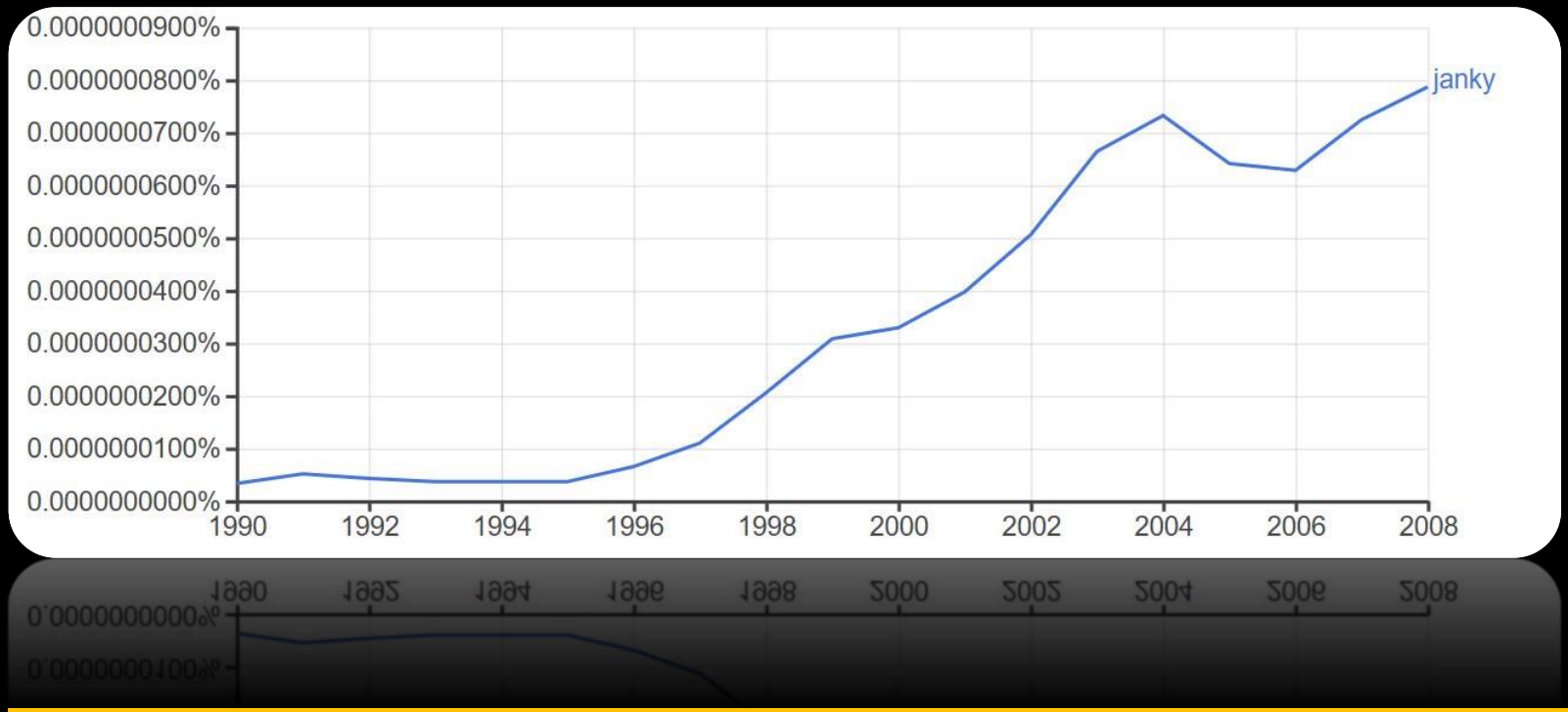
Text analysis

≈ Natural Language Processing, or "*How to do cool stuff with words.*"

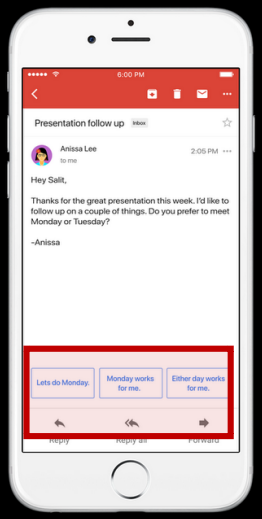


Emily Rae Sabo Data Camp | June 19, 2019

2 objectives for this session:

- ✓ What is NLP /Text Analysis and why would I use it?
- ✓ What tools are out there for me to use?

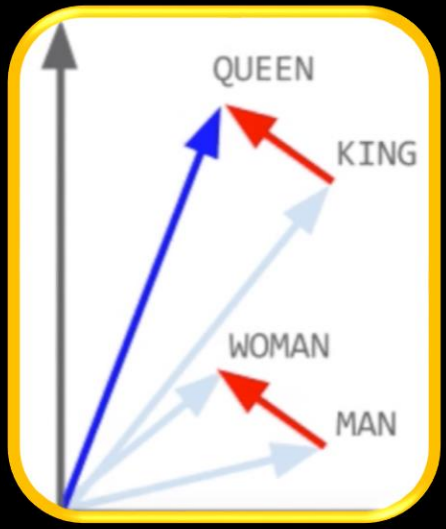
What is NLP used for?



Predicting language

Translating language

Finding patterns in language



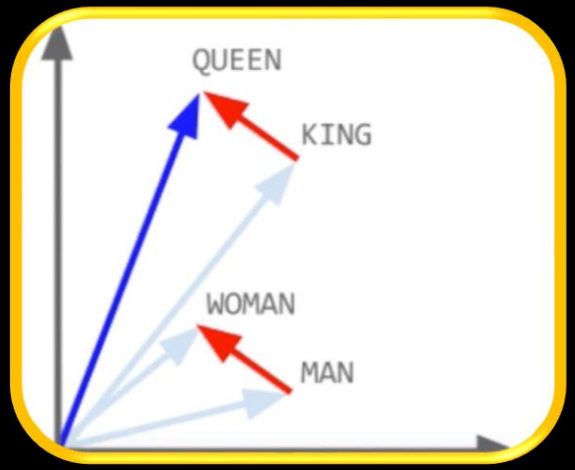
Measuring meaning in language

How to apply Text Analysis

Finding patterns in language

- Change over time with Google Ngram
- Topic Modeling with Gensim, NLTK
- String matching and token extraction with RegEx

Measuring meaning in language

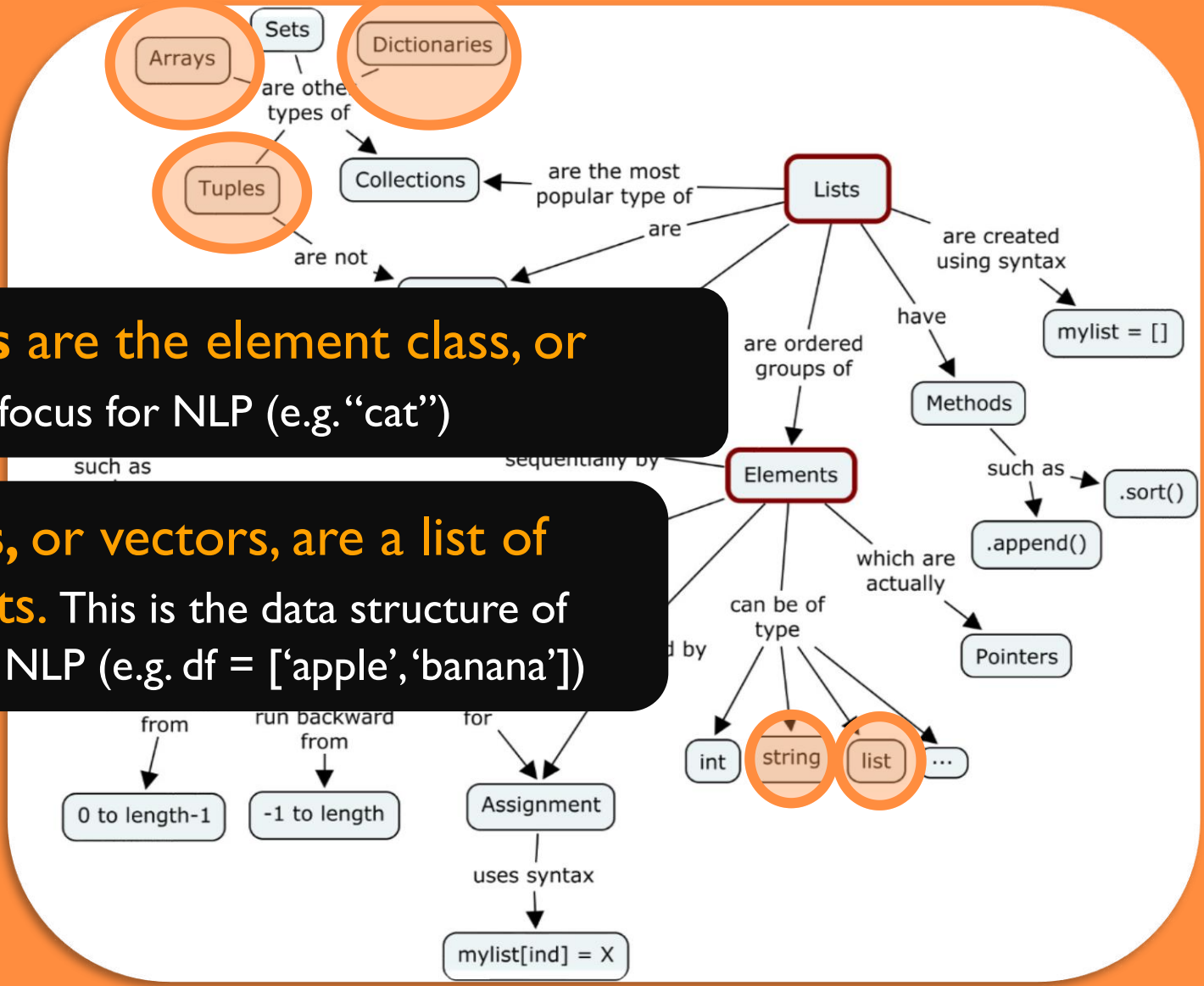


- Vector space modeling with word-embedded vectors like Word2Vec in *Gensim* or GloVe in *SpaCy*

Python's basic elements & data structures

Strings are the element class, or type, of focus for NLP (e.g. "cat")

Arrays, or vectors, are a list of elements. This is the data structure of focus for NLP (e.g. df = ['apple', 'banana'])



4 TAKE-AWAYS

1. **Google Ngram Viewer** is a quick 'n dirty tool for measuring word frequency change over time.
2. **Topic modeling** is a dimensionality reduction technique used to reveal "topics" in a document.
3. **Regular Expressions (RegEx) is the syntax you use** to do string matching, text cleaning, and token extraction.
4. **Word-embedded vectors** are decomposed matrices from a huge word matrix that tells you about word meaning.

How to measure changes in word frequency over time?

Google Ngram Viewer

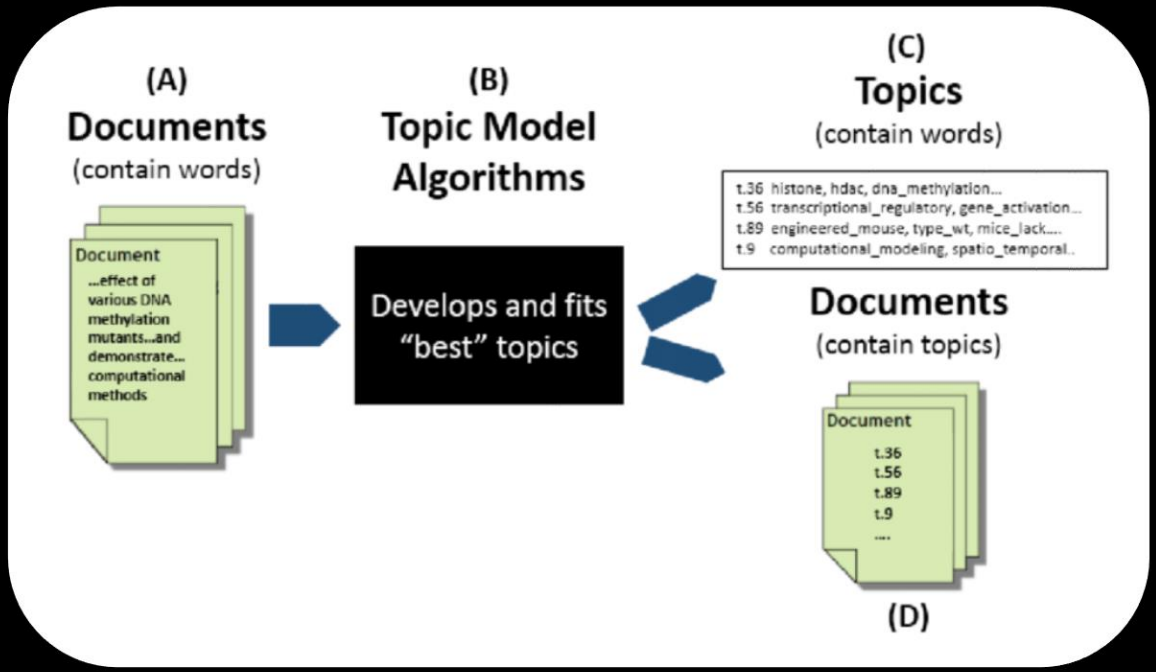
- The founding tool of "culturomics"
- Advantages vs. limitations?
- Share one way you could imagine using this in your research.
- Go and play!
  - https://books.google.com/ngrams
  - https://books.google.com/ngrams/info

What is Topic Modeling?

It is an **unsupervised approach** used for finding and observing the bunch of words (called "topics") in large clusters of texts."
Bansal (2016)

Click here for a good starter on Topic Modeling in Python with NLTK and Gensim

- It's a dimensionality reduction technique used to discover the hidden or abract "topics" that occur in a document or collection of documents.
- Techniques you may have heard of before: LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation)



What are Regular Expressions, or RegEx?

```
\d{3}[-.] \d{3}[-.] \d{4}

M(r|s|rs)\.?\s[A-Z]\w*
```

What are Regular Expressions, or RegEx?

- Literal s vs. meta ^ characters (e.g. ^s)
- Wildcards s . . . .
- Character sets [a-z]
- Character groups (a|z)
- Quantifiers s*

Examples:

- **Finding phone number patterns**

```
\d\d\d. \d\d\d. \d\d\d\d

\d\d\d[-.]\d\d\d[-.]\d\d\d\d

\d{3}[-.] \d{3}[-.] \d{4}
```

- **What string pattern will this RegEx code match?**

```
M(r|s|rs)\.?\s[A-Z]\w*
```

What are Regular Expressions, or RegEx?

2 options for you to explore RegEx:

- **Work through a tutorial:**
  https://regexone.com/
  https://www.tutorialspoint.com/python/python_reg_expressions.htm

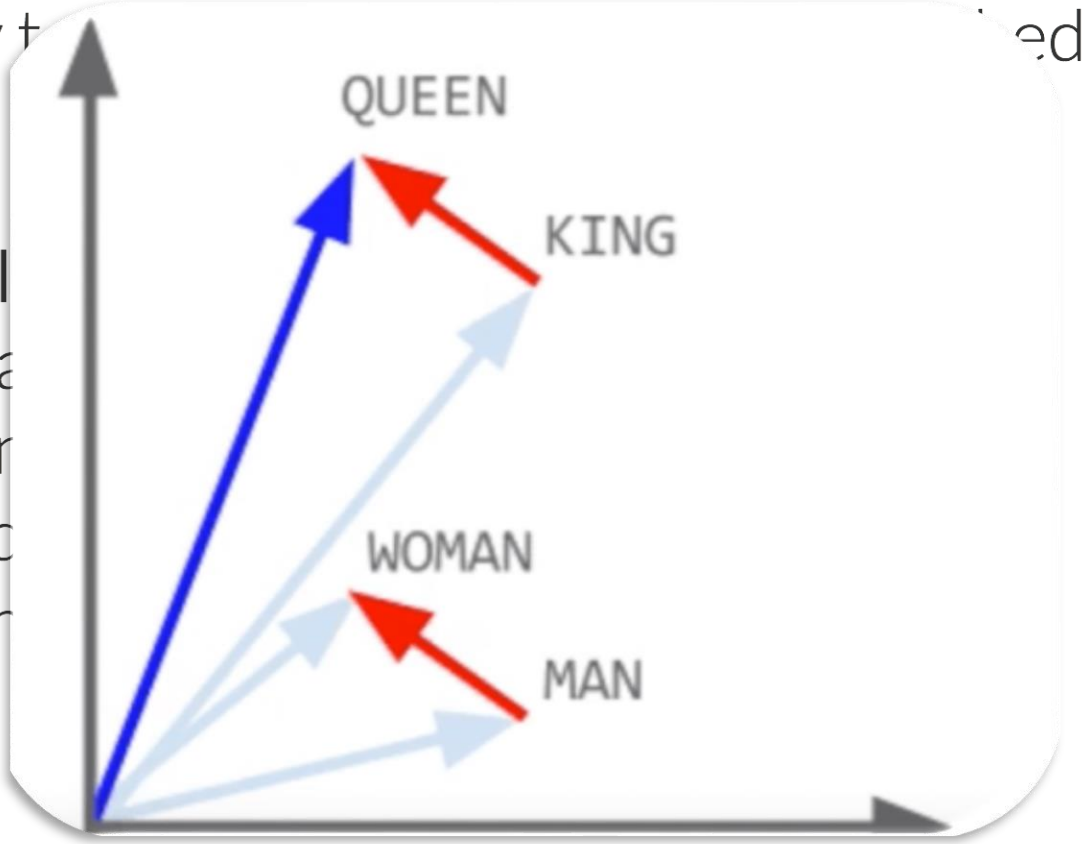
- **Play in Jupyter, using your RegEx cheat sheet handout as a guide.**
  Start by creating your own mini-corpus (~20 words) and write RegEx code to match a string from your corpus.

Pro-tip reminders: Be computational *and* creative in your approach. There are an infinite number of ways to accomplish a string matching task! Define your task clearly (functional level) then start coding.

Vector Space Modeling, Word-embedded vectors & Cosine Similarity

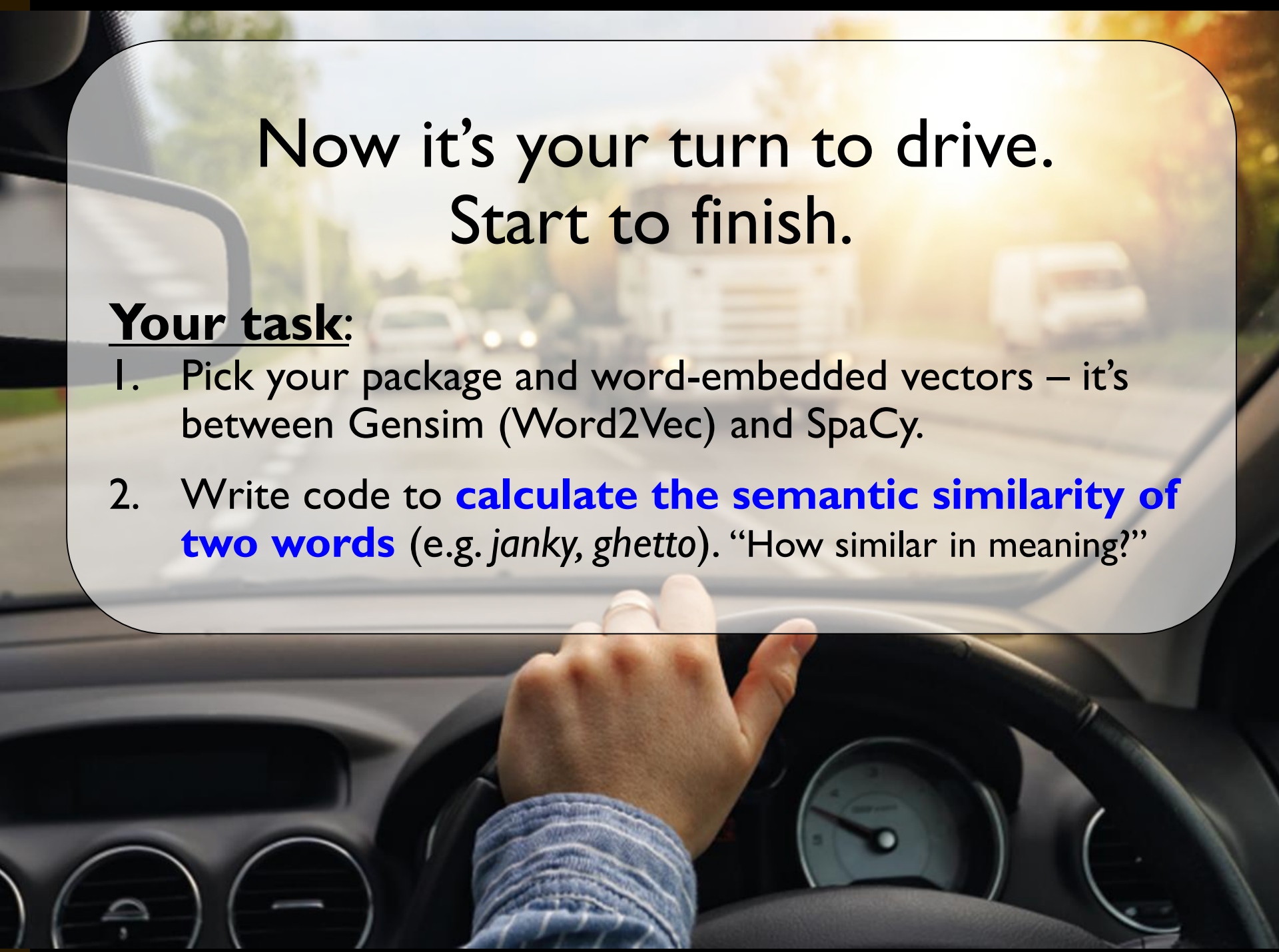
Try t edding

Vocabul
Man, woma
girl, prin
princess, c
king, mon



Quantifying word meaning

```
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 26 11:07:47 2019

@author: emily
"""

import spacy

nlp = spacy.load('en_core_web_lg')

#%%
#Calculate semantic similarity
tokens = nlp('god data')
tokens[0].similarity(tokens[1])

#%%
#Find closest semantic neighbors
def most_similar(word):
    queries = (w for w in word.vocab if w.is_lower == word.is_lower and
w.prob >= -15)
    by_similarity = sorted(queries, key=lambda w: word.similarity(w),
reverse=True)
    return by_similarity[:40]

[w.lower_ for w in most_similar(nlp.vocab['mandate'])]
```



Now it's your turn to drive. Start to finish.

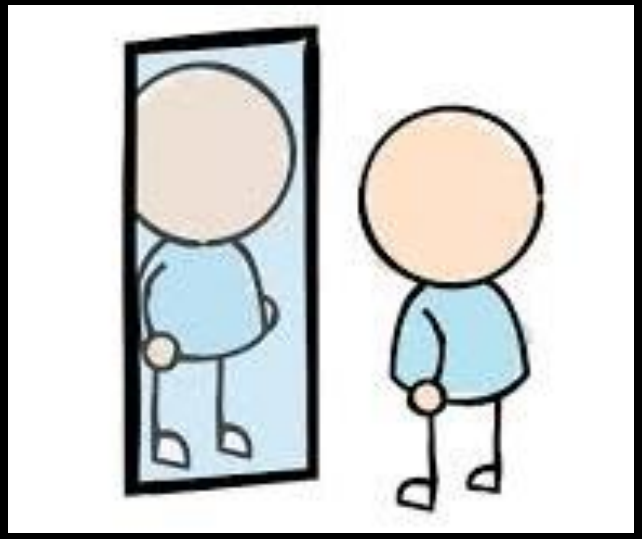
Your task:

1. Pick your package and word-embedded vectors – it's between Gensim (Word2Vec) and SpaCy.
2. Write code to **calculate the semantic similarity of two words** (e.g. *janky, ghetto*). "How similar in meaning?"

4 TAKE-AWAYS

1. **Google Ngram Viewer** is a quick 'n dirty tool for measuring word frequency change over time.
2. **Topic modeling** is a dimensionality reduction technique used to reveal "topics" in a document.
3. **Regular Expressions (RegEx) is the syntax you use** to do string matching, text cleaning, and token extraction.
4. **Word-embedded vectors** are decomposed matrices from a huge word matrix that tells you about word meaning.

CHECK-IN:



1. So far, what is the most insightful thing you've learned during camp?
2. What is the one thing that's still the muddiest for you?

Thank you!

Come to a FREE Nerd Nite talk
I'm doing about linguistics on
Thursday, June 20th at LIVE, 7pm:

The 13 Things You Need to Know about Language.

LANGUAGE MATTERS
The University of Michigan

Emily Rae Sabo
@StandupLinguist