LING 388: Computers and Language

Lecture 22

Today's Topic

- A word about Term Projects
- `nltk.pos_tag(`*`words`*`)`

Term Project Proposal

Syllabus (Lecture 1)

- approx. 10-14 homeworks in total
  - **75%** of the grade
- Term project
  - e.g. build some cool application or do an experiment
  - slightly premature: I haven't introduced all the tools yet …
  - **25%** of the grade
- Procedure (if you're sure)
  - Send me a proposal, e.g. a paragraph or a page describing what you want to do. If I say okay, go ahead

nltk.pos_tag(*words*)

nltk.tag.pos_tag

nltk.tag.pos_tag(*tokens*, *tagset=None*, *lang='eng'*) [source]

Use NLTK's currently recommended part of speech tagger to tag the given list of tokens.

```
>>> from nltk.tag import pos_tag
>>> from nltk.tokenize import word_tokenize
>>> pos_tag(word_tokenize("John's big idea isn't all that bad."))
[('John', 'NNP'), ("'s", 'POS'), ('big', 'JJ'), ('idea', 'NN'), ('is', 'VBZ'),
("n't", 'RB'), ('all', 'PDT'), ('that', 'DT'), ('bad', 'JJ'), ('.', '.')]
>>> pos_tag(word_tokenize("John's big idea isn't all that bad."), tagset='universal')
[('John', 'NOUN'), ("'s", 'PRT'), ('big', 'ADJ'), ('idea', 'NOUN'), ('is', 'VERB'),
("n't", 'ADV'), ('all', 'DET'), ('that', 'DET'), ('bad', 'ADJ'), ('.', '.')]
```

NB. Use pos_tag_sents() for efficient tagging of more than one sentence.

Parameters

- **tokens** (*list(str)*) – Sequence of tokens to be tagged
- **tagset** (*str*) – the tagset to be used, e.g. universal, wsj, brown
- **lang** (*str*) – the ISO 639 code of the language, e.g. 'eng' for English, 'rus' for Russian

Tagset: Universal

Several tagged corpora support access to a simplified, universal tagset, e.g. where all nouns tags are collapsed to a single category `NOUN`:

```
>>> print(brown.tagged_sents(tagset='universal'))
[[('The', 'DET'), ('Fulton', 'NOUN'), ('County', 'NOUN'), ('Grand', 'ADJ'), ('Jury', 'NO
 [('The', 'DET'), ('jury', 'NOUN'), ('further', 'ADV'), ('said', 'VERB'), ('in', 'ADP'),
>>> from nltk.corpus import conll2000, switchboard
>>> print(conll2000.tagged_words(tagset='universal'))
[('Confidence', 'NOUN'), ('in', 'ADP'), ...]
```

Brown and PTB Mappings:

- https://github.com/slavpetrov/universal-pos-tags/blob/fca8727e9424255f0732d1bc437f432f45a0c166/en-brown.map
- https://github.com/slavpetrov/universal-pos-tags/blob/c8e49bf1654d337d55553fabd75b4073596feac7/en-ptb.map

```
10  Interface for converting POS tags from various treebanks
11  to the universal tagset of Petrov, Das, & McDonald.
12
13  The tagset consists of the following 12 coarse tags:
14
15  VERB - verbs (all tenses and modes)
16  NOUN - nouns (common and proper)
17  PRON - pronouns
18  ADJ - adjectives
19  ADV - adverbs
20  ADP - adpositions (prepositions and postpositions)
21  CONJ - conjunctions
22  DET - determiners
23  NUM - cardinal numbers
24  PRT - particles or other function words
25  X - other: foreign words, typos, abbreviations
26  . - punctuation
```

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*

Figure 8.2 Penn Treebank part-of-speech tags.

J&M 3rd ed. draft

Penn Part-of-Speech (POS) Tagset

Tagset: Penn to Universal mapping

https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

!	.	CC	CONJ	MD	VERB	VB	VERB	PRT	PRT
#	.	CD	NUM	NN	NOUN	VBD	VERB	RB	ADV
$	.	CD|RB	X	NNP	NOUN	VBD|VBN		RBR	ADV
"	.	DT	DET	NNPS	NOUN		VERB	RBS	ADV
(	.	EX	DET	NNS	NOUN	VBG	VERB	RB|RP	ADV
)	.	FW	X	NN|NNS	NOUN	VBG|NN	VERB	RB|VBG	ADV
,	.	IN	ADP	NN|SYM	NOUN	VBN	VERB	RN	X
-LRB-	.	IN|RP	ADP	NN|VBG	NOUN	VBP	VERB	RP	PRT
-RRB-	.	JJ	ADJ	NP	NOUN	VBP|TO	VERB	SYM	X
.	.	JJR	ADJ	PDT	DET	VBZ	VERB	TO	PRT
:	.	JJRJR	ADJ	POS	PRT	VP	VERB	UH	X
?	.	JJS	ADJ	PRP	PRON			WDT	DET
LS	X	JJ|RB	ADJ	PRP$	PRON			WH	X
``	.	JJ|VBG	ADJ	PRP|VBP	PRON			WP	PRON
								WP$	PRON
								WRB	ADV

nltk.pos_tag(*words*)

- Example:

```
>>> import nltk
>>> raw1 = open('ot.txt').read() % Oliver Twist
>>> words1 = nltk.word_tokenize(raw1)
>>> len(words1)
199779
>>> twl = nltk.pos_tag(words1)
>>> twl[-10:]
[('a', 'DT'), ('Church', 'NNP'), (',', ','), ('and', 'CC'),
('she', 'PRP'), ('was', 'VBD'), ('weak', 'JJ'), ('and', 'CC'),
('erring', 'VBG'), ('.', '.')]
```

`nltk.pos_tag(`*`words`*`)`

- How to extract the adjectives and look at the frequency distribution?
  - use a *conditional* list comprehension over the `nltk.pos_tag(words1)` list.
  - What is the condition?
    - Each tuple has form `(word, tag)`.
    - Tuples can be indexed: e.g. `[0]` for `word` or `[1]` for `tag`.
    - Relevant tag is `'JJ'`.
    - Condition is `tuple[1] == 'JJ'`.
  - `[tuple[0] for tuple in nltk.pos_tag(words1) if tuple[1] == 'JJ']`
  - `[word for (word,tag) in nltk.pos_tag(words1) if tag == 'JJ']`

nltk.pos_tag(*words*)

- Example:

```
>>> adjs = [tuple[0] for tuple in nltk.pos_tag(words1) if
tuple[1] == 'JJ']
>>> len(adjs)
12439
>>> fd = nltk.FreqDist(adjs)
>>> fd
FreqDist({'’': 514, '“': 457, 'old': 434, '”': 422, 'young':
278, 'little': 262, 'other': 236, 'great': 229, 's': 229,
'good': 189, ...})
```

nltk.pos_tag(*words*)

- Recall punctuation removal from Homework 7 Review?

```
>>> def isword(x):
...     return any(c.isalpha() for c in x)
...
```

- Get the adjectives (without punctuation):

```
>>> adjs = [word for (word,tag) in nltk.pos_tag(words1) if tag == 'JJ' and isword(word)]
>>> len(adjs)
11035
fd = nltk.FreqDist(adjs)
>>> fd
FreqDist({'old': 434, 'young': 278, 'little': 262, 'other': 236, 'great': 229, 's': 229, 'good': 189, 'same': 147, 'such': 138, 'll': 136, ...})
>>> fd.most_common(20)
[('old', 434), ('young', 278), ('little', 262), ('other', 236), ('great', 229), ('s', 229), ('good', 189), ('same', 147), ('such', 138), ('ll', 136), ('dear', 135), ('own', 131), ('many', 130), ('much', 127), ('first', 118), ('few', 110), ('last', 90), ('long', 87), ('small', 79), ('poor', 75)]
```

Worked Example

- Let's compare the adjectives used by Charles Dickens in *Oliver Twist* and *Nicholas Nickleby*.

```
>>> raw2 = open('nn.txt').read()
>>> words2 = nltk.word_tokenize(raw2)
>>> adjs2 = [word for (word,tag) in nltk.pos_tag(words2) if tag ==
'JJ' and isword(word)]
>>> len(words2)
396970
>>> len(adjs2)
22636
>>> fd2 = nltk.FreqDist(adjs2)
>>> fd2.most_common(20)
[('little', 713), ('old', 541), ('other', 539), ('young', 525),
('great', 521), ('such', 508), ('good', 382), ('dear', 329),
('own', 327), ('same', 320), ('much', 319), ('many', 318), ('last',
312), ('first', 253), ('poor', 218), ('long', 180), ('few', 162),
('short', 155), ('sure', 150), ('new', 141)]
```

Worked Example

- How to plot the histograms?
  - y-axis: should be relative frequency, not raw counts
    - # of adjectives: ot: 11035, nn: 22636
  - x-axis: should be the adjectives

```
>>> awords2 = [tuple[0] for tuple in fd2.most_common(20)]
>>> awords = [tuple[0] for tuple in fd.most_common(20)]
>>> awords
['old', 'young', 'little', 'other', 'great', 's', 'good', 'same',
'such', 'll', 'dear', 'own', 'many', 'much', 'first', 'few', 'last',
'long', 'small', 'poor']
>>> awords2
['little', 'old', 'other', 'young', 'great', 'such', 'good', 'dear',
'own', 'same', 'much', 'many', 'last', 'first', 'poor', 'long', 'few',
'short', 'sure', 'new']
>>> x = list(set(awords).union(set(awords2)))
['own', 'great', 'll', 'little', 'much', 'dear', 'sure', 'short',
'many', 'first', 'young', 's', 'poor', 'last', 'few', 'small', 'new',
'other', 'same', 'good','such', 'long', 'old']
```

Worked Example

Histograms:

- when counts are known, use `plt.bar(x-values, heights)`
- `plt.hist(words, bins)` counts # occurrences of words for you

matplotlib.pyplot.bar

`matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)` [source]

Make a bar plot.

The bars are positioned at *x* with the given *align*ment. Their dimensions are given by *height* and *width*. The vertical baseline is *bottom* (default 0).

Many parameters can take either a single value applying to all bars or a sequence of values, one for each bar.

Parameters:

x : ***float or array-like***

The x coordinates of the bars. See also *align* for the alignment of the bars to the coordinates.

height : ***float or array-like***

The height(s) of the bars.
Note that if *bottom* has units (e.g. datetime), *height* should be in units that are a difference from the value of *bottom* (e.g. timedelta).

Worked Example

- Histograms:

```
>>> import matplotlib.pyplot as plt
>>> plt.bar(range(len(x)),[fd[word]/len(fd) for word in
x],fill=False,edgecolor='b',label='Oliver Twist')
>>> plt.bar(range(len(x)),[fd2[word]/len(fd2) for word in
x],fill=False,edgecolor='g',label='Nicholas Nickleby')
>>> plt.xticks(range(len(x)),x,rotation=90)
>>> plt.legend()
>>> plt.show()
```

`[fd[word]/len(fd) for word in x]`
`fd[`*word*`]` = count for *word*
`len(`*fd*`)` = total count in `FreqDist()`
`fd[`*word*`]/len(`*fd*`)` = proportion of total count for *word*

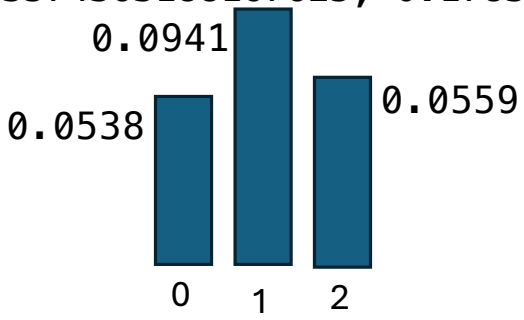
Worked Example

- We supply `plt.bar()` with two arguments:

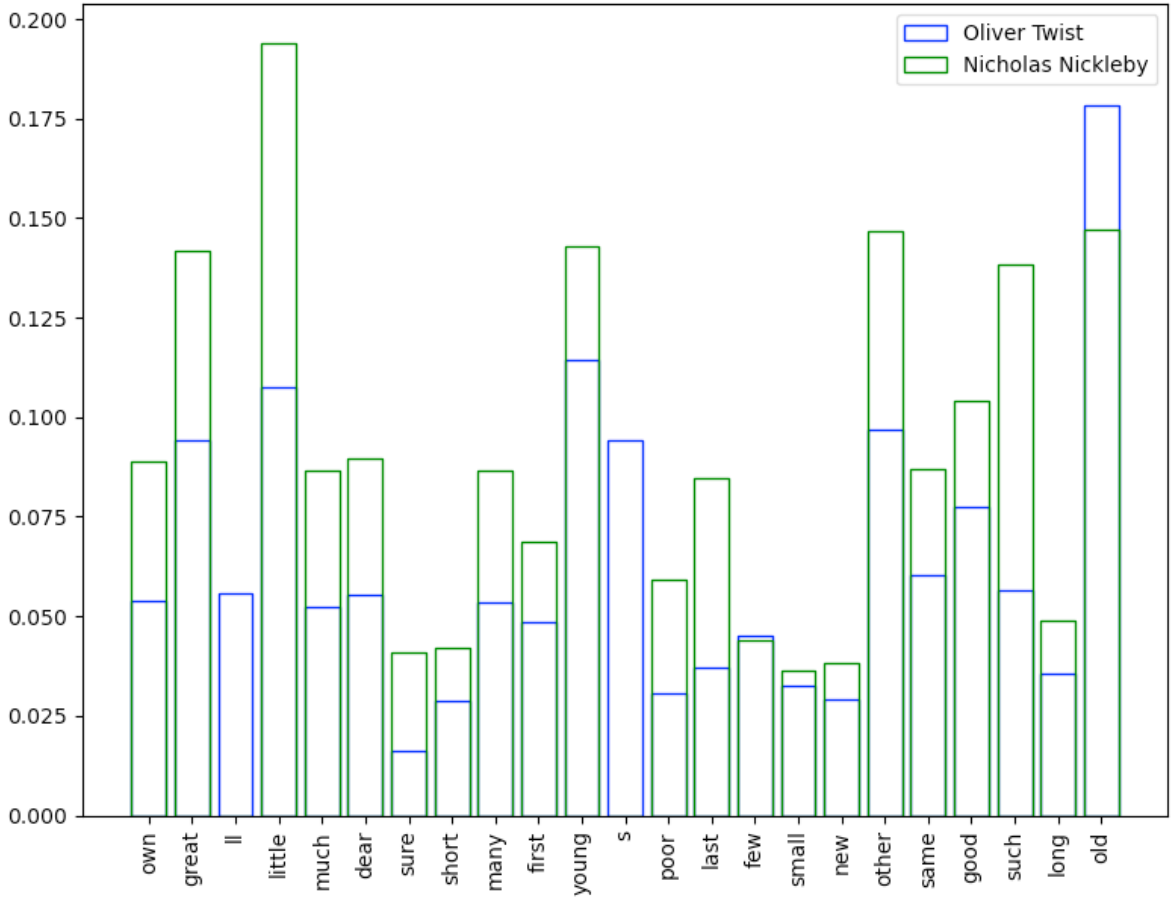
```
>>> range(len(x))
range(0, 23) = [0,1,2,3,…]
```

- and

```
>>> [fd[word]/len(fd) for word in x]
[0.05382087099424815, 0.0940838126540673 7, 0.05587510271158587,
0.10764174198849 63, 0.05217748562037798, 0.055464256368118324,
0.016023007395234 18, 0.028759244042728 02, 0.053410024650780 61,
0.048479868529170 09, 0.114215283483977, 0.094083812654067 37,
0.030813475760065 736, 0.03697617091207888, 0.045193097781429 74,
0.032456861133935 91, 0.029170090386195564, 0.096959737058340 18,
0.060394412489728 84, 0.077649958915365 65, 0.056696795398520954,
0.035743631881676 25, 0.17830731306491374]
```



Worked Example: top-20 adjectives



Worked Example

matplotlib.pyplot.xticks

matplotlib.pyplot.xticks(*ticks=None, labels=None, *, minor=False, **kwargs*) [source]

Get or set the current tick locations and labels of the x-axis.

Pass no arguments to return the current values without modifying them.

Parameters:

ticks : ***array-like, optional***

The list of xtick locations. Passing an empty list removes all xticks.

labels : ***array-like, optional***

The labels to place at the given *ticks* locations. This argument can only be passed if *ticks* is passed as well.

minor : ***bool, default: False***

If `False`, get/set the major ticks/labels; if `True`, the minor ticks/labels.

****kwargs**

`Text` properties can be used to control the appearance of the labels.

Examples

```
>>> locs, labels = xticks()  # Get the current locations and labels.
>>> xticks(np.arange(0, 1, step=0.2))  # Set label locations.
>>> xticks(np.arange(3), ['Tom', 'Dick', 'Sue'])  # Set text labels.
>>> xticks([0, 1, 2], ['January', 'February', 'March'],
...         rotation=20)  # Set text labels and properties.
>>> xticks([])  # Disable xticks.
```

Worked Example

```
plt.bar(x-values, heights)
```

Other Parameters:

color : *color or list of color, optional*

The colors of the bar faces.

edgecolor : *color or list of color, optional*

The colors of the bar edges.

linewidth : *float or array-like, optional*

Width of the bar edge(s). If 0, don't draw edges.

tick_label : *str or list of str, optional*

The tick labels of the bars. Default: None (Use default numeric labels.)

label : *str or list of str, optional*

A single label is attached to the resulting `BarContainer` as a label for the whole dataset. If a list is provided, it must be the same length as *x* and labels the individual bars. Repeated labels are not de-duplicated and will cause repeated label entries, so this is best used when bars also differ in style (e.g., by passing a list to *color*.)

Worked Example

https://matplotlib.org/stable/gallery/color/named_colors.html#sphx-glr-gallery-color-named-colors-py

Base colors

```
plot_colortable(mcolors.BASE_COLORS, ncols=3, sort_colors=False)
```

Worked Example

```
plt.bar(x-values, heights)
```

****kwargs** : `Rectangle` ***properties***

Property	Description
`agg_filter`	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array and two offsets from the bottom left corner of the image
`alpha`	scalar or None
`angle`	unknown
`animated`	bool
`antialiased` or aa	bool or None
`bounds`	(left, bottom, width, height)
`capstyle`	`CapStyle` or {'butt', 'projecting', 'round'}
`clip_box`	`BboxBase` or None
`clip_on`	bool
`clip_path`	Patch or (Path, Transform) or None
`color`	color
`edgecolor` or ec	color or None
`facecolor` or fc	color or None
`figure`	`Figure`
`fill`	bool