Natural Language Processing

Christan Grant, Ph.D.

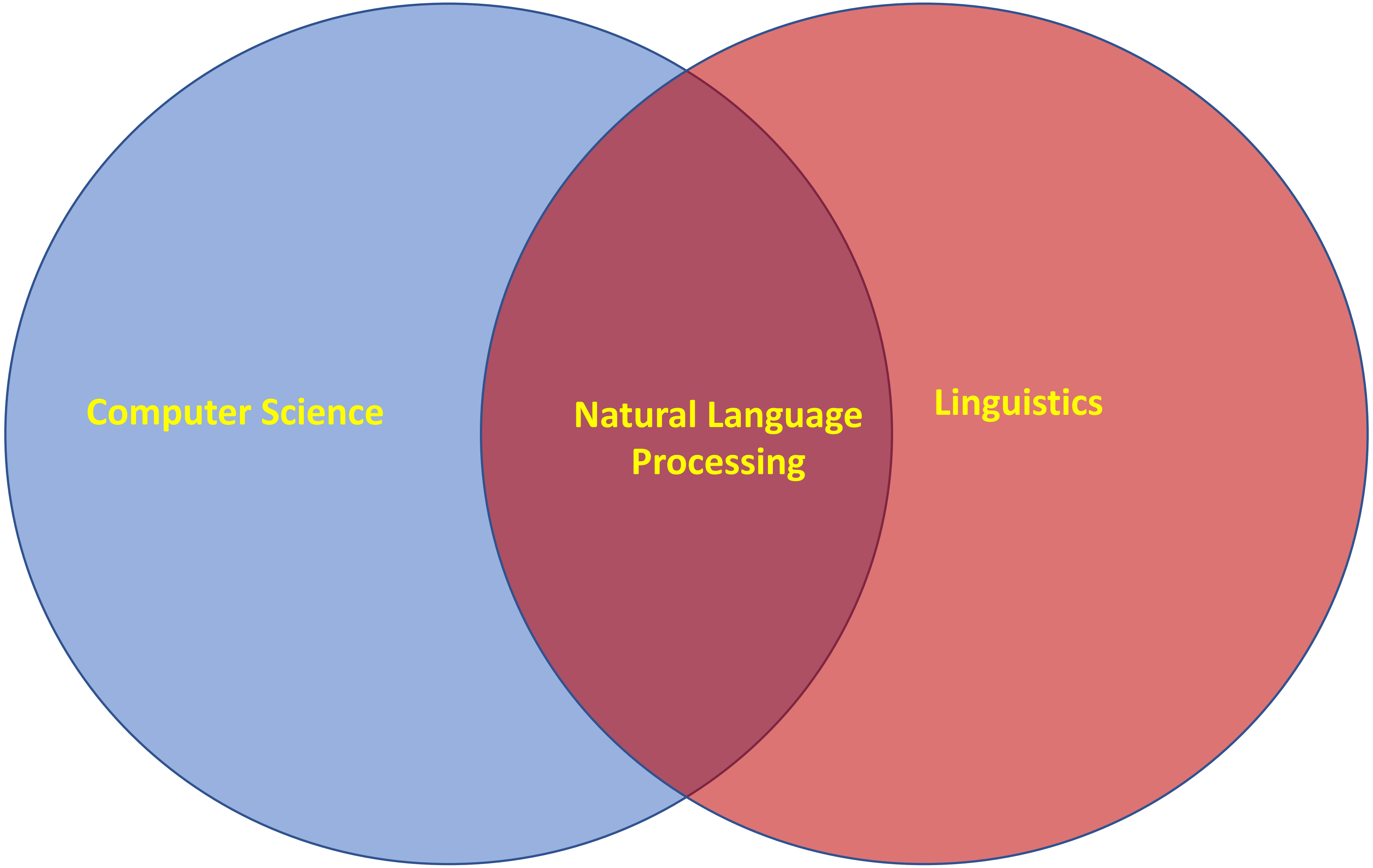
Data Bite Summer 2021

oudalab.com

DataBite Summer 2021 Schedule

Day 1 Welcome	**Day 2** Introduction to Python	**Day 3** Introduction to Probability	**Day 4** Model Olympics
Day 5 Socratic Seminar	**Day 6** Bias and Fairness	**Day 7** Natural Language Processing	**Day 8** Deep Learning

What is Natural Language Processing?

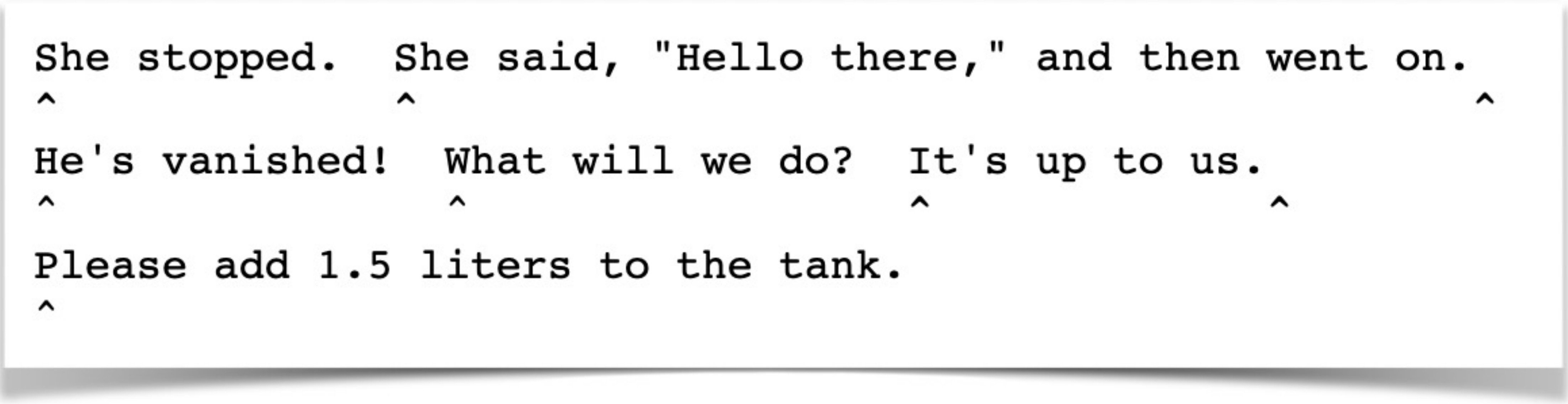


Natural Language Processing (NLP) Tasks

- Conversational Agents
- Natural Language Understanding
- Question Answering
- Text Summarization
- Knowledge Base Construction
- Machine Translation
- Information Retrieval
- Text Classification
- Sentiment Analysis
- Entity Resolution
- Coreference Resolution
- Named Entity Recognition
- Relation Extraction
- Entity Linking
- Word Sense Disambiguation
- Part of Speech Tagging
- Spelling Correction
- Text-to-Speech
- Speech-to-Text
- And more…

Sentence Segmentation

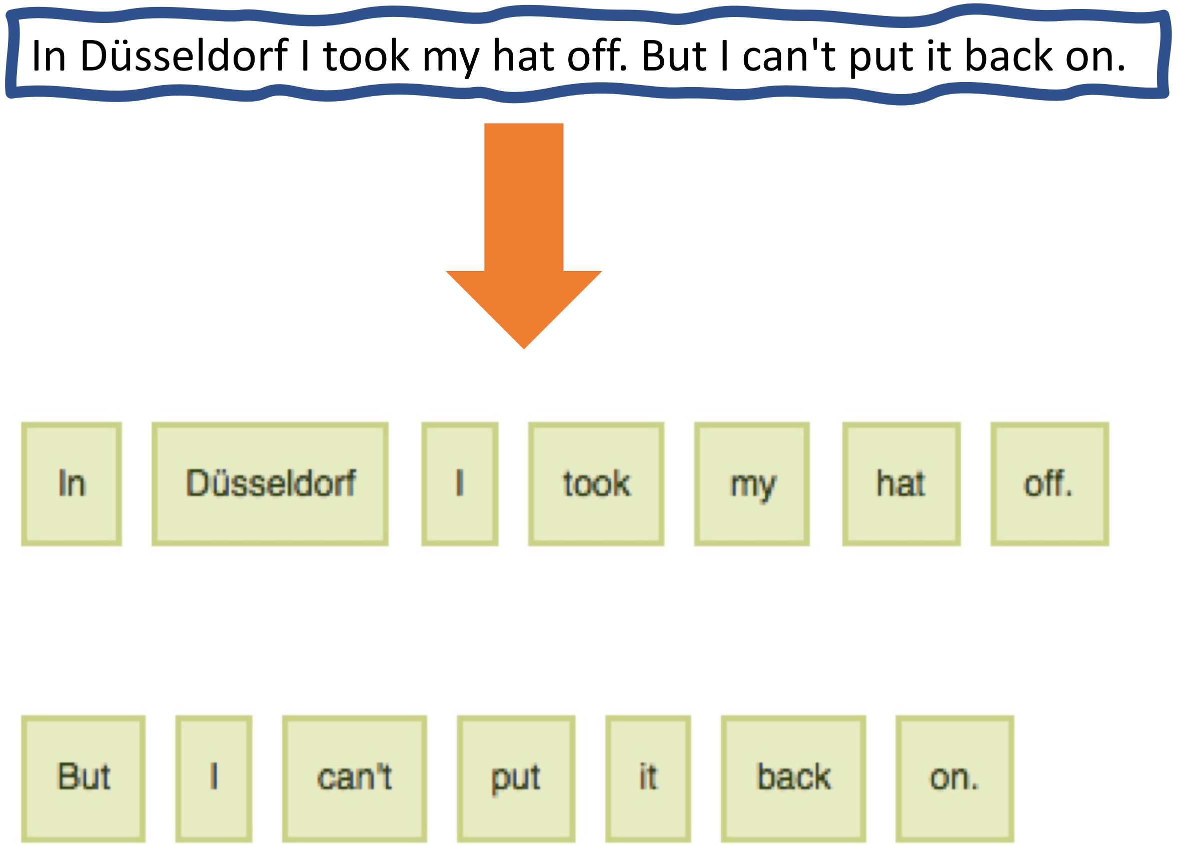
Split a Textual Document into sentences



```
She stopped.  She said, "Hello there," and then went on.
^             ^                                         ^
He's vanished!  What will we do?  It's up to us.
^               ^                 ^             ^
Please add 1.5 liters to the tank.
^
```

Word Tokenization

Split a sentence into tokens.



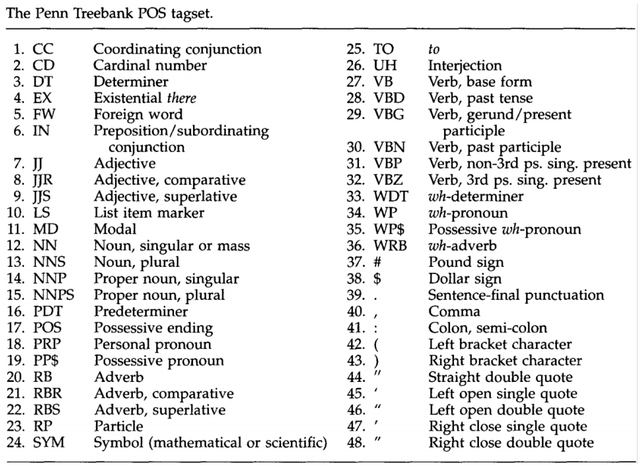
In Düsseldorf I took my hat off. But I can't put it back on.

In | Düsseldorf | I | took | my | hat | off.

But | I | can't | put | it | back | on.

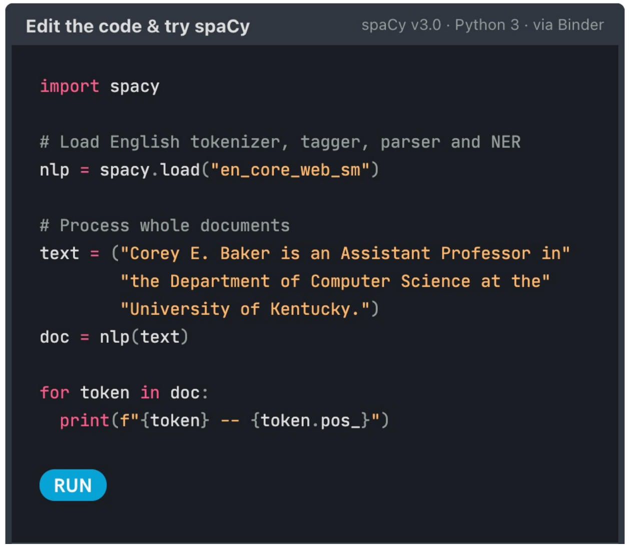
Part-of-Speech Tagging

Classify word tokens into Parts of Speech.

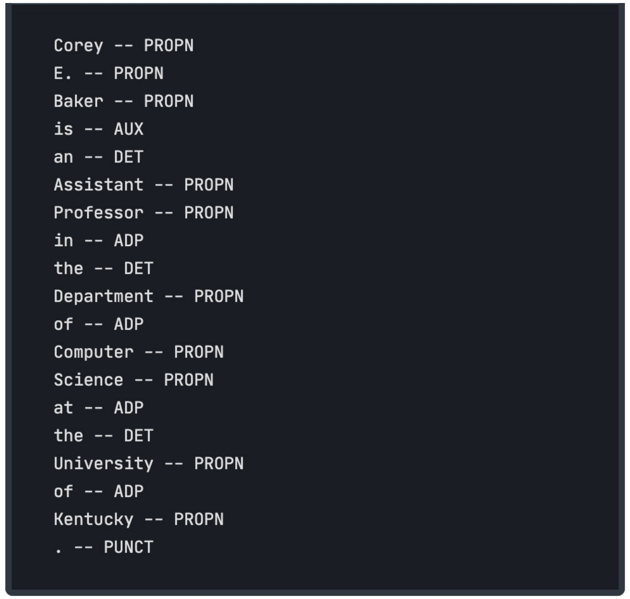


The Penn Treebank POS tagset.

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/subordinating conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-3rd ps. sing. present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd ps. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	‘	Left open single quote
22.	RBS	Adverb, superlative	46.	“	Left open double quote
23.	RP	Particle	47.	’	Right close single quote
24.	SYM	Symbol (mathematical or scientific)	48.	”	Right close double quote



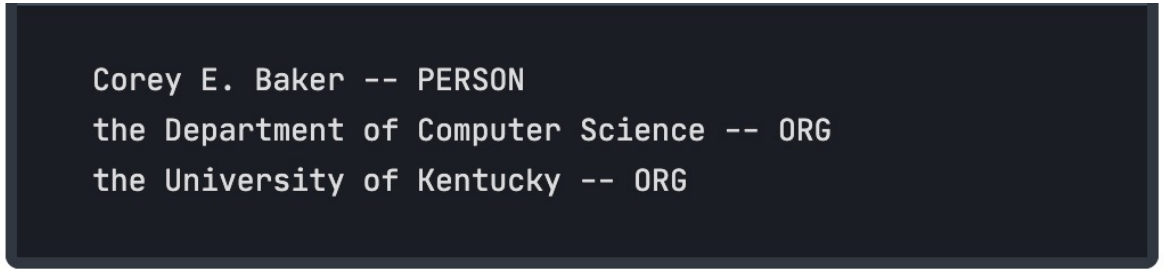
```python
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("Corey E. Baker is an Assistant Professor in"
        "the Department of Computer Science at the"
        "University of Kentucky.")
doc = nlp(text)

for token in doc:
  print(f"{token} -- {token.pos_}")
```



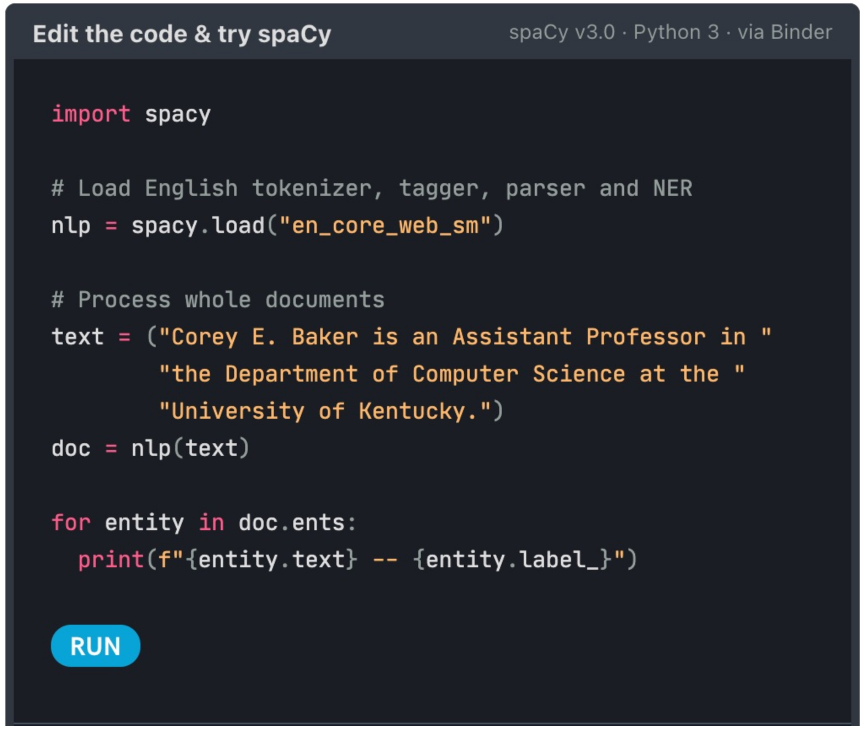
```
Corey -- PROPN
E. -- PROPN
Baker -- PROPN
is -- AUX
an -- DET
Assistant -- PROPN
Professor -- PROPN
in -- ADP
the -- DET
Department -- PROPN
of -- ADP
Computer -- PROPN
Science -- PROPN
at -- ADP
the -- DET
University -- PROPN
of -- ADP
Kentucky -- PROPN
. -- PUNCT
```

Named Entity Recognition

Identify the tokens in a sentence that correspond to an Entity.



```
Corey E. Baker -- PERSON
the Department of Computer Science -- ORG
the University of Kentucky -- ORG
```

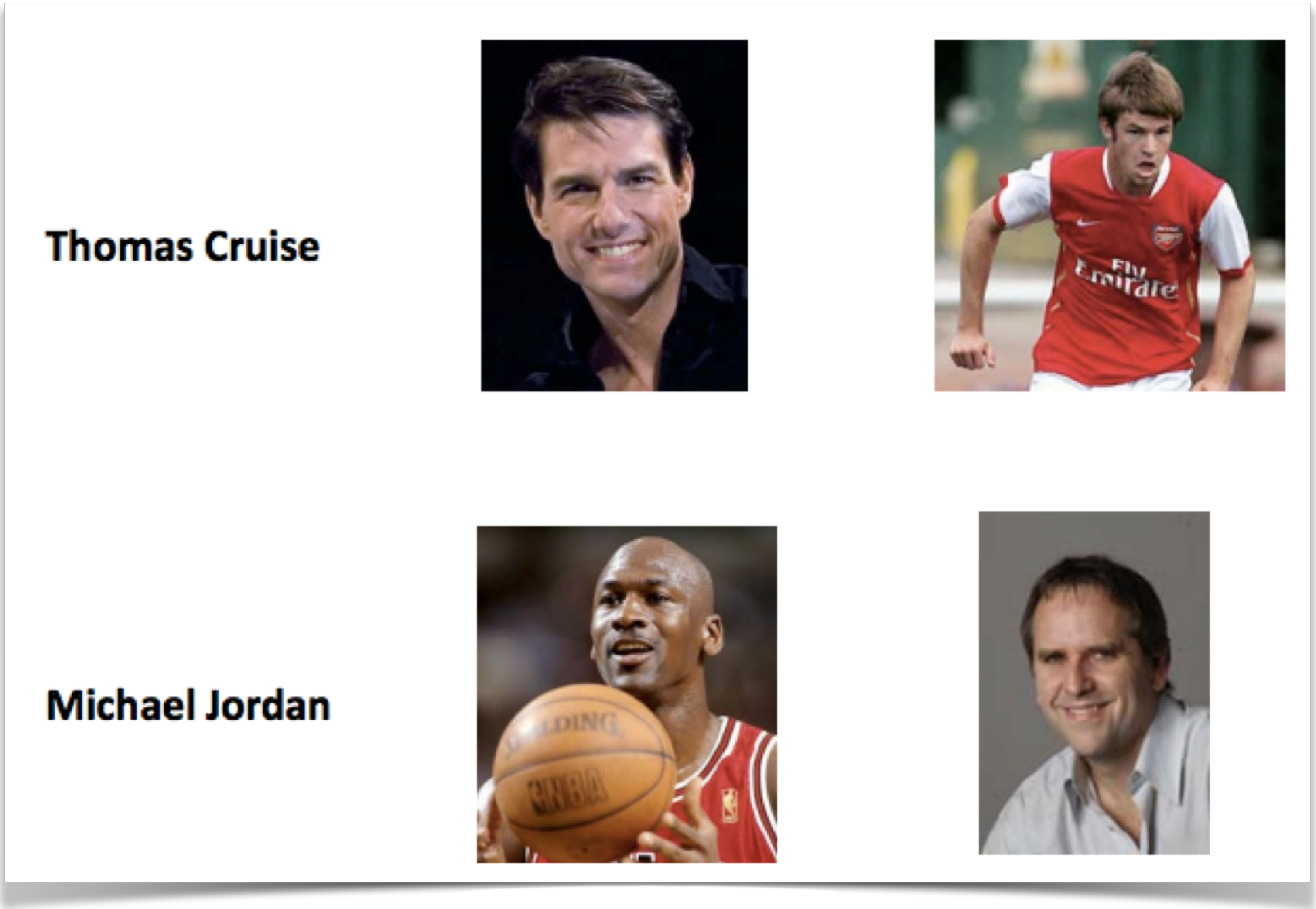


```python
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("Corey E. Baker is an Assistant Professor in "
        "the Department of Computer Science at the "
        "University of Kentucky.")
doc = nlp(text)

for entity in doc.ents:
  print(f"{entity.text} -- {entity.label_}")
```

TYPE	TAG	SAMPLE CATEGORIES
People	PER	Individuals, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes seas
Geo-Political Entity	GPE	Countries states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicles	VEH	Planes, trains, and automobiles

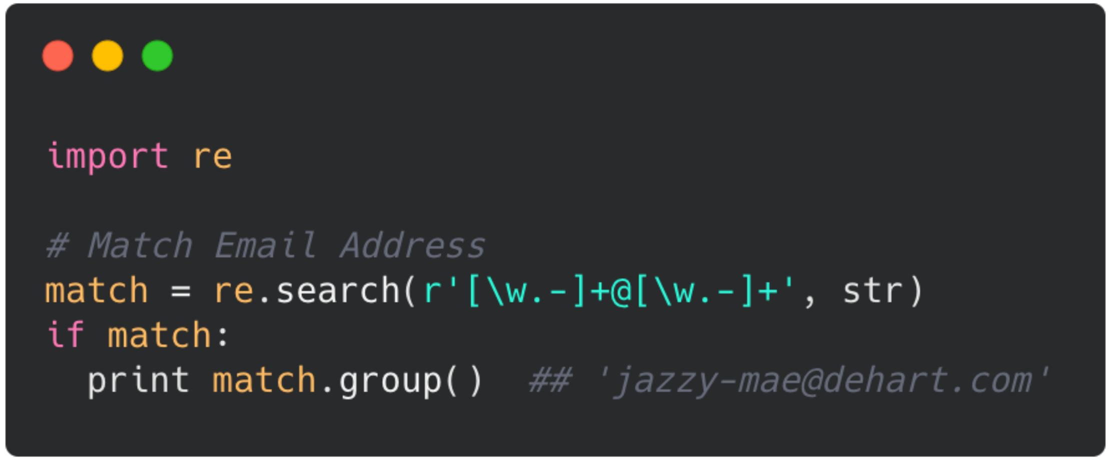
Entity Resolution

Connect mentions of noun phrases with real world objects.



Text Processing before Machine learning

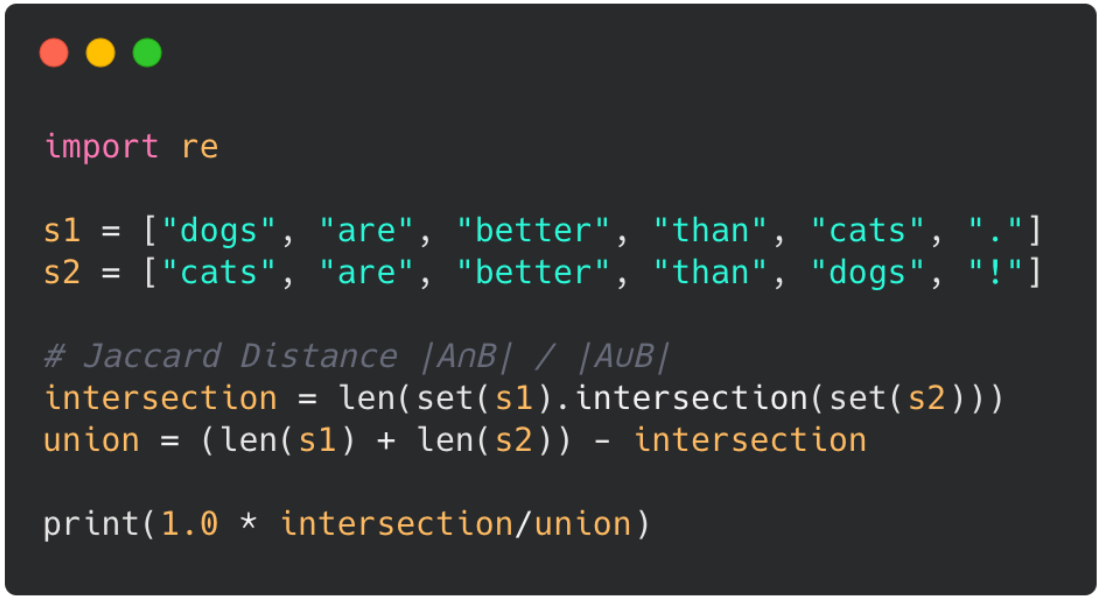
Regular expressions – A language for matching patterns in text. (More on this in another module)



```
import re

# Match Email Address
match = re.search(r'[\w.-]+@[\w.-]+', str)
if match:
  print match.group()  ## 'jazzy-mae@dehart.com'
```

Text Processing before Machine learning

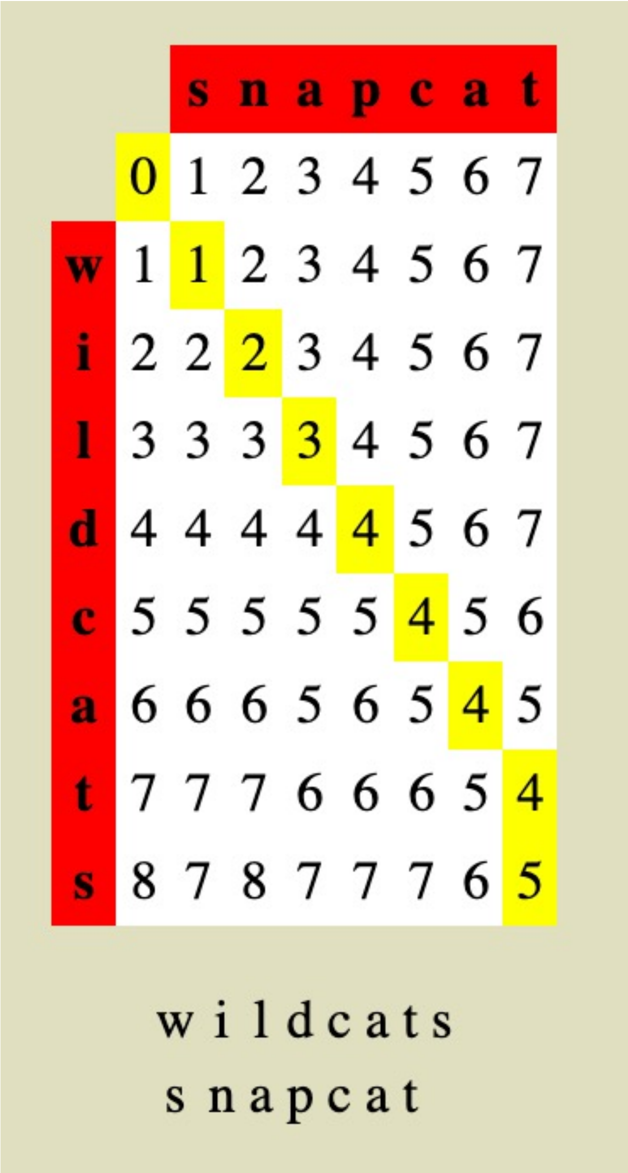
Jaccard Distance – Measure the overlap between two sentences.



```python
import re

s1 = ["dogs", "are", "better", "than", "cats", "."]
s2 = ["cats", "are", "better", "than", "dogs", "!"]

# Jaccard Distance |A∩B| / |A∪B|
intersection = len(set(s1).intersection(set(s2)))
union = (len(s1) + len(s2)) - intersection

print(1.0 * intersection/union)
```

Text Processing before Machine learning

- Minimum Edit Distance – Compute the number of edits to transform one string to another.
- Each letter/word is compared and and weighted.
- A penalty is is given if a deletion, insert, or substitution is made.



		s	n	a	p	c	a	t
	0	1	2	3	4	5	6	7
w	1	1	2	3	4	5	6	7
i	2	2	2	3	4	5	6	7
l	3	3	3	3	4	5	6	7
d	4	4	4	4	4	5	6	7
c	5	5	5	5	5	4	5	6
a	6	6	6	5	6	5	4	5
t	7	7	7	6	6	6	5	4
s	8	7	8	7	7	7	6	5

wildcats
snapcat

Feature Extraction / Vectorization

The first step in NLP is *data cleaning* 🧹🧼.

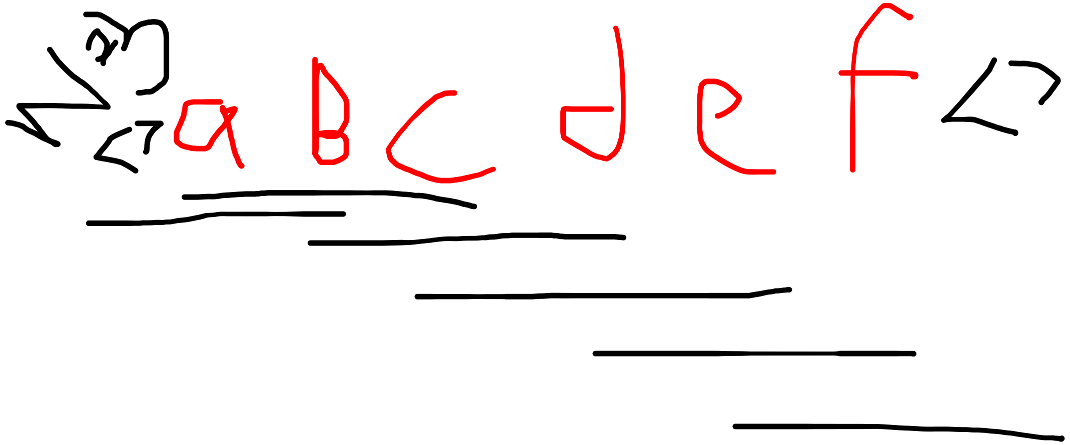
- The **most** time-consuming step [We’ll skip this discussion for now].

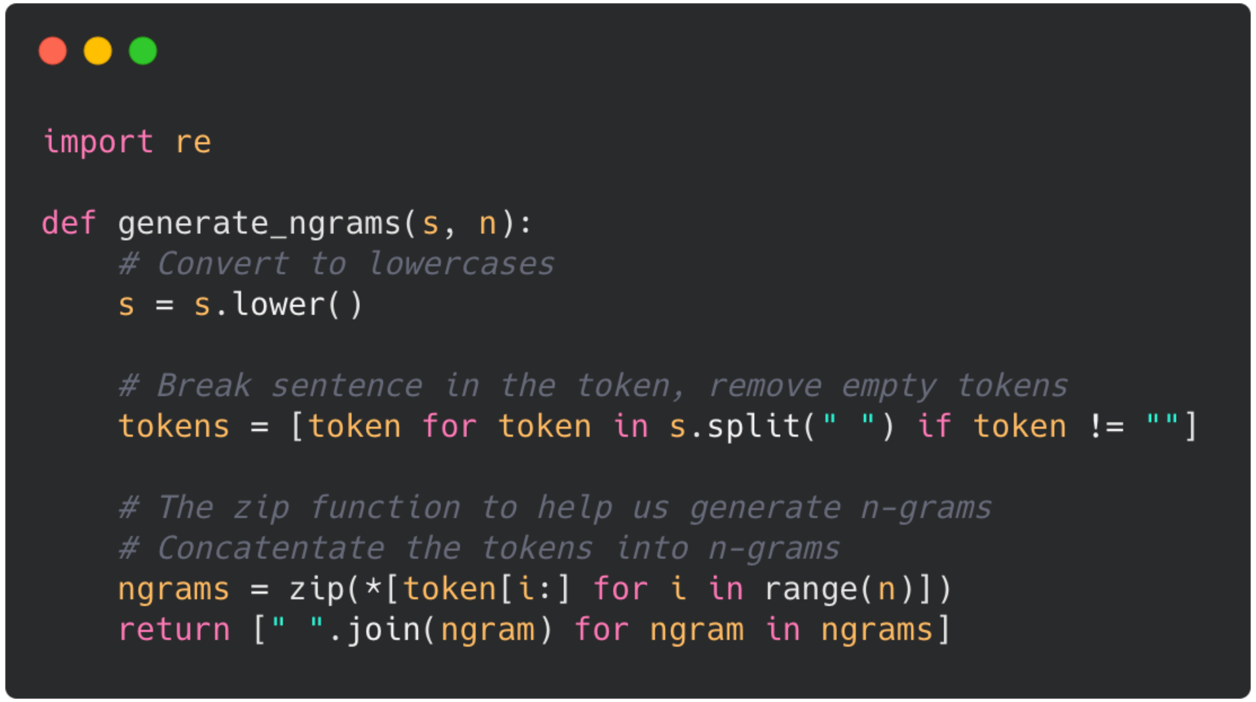
Next, is feature extraction or **vectorization**.

- Once the data is in vector form, we can use the general machine learning tools.

Vectorization

Words to N-Grams





```
import re

def generate_ngrams(s, n):
    # Convert to lowercases
    s = s.lower()

    # Break sentence in the token, remove empty tokens
    tokens = [token for token in s.split(" ") if token != ""]

    # The zip function to help us generate n-grams
    # Concatentate the tokens into n-grams
    ngrams = zip(*[token[i:] for i in range(n)])
    return [" ".join(ngram) for ngram in ngrams]
```

https://web.stanford.edu/~jurafsky/slp3/4.pdf

Vectorization – Term Importance

Term Frequency (**tf**)

Number of times a term **t** appears in a document **d**.

Document Frequency (**df**)

Number of documents that contain the word **t**.

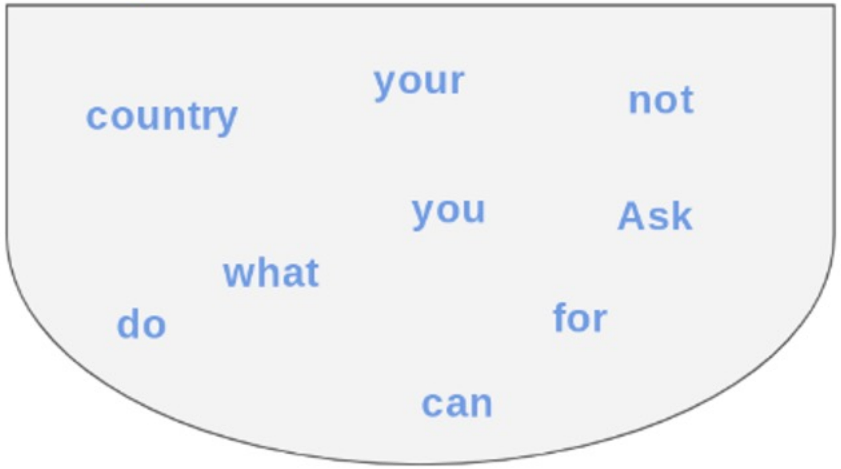
(Term Frequency) ✖ (*Inverse* Document frequency)

tf * idf

Bag of Words Models vs Sequences Models

Count Vectorizer / TFIDF Vectorizer

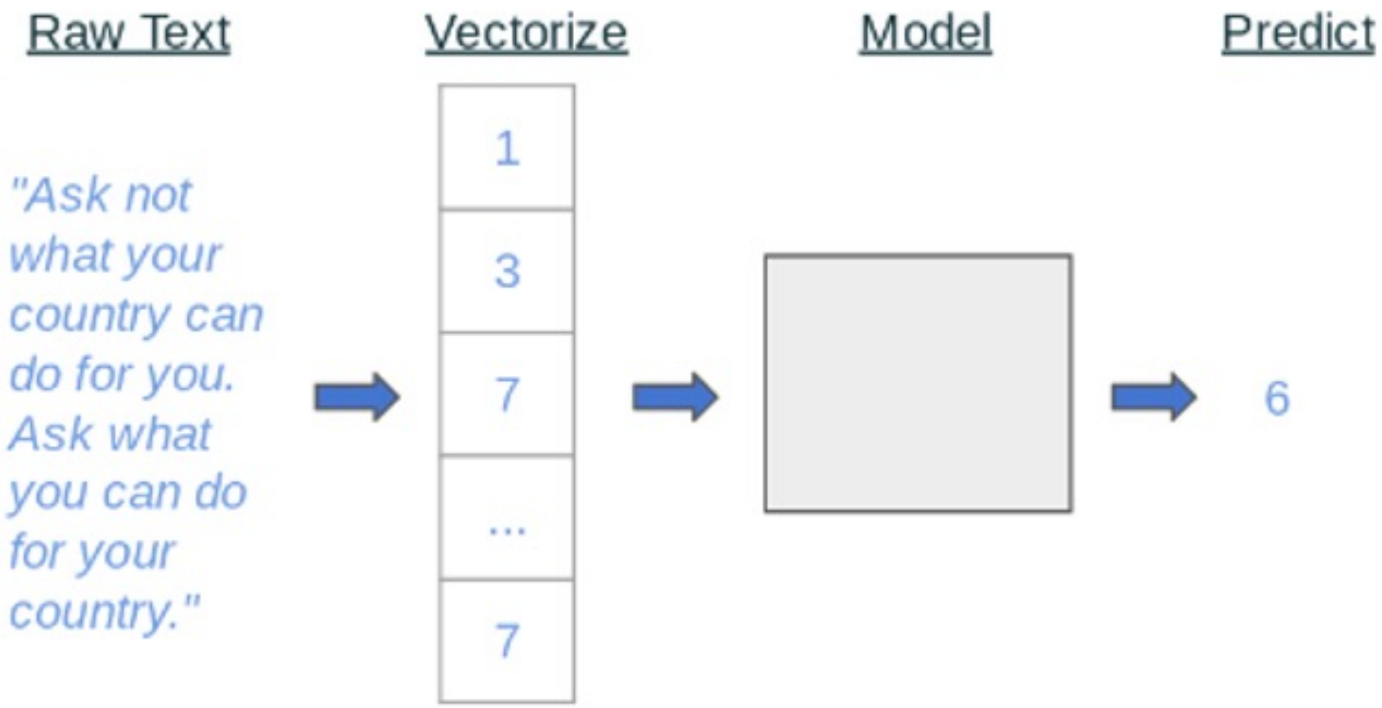
"Ask not what your country can do for you. Ask what you can do for your country."



RNN/LSTM/GRU/1D Cov

"Ask not what your country can do for ..."

NLP Pipeline


Raw Text
Vectorize
Model
Predict
"Ask not what your country can do for you. Ask what you can do for your country."
1
3
7
...
7
6