
pitilessly ...
```

- `nltk.help.upenn_tagset(‘NN.*’)`

```
NN: noun, common, singular or mass
common-carrier cabbage knuckle-duster Casino afghan shed thermostat investment
slide humour falloff slick wind hyena machinist ...
NNP: noun, proper, singular
Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
```

NLTK BUILT-IN TAGGER

There is an **English POS tagger** that is built-in in NLTK

- It uses the Penn Treebank tagset by default

To use the built-in tagger:

```
nltk.pos_tag()  #Tag an individual tokenized sentence

nltk.pos_tag_sents()  #Tag a list of tokenized sentences
```

POS tagging accuracy is about 97%

However, we need to learn how to build our own baseline POS tagger

- We can easily reach 90% accuracy (baseline – most stupid method)
  1. Tag every word with its most frequent tag
  2. Tag unknown words as nouns

POS TAGGING IS DIFFICULT

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

- 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

They can fish.

SOURCES OF INFORMATION

What are the main sources of information for POS tagging?

- **Knowledge of neighboring words** (contextual information)

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

POS-TAGGED CORPORA

Manually POS-tagged large-scale corpora were instrumental in advancing statistical NLP technologies

The Brown Corpus

- 1 million words (1.16m after tokenization)

The Penn Treebank Corpus

- 1 million words (1st volume)
- A small portion (100K words) is included in NLTK data
- MANY subsequent volumes, in many different languages

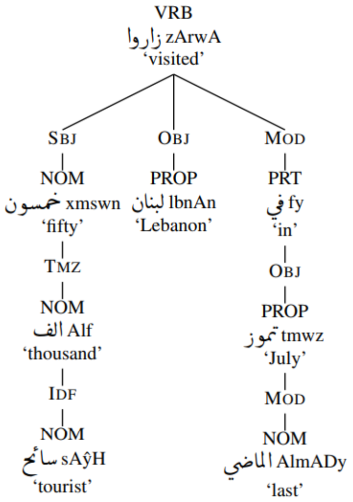


Figure 1: CATiB annotation for the sentence خمسون الف سائح زاروا لبنان في تموز الماضي

CATiB – Columbia Arabic Treebank

http://aclweb.org/anthology/P/P09/P09-2056.pdf

All student are required to download the two corpora on NLTK

FINDING THE TAGSET FOR A CORPUS IN NLTK

```
>>> from nltk.corpus import brown
>>> brown_news_tagged = brown.tagged_words(categories=‘news’, simplify_tags=True)
>>> tag_fd = nltk.FreqDist(tag for (word, tag) in brown_news_tagged)
>>> tag_fd.keys()
['N', 'P', 'DET', 'NP', 'V', 'ADJ', ',', '.', 'CNJ', 'PRO', 'ADV', 'VD', ...]
```

EXERCISE

Using NLTK, write a program that finds the most common 5 tags in the Brown Corpus

- Hint: you may use nltk.FreqDist() to get the frequency distribution of tags

AUTOMATIC TAGGING IN NLTK

- Default Tagger
- Unigram Tagger
- Bigram tagger
- Higher order n-gram tagger
- Regular Expression Tagger
- Combining taggers using backoff
- Classifier-based tagger

DEFAULT TAGGER

The “Default” tagger **tags each token with the most common tag**

You can easily find out that NOUN is the most common tag in the manually tagged corpora

Assume that around 13% of the words are NOUNs

Thus, if we create a default tagger that tags everything as NOUN, we can achieve a tagging accuracy of 13%

Disadvantage: Poor coverage

EVALUATING A DEFAULT TAGGER

Gold standard: Corpus that has been manually annotated – ***ground truth***

The gold standard corpus is used to evaluate the performance

```
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories=‘news’)
>>> brown_sents = brown.sents(categories=‘news’)
>>> default_tagger = nltk.DefaultTagger(‘NN’)
>>> default_tagger.tag(brown_sents)
>>> default_tagger.evaluate(brown_tagged_sents)
0.13089484257215028
```

STATISTICAL TAGGING - UNIGRAM TAGGER

Idea: **use the most frequent tag for every word**

- Assign the tag that is most likely for that particular token
- Train the tagger by specifying tagged sentence data as a parameter when we initialize the tagger
- Separate training and testing data
  - Usually have a training and test component; ideally a portion held out for final evaluation

Gold standard: Corpus that has been manually annotated

UNIGRAM TAGGER
1- DATA SET PARTITIONING

```
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories=‘news’)
>>> size = int(len(brown_tagged_sents) * 0.9)
>>> size
4160
>>> train_sents = brown_tagged_sents[:size]
>>> test_sents = brown_tagged_sents[size:]
>>> unigram_tagger = nltk.UnigramTagger(train_sents)
>>> unigram_tagger.evaluate(test_sents)
0.8120203329014252
```

UNIGRAM TAGGER
2- TAGGING

```
>>> brown_sents = brown.sents(categories =‘news’)
>>> unigram_tagger.tag(brown_sents[2007])
>>> unigram_tagger.evaluate(brown_tagged_sents)
```

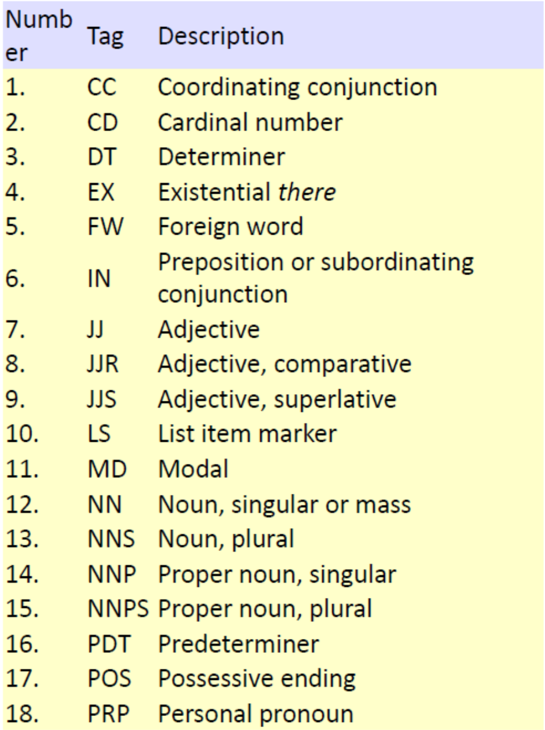
Try this code without training/testing separation and again with training/testing separation and output the accuracy level of the tagger

UNIGRAM CAN FAIL WITH AMBIGUOUS WORDS

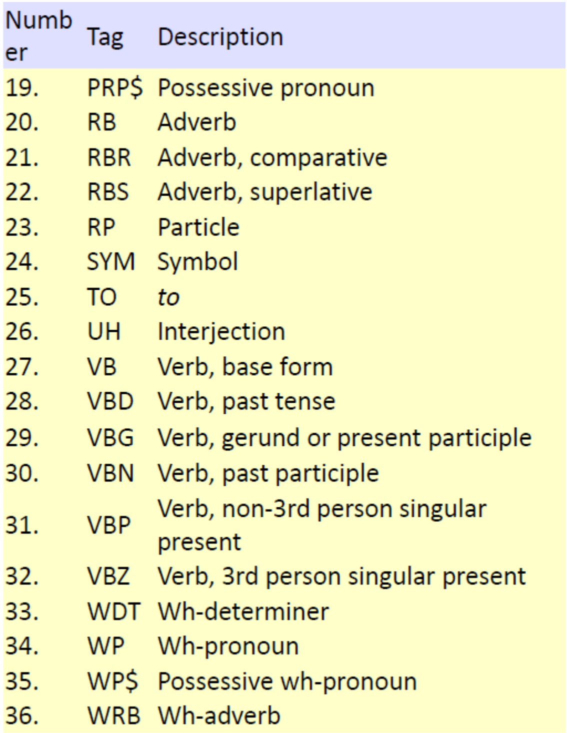
I have a question (should be NN)

I wanted to question him (Should be VB)

They can fish ➔ ? ? ?



Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun



Number	Tag	Description
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

STATISTICAL TAGGING - BIGRAM TAGGER

Bigram frequency can improve tagging accuracy by considering the POS tag of the preceding word when tagging the current word

Basic idea: **Choose the tag t_i for word w_i that maximizes the probability of t_i given the tag of the previous word t_{i-1} and w_i**

STATISTICAL TAGGING - BIGRAM TAGGER

```
>>> bigram_tagger = nltk.BigramTagger(train_sents)
>>> bigram_tagger.tag(brown_sents[2007])
[('Various', 'JJ'), ('of', 'IN'), ('the', 'AT'), ('apartments', 'NNS'), ('are', 'BER'),
('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'), ('type', 'NN'), (',', ','), ('being',
'BEG'), ('on', 'IN'), ('the', 'AT'), ('ground', 'NN'), ('floor', 'NN'), ('so', 'CS'),
('that', 'CS'), ('entrance', 'NN'), ('is', 'BEZ'), ('direct', 'JJ'), ('.', '.')]
>>> unseen_sent = brown_sents[4203]
>>> bigram_tagger.tag(unseen_sent)
[('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'), ('Congo', 'NP'),
('is', 'BEZ'), ('13.5', None), ('million', None), (',', None), ('divided', None),
('into', None), ('at', None), ('least', None), ('seven', None), ('major', None), ('``',
None), ('culture', None), ('clusters', None), ("''", None), ('and', None),
('innumerable', None), ('tribes', None), ('speaking', None), ('400', None), ('separate',
None), ('dialects', None), ('.', None)]
>>> bigram_tagger.evaluate(test_sents)
0.10276088906608193
```

STATISTICAL TAGGING - BIGRAM TAGGER

```
>>> bigram_tagger = nltk.BigramTagger(train_sents)
>>> bigram_tagger.tag(brown_sents[2007])
[('Various', 'JJ'), ('of', 'IN'), ('the', 'AT'), ('apartments', 'NNS'), ('are', 'BER'),
('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'), ('type', 'NN'), (',', ','), ('being',
'BEG'), ('on', 'IN'), ('the', 'AT'), ('ground', 'NN'), ('floor', 'NN'), ('so', 'CS'),
('that', 'CS'), ('entrance', 'NN'), ('is', 'BEZ'), ('direct', 'JJ'), ('.', '.')]
>>> unseen_sent = brown_sents[4203]
>>> bigram_tagger.tag(unseen_sent)
[('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'), ('Congo', 'NP'),
('is', 'BEZ'), ('13.5', None), ('million', None), (',', None), ('divided', None),
('into', None), ('at', None), ('least', None), ('seven', None), ('major', None), ('``',
None), ('culture', None), ('clusters', None), ("''", None), ('and', None),
('innumerable', None), ('tribes', None), ('speaking', None), ('400', None), ('separate',
None), ('dialects', None), ('.', None)]
```

```
>>> bigram_tagger.evaluate(test_sents)
0.10276088906608193
```

BIGRAM TAGGER CAN FAIL EVEN WITH PRIOR KNOWLEDGE

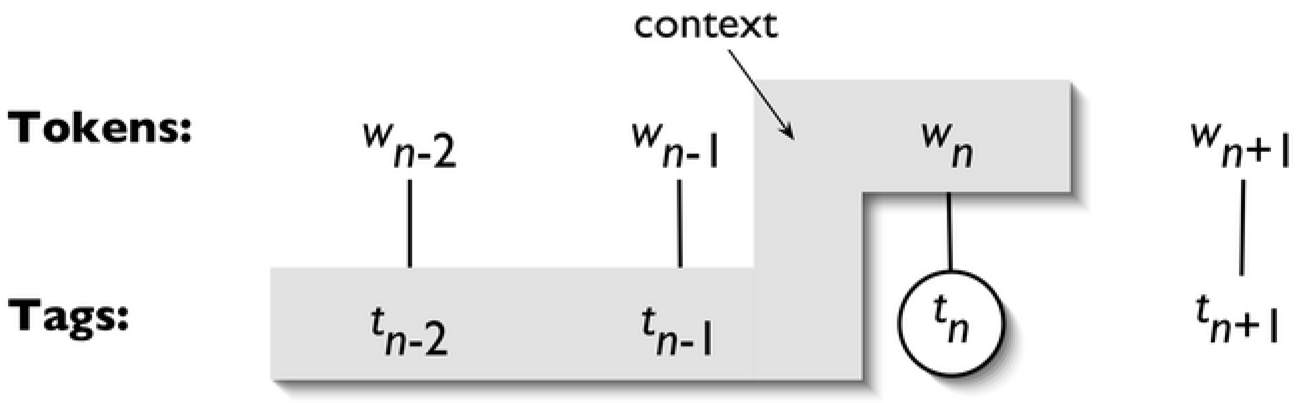
1. January was a *cold* month (JJ)
2. I had a *cold* (NN)

In the two above cases, the word *cold* was preceded with “a”. However, POS tag should be different

STATISTICAL TAGGING – TRIGRAM AND N-GRAMS TAGGERS

Context is the current word together with the POS tags of the $n-1$ preceding tokens

- $n=1$ ➔ unigram tagger
- $n=2$ ➔ bigram tagger
- $n=3$ ➔ trigram tagger
- $n>3$ ➔ n-gram tagger



n-gram taggers **should not consider context that crosses a sentence boundary**

Food for Thought: What will happen as we keep increasing n?

REGULAR EXPRESSIONS

A formal language for specifying text strings

How can we search for any of these?

- woodchuck
- woodchucks
- Woodchuck
- Woodchucks

REGULAR EXPRESSIONS: DISJUNCTIONS

(Any) 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]

- Caret 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 caret b*	Look up a^b now

REGULAR EXPRESSIONS: MORE DISJUNCTION

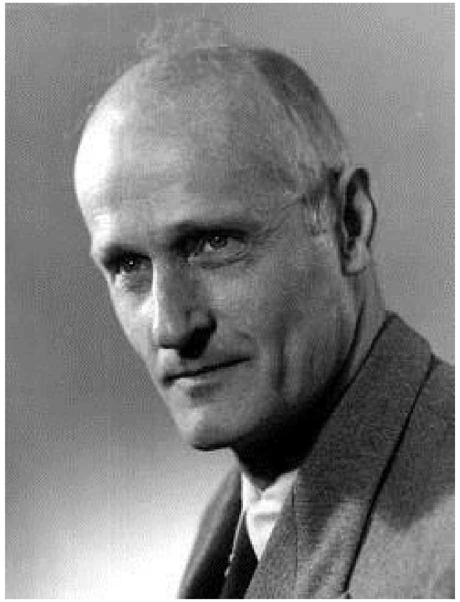
Woodchucks is another name for groundhog!

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	1 char in place	begin begun beg3n



Stephen C. Kleene

Leader in computer science and inventor of Regular Expression

Kleene *, Kleene +

REGULAR EXPRESSIONS: ANCHORS ^ $

^ beginning of line

$ end of line

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Z]	1 “Hello”
\.$	The end.
.$	The end? The end!

EXAMPLE

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

the >>> Misses capitalized examples (false negatives)

[tT]he >>> Incorrectly returns other or theology (false positives)

[^a-zA-Z][tT]he[^a-zA-Z] >>> is this a good regex for the search query?

(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$) >>> parentheses supersede other operators

MORE EXAMPLES

Can you guess what expressions will match the following regexes?

(Note: \b means word boundary)

- \b$[0-9]+(\.[0-9][0-9])?\b
- \b[0-9]+␣*(GHz|[Gg]igahertz)\b
- \b[0-9]+(\.[0-9]+)?␣*(GB|[Gg]igabytes?)\b

REGULAR EXPRESSION TAGGING

Consider morphology using human-defined patterns

- Ends in 'ly' → ADV
- Ends in 'ed' → VERB

```
>>> patterns = [
(r'.*ing$', 'VBG'), # gerunds
(r'.*ed$', 'VBD'), # simple past
(r'.*es$', 'VBZ'), # 3rd singular present
(r'.*ould$', 'MD'), # modals
(r'.*\'s$', 'NN$'), # possessive nouns
(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal num.
(r'^[A-Z][a-z]*s$', 'NPS'), # plural proper nouns
(r'^[A-Z][a-z]*[^s]$', 'NP'), # singular proper nouns
(r'.*s$', 'NNS'), # plural nouns
(r'.*', 'NN') ] # nouns (default)
>>> re_tagger = nltk.RegexpTagger(patterns)
>>> re_tagger.tag('Akbar and Jedis tweeted'.split())
[('Akbar', 'NP'), ('and', 'NN'), ('Jedis', 'NPS'), ('tweeted', 'VBD')]
```

The RE tagger can increase tagging accuracy to ~20%

Disadvantage of RE taggers:

Can be wrong: "fly" is not an adverb

Not every word has an identifiable morphological marker

EVALUATING RE TAGGER IN NLTK

```
>>> regexp_tagger = nltk.RegexpTagger(patterns)

>>> regexp_tagger.tag(brown_sents[3])

[(‘''’, ‘NN’), (‘Only’, ‘NN’), (‘a’, ‘NN’), (‘relative’, ‘NN’),
(‘handful’, ‘NN’), (‘of’, ‘NN’), (‘such’, ‘NN’), (‘reports’, ‘NNS’),
(‘was’,‘NNS’), (‘received’, ‘VBD’), (‘''’, ‘NN’), (‘,’, ‘NN’),
(‘the’, ‘NN’), (‘jury’, ‘NN’), (‘said’, ‘NN’), (‘,’, ‘NN’), (‘''’,
‘NN’), (‘considering’, ‘VBG’), (‘the’, ‘NN’), (‘widespread’, ‘NN’),
...]

>>> regexp_tagger.evaluate(brown_tagged_sents)

0.20326391789486245
```

COMBINING TAGGERS

We can combine different taggers

- If a given context is not found in a tri-gram tagger, we **back off** to the lower order model (bi-gram tagger)
  - The great depression → None → The great depression
- If context is still not found in the bigram tagger, we **back off** to unigram tagger
  - The great depression → None → The great depression
- If the word is not found, we **back off** to RE tagger
  - The great depression → None → `(r'.*sion$', ‘NN’), # noun`
- Finally, if the word doesn’t match any regex pattern, we back off to the default tagger

CLASSIFIER-BASED TAGGER (FEATURE-BASED)

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 classification) model to predict tag

- Maxent P(t|w): 93.7% overall / 82.6% unknown

POS TAGGING ACCURACIES

Rough accuracies:

Most freq tag:	~90% / ~50% (overall accuracy / unknown words)
Trigram HMM:	~95% / ~55%
TnT (HMM++):	96.2% / 86.0%
Maxent P(t|w):	93.7% / 82.6%
MEMM:	96.9% / 86.9%
Bidirectional dependencies:	97.2% / 90.0%
Upper bound:	~98% (human agreement)

EVALUATION

The **accuracy** metric is a pretty standard metric

However, there is another way to investigate the performance of a tagger
→ study tagger's mistakes

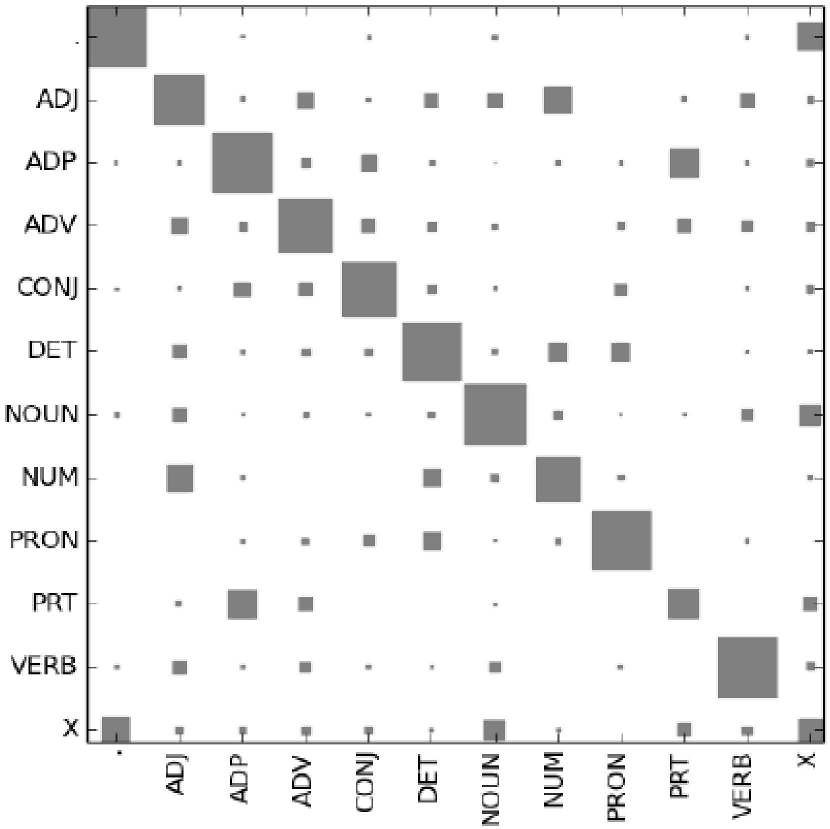
Some tags may be harder than others to assign, and it might be possible to treat them specially by pre- or post-processing the data

A convenient way to look at tagging errors is the confusion matrix

- charts expected tags (the gold standard) against actual tags generated by a tagger

CONFUSION MATRIX

Actual Tags



Predicted tags

CONFUSION MATRIX USING NLTK

```
>>> test_tags = [tag for sent in brown.sents(categories=‘editorial’)
                        for (word, tag) in yourtagger.tag(sent)]
>>> gold_tags = [tag for (word, tag) in brown.tagged_words(categories=‘editorial’)]
>>> print(nltk.ConfusionMatrix(gold_tags, test_tags))
```

NEXT TIME

Word Embeddings

REFERENCES

This lecture is heavily relying on the following courses:

- NLTK book
- Natural Language Processing Lecture Slides from the Stanford Coursera course by Dan Jurafsky and Christopher Manning