CSC401 Natural Language Computing

Tutorial: Assignment 1

Jan 19, 2018

TA: Willie Chang

(Slides adapted from Stefania Raimondo, Erin Grant, Siavash Kazemian, Varada Kolhatkar, Ka-Chun Won, and Aryan Arbabi)

Mascots: r/sandersforpresident (left) and r/the_donald (right)

Goal

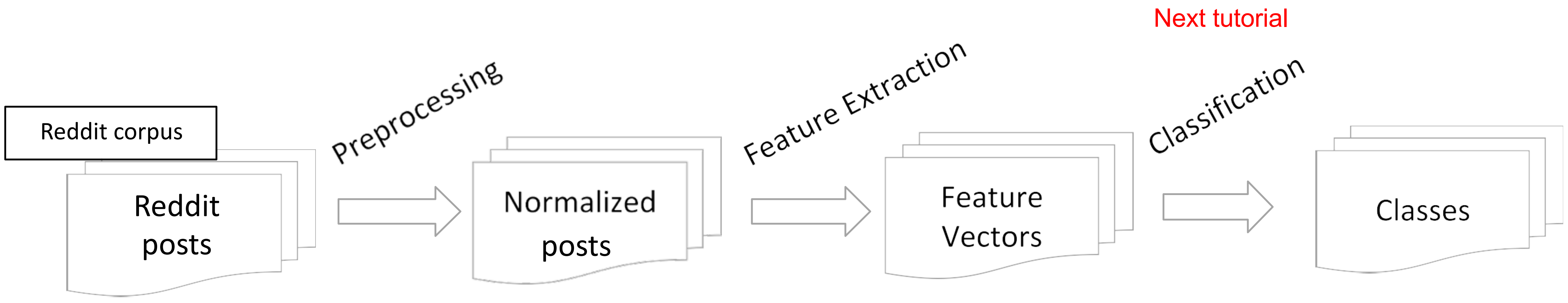
To perform **sentiment analysis** on Reddit posts and comments.

In this assignment, you will classify them according to their political leaning.

Input	Output
“‘States' rights’ only apply when Democrats are in power.”	Left
“Leftists don't understand just how fed up the people are.”	Right

Commenters: u/spaceghoti (left), u/Donnaguska (right)

Methodology



Reddit corpus

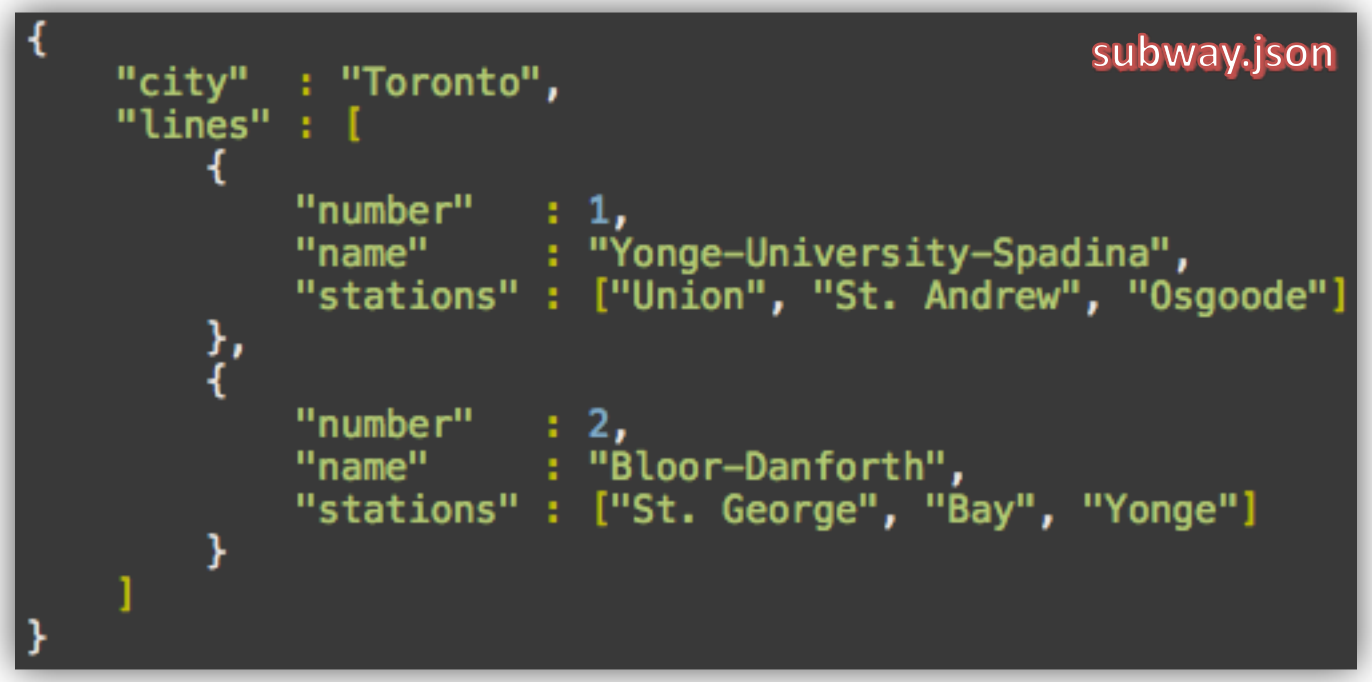
- /u/cs401/A1/data/
  - Over a million posts (several hundred thousand per category)
  - You will only process 40,000 posts (10,000 per category)

- Format: JSON

a1_preproc.py takes a not-yet-preprocessed JSON-like file provided as input and outputs a JSON file with preprocessed text.

Icon: Smashicons on flaticon.com

JSON in Python



subway.json

```
{
    "city"  : "Toronto",
    "lines" : [
        {
            "number"   : 1,
            "name"     : "Yonge-University-Spadina",
            "stations" : ["Union", "St. Andrew", "Osgoode"]
        },
        {
            "number"   : 2,
            "name"     : "Bloor-Danforth",
            "stations" : ["St. George", "Bay", "Yonge"]
        }
    ]
}
```

```
import json, pprint

file = open("subway.json")
data = json.load(file)

print(data["lines"][0]["name"])
# Prints "Yonge-University-Spadina"

pprint.pprint(data)
# Prints the entire JSON
```

Your script must output valid JSON

```
[{
    "id": "nf9w3g",
    "body": "......",
    "cat": "Right"
                 ↑ no comma
}, {
    "id": "b92j4a",
    "body": "......",
    "cat": "Left"
}]
 ↑ no comma
```

This includes, but is not limited to:

- Keys and values in double quotes
- No comma after the last value in an object or array
- Use UTF-8; Python 3 on CDF uses this by default, so don’t change it
- Escape quotation marks in values
- In values, use newlines \n rather than actual line breaks

Tip: Use jsonlint.com to validate!

part 1

Preprocessing

Key Points

1. Clean up the text to remove noise
2. Tokenization (separate words and punctuation into tokens)
    - Also, delimit sentences
3. Parts-of-speech tagging and lemmatization

Removing HTML/URLs

- Regex is your friend!

- For fixed patterns, you can use *string replace*
  - Example: my_string.replace("Air Canada Centre", "Scotiabank Arena")
  - Note: strings are immutable
- For variable patterns, you'll need *regular expressions*
  - Example: for URLs, use re.sub
  - Note: re is greedy!

    If you're trying to remove HTML tags from the string "<title>New Page</title>" with the regex "<.+>", you'll end up with an empty string.

RegEx

```
import re

# Compiling a pattern that looks for natural numbers without commas
pattern = re.compile("\d+")

# Find all substrings where the pattern matches
pattern.findall("Highway 401 continues in Quebec as Autoroute 20")
# ["401", "20"]

# Search for pattern, then replace
pattern = re.compile("\d{3}-\d{3}-\d{4}")
pattern.sub("XXX-XXX-XXXX", "Call 718-387-6962")
# "Call XXX-XXX-XXXX"
```

More examples: http://www.cs.toronto.edu/~frank/csc401/tutorials/401_python_web/regexp.html. A regex found online can be used as long as it is cited, with adequate documentation with regards to how it works **(no student collaborations)**.

Sentence Boundaries

Add a newline (“\n”) between each sentence.

- Sentences end with ‘.’, ‘?’, or ‘!’
- But not all periods are EOS (e.g. abbreviations)

  e.g., How much does the U.S. president get paid?
- But some abbreviations *are* EOS

  e.g., After the UK tour ends next week, he returns to the U.S.
- Possible solution: consider checking if the following letter is lowercase

  But what about: e.g., After U.S. Attorney General…
- List of common abbreviations:
  - /u/cs401/Wordlists/abbrev.english

Sentence Boundaries

- Don’t break multiple times for multiple punctuation(e.g. !!!)
- But not all ellipsis are EOS
  e.g., I dunno Manny… do you want to go?
- Quotations: after the punctuation, but part of the sentence
  e.g., “You remind me,” she remarked, “of your mother.”

- There is no perfect sentence parser!
- See Manning and Schütze, Section 4.2.4 for some good ideas

Tokenization: Splitting sentences into tokens

- Simple words: Use line.strip().split()

  e.g., ‘an apple’ → [‘an’, ‘apple’]

- Punctuation should be it’s own token

  e.g., ‘she said,’ → [‘she’, ‘said’, ‘,’]

- But not always…

  e.g., ‘paid $10,000’ → [‘paid’, ‘$’, ‘10,000’]

- Including clitics and contractions

  e.g., “can’t” → [“ca”, “n’t”]

Don't use spaCy for tokenization

Tokenization (con’t)

- Possessives

  $\downarrow$ Apostrophe

  e.g., “she’s” $\longrightarrow$ [“she”, “’s”]
- Compounds (your choice)

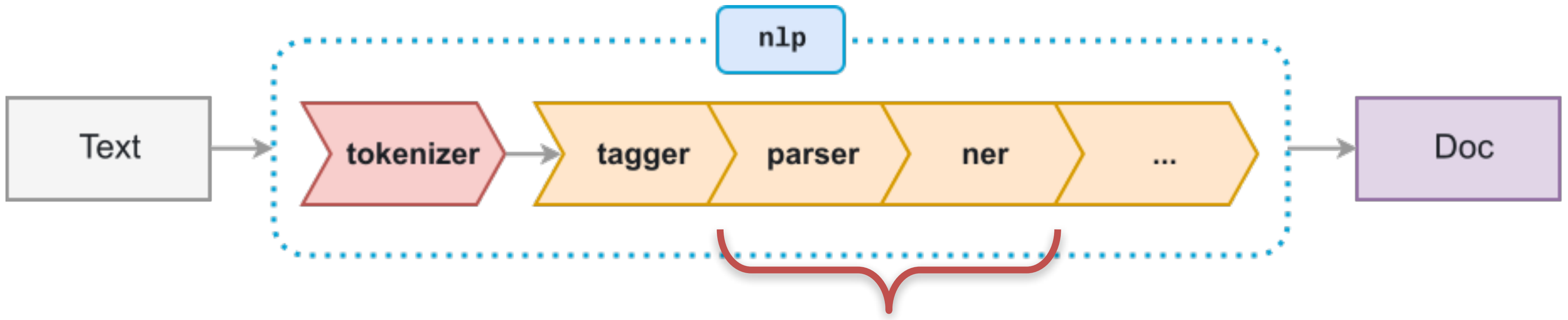
  e.g., time-consuming
- Don’t break up ellipsis...

Don't use spaCy for tokenization

POS Tagging

spaCy

```
import spacy
nlp = spacy.load('en', disable=['parser', 'ner'])
```



We don’t need parser or ner, so we disable them

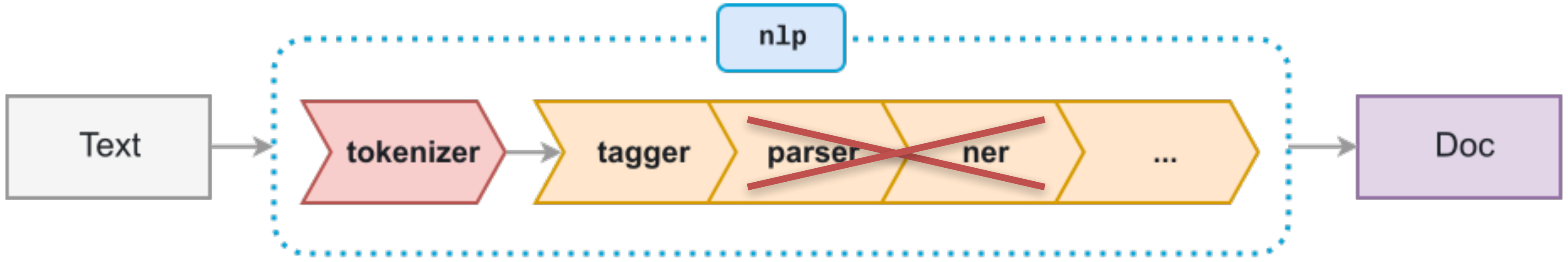
Figure: https://spacy.io/usage/processing-pipelines

POS Tagging

spaCy

spaCy has a built-in tokenizer, but you need to implement your own in the previous step, then pass in the tokens directly.

```
doc = spacy.tokens.Doc(nlp.vocab, words=["i", "'m", "very", "highly", "educated"])
```



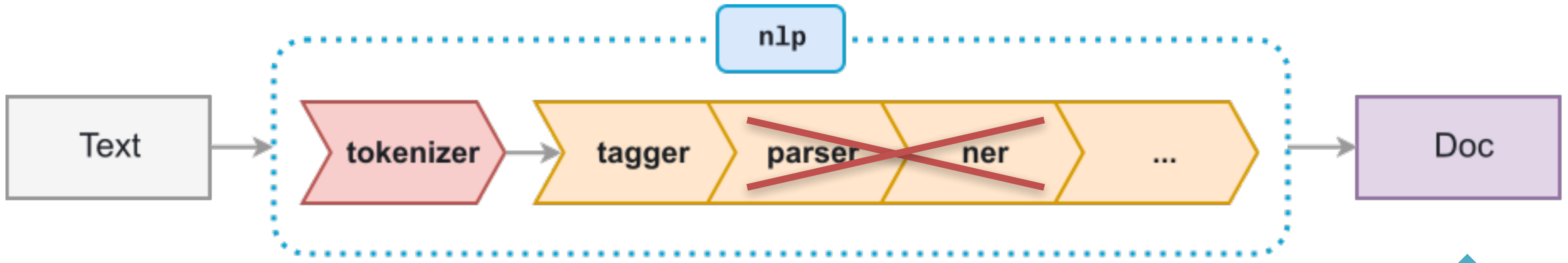
spaCy performs POS tagging automatically when you pass in a sentence.
However, spaCy does not automatically tag POS if you provide your own tokens.
So, you'll need to call the POS tagger manually.

```
doc = nlp.tagger(doc)
```

Figure: https://spacy.io/usage/processing-pipelines

POS Tagging

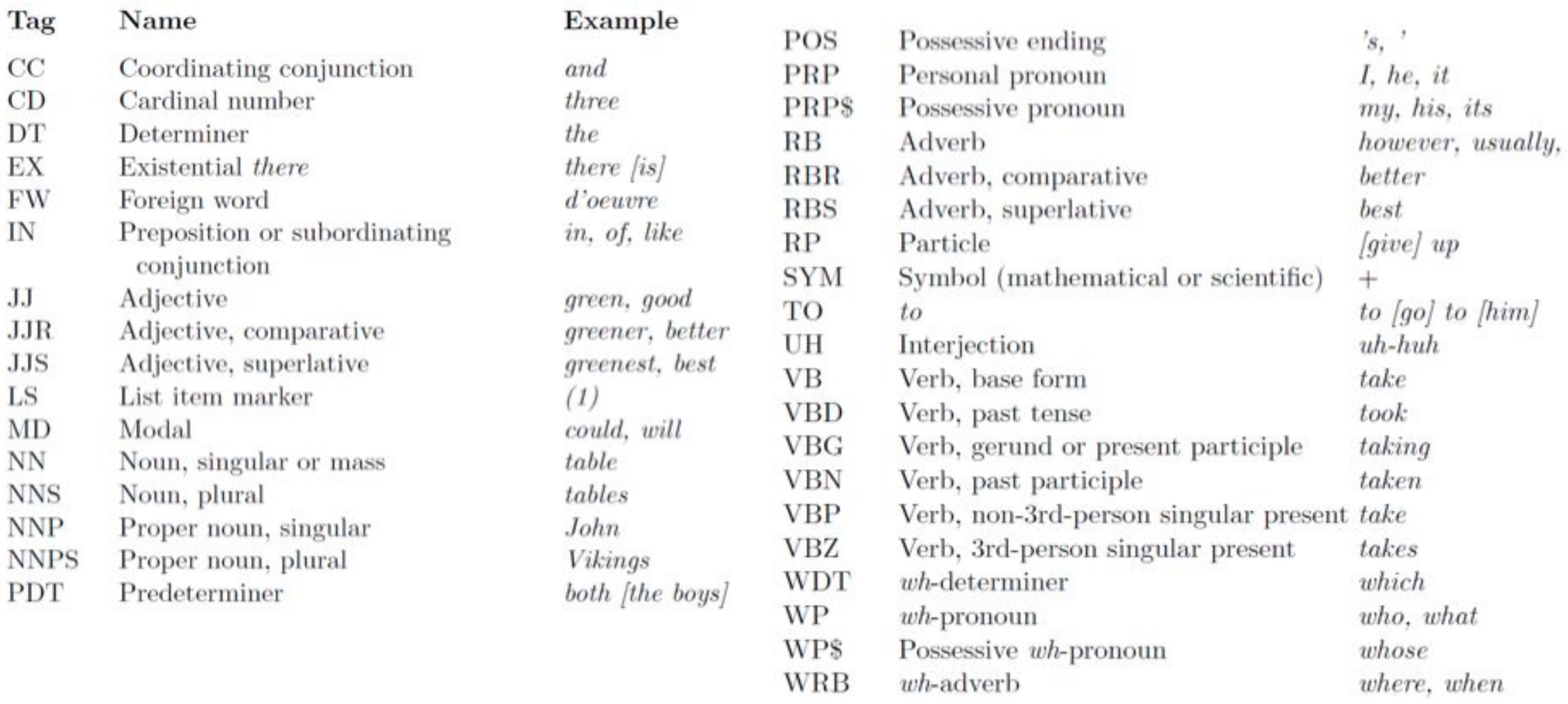
spaCy



[print(token.text) for token in doc]	[print(token.tag_) for token in doc]	[print(token.lemma_) for token in doc]
i	PRP	i
'm	VBP	be
very	RB	very
highly	RB	highly
educated	VBN	educate

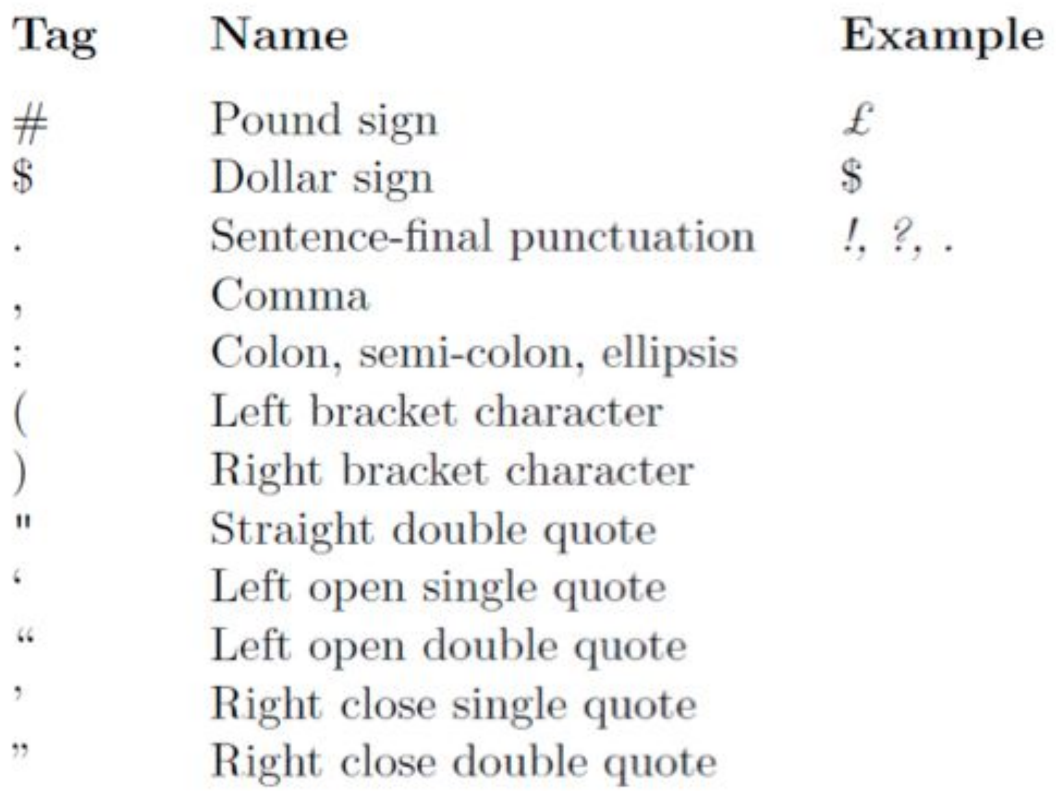
Figure: https://spacy.io/usage/processing-pipelines

Tag list (see handout)



Tag	Name	Example
CC	Coordinating conjunction	*and*
CD	Cardinal number	*three*
DT	Determiner	*the*
EX	Existential *there*	*there [is]*
FW	Foreign word	*d'oeuvre*
IN	Preposition or subordinating conjunction	*in, of, like*
JJ	Adjective	*green, good*
JJR	Adjective, comparative	*greener, better*
JJS	Adjective, superlative	*greenest, best*
LS	List item marker	*(1)*
MD	Modal	*could, will*
NN	Noun, singular or mass	*table*
NNS	Noun, plural	*tables*
NNP	Proper noun, singular	*John*
NNPS	Proper noun, plural	*Vikings*
PDT	Predeterminer	*both [the boys]*
POS	Possessive ending	*'s, '*
PRP	Personal pronoun	*I, he, it*
PRP$	Possessive pronoun	*my, his, its*
RB	Adverb	*however, usually,*
RBR	Adverb, comparative	*better*
RBS	Adverb, superlative	*best*
RP	Particle	*[give] up*
SYM	Symbol (mathematical or scientific)	+
TO	*to*	*to [go] to [him]*
UH	Interjection	*uh-huh*
VB	Verb, base form	*take*
VBD	Verb, past tense	*took*
VBG	Verb, gerund or present participle	*taking*
VBN	Verb, past participle	*taken*
VBP	Verb, non-3rd-person singular present	*take*
VBZ	Verb, 3rd-person singular present	*takes*
WDT	*wh*-determiner	*which*
WP	*wh*-pronoun	*who, what*
WP$	Possessive *wh*-pronoun	*whose*
WRB	*wh*-adverb	*where, when*

Tag list (see handout)



Tag	Name	Example
#	Pound sign	£
$	Dollar sign	$
.	Sentence-final punctuation	*!, ?, .*
,	Comma	
:	Colon, semi-colon, ellipsis	
(	Left bracket character	
)	Right bracket character	
"	Straight double quote	
‘	Left open single quote	
“	Left open double quote	
’	Right close single quote	
”	Right close double quote	

- Space between tokens `(“ ”.join(tokens))`

Code Modularity

Please write well-structured code; this includes making your code modular.

We will be running a suite of auto-graders, some of which bypass main() in order to test specific steps of your preprocessor.

Because main() is not called, don't change any global variable in main() that will be used by preproc1().

```
clitics = []

def preproc1(comment, steps=range(1,11)):
        # Get some string in clitics

def main(args):
        ~~clitics.append("n't")~~   # AVOID!
```

Feature Extraction

part 2

Input: JSON   Output: NPZ (NumPy array)

Three types of tasks

- Count the number of tokens meeting a specified criterion in a post
  - Consider all relevant tags (e.g. RB, RBR, and RBS are all adverbs).
  - Check /u/cs401/Wordlists/ (e.g. for slang and 1st/2nd/3rd-person pronouns).
- Lexical norms
  - Look up each word in the respective norms file.
  - Do not consider words missing from the norms file.
- Linguistic Inquiry & Word Count (LIWC) / Receptiviti – *personality and word choice*
  - Find the post ID in the /u/cs401/A1/feats/*CategoryName*_IDs.txt file.
  - Copy the 144 elements from *CategoryName*_feats.dat.npy starting at element 144 times the line index where you found the post ID in *CategoryName*_IDs.txt.
  - **Line index starts at 0**, which is the first actual line in the *CategoryName*_IDs.txt file!

Feature Definitions

- Coordinating conjunctions (CC):
  - *and, but, for, nor, or, so*, and *yet*
- Past and future tense verbs
  - You should be able to come up with some rules for most cases...
  - Watch out for irregular verbs!
    - As well as verb tokens with the "n't" clitic removed (e.g. can't → ca, n't)
  - Perfective aspect (*has*/*have eaten*) should be counted as **one** token.
  - Present-tense verbs can be used to describe future events; don't count them as future-tense.
    - "She is moving to Vancouver next month."

Feature Definitions

- Number of common nouns (NN, NNS)
- Number of proper nouns (NNP, NNPS)
- Number of Adverbs (RB, RBR, RBS)
- Number of wh-words (WDT, WP, WP$, WRB)
    - **Use the tagger output!**

Tips

- Write clean, well-documented, and efficient (for part 2) code.
- Sanity check often
- Have a look at the corpus
- Use your best judgement– check how these tools handle specific cases:
    - https://code.google.com/p/splitta/
    - http://nlp.stanford.edu/software/tokenizer.shtml

Finish Part 1 ASAP!

- Get it working. Don’t worry about perfecting it. There’s no such thing as a perfect parser.

More Tips

- Python features you may want to use:
  - Dictionaries, regular expressions
  - String formatting (% operator)

- Do **not** hardcode file paths in your home directory.
  - Instead, reference CSC401 folders such as */u/cs401/Wordlists/*.
  - Or place the necessary data somewhere in the files you submit.

- Before tackling the bonus, ensure the rest of the assignment is done well.