We'll focus on text classification

Example: sentiment analysis

- Input: text of review
- Output: class of sentiment
  - e.g. 2 classes: positive vs negative
- Positive example:
  - The hotel is really beautiful. Very nice and helpful service at the front desk.
- Negative example:
  - We had problems to get the Wi-Fi working. The pool area was occupied with young party animals. So the area wasn't fun for us.

Text preprocessing

What is text?

You can think of text as a sequence of

- Characters
- **Words**
- Phrases and named entities
- Sentences
- Paragraphs
- …

What is a word?

It seems natural to think of a text as a sequence of words

- A word is a meaningful sequence of characters

How to find the boundaries of words?

- In English we can split a sentence by spaces or punctuation

Input: Friends, Romans, Countrymen, lend me your ears;

Output:

Friends	Romans	Countrymen	lend	me	your	ears

- In German there are compound words which are written without spaces
  - "Rechtsschutzversicherungsgesellschaften" stands for "insurance companies which provide legal protection"
- In Japanese there are no spaces at all!
  - Butyoucanstillreaditright?

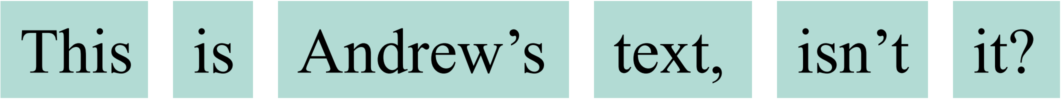
Tokenization

Tokenization is a process that splits an input sequence into so-called tokens

- You can think of a token as a useful unit for semantic processing
- Can be a word, sentence, paragraph, etc.

An example of simple whitespace tokenizer

- nltk.tokenize.WhitespaceTokenizer



This | is | Andrew’s | text, | isn’t | it?

- Problem: “it” and “it?” are different tokens with same meaning

Tokenization

Let’s try to also split by punctuation

- nltk.tokenize.WordPunctTokenizer

This	is	Andrew	’	s	text	,	isn	’	t	it	?

- Problem: “s”, “isn”, “t” are not very meaningful

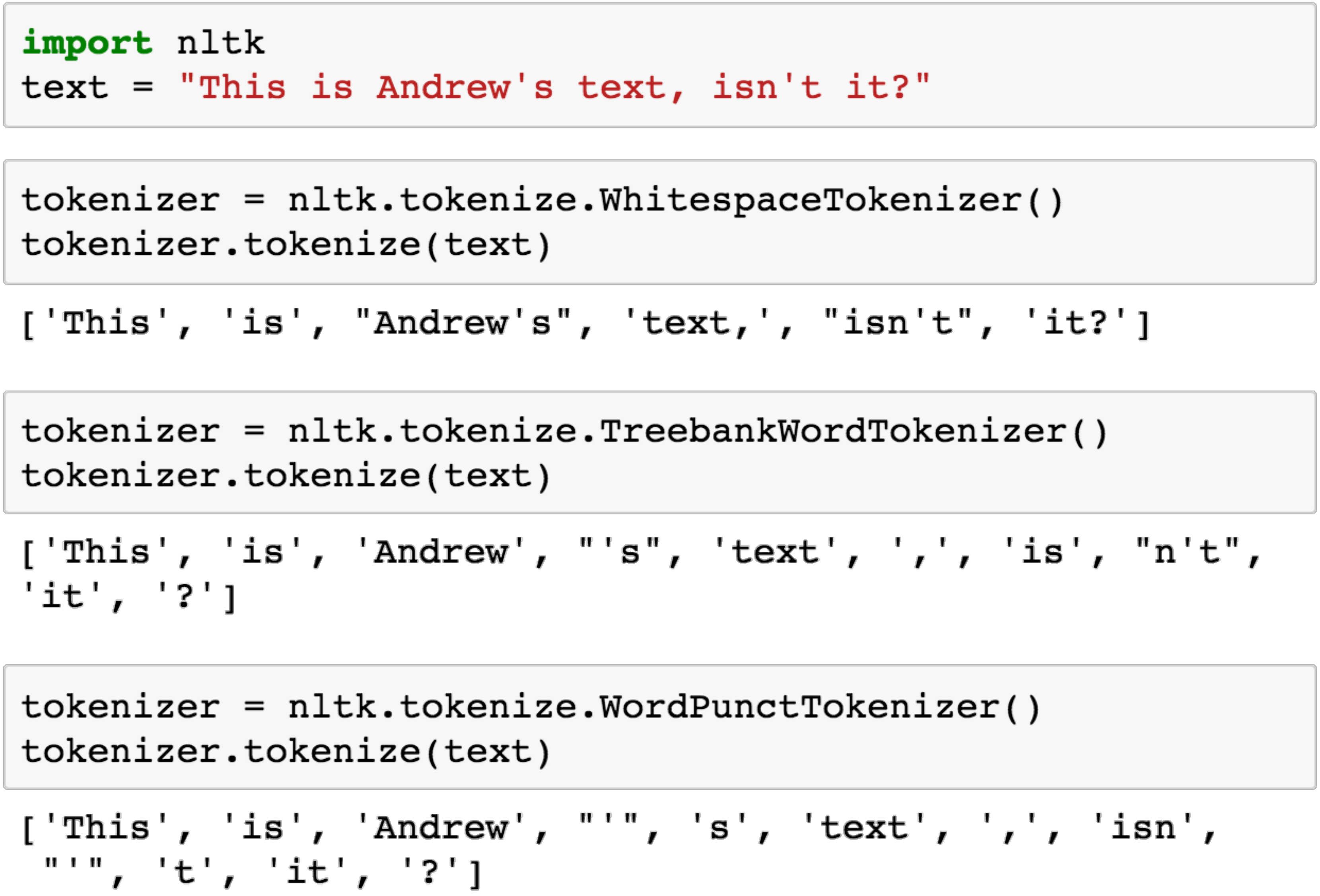
We can come up with a set of rules

- nltk.tokenize.TreebankWordTokenizer

This	is	Andrew	’s	text	,	is	n’t	it	?

- “’s” and “n’t” are more meaningful for processing

Python tokenization example



```
import nltk
text = "This is Andrew's text, isn't it?"
```

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', "Andrew's", 'text,', "isn't", 'it?']
```

```
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'s", 'text', ',', 'is', "n't",
'it', '?']
```

```
tokenizer = nltk.tokenize.WordPunctTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'", 's', 'text', ',', 'isn',
 "'", 't', 'it', '?']
```

http://text-processing.com/demo/tokenize/

Token normalization

We may want the same token for different forms of the word

- wolf, wolves → wolf
- talk, talks → talk

Stemming

- A process of removing and replacing suffixes to get to the root form of the word, which is called the **stem**
- Usually refers to heuristics that chop off suffixes

Lemmatization

- Usually refers to doing things properly with the use of a vocabulary and morphological analysis
- Returns the base or dictionary form of a word, which is known as the **lemma**

Stemming example

Porter's stemmer

- 5 heuristic phases of word reductions, applied sequentially
- Example of phase 1 rules:

Rule	**Example**
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

- nltk.stem.PorterStemmer
- Examples:
  - feet → feet   cats → cat
  - wolves → wolv   talked → talk
- Problem: fails on irregular forms, produces non-words

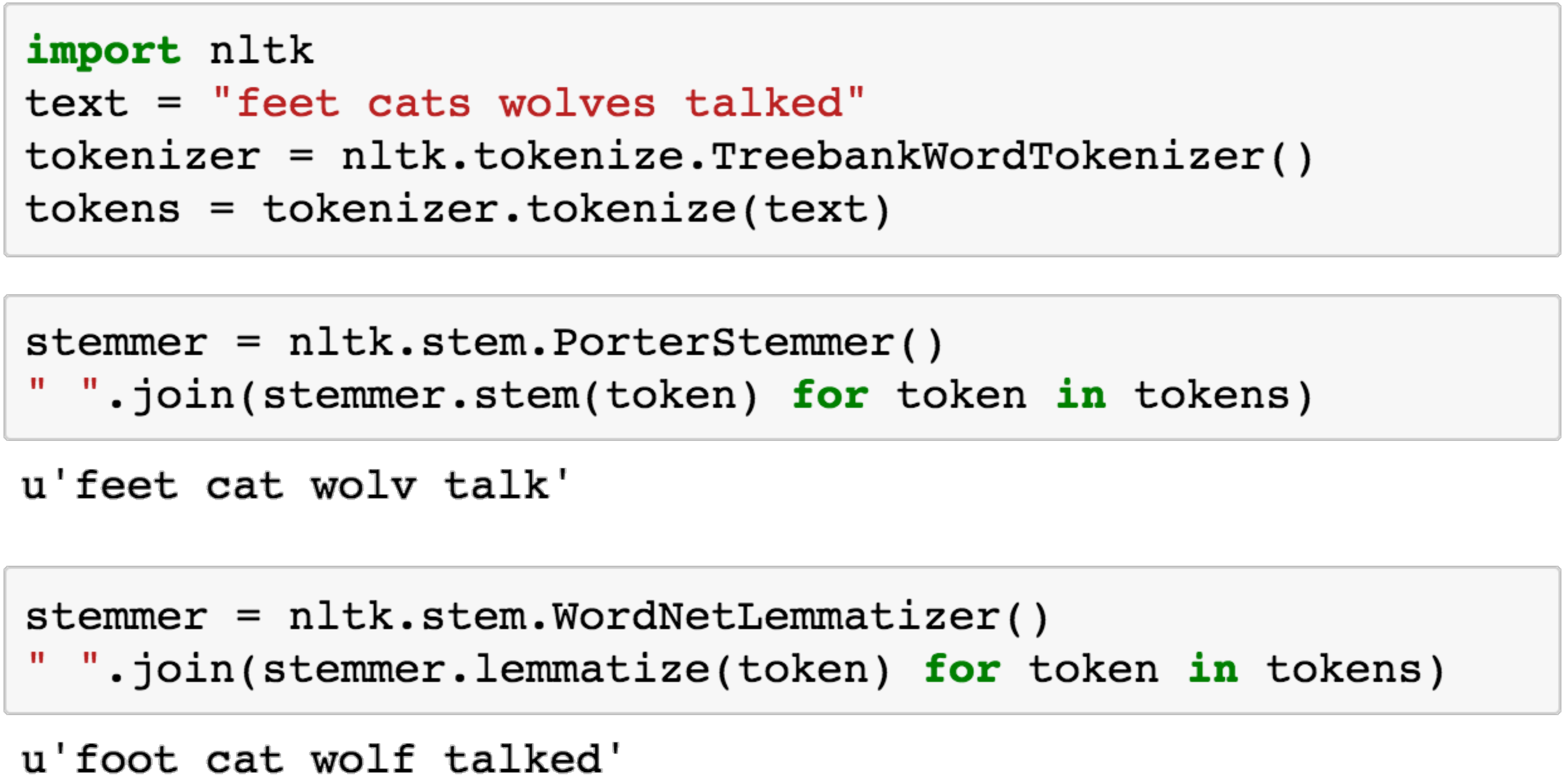
Lemmatization example

WordNet lemmatizer

- Uses the WordNet Database to lookup lemmas
- nltk.stem.WordNetLemmatizer
- Examples:
  - feet → foot cats → cat
  - wolves → wolf talked → talked
- Problems: not all forms are reduced

- Takeaway: we need to try stemming or lemmatization and choose best for our task

Python stemming example



```
import nltk
text = "feet cats wolves talked"
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)
```

```
stemmer = nltk.stem.PorterStemmer()
" ".join(stemmer.stem(token) for token in tokens)
```

```
u'feet cat wolv talk'
```

```
stemmer = nltk.stem.WordNetLemmatizer()
" ".join(stemmer.lemmatize(token) for token in tokens)
```

```
u'foot cat wolf talked'
```

Further normalization

Normalizing capital letters

- Us, us → us (if both are pronoun)
- us, US (could be pronoun and country)
- We can use heuristics:
  - lowercasing the beginning of the sentence
  - lowercasing words in titles
  - leave mid-sentence words as they are
- Or we can use machine learning to retrieve true casing → hard

Acronyms

- eta, e.t.a., E.T.A. → E.T.A.
- We can write a bunch of regular expressions → hard

Summary

- We can think of text as a sequence of tokens
- Tokenization is a process of extracting those tokens
- We can normalize tokens using stemming or lemmatization
- We can also normalize casing and acronyms
- In the next video we will transform extracted tokens into features for our model