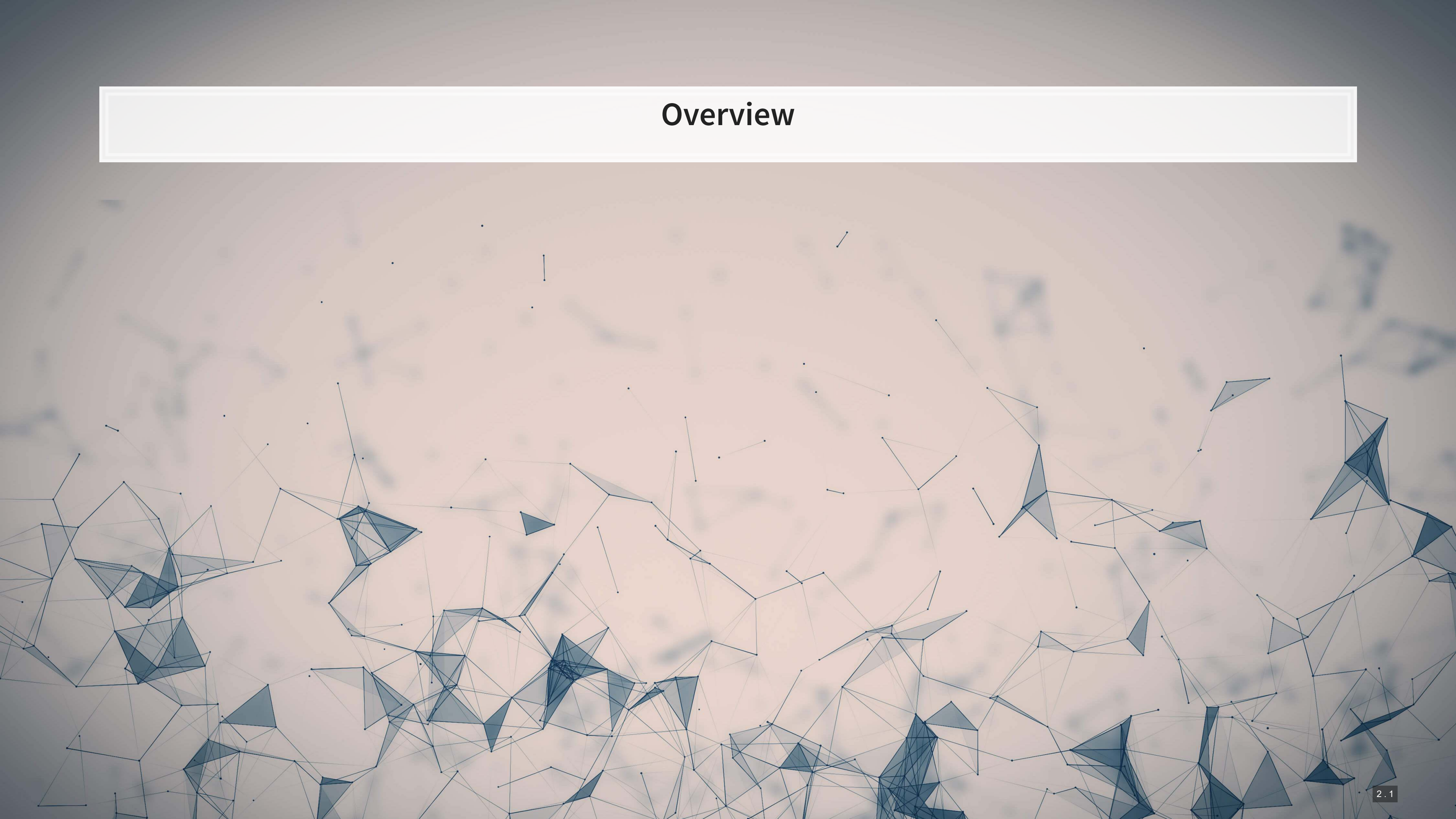


ML for SS: Linguistic analysis

Session 5

Dr. Richard M. Crowley
rcrowley@smu.edu.sg
<http://rmc.link/>

Overview



Papers

Jurafsky et al. 2014

- A fairly simple paper examining linguistic patterns in consumer reviews of restaurants

Hope, Hu and Lu (2016)

- A use of Named Entity Recognition (NER) in a fairly simple way.
 - Names of persons, locations, or organizations
 - Percentages and monetary values
 - Times and dates

Technical Discussion: Linguistics

Python

- [NLTK](#) for standard/statistical approaches
- [SpaCy](#) for machine learning pipelines
- [Stanza](#) for Stanford NLP methods
 - They have some interesting models for narrower uses
- [BeautifulSoup](#) for HTML parsing

R

- Call python's SpaCy package from R using `spacyr`
- `rvest` for HTML parsing

Python is generally a bit stronger for these topics, unless your data is clean and fairly small.

There is a fully worked out solution for using python, data and dictionaries are on eLearn.

Main application: Analyzing Wall Street Journal articles

- On eLearn you will find a full issue of the WSJ in text format

Linguistic models using NLTK and SpaCy

- Tokenization and break documents into smaller chunks
- Part of speech tagging (grammar)
- Dependency parsing
- Named Entity Recognition (NER)
- Lemmatization

Using NLP parsers

NLTK

- [NLTK](#) stands for Natural Language Toolkit
- It provides a bunch of handy things for text analytics
 1. Corpora that are used in research and algorithm development
 - Tagged corpora are particularly valuable
 2. Models for things like dependency parsing
 3. Useful functions for working with text

Setting up NLTK

- When using a resource from [NLTK](#), we will often have install needed datasets

Useful parts to download using `nltk.download()`

- `'punkt'`: Used for tokenizing words (splitting apart words in a document)
- `'brown'`: A corpus that contains part of speech information based on news articles
 - Can be used to train a part of speech tagger
- `'averaged_perceptron_tagger'`: An ML model for applying part of speech tags
- `'universal_tagset'`: If you only need simple part of speech labels, this is easier to work with
- `'treebank'`: Like `'brown'` above, but based on WSJ

Tokenizing

```
text = 'A U.S. appeals court will hear oral arguments today in a suit by Verizon challenging FCC "net-neutrality" rules.'  
tokens = nltk.tokenize.word_tokenize(text)  
print(tokens)
```

```
## ['A', 'U.S.', 'appeals', 'court', 'will', 'hear', 'oral', 'arguments', 'today', 'in', 'a', 'suit', 'by', 'Verizon', 'challe
```

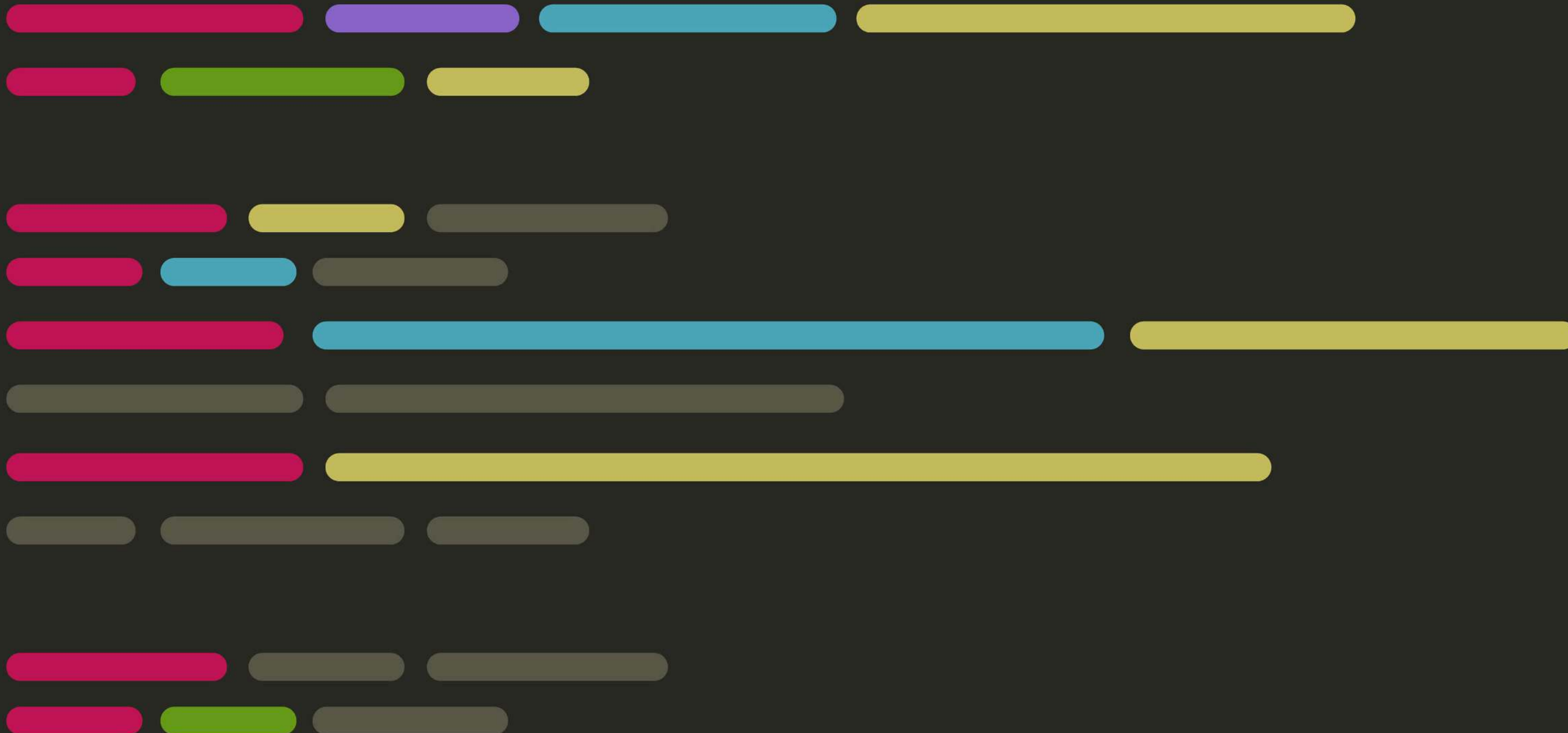
Part of Speech tagging

```
text = 'A U.S. appeals court will hear oral arguments today in a suit by Verizon challenging FCC "net-neutrality" rules.'  
tokens = nltk.tokenize.word_tokenize(text)  
  
# Requires: nltk.download('brown')  
brown_news_tagged = nltk.corpus.brown.tagged_sents(categories='news', tagset='brown')  
pos_tagger = nltk.UnigramTagger(brown_news_tagged)  
  
tagged = pos_tagger.tag(tokens)  
print(tagged)
```

```
## [('A', 'AT'), ('U.S.', 'NP'), ('appeals', 'NNS'), ('court', 'NN'), ('will', 'MD'), ('hear', 'VB'), ('oral', None), ('argume
```

More details included in the Python file

- Using other PoS taggers (perceptron and Bigram)
- Extracting parts of speech from a full corpus
- Building a multi-level PoS tagger



SpaCy

- SpaCy provides a machine-learning based approach to many of the things NLTK does
- SpaCy is also perhaps a bit more user-friendly

```
import spacy

# python -m spacy download en_core_web_sm
nlp = spacy.load("en_core_web_sm")
# pipes enabled by default: tok2vec, tagger, parser, ner, attribute_ruler, lemmatizer]

doc = nlp(text)
```



Parse trees in SpaCy

- SpaCy has a visualization module called displaCy
- With this, we can quickly see how a sentence is structured
- To run it in a Jupyter notebook, use the below code:

```
sent = nlp("""Citi intends to release a revised Quarterly Financial Data  
Supplement reflecting this realignment prior to the release of first quarter of  
2014 earnings information.""")  
spacy.displacy.render(sent, style="dep", jupyter=True, options={'compact':True})
```

Take a look at the code file to see the output

NER: Named Entity Recognition

- During the `nlp()` call earlier, spaCy automatically did named entity recognition'
- Using an ML algorithm + the dependency tree, it tries to determine any proper nouns in the document
 - It also tries to label them
- You can visualize these as well with `displayCy`

```
spacy.displacy.render(sent, style="ent", jupyter=True)
```



Citi ORG intends to release a revised Quarterly Financial Data Supplement ORG reflecting this realignment prior to the release of first quarter DATE of 2014 DATE earnings information.

More details included in the Python file

- Using `nlp.pipe()` instead of `nlp()`
 - Allows you to apply a process to a corpus all at once (as a generator)
- Sentence boundary detection
- PoS tagging in SpaCy
- Lemmatization
- Extracting all entities from a corpus

Parsing HTML

Overview

- As this part is code-heavy, we will do it in Jupyter
- The main idea is:
 1. Grab the main page of the website using `requests`
 2. Structure it with `beautifulsoup4` so we can traverse the page
 3. Grab the links to and names of standards, along with the publication years
 4. Traverse the links
 5. Extract the pdf locations from the traversed pages
 6. Grab the pdf files

Addendum: Using R

- HTML files
 - You can load from a URL using `httr` or `RCurl`
 - You can use `XML` or `rvest` to parse out specific pieces of html files
- JSON files
 - You can process JSON data using `jsonlite`
- PDF files
 - Use `pdftools` to extract text into a vector of pages of text
 - Use `tabulizer` to extract tables straight from PDF files!
 - This is very painful to code by hand without this package
 - The package itself is a bit difficult to install, requiring Java and `rJava`, though

Conclusion



Wrap-up

Linguistics is largely handled by importing specialized libraries

- NLTK for traditional measures
- SpaCy for more powerful, ML-based measures
- Stanza for Stanford NLP measures

Easy to calculate many different measures, such as grammar/parts of speech or entities (NER)

Packages used for these slides

Python

- bs4
- nltk
- numpy
- requests
- spacy

R

- knitr
- reticulate
- revealjs

References

- Hope, Ole-Kristian, Danqi Hu, and Hai Lu. “The benefits of specific risk-factor disclosures.” *Review of Accounting Studies* 21, no. 4 (2016): 1005-1045.
- Jurafsky, Dan, Victor Chahuneau, Bryan R. Routledge, and Noah A. Smith. “Narrative framing of consumer sentiment in online restaurant reviews.” *First Monday* (2014).