# Data & Text Analysis

*POS, NER, spacy*

IMT 547 - Social Media Data Mining and Analysis

11-Feb-2021 (Week 6, Day 12)

## Today's Topics

- Representing Text Data - DTM
- Lab leftover
- NER, POS
- Spacy
- Lab
- Project team
- PS2 discussion

## Beautiful Soup cheatsheet

Cheatsheet

http://akul.me/blog/2016/beautifulsoup-cheatsheet/

Tutorial: Web Scraping with Python Using Beautiful Soup

https://www.dataquest.io/blog/web-scraping-tutorial-python/

Beautiful Soup documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

## Representing Text Data

1. Corpus
2. Document-Term Matrix

# Representing Text Data

**Corpus**: a collection of texts

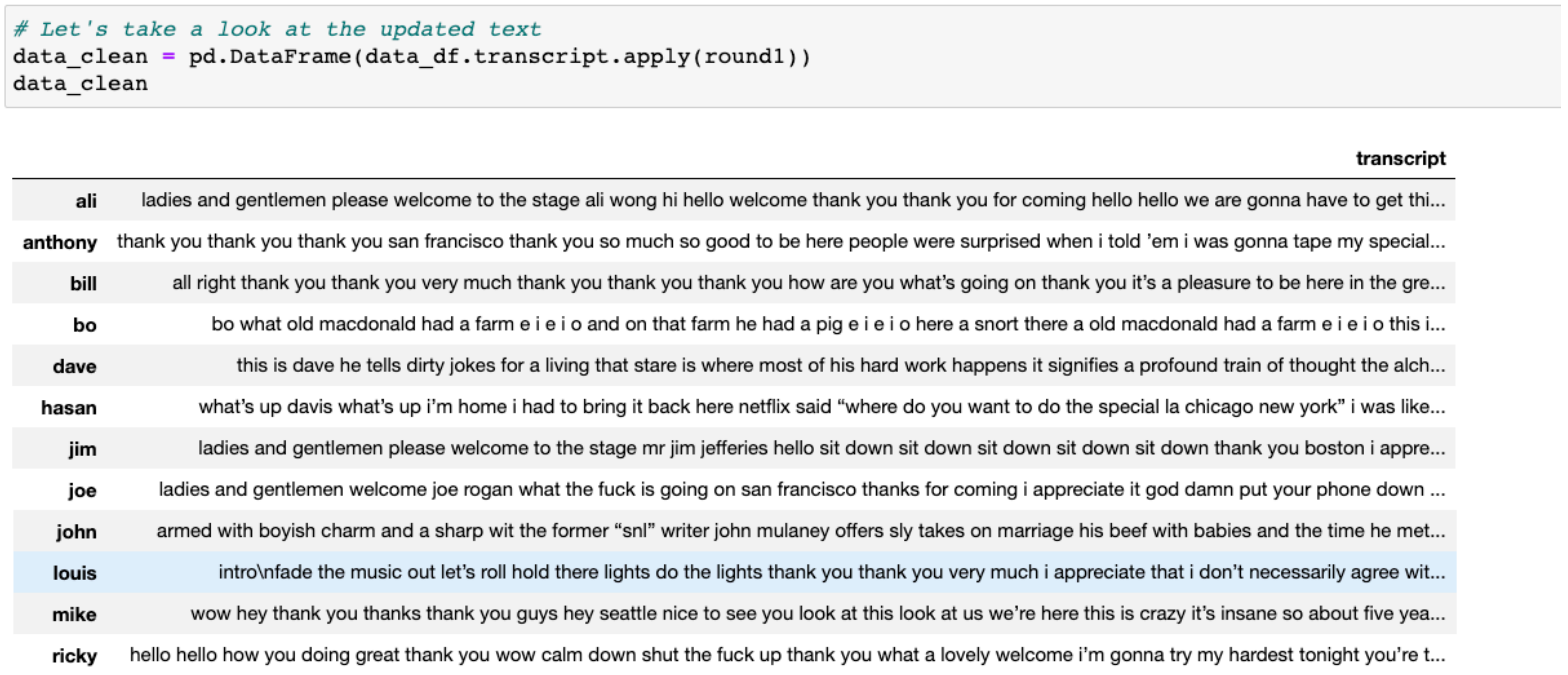
**Corpus of movie reviews:** positive or negative

Unbelievably disappointing

Richly applied satire, great plot twists.

This is the greatest screwball comedy ever filmed

It was pathetic. The worst part about it was the boxing scenes



```
# Let's take a look at the updated text
data_clean = pd.DataFrame(data_df.transcript.apply(round1))
data_clean
```

| | transcript |
|---|---|
| **ali** | ladies and gentlemen please welcome to the stage ali wong hi hello welcome thank you thank you for coming hello hello we are gonna have to get thi... |
| **anthony** | thank you thank you thank you san francisco thank you so much so good to be here people were surprised when i told 'em i was gonna tape my special... |
| **bill** | all right thank you thank you very much thank you thank you thank you how are you what's going on thank you it's a pleasure to be here in the gre... |
| **bo** | bo what old macdonald had a farm e i e i o and on that farm he had a pig e i e i o here a snort there a old macdonald had a farm e i e i o this i... |
| **dave** | this is dave he tells dirty jokes for a living that stare is where most of his hard work happens it signifies a profound train of thought the alch... |
| **hasan** | what's up davis what's up i'm home i had to bring it back here netflix said "where do you want to do the special la chicago new york" i was like... |
| **jim** | ladies and gentlemen please welcome to the stage mr jim jefferies hello sit down sit down sit down sit down sit down thank you boston i appre... |
| **joe** | ladies and gentlemen welcome joe rogan what the fuck is going on san francisco thanks for coming i appreciate it god damn put your phone down ... |
| **john** | armed with boyish charm and a sharp wit the former "snl" writer john mulaney offers sly takes on marriage his beef with babies and the time he met... |
| **louis** | intro\nfade the music out let's roll hold there lights do the lights thank you thank you very much i appreciate that i don't necessarily agree wit... |
| **mike** | wow hey thank you thanks thank you guys hey seattle nice to see you look at this look at us we're here this is crazy it's insane so about five yea... |
| **ricky** | hello hello how you doing great thank you wow calm down shut the fuck up thank you what a lovely welcome i'm gonna try my hardest tonight you're t... |

From lab last class

# Representing Text Data

Document-Term Matrix

| ***Tokens*** | ***"great"*** | ***"worst"*** | ***"plot"*** | **....** |
|---|---|---|---|---|
| Unbelievably disappointing. Worst worst movie ever. | | 2 | | |
| Richly applied satire, great plot twists. | | | | |
| This is such a great comedy | | | | |
| It was pathetic. The worst part about it was the boxing scenes | | | | |
| ..... | | | | |

# Representing Text Data

Document-Term Matrix

**How to create one?**

1. Clean text
2. Tokenize text
3. Document-Term Matrix: Put into a matrix so a machine can read it

***Tokens***

| | **"great"** | **"worst"** | **"plot"** | **....** |
|---|---|---|---|---|
| Unbelievably disappointing. Worst worst movie ever. | | 2 | | |
| Richly applied satire, great plot twists. | | | | |
| This is such a great comedy | | | | |
| It was pathetic. The worst part about it was the boxing scenes | | | | |

# Representing Text Data

Document-Term Matrix

**Step 1: data cleaning**

All right, Petunia. Wish me luck out there. You will die on August 7th. 2037.

How would a computer read this? It needs to be **cleaned first**

Common data cleaning steps that we can do on this data (using regular expressions, **python re**)

- Remove punctuation
- Remove numbers
- Lowercase letters

all right petunia wish me luck out there you will die on august

# Representing Text Data

Document-Term Matrix

**Step 2: tokenize the data**

all right petunia wish me luck out there you will die on august **Cleaned data**

How would a computer read this? It needs to be cleaned first, **then tokenized**

**What's tokenization?** Split text into smaller pieces (or tokens). The most common token size is a word. It can also be sentence, bigrams, trigrams

| | |
|---|---|
| all | right |
| luck | out |
| die | …… |

Now you can do **some more cleaning** — removing stop words (which carry very little meaning)

# Representing Text Data

Document-Term Matrix

**Step 2: tokenize the data, followed by some more cleaning**

all right petunia wish me luck out there you will die on august

How would a computer read this? It needs to be cleaned first, **then tokenized**

right petunia wish much die august

**Bag of words model**: This representation of the text is called a **bag of words model.** Simple format that ignores order

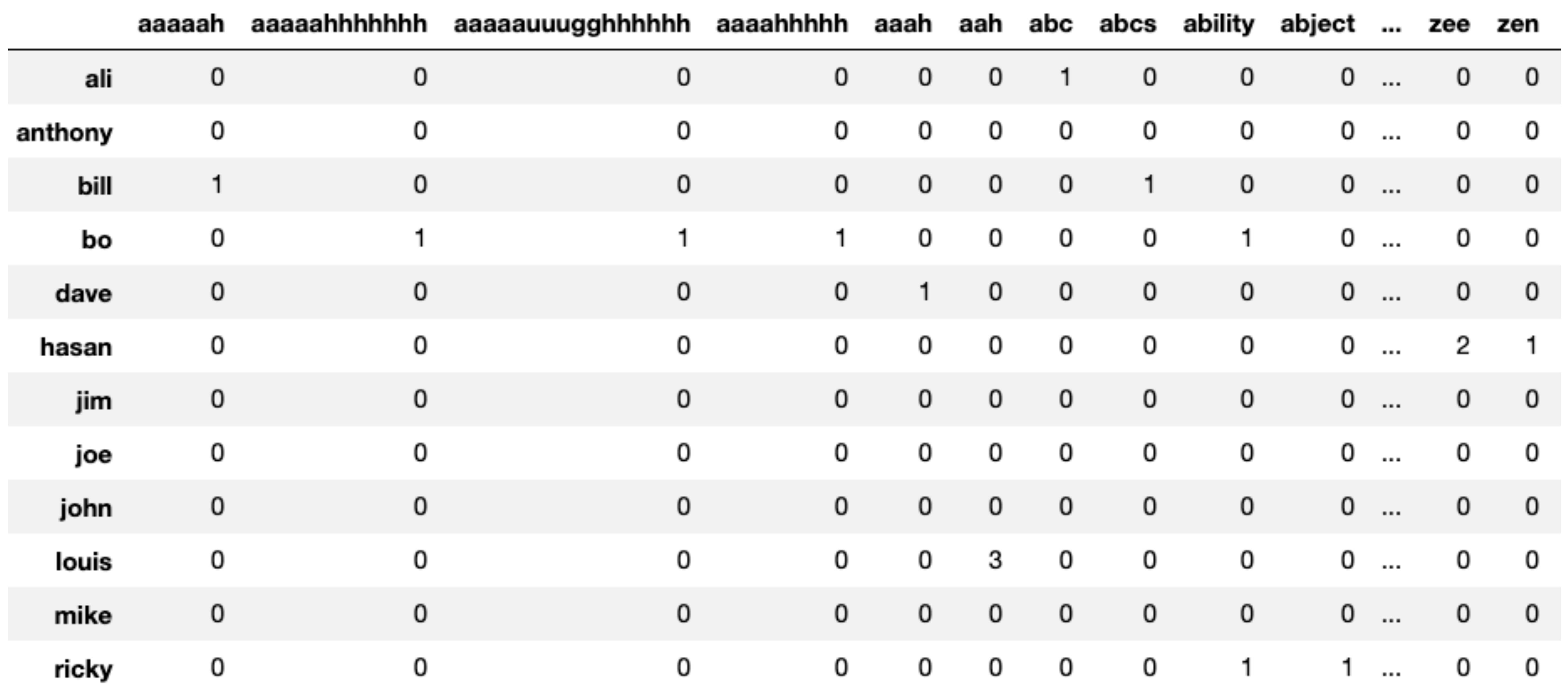
# Representing Text Data

Document-Term Matrix

**Step 3: put into a matrix**

all right petunia wish me luck out there you will die on august

How would a computer read this? It needs to be cleaned first, then tokenized, and **put into matrix**



| | aaaaah | aaaaahhhhhhh | aaaaauuugghhhhhh | aaaahhhhh | aaah | aah | abc | abcs | ability | abject | ... | zee | zen |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ali** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| **anthony** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **bill** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 |
| **bo** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 |
| **dave** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **hasan** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 2 | 1 |
| **jim** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **joe** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **john** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **louis** | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **mike** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| **ricky** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | ... | 0 | 0 |

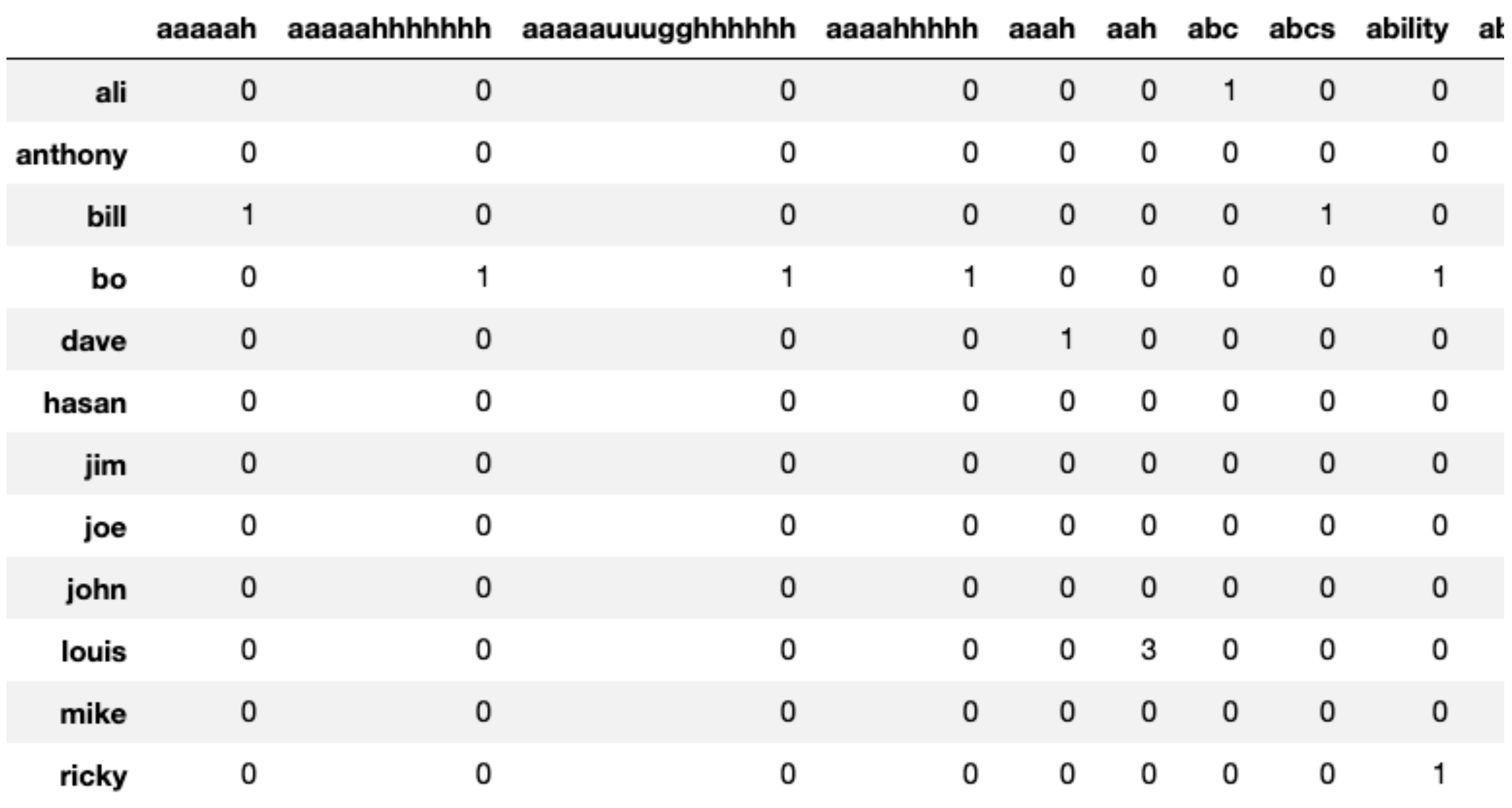
# Representing Text Data

Document term matrix:

- Each row is a different document

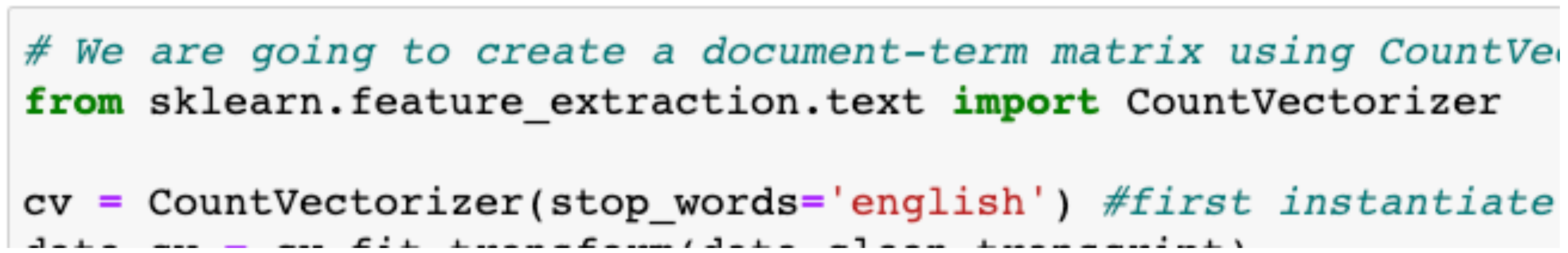
- Each column is a different term (usually words, can be bigrams etc.)

- The values are word counts



| | aaaaah | aaaaahhhhhhh | aaaaauuugghhhhhh | aaaahhhhh | aaah | aah | abc | abcs | ability | ab |
|---|---|---|---|---|---|---|---|---|---|---|
| ali | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| anthony | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| bill | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| bo | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |
| dave | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| hasan | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| jim | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| joe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| john | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| louis | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | |
| mike | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ricky | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |

**How do you do it programmatically?**



```
# We are going to create a document-term matrix using CountVe
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words='english') #first instantiate
```

## Leftover lab

## Getting simple structured information from text

Named Entity Recognition

POS

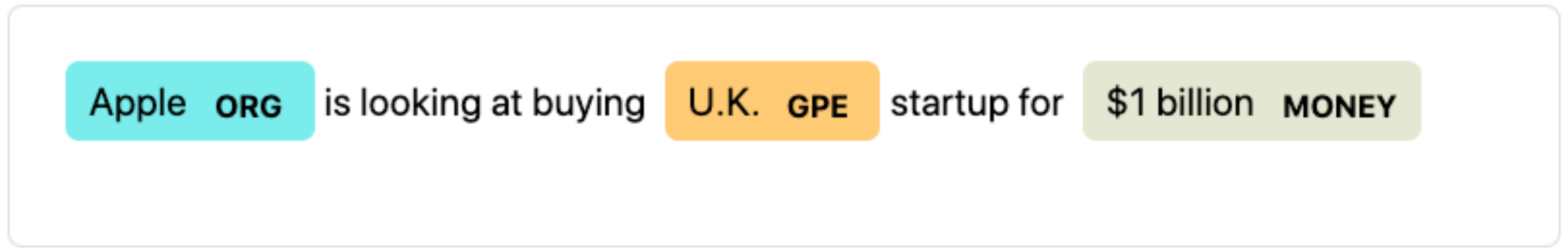
Goal of Information Extraction (IE) systems:

- Find an understand relevant parts of text
- Gather information from many pieces text

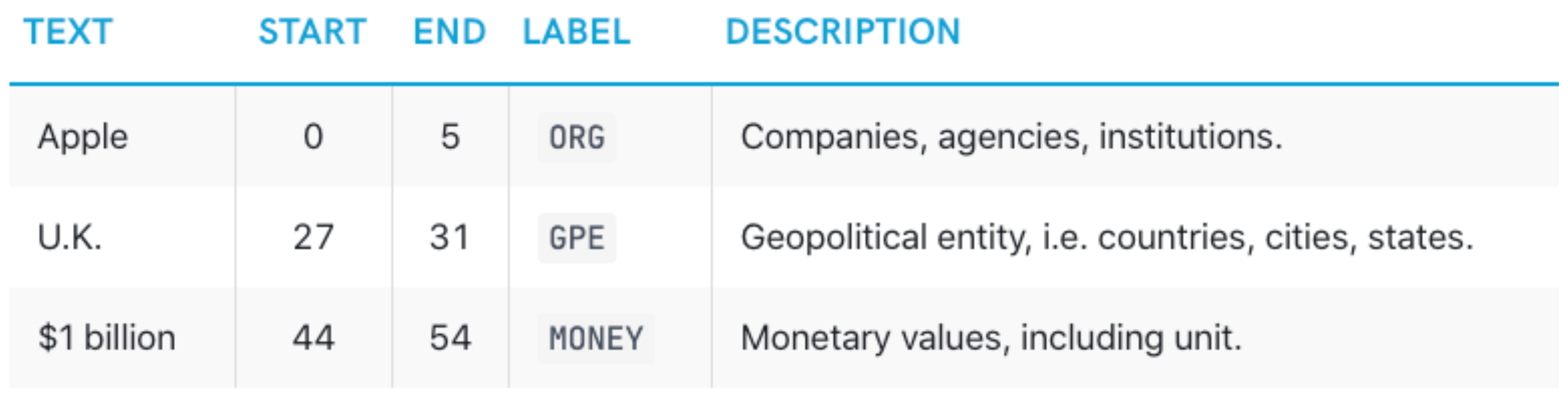
GOAL: Produce a structured representation of the relevant information

# Named Entity Recognition (NER)

A named entity is a "real-world object" that's assigned a name – for example, a person, a country,….


Apple ORG is looking at buying U.K. GPE startup for $1 billion MONEY

NER - find and classify entities in a text



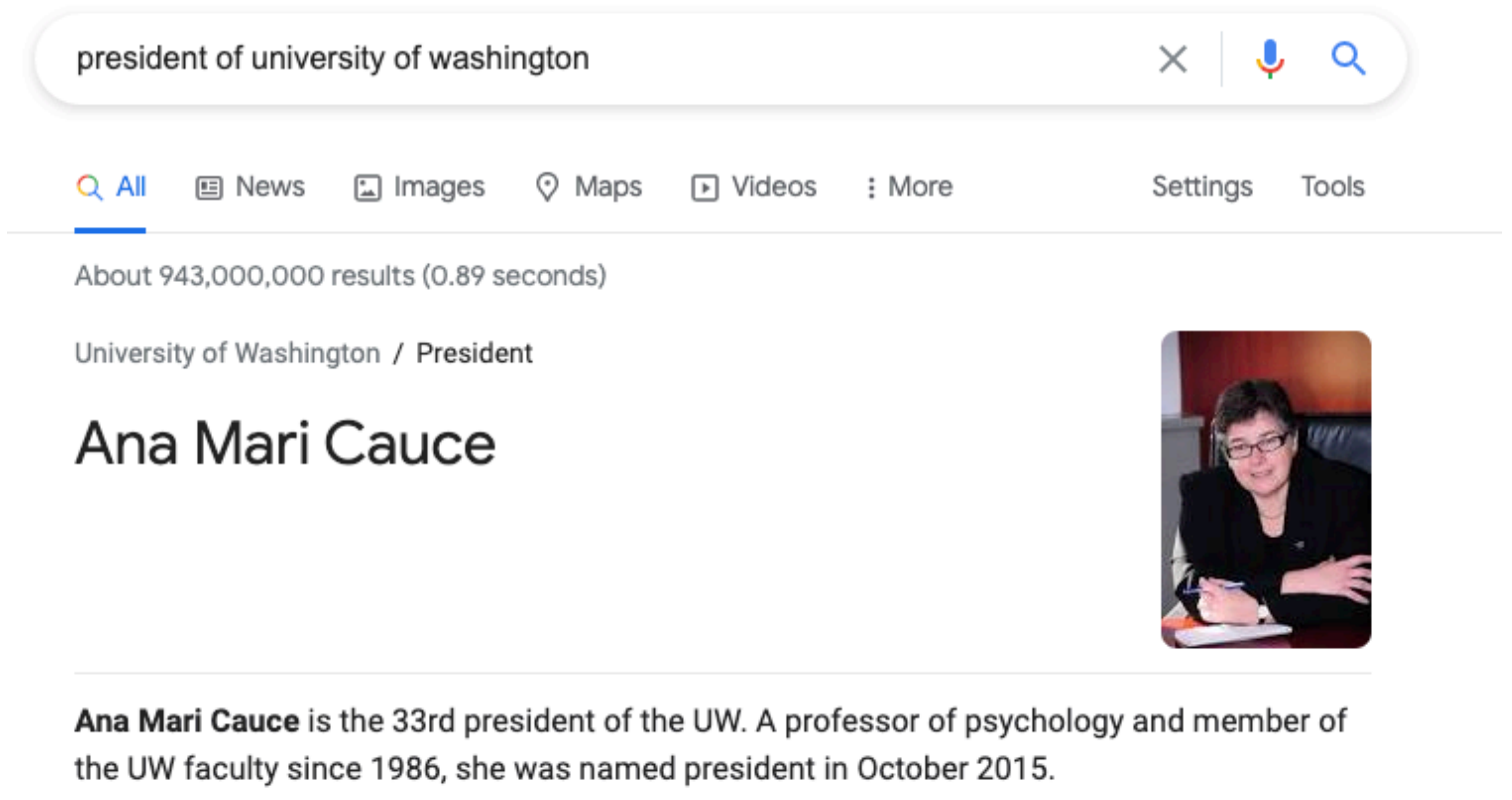
| TEXT | START | END | LABEL | DESCRIPTION |
|---|---|---|---|---|
| Apple | 0 | 5 | ORG | Companies, agencies, institutions. |
| U.K. | 27 | 31 | GPE | Geopolitical entity, i.e. countries, cities, states. |
| $1 billion | 44 | 54 | MONEY | Monetary values, including unit. |

*Hovy Text Analysis in python for social scientists, Chapter 2.4*

https://spacy.io/usage/linguistic-features#named-entities

# Named Entity Recognition (NER)

Lots of real-world application



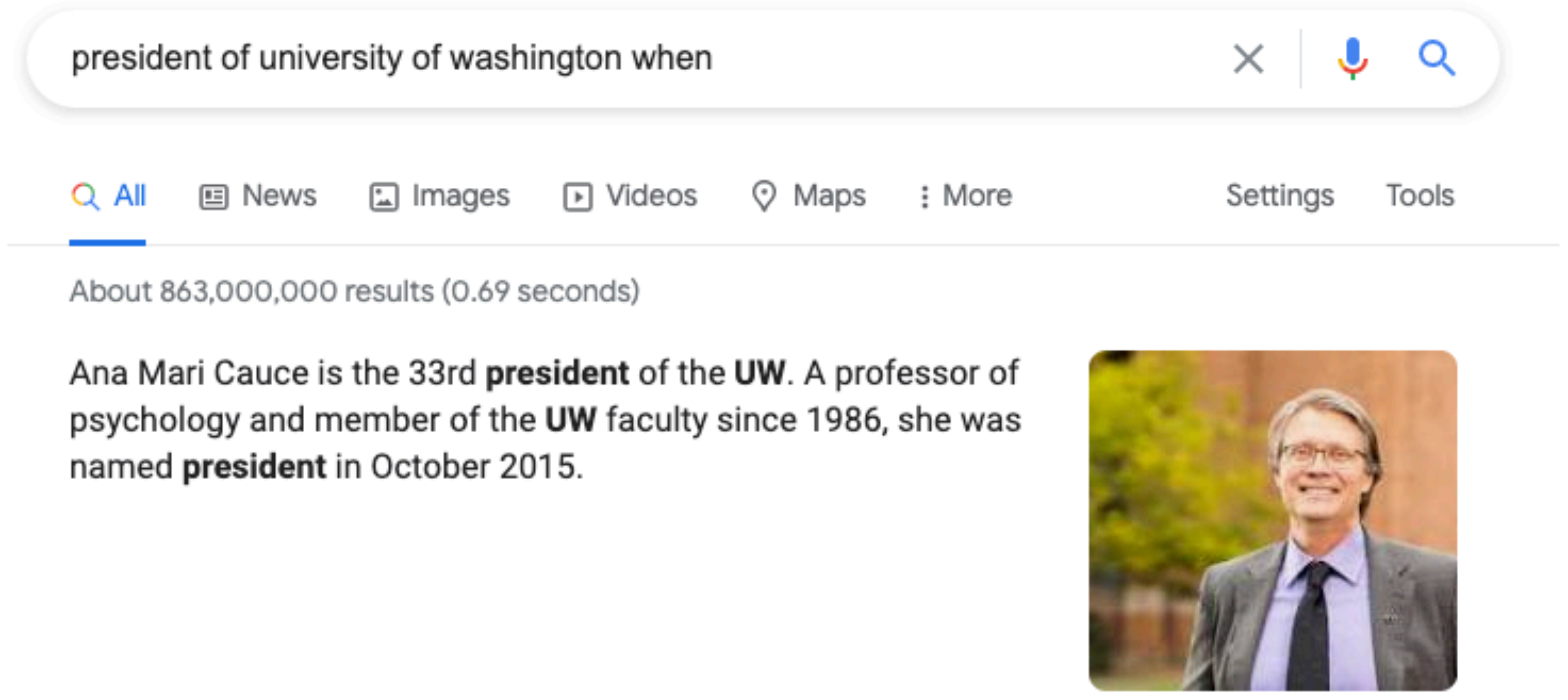
president of university of washington

About 943,000,000 results (0.89 seconds)

University of Washington / President

Ana Mari Cauce

**Ana Mari Cauce** is the 33rd president of the UW. A professor of psychology and member of the UW faculty since 1986, she was named president in October 2015.



president of university of washington when

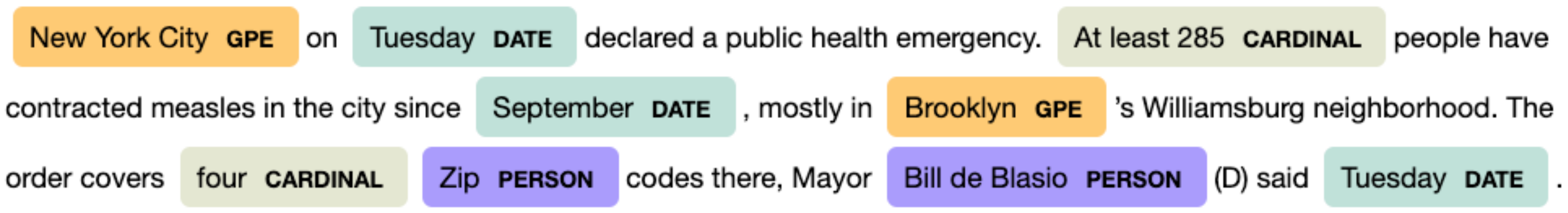
About 863,000,000 results (0.69 seconds)

Ana Mari Cauce is the 33rd **president** of the **UW**. A professor of psychology and member of the **UW** faculty since 1986, she was named **president** in October 2015.

# Named Entity Recognition (NER)

Can you identify the named entities in this text?

New York City on Tuesday declared a public health emergency. At least 285 people have contracted measles in the city since September, mostly in Brooklyn's Williamsburg neighborhood. The order covers four Zip codes there, Mayor Bill de Blasio (D) said Tuesday.



New York City GPE on Tuesday DATE declared a public health emergency. At least 285 CARDINAL people have contracted measles in the city since September DATE , mostly in Brooklyn GPE 's Williamsburg neighborhood. The order covers four CARDINAL Zip PERSON codes there, Mayor Bill de Blasio PERSON (D) said Tuesday DATE .

# Parts of Speech

e.g., nouns, verbs, and adjectives (most languages have more categories than just these three)

- ADJ: adjectives. They modify nouns to specify their properties. Examples: *awesome*, *red*, *boring*
- ADV: adverbs. They modify verbs, but also serve as question markers. Examples: *quietly*, *where*, *never*
- INTJ: interjections. Exclamations of some sort. Examples: *ouch*, *shhh*, *oi*

- NOUN: nouns. Entities in the world. Examples: *book*, *war*, *shark*
- PROPN: proper nouns. Names of entities, a subclass of nouns. Examples: *Rosa*, *Twitter*, *CNN*
- VERB: full verbs. Events in the world. Examples: *codes*, *submitted*, *succeed*

*Hovy Text Analysis in python for social scientists, Chapter 2.2*

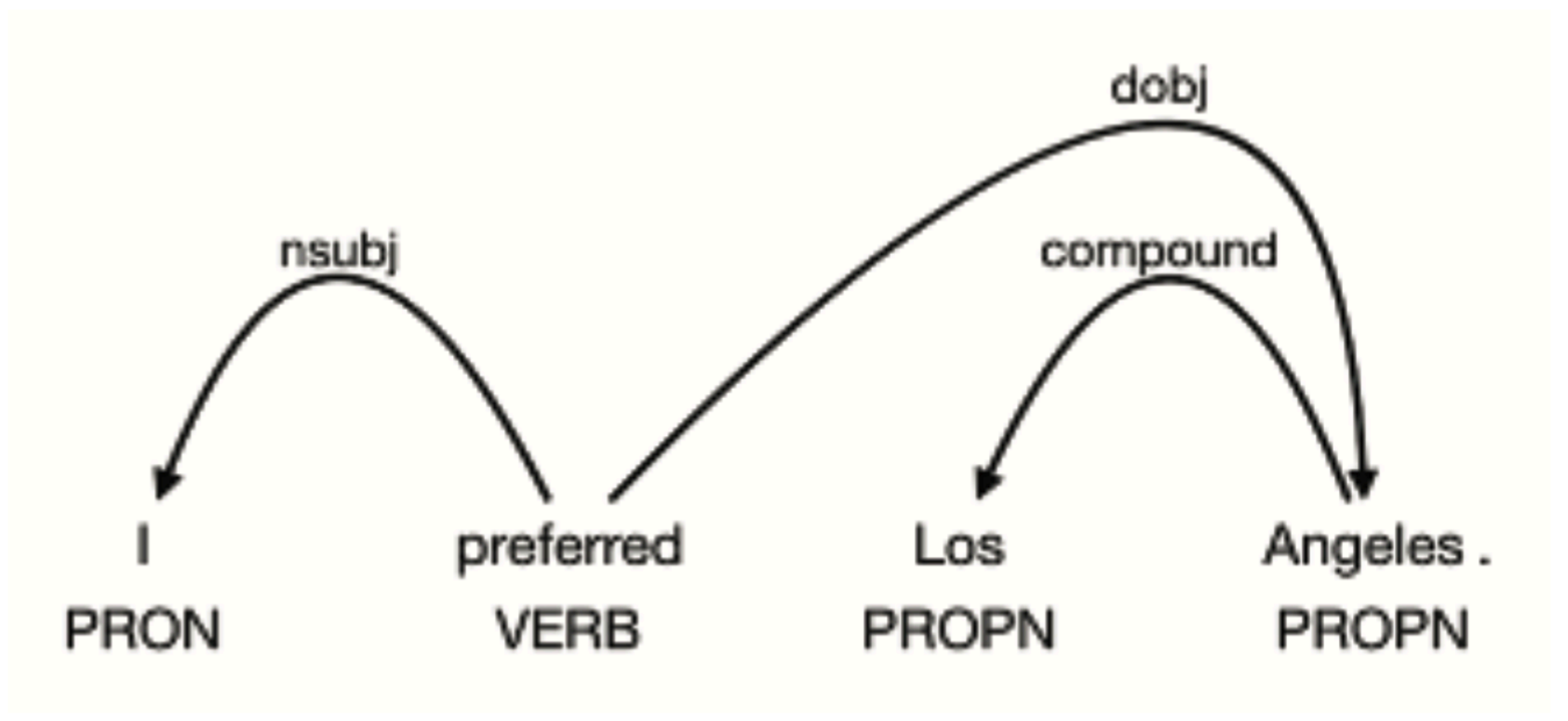
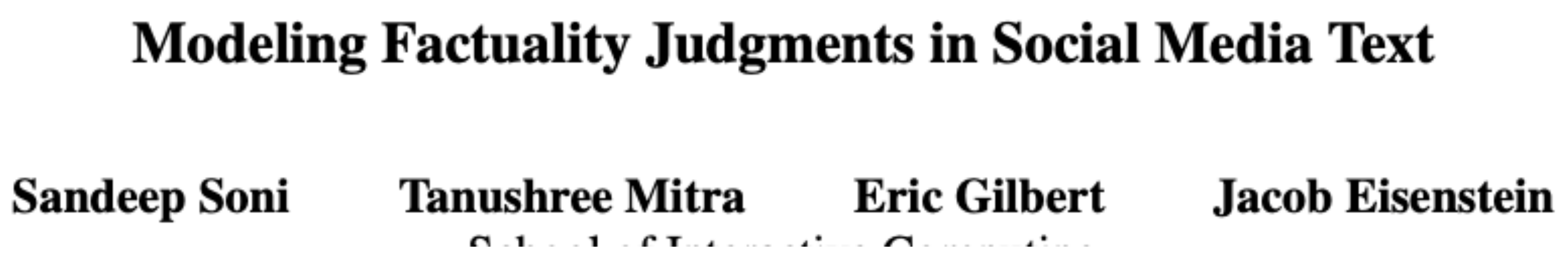


**Figure 1** Example of a dependency parse.

# Parts of Speech

POS in action in a DS project



**Modeling Factuality Judgments in Social Media Text**

**Sandeep Soni** **Tanushree Mitra** **Eric Gilbert** **Jacob Eisenstein**

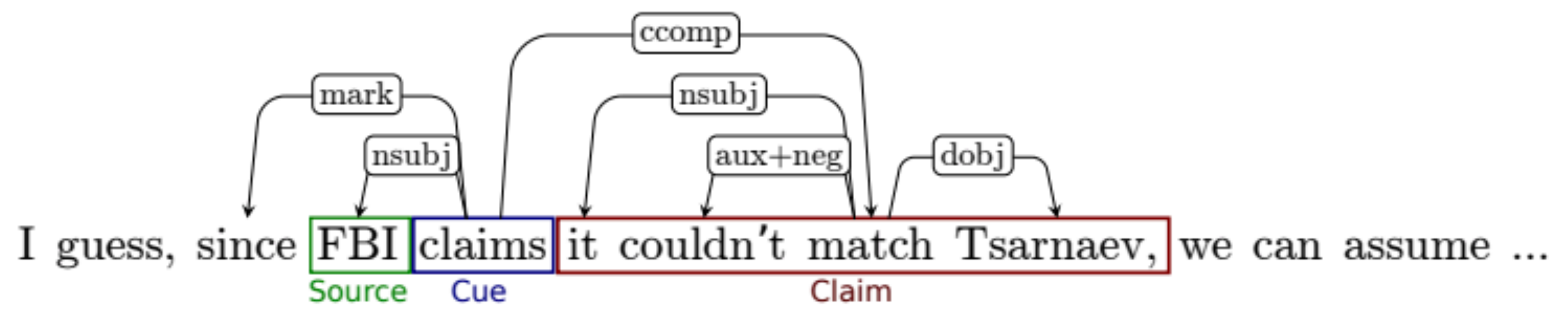
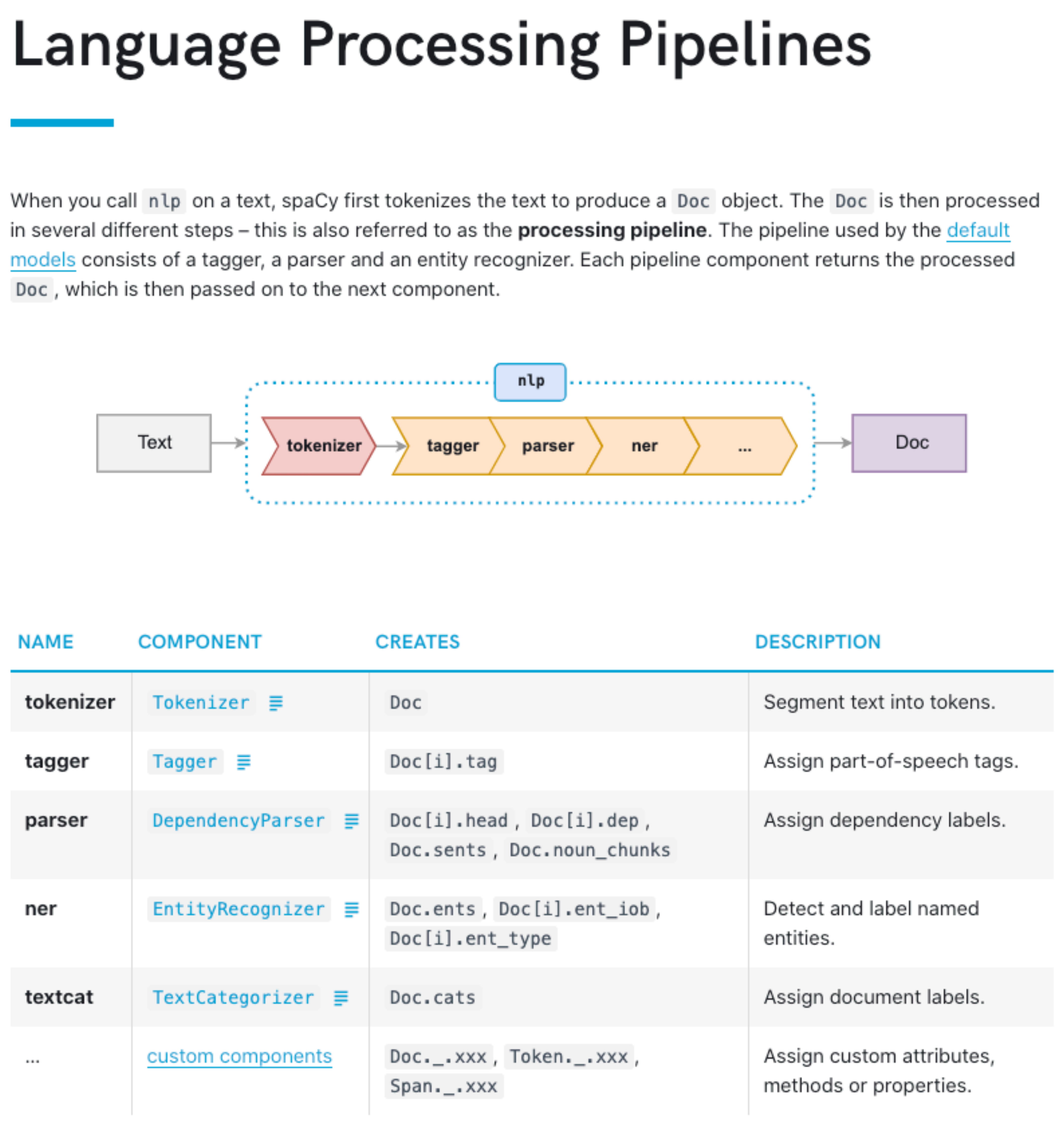

ccomp
mark
nsubj
nsubj
aux+neg
dobj
I guess, since FBI claims it couldn't match Tsarnaev, we can assume ...
Source
Cue
Claim

Figure 4: Dependency parse of an example message, with claim, source, and cue.

RQ: How do journalists mark quoted content as certain or uncertain, and how do readers interpret these signals?

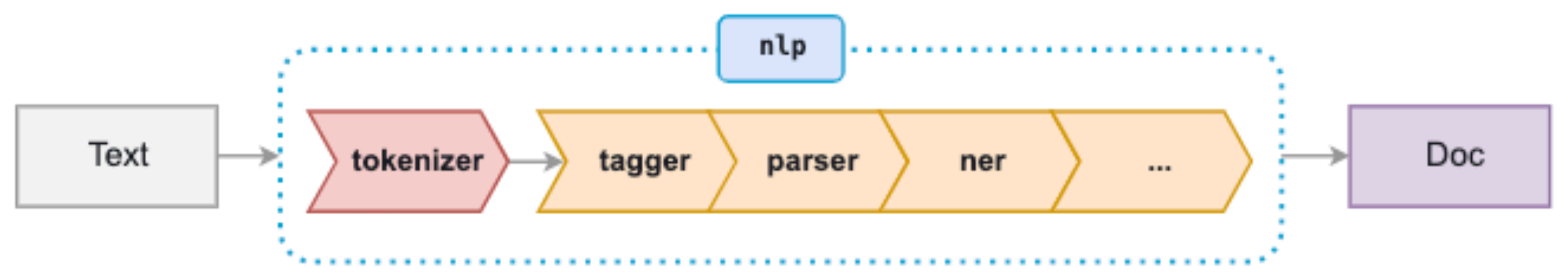
# Language processing with Spacy

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



## Language Processing Pipelines

When you call `nlp` on a text, spaCy first tokenizes the text to produce a `Doc` object. The `Doc` is then processed in several different steps – this is also referred to as the **processing pipeline**. The pipeline used by the default models consists of a tagger, a parser and an entity recognizer. Each pipeline component returns the processed `Doc`, which is then passed on to the next component.



| NAME | COMPONENT | CREATES | DESCRIPTION |
|---|---|---|---|
| **tokenizer** | `Tokenizer` | `Doc` | Segment text into tokens. |
| **tagger** | `Tagger` | `Doc[i].tag` | Assign part-of-speech tags. |
| **parser** | `DependencyParser` | `Doc[i].head`, `Doc[i].dep`, `Doc.sents`, `Doc.noun_chunks` | Assign dependency labels. |
| **ner** | `EntityRecognizer` | `Doc.ents`, `Doc[i].ent_iob`, `Doc[i].ent_type` | Detect and label named entities. |
| **textcat** | `TextCategorizer` | `Doc.cats` | Assign document labels. |
| ... | custom components | `Doc._.xxx`, `Token._.xxx`, `Span._.xxx` | Assign custom attributes, methods or properties. |

# Part-of-speech tagging NEEDS MODEL

After tokenization, spaCy can **parse** and **tag** a given `Doc`. This is where the trained pipeline and its statistical models come in, which enable spaCy to **make predictions** of which tag or label most likely applies in this context. A trained component includes binary data that is produced by showing a system enough examples for it to make predictions that generalize across the language – for example, a word following “the” in English is most likely a noun.

Linguistic annotations are available as `Token` attributes. Like many NLP libraries, spaCy **encodes all strings to hash values** to reduce memory usage and improve efficiency. So to get the readable string representation of an attribute, we need to add an underscore `_` to its name:

**Editable Code** spaCy v3.0 · Python 3 · via Binder

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
            token.shape_, token.is_alpha, token.is_stop)
```

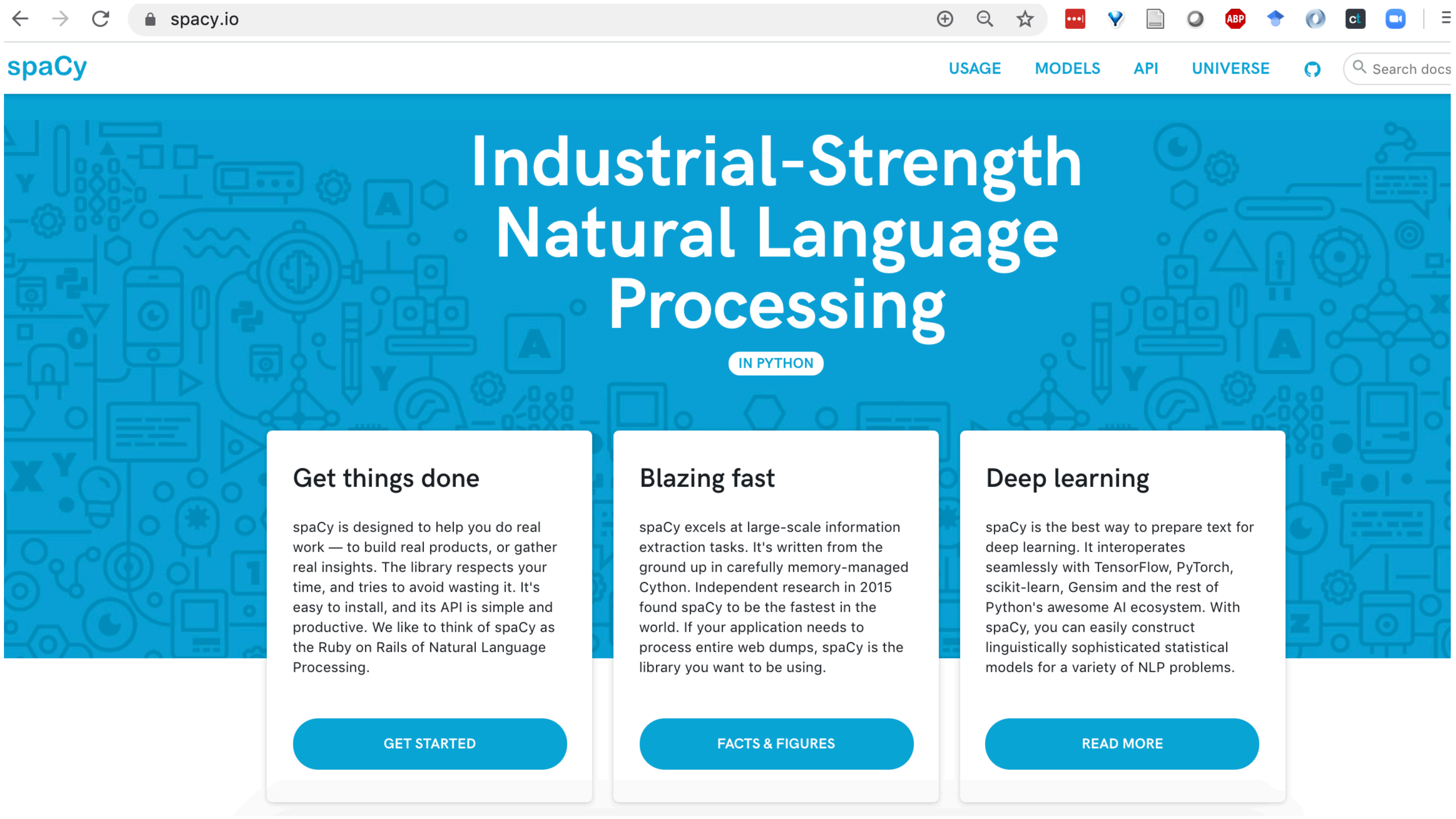
RUN

| TEXT | LEMMA | POS | TAG | DEP | SHAPE | ALPHA | STOP |
|---|---|---|---|---|---|---|---|
| Apple | apple | PROPN | NNP | nsubj | Xxxxx | True | False |
| is | be | AUX | VBZ | aux | xx | True | True |
| looking | look | VERB | VBG | ROOT | xxxx | True | False |
| at | at | ADP | IN | prep | xx | True | True |
| buying | buy | VERB | VBG | pcomp | xxxx | True | False |
| U.K. | u.k. | PROPN | NNP | compound | X.X. | False | False |

**Text:** The original word text.
**Lemma:** The base form of the word.
**POS:** The simple UPOS part-of-speech tag.
**Tag:** The detailed part-of-speech tag.
**Dep:** Syntactic dependency, i.e. the relation between tokens.
**Shape:** The word shape – capitalization, punctuation, digits.
**is alpha:** Is the token an alpha character?
**is stop:** Is the token part of a stop list, i.e. the most common words of the language?

## BREAK

Be back at 9 :35



Industrial-Strength Natural Language Processing

IN PYTHON

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

GET STARTED

Blazing fast

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spaCy to be the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

FACTS & FIGURES

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

READ MORE

## Lab

## Project Group

Work on this sheet as a group: https://docs.google.com/document/d/1JpSFaH5x7_BOCrzOkhoUgzBAWs3JBeQeHDv_f-Fs1IY/edit?usp=sharing

## PS2 discussion