LG 467 Computers in Linguistics

[1-2021] Topic 5: POS tagging

Sakol Suethanapornkul

Previously...

We can count frequencies of individual items with `FreqDist`:

Code 6.2

```
import nltk

f = open('text1.txt')
txt = f.read()
f.close()

tokens = nltk.word_tokenize(txt)
freq   = nltk.FreqDist(tokens)

freq.most_common(15)
freq.plot()
freq.plot(cumulative = True)
```

Previously...

You can define your own functions to automate tasks. Functions can take multiple parameters plus a docstring:

Code 6.11

```
def ttr(lst, digits):
    """Compute type-token ratio on a list of strings,
    accepts one list and digits to round."""
    result = len(set(lst))/len(lst)
    return round(result, digits)

ttr(text1, 3)
ttr(text2, 4)
help(ttr)
```

Previously...

Language is not random. Things pattern together.

- We can ask what the next word might be given context
- Simple and efficient way of modeling context is using ***n-grams***
- N-grams are units of n sequences
  - Characters: 'custard'
    - *(c, u, s, t, a, r, d)* *(cu, us, st, ta, ar, rd)* *(cus, ust, sta, tar, ard)*
  - Words: 'He is eating fried rice'
    - *(he, is, eating, fried, rice)* *(he is, is eating, eating fried, fried rice)*

Previously...

N-grams to model language data

- "Turn in your ________" → *p(____ |"turn in your")*
  - decide on a useful context size (2, 3, 4, or ?)
  - save analyses not of individual words but of words given the previous n words

Previously...

Bigram counts from the Berkeley Restaurant Project:

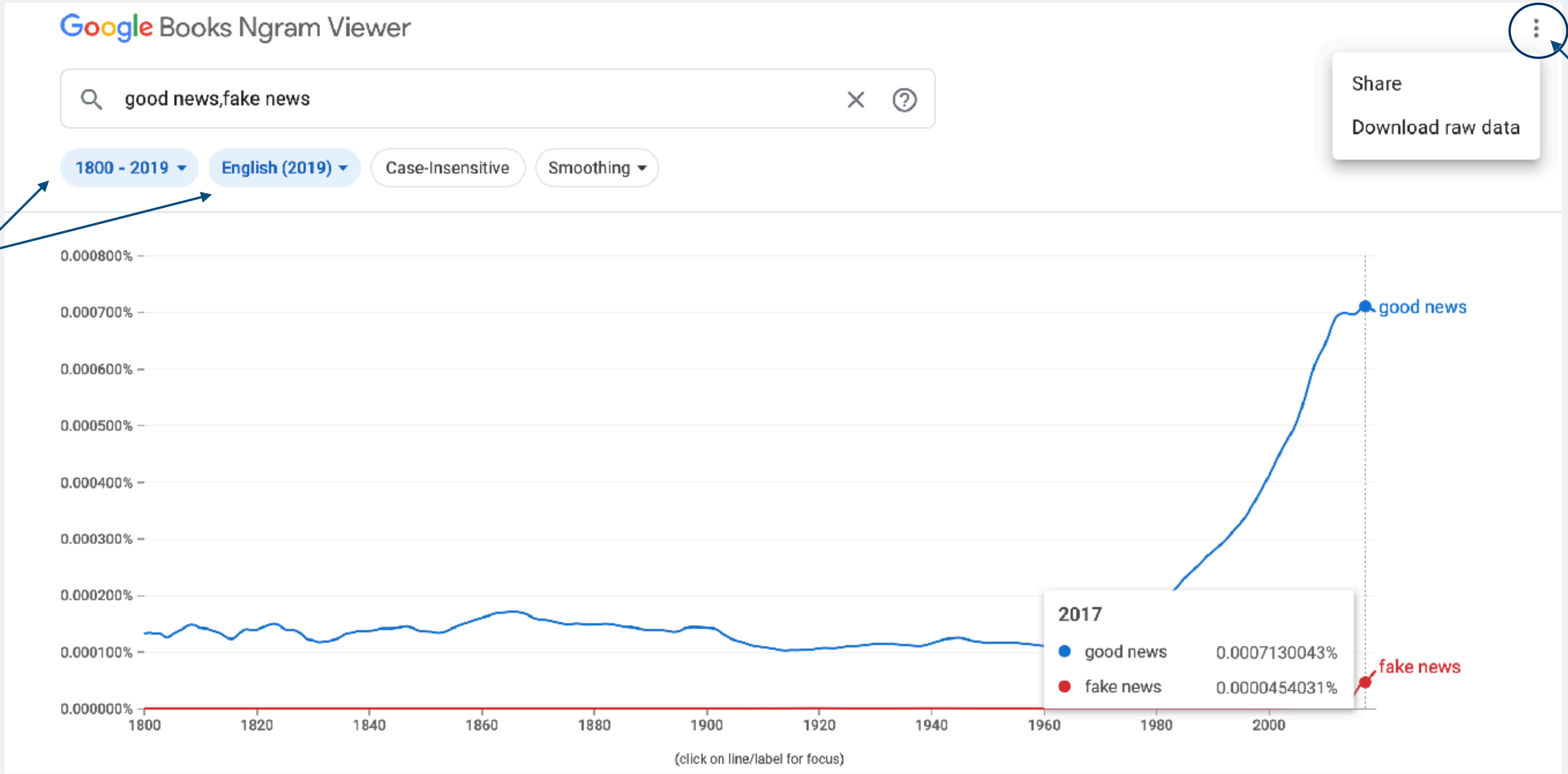
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Source: Figure 3.1 in Jurafsky & Martin [chapter 3]

Previously...

Check out: Google N-grams and Google Books ngram viewer

Toggle to change



Click on this

Previously...

We can generate bigram (or any n-gram) counts with NLTK:

Code 6.12

```
import nltk

nltk.bigrams(text1)

list(nltk.bigrams(text1))

nltk.ngrams(text1, 2)
list(nltk.ngrams(text1, 2))

bigrams = list(nltk.ngrams(text1, 2))
bicount = FreqDist(bigrams)
```

Beneath the surface

Our token n-gram models represent transitions between observed characters/words

- We can call them **Visible Markov Models (VMMs)**

For things that are overt, we are also interested in the probabilities of **hidden** categories

- We will need **Hidden Markov Models** (**HMMs**)

Beneath the surface

Did I hear hidden categories? What did you mean?

- We may not be interested in a phrase like 'awesome news'
- Instead, we might want to know the likelihood of adjectives followed by nouns (ADJ + NOUN)
- How can we look at categories that are not in the data explicitly?

Parts of speech (POS)

The idea that words can be classified into grammatical categories has been around for two millennia

- **part of speech**, word classes, **POS**, **POS tags**

8 parts of speech attributed to Dionysius Thrax of Alexandria (c. 1st C. BCE):

- noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- These categories are relevant for NLP today.

Adapted from: Jurafsky & Martin [chapter 8 PPT]

Parts of speech (POS)

Parts of speech are defined based on:

- grammatical relationship with their neighboring words
- Morphological properties about their affixes (-*ness*, -*able*, -*ed*)

Adapted from: Jurafsky & Martin [chapter 8 PPT]

Two classes of English words

1. Closed class words
   - Relatively fixed membership
   - Usually function words:
     - determiners: *this*, *that*, *a*, *an*, *the*
     - pronouns: *she*, *I*, *me*, *them*, *her*, *our*, *who*, *whose*
     - prepositions: *on*, *under*, *over*, *at*, *with*, …
     - conjunctions: *and*, *but*, *while*, *because*, *that*, …
     - particles: *go over*, *turn in*, …

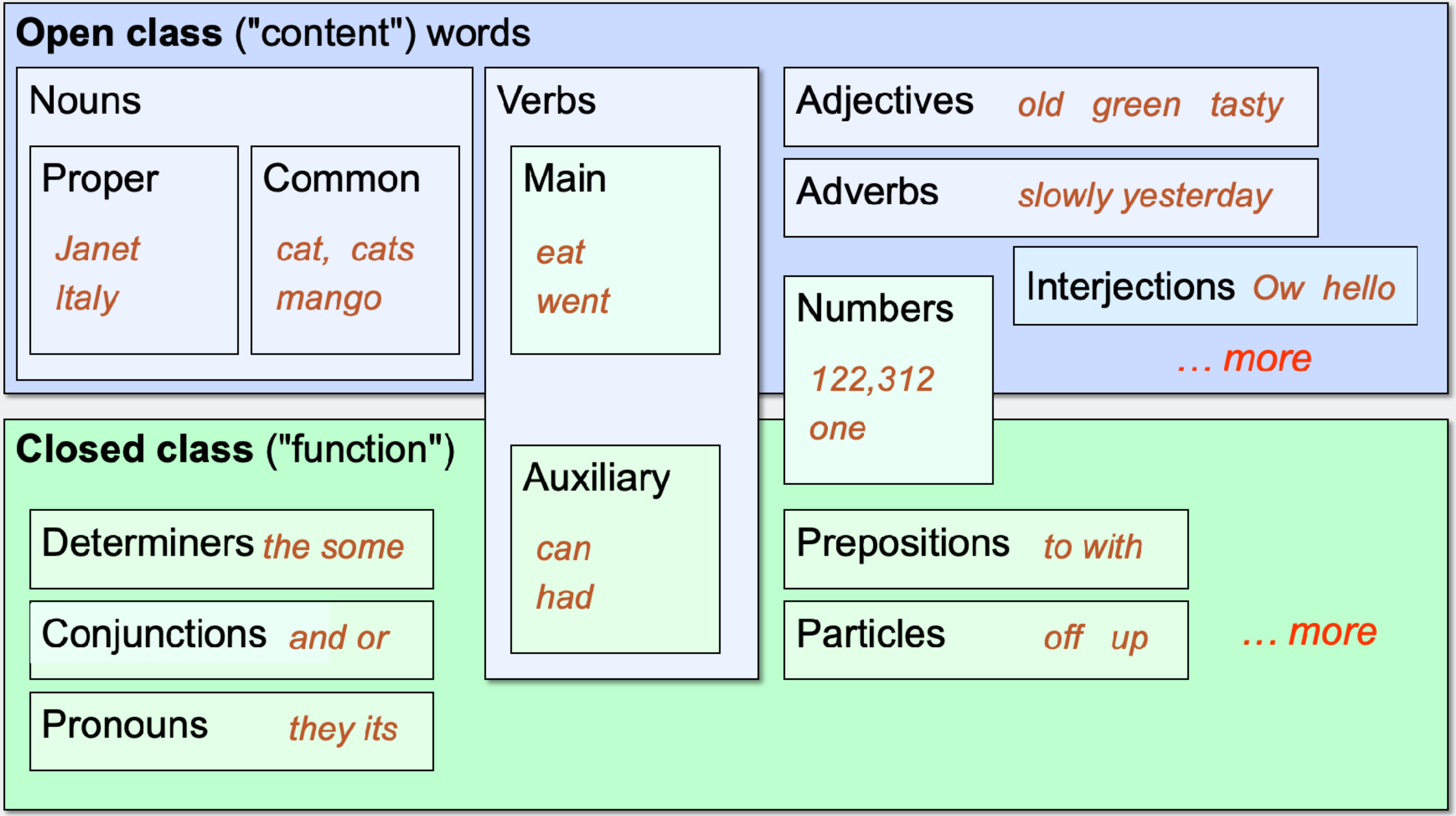
Adapted from: Jurafsky & Martin [chapter 8 PPT]

Two classes of English words

2. Open class words

- Usually content words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: *oh*, *ouch*, *uh-huh*, *yes*, *no*, *hello*
- New nouns and verbs like *iPhone*, *facebook*, *mansplain*, *google*

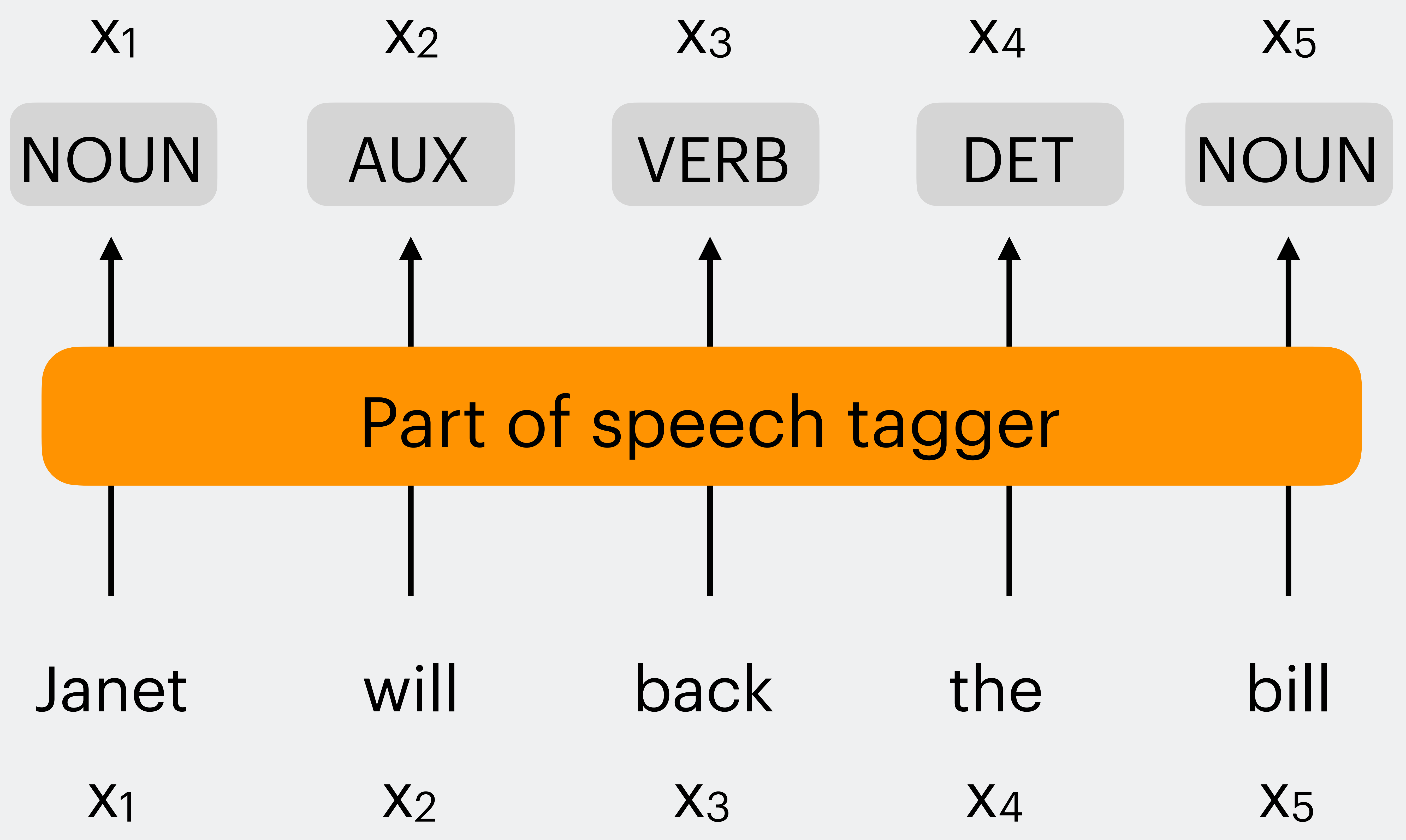
Adapted from: Jurafsky & Martin [chapter 8 PPT]



Adapted from: Jurafsky & Martin [chapter 8 PPT]

POS tagging

POS tagging = assigning a part of speech to each word in a text



Adapted from: Jurafsky & Martin [chapter 8 PPT]

POS tagging

Tagging is a disambiguation task

- Words often have more than one POS
- The goal is to find the correct tag for the situation

Case 1: 'book'

- *Book* a flight
- Whose *book* is it?

Case 2: 'tired'

- He *tired* me out
- I'm just *tired*

Case 3: 'that'

- Hand me *that* book
- I think *that* you liked it

Adapted from: Jurafsky & Martin's chapter 8 PPT

POS tagging

Though tagging is challenging, the accuracy of POS tagging algorithms is extremely high (97% or higher)

- 55-67% of word tokens in running text are ambiguous!

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	**(86%)**	45,799	**(85%)**
Ambiguous	(2+ tags)	7,025	**(14%)**	8,050	**(15%)**
Tokens:					
Unambiguous	(1 tag)	577,421	**(45%)**	384,349	**(33%)**
Ambiguous	(2+ tags)	711,780	**(55%)**	786,646	**(67%)**

Source: Figure 8.4 in Jurafsky & Martin [chapter 8]

Tagsets for English

Common in the US:

- Penn Treebank Tagset (PTB) 45 tags
- Brown Tagset 87 tags

Common in the UK:

- CLAWS 5 (or C5) 61 tags

CLAWS = Constituent Likelihood Automatic Word-tagging System (for BNC)

Penn Treebank Tags

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Source: Figure 8.2 in Jurafsky & Martin [chapter 8]

Practice

Try tagging the following sentences with the PTB tags:

1. The/ grand/ jury/ commented/ on/ a/ number/ of/ other/ topics/ ./
2. Although/ preliminary/ findings/ were/ reported/ more/ than/ a/ year/ ago/ ,/ the/ latest/ results/ appear/ in/ today/ 's/ New/ England/ Journal/ of/ Medicine/ ./

Practice

Try tagging the following sentences with the PTB tags:

1. The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN** a/**DT** number/**NN** of/**IN** other/**JJ** topics/**NNS** ./**.**

2. Although/**IN** preliminary/**JJ** findings/**NNS** were/**VBD** reported/**VBN** more/**RBR** than/**IN** a/**DT** year/**NN** ago/**IN** ,/**,** the/**DT** latest/**JJS** results/**NNS** appear/**VBP** in/**IN** today/**NN** 's/**POS** New/**NNP** England/**NNP** Journal/**NNP** of/**IN** Medicine/**NNP** ./**.**

Universal Dependencies Tagset

	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	PUNCT	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Source: Figure 8.1 in Jurafsky & Martin [chapter 8]

Universal Dependencies Tagset for Thai

Abbreviation	Part-of-Speech tag	Examples
ADJ	Adjective	ใหม่, พิเศษ , ก่อน, มาก, สูง
ADP	Adposition	แม้, ว่า, เมื่อ, ของ, สำหรับ
ADV	Adverb	ก่อน, ก็, เล็กน้อย, เลย, สุด
AUX	Auxiliary	เป็น, ใช่, คือ, คล้าย
CCONJ	Coordinating conjunction	แต่, และ, หรือ
DET	Determiner	ที่, นี้, ซึ่ง, ทั้ง, ทุก, หลาย
INTJ	Interjection	อุ้ย, โอ้ย
NOUN	Noun	กำมือ, พวก, สนาม, กีฬา, บัญชี
NUM	Numeral	5,000, 103.7, 2004, หนึ่ง, ร้อย
PART	Particle	มา ขึ้น ไม่ ได้ เข้า
PRON	Pronoun	เรา, เขา, ตัวเอง, ใคร, เธอ
PROPN	Proper noun	โอบามา, แคปิตอลฮิล, จีโอพี, ไมเคิล
PUNCT	Punctuation	(,), ", ', :
SCONJ	Subordinating conjunction	หาก
VERB	Verb	เปิด, ให้, ใช้, เผชิญ, อ่าน

- Universal Dependencies (UD) is one of the tagsets in PyThaiNLP
- In PyThaiNLP, this tagset is known as Parallel Universal Dependencies (PUD)

Source: PyThaiNLP

POS tagging in NLTK

An off-the-shelf tagger is available for English:

Code 7.1

```
from nltk import pos_tag, word_tokenize

text  = "John's big idea isn't all that bad."
token = word_tokenize(text)
pos   = pos_tag(token)

print(pos)
```

Question: What tagset is this?

POS tagging in NLTK

What can we do with pos? We can separate tags from tokens

Code 7.2

```
from nltk import FreqDist

tok = [tok for (tok, tag) in pos]
tag = [tag for (tok, tag) in pos]

# Then, you may choose to count
FreqDist(tok)
FreqDist(tag)
```

Our plan next week...

- Part-of-speech (POS) tagging
  - More tagging!
  - Please install SpaCy
    - Windows:
      - Use Anaconda Prompt to install (Here)
      - Follow the instructions on SpaCy (Here)
    - Mac:
      - Use Terminal