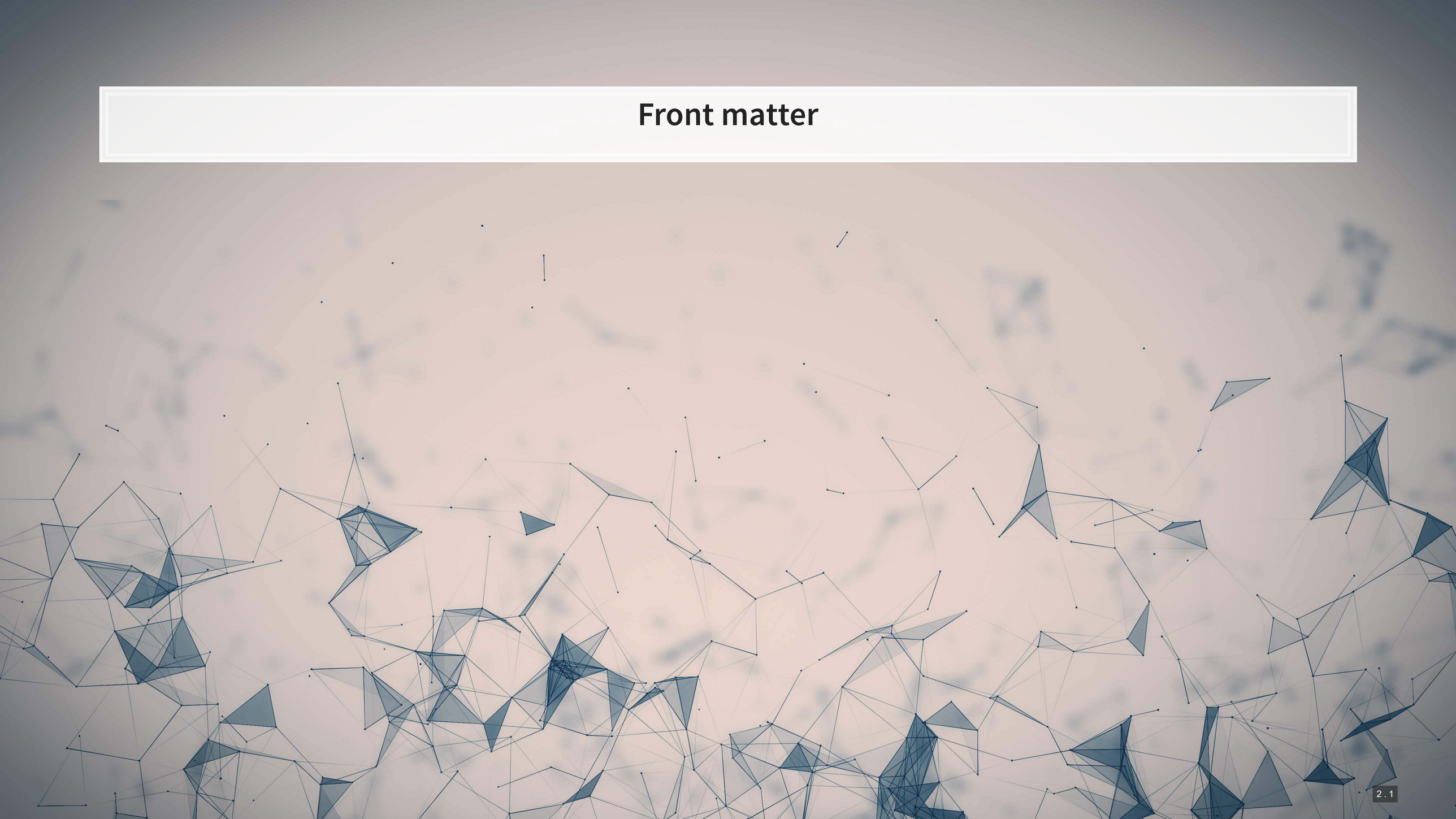
ACCT 420: ML and AI for numeric and text data

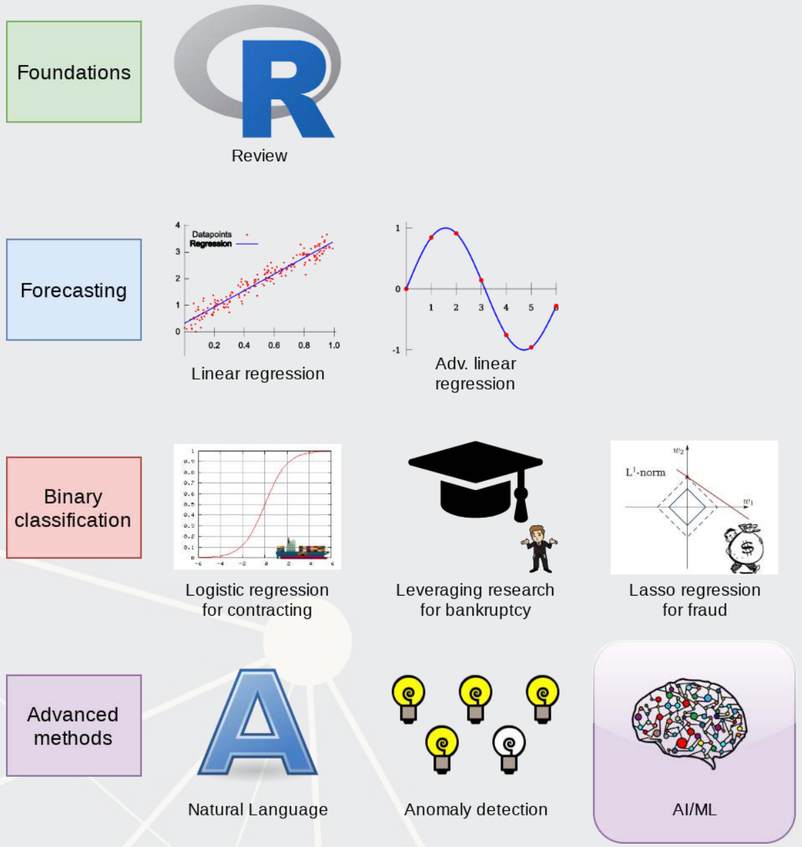
Session 10

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

Front matter



Learning objectives



- **Theory:**
  - Neural Networks (broad overview)
  - Vector space methods
- **Application:**
  - Neural networks for understanding textual data
  - Top managements' tweets
- **Methodology:**
  - Vector methods
  - 6 types of neural networks
  - Others

Languages for ML/AI

R for ML/AI

Older methods

- `caret`
- `randomForest`
- `nnet`
- `e1071`

Best-in-class

- `glmnet`: LASSO and elastic nets
- `xgboost`: XGBoost
- `Prophet`: ML for time series forecasting
- `keras`: Plugs into python's Keras
- `H2O4GPU`: Plugs into python's H2O
- `spacyr`: Plugs into python's SpaCy

Python for ML/AI

Older methods

- Sci-kit learn – one stop shop for most older libraries
- RPy2
- scipy + numpy + pandas + statsmodels
  - Add Theano in for GPU compute

Best-in-class

- **TENSORFLOW** (Google)
  - Can do everything
- pytorch – python specific Torch port
- gensim: “Topic modelling for humans”
- H2O (H2O)
- caffe (Berkley)
- caffe2 (Facebook)
- SpaCy – Fast NLP processing
- CoreNLP – through various wrappers to the Java library

Others for ML/AI

- C/C++: Also a first class language for TensorFlow!
  - Really fast – precompiled
  - Much more difficult to code in
- Swift: Strong TensorFlow support
- Javascript: Improving support from TensorFlow and others

Why do I keep mentioning TensorFlow?

- It can run almost ANY ML/AI/NN algorithm
- It has APIs for easier access like Keras
- Comparatively easy GPU setup
- It can deploy anywhere
  - Python & C/C++ built in
  - Swift, R Haskell, and Rust bindings
  - TensorFlow light for mobile deployment
  - TensorFlow.js for web deployment

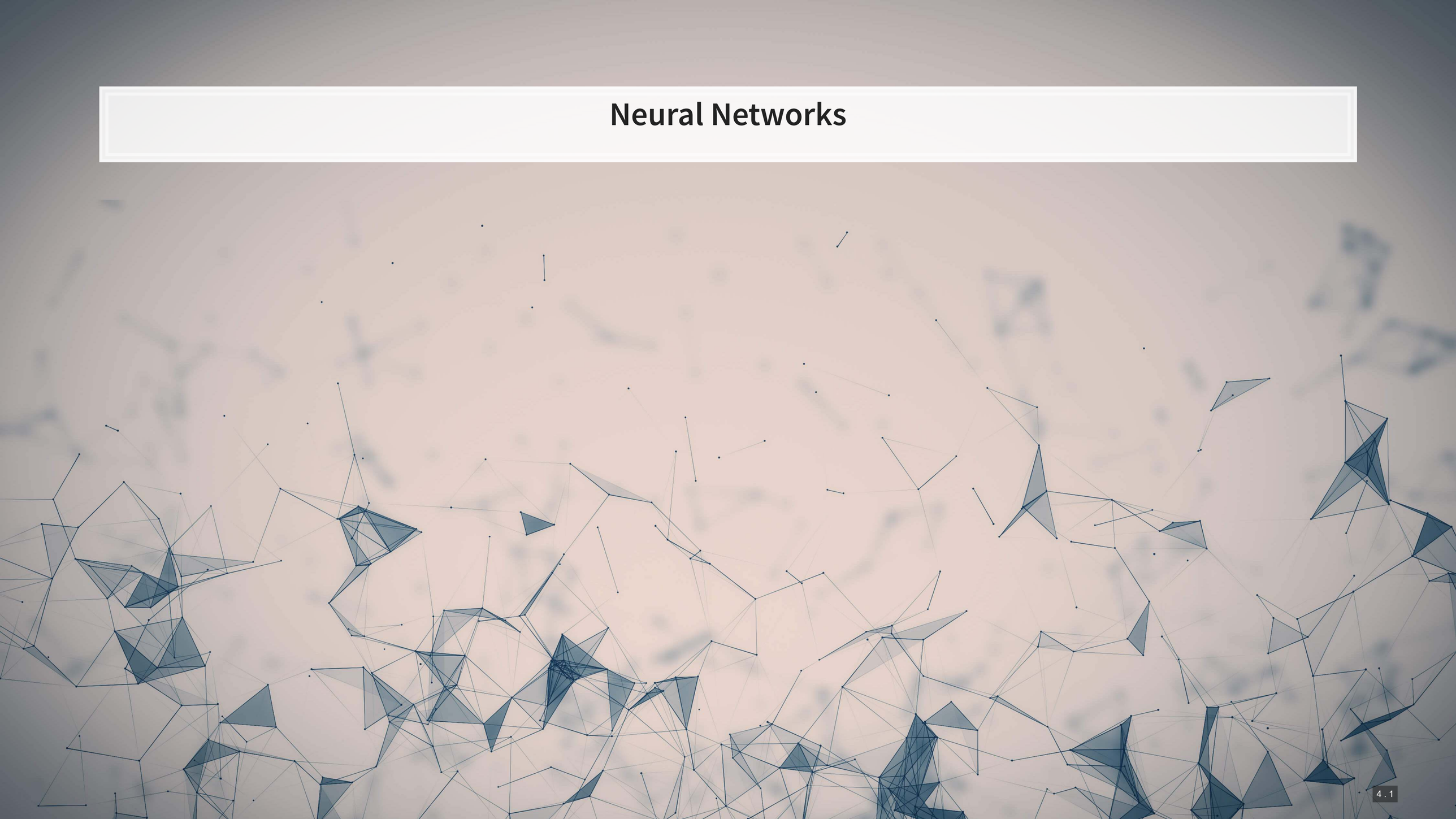
Why do I keep mentioning TensorFlow?

- It has strong support from Google and others
  - TensorFlow Hub – Premade algorithms for text, image, and video
  - tensorflow/models – Premade code examples
    - The research folder contains an amazing set of resources
  - trax – AI research models from Google Brain

Other notable frameworks

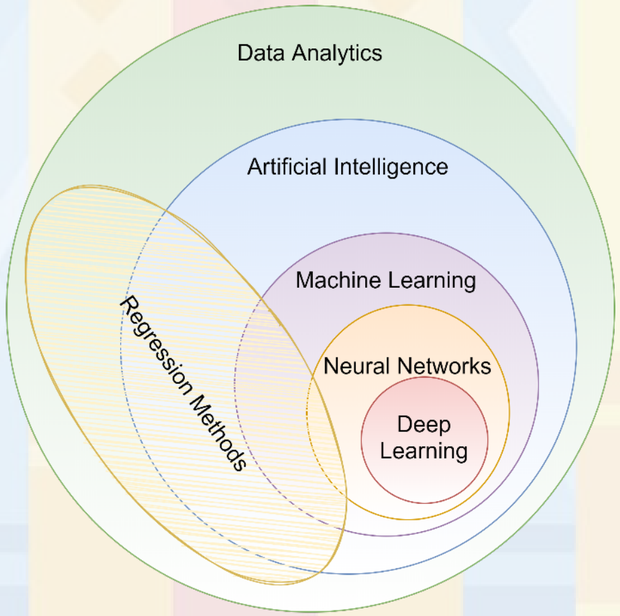
- Caffe
  - Python, C/C++, Matlab
  - Good for image processing
- Caffe2
  - C++ and Python
  - Still largely image oriented
- Microsoft Cognitive Toolkit
  - Python, C++
  - Scales well, good for NLP
- Torch and Pytorch
  - For Lua and python
  - fast.ai, ELF, and AllenNLP
- H20
  - Python based
  - Integration with R, Scala…

Neural Networks



What are neural networks?

- The phrase *neural network* is thrown around almost like a buzz word
- *Neural networks* are actually a specific type class algorithms
  - There are many implementations with different primary uses

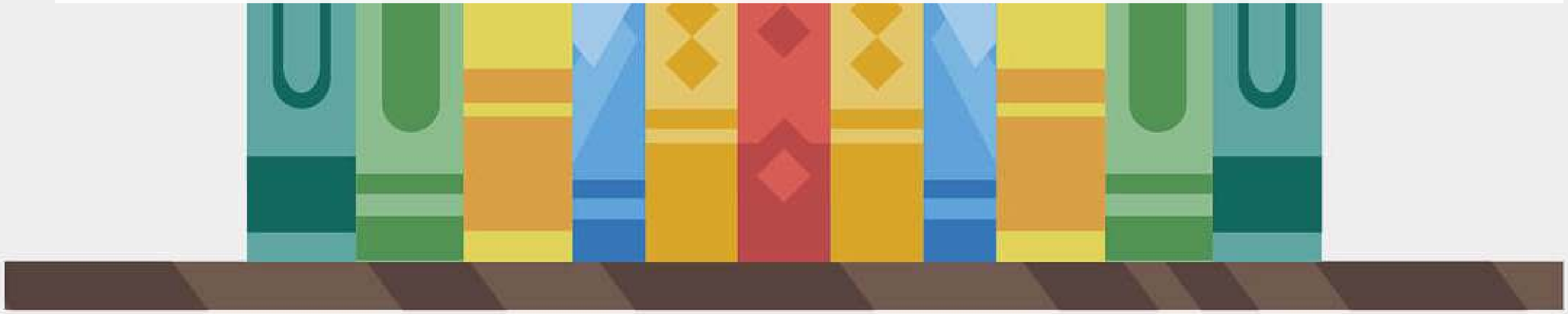


What are neural networks?

- Originally, the goal was to construct an algorithm that behaves like a human brain
  - Thus the name
- Current methods don't quite reflect human brains, however:
  1. We don't fully understand how our brains work, which makes replication rather difficult
  2. Most neural networks are constructed for specialized tasks (not general tasks)
  3. Some (but not all) neural networks use tools our brain may not have
     - I.e., **backpropogation** is potentially possible in brains, but it is not pinned down how such a function occurs (if it does occur)

What are neural networks?

- Neural networks are a method by which a computer can learn from observational data
- In practice:
  - They were not computationally worthwhile until the mid 2000s
  - They have been known since the 1950s (perceptrons)
  - They can be used to construct algorithms that, at times, perform better than humans themselves
    - But these algorithms are often quite computationally intense, complex, and difficult to understand
  - Much work has been and is being done to make them more accessible

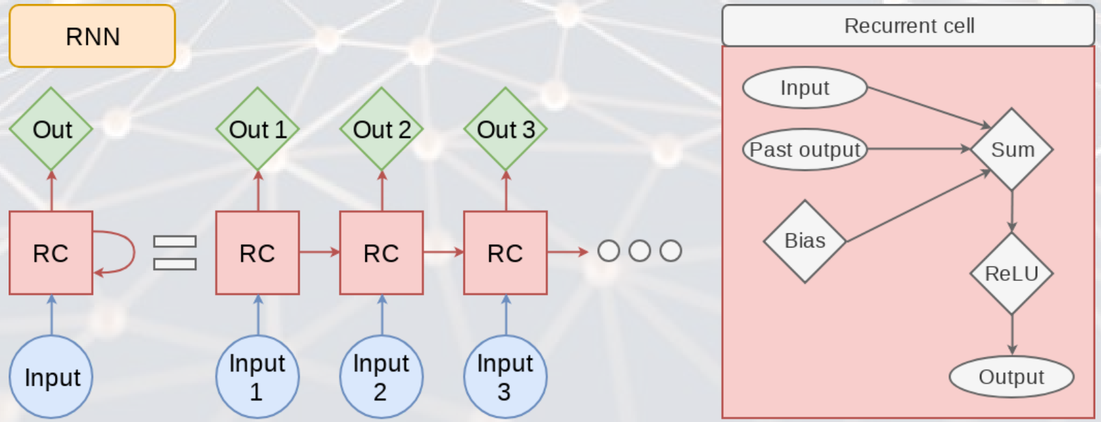


Types of neural networks

- There are *a lot* of neural network types
  - See The "Neural Network Zoo"
- Some of the more interesting ones which we will see or have seen:
  - RNN: Recurrent Neural Network
  - LSTM: Long/Short Term Memory
  - CNN: Convolutional Neural Network
  - DAN: Deep Averaging Network
  - GAN: Generative Adversarial Network
- Others worth noting
  - VAE (Variational Autoencoder): Generating *new* data from datasets
- Not in the Zoo, but of note:
  - Transformer: Networks with "attention"
    - From Attention is All You Need

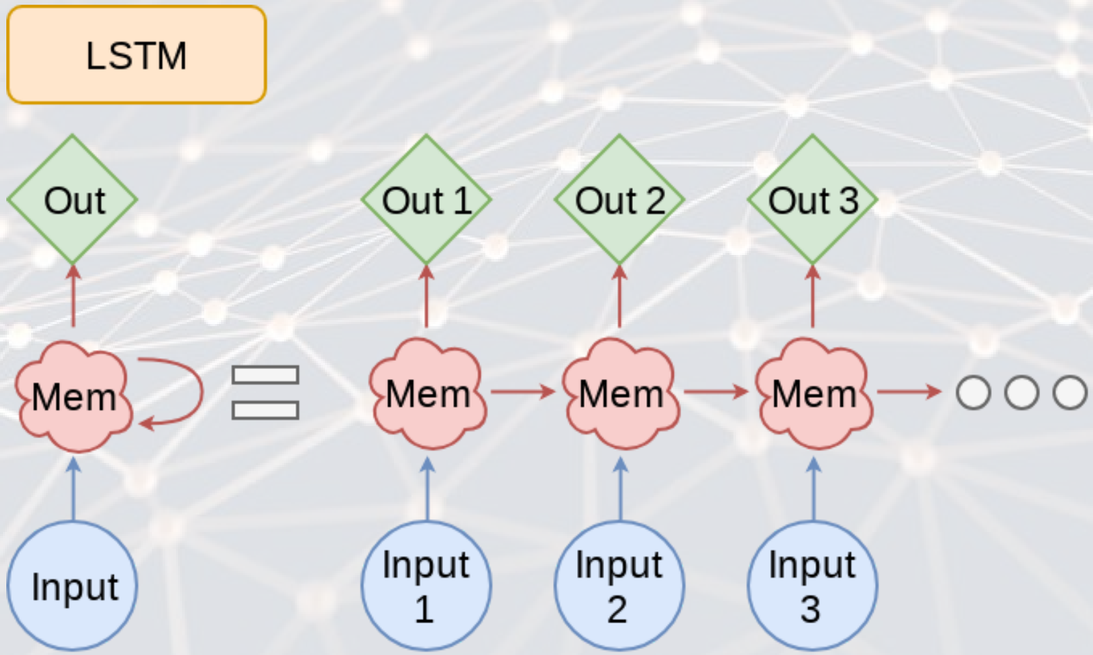
RNN: Recurrent NN

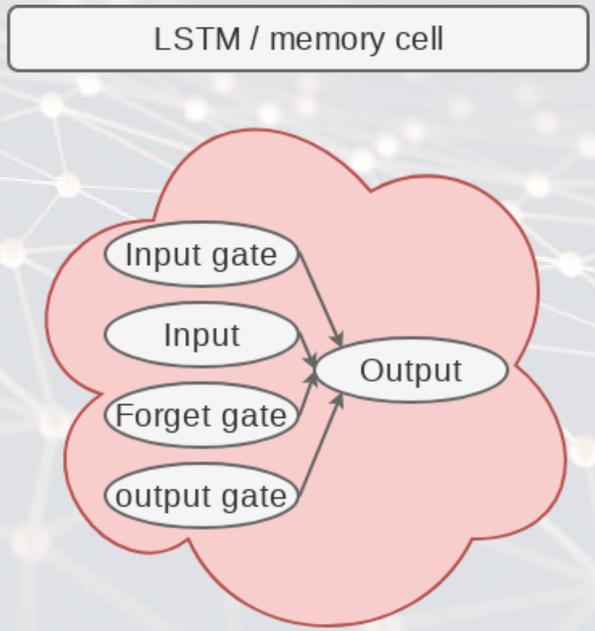
- Recurrent neural networks embed a history of information in the network
  - The previous computation affects the next one
  - Leads to a *short term memory*
- Used for speech recognition, image captioning, anomaly detection, and many others
  - Also the foundation of LSTM
  - SketchRNN (live demo)



LSTM: Long Short Term Memory

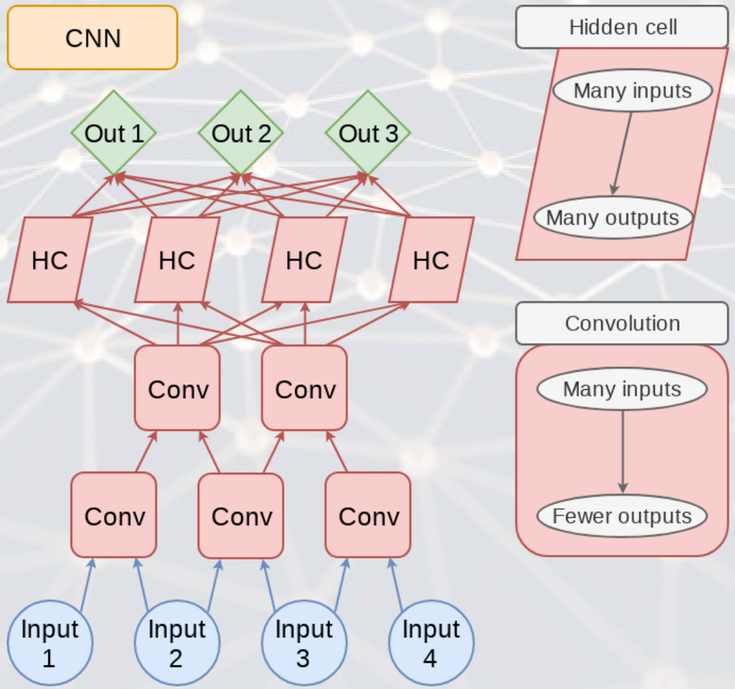
- LSTM improves the *long term memory* of the network while explicitly modeling a *short term memory*
- Used wherever RNNs are used, and then some
  - Ex.: Seq2seq (machine translation)





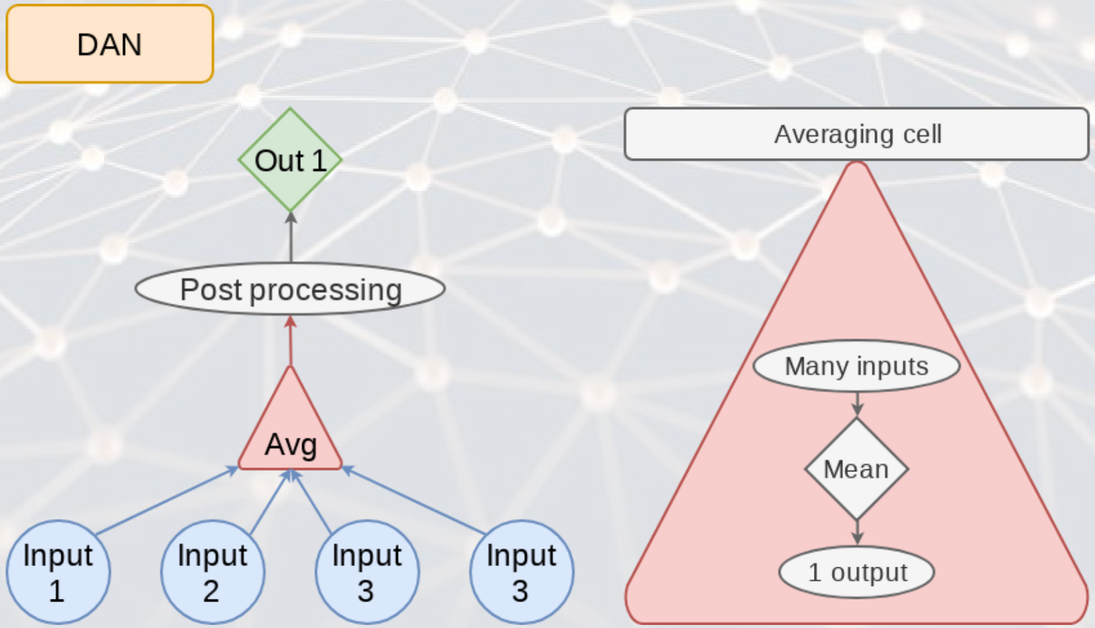
CNN: Convolutional NN

- Networks that excel at object detection (in images)
- Can be applied to other data as well
- Ex.: Inception-v3



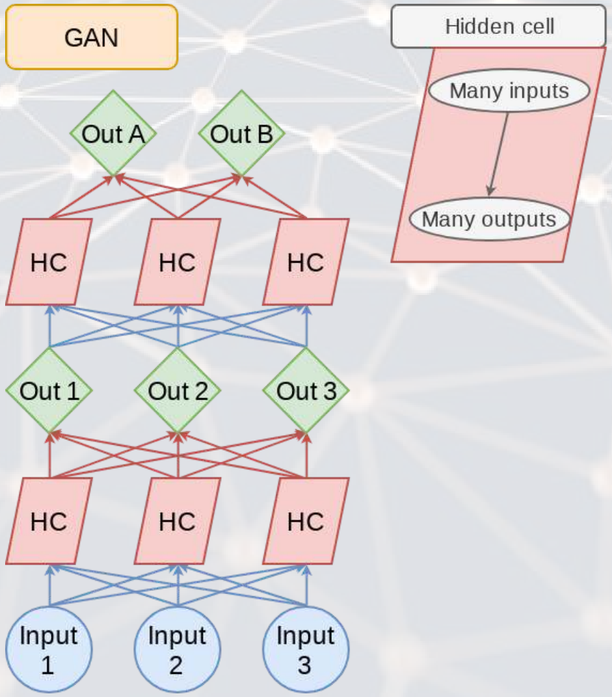
DAN: Deep Averaging Network

- DANs are simple networks that simply average their inputs
- Averaged inputs are then processed a few times
- These networks have found a home in NLP
  - Ex.: Universal Sentence Encoder



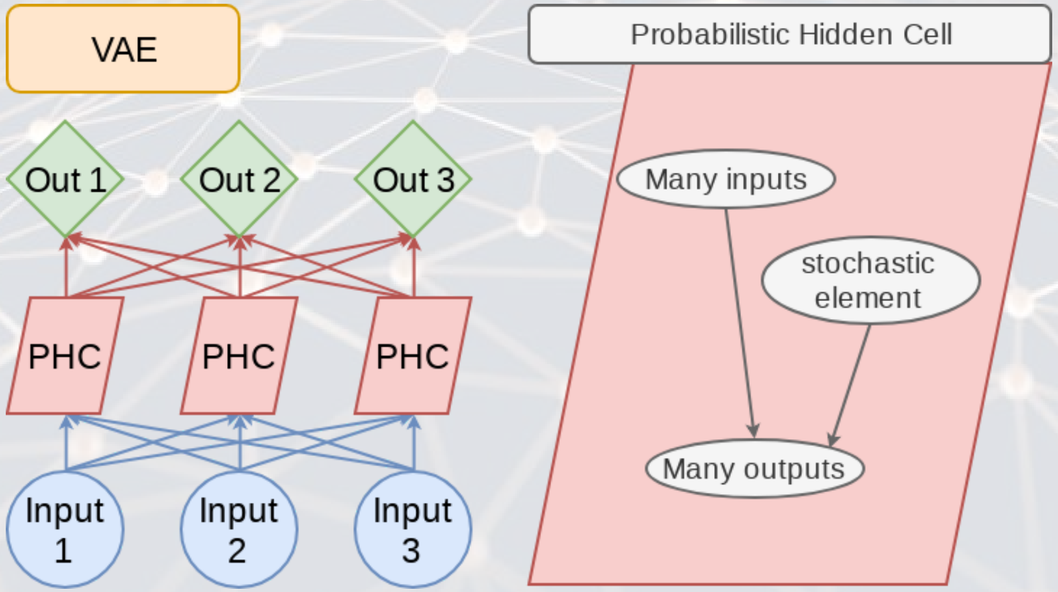
GAN: Generative Adversarial Network

- Feature two networks working against each other
- Many novel uses
  - Ex.: The anonymization GAN we saw
  - Ex.: Aging images



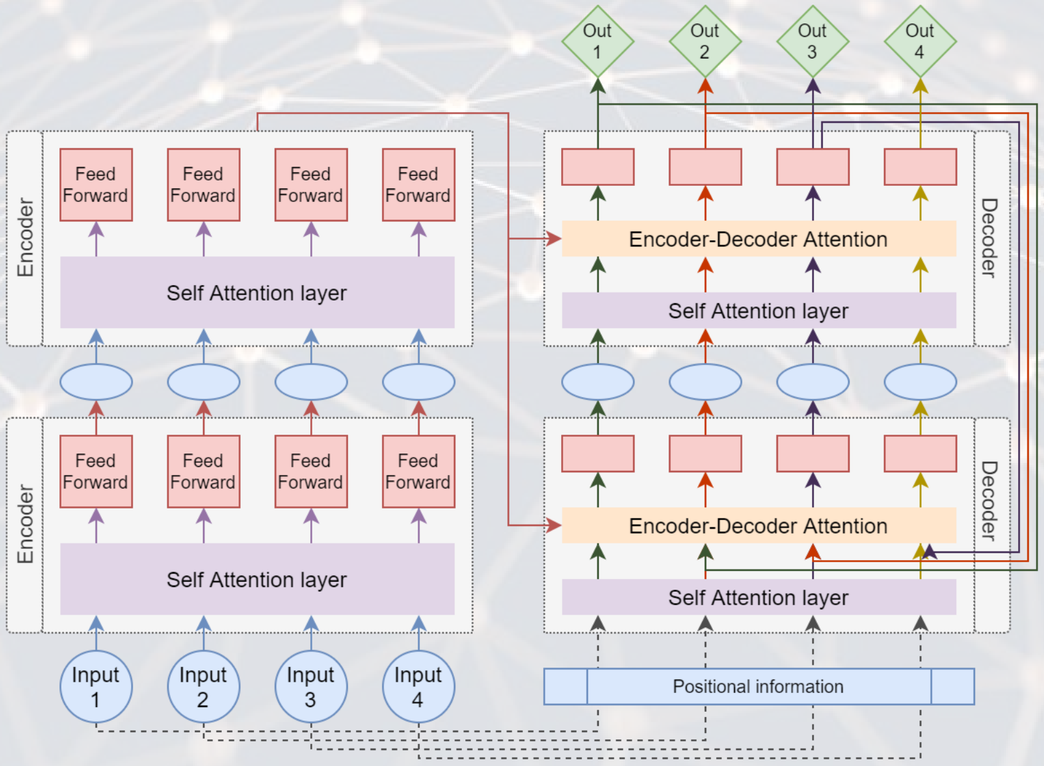
VAE: Variational Autoencoder

- An autoencoder (AE) is an algorithm that can recreate input data
- Variational means this type of AE can vary other aspects to generate completely new output
  - Good for creating fake data
- Like a simpler, noisier GAN

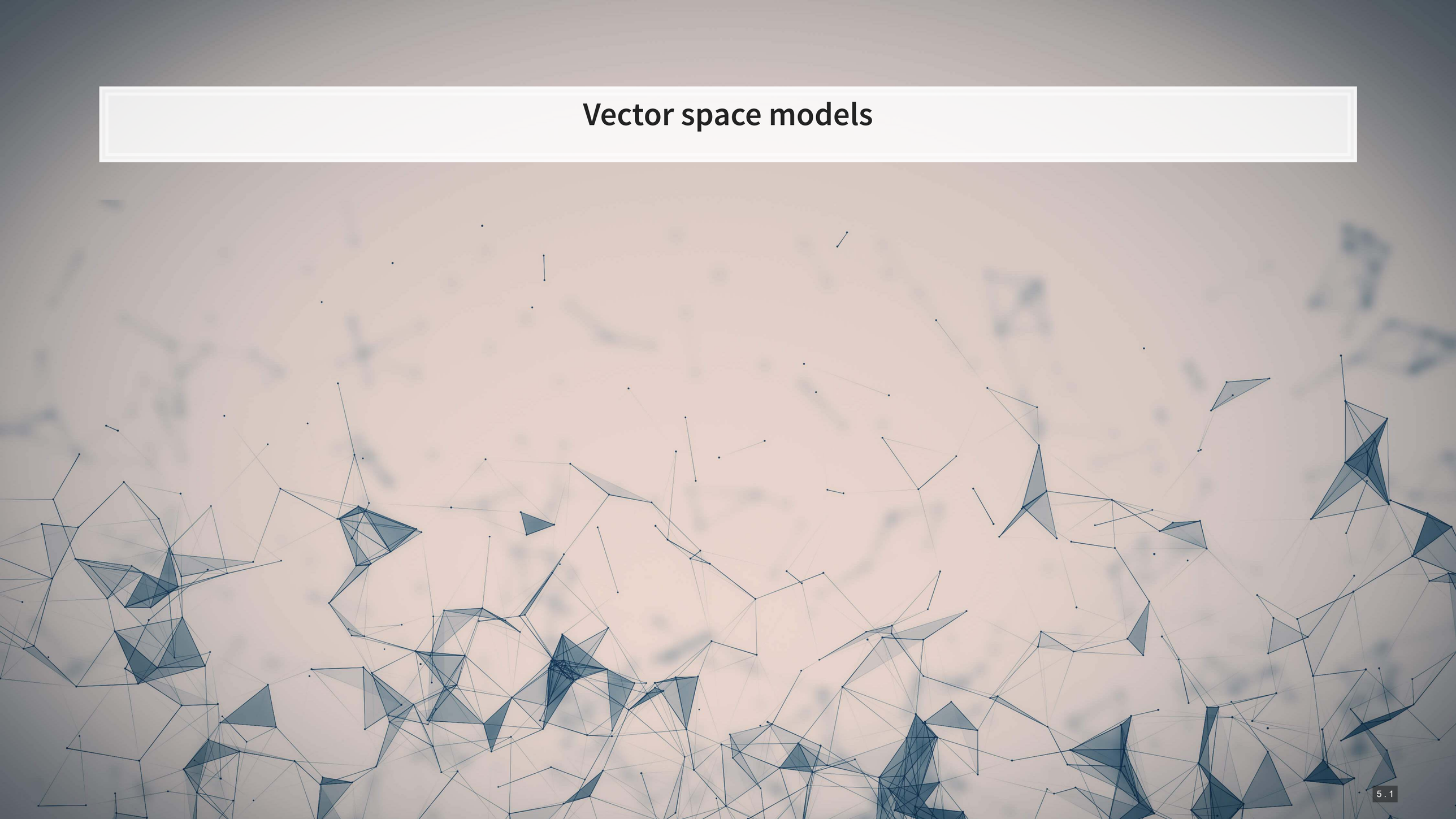


Transformer

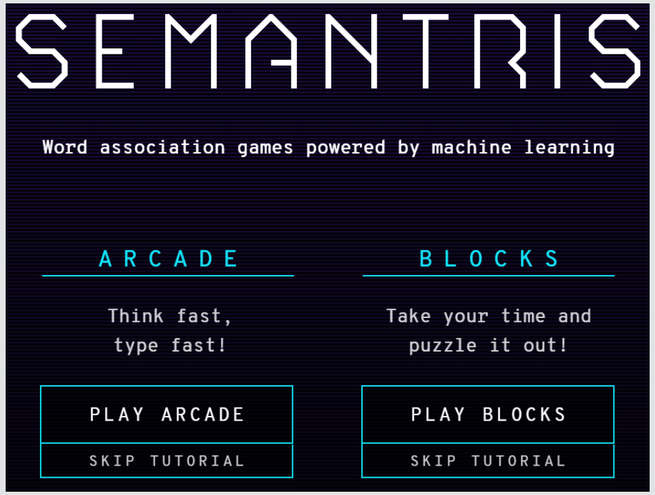
- Shares some similarities with RNN and LSTM: Focuses on attention
- Currently being applied to solve many types of problems
- Examples: BERT, GPT-3, XLNEt

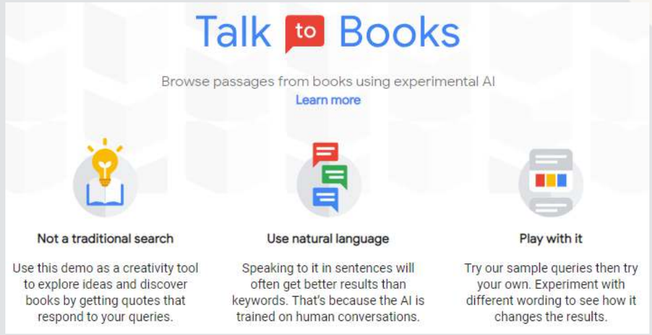


Vector space models



Motivating examples





What are “vector space models”

- Different ways of converting some abstract information into numeric information
  - Focus on maintaining some of the underlying structure of the abstract information
- Examples (in chronological order):
  - Word vectors:
    - Word2vec
    - GloVe
  - Paragraph/document vectors:
    - Doc2Vec
  - Sentence vectors:
    - Universal Sentence Encoder

Word vectors

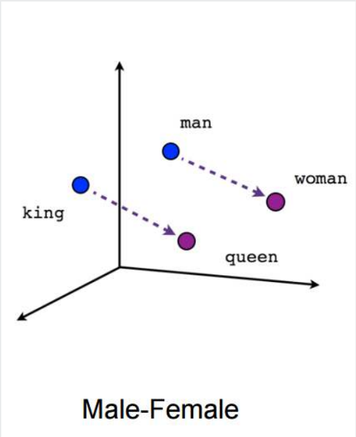
- Instead of coding individual words, encode word meaning
- The idea:
  - Our old way (encode words as IDs from 1 to N) doesn’t understand relationships such as:
    - Spatial
    - Categorical
    - Grammatical (weakly when using stemming)
    - Social
    - etc.
  - Word vectors try to encapsulate all of the above
    - They do this by encoding words as a vector of different features

Word vectors: Simple example

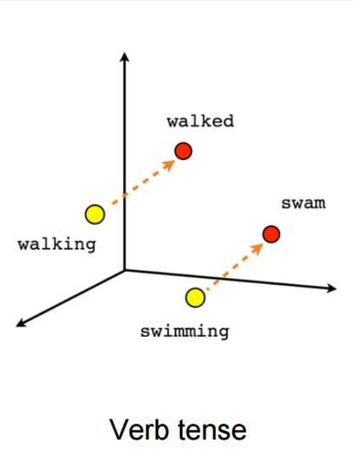
words	f_animal	f_people	f_location
dog	0.5	0.3	-0.3
cat	0.5	0.1	-0.3
Bill	0.1	0.9	-0.4
turkey	0.5	-0.2	-0.3
Turkey	-0.5	0.1	0.7
Singapore	-0.5	0.1	0.8

- The above is an idealized example
- Notice how we can tell apart different animals based on their relationship with people
- Notice how we can distinguish turkey (the animal) from Turkey (the country) as well

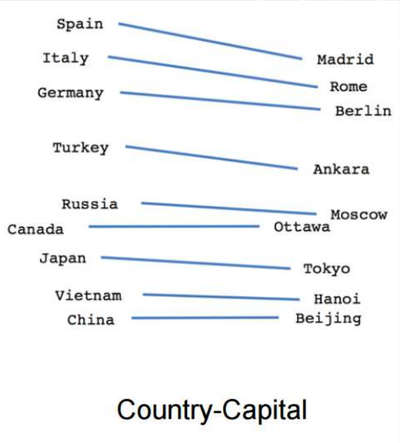
What it retains: word2vec



Male-Female

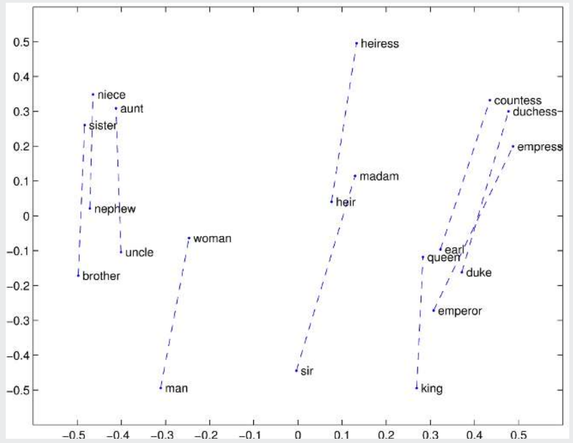


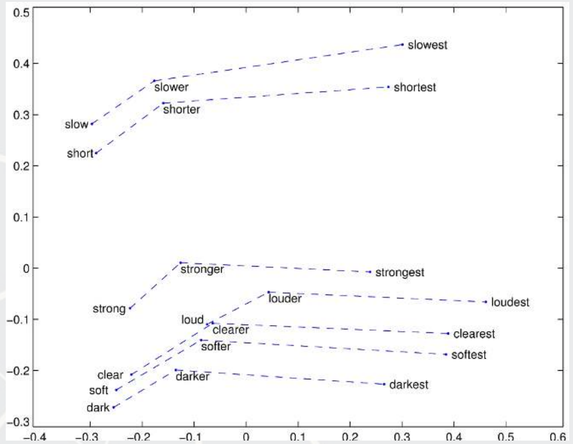
Verb tense

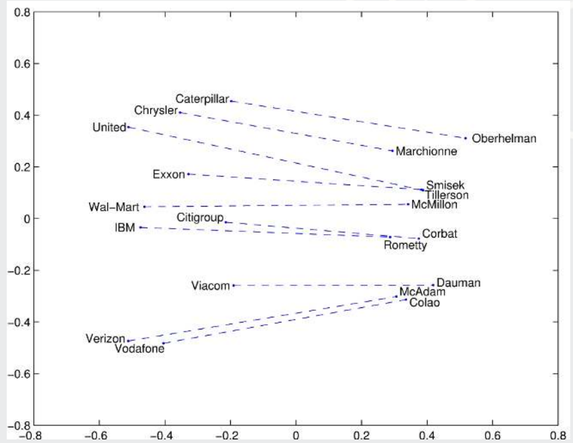


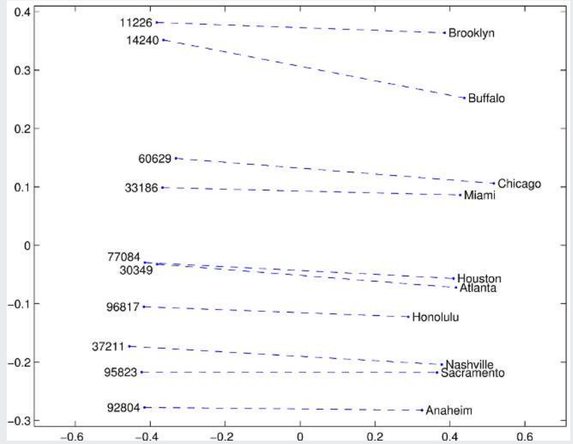
Country-Capital

What it retains: GloVe







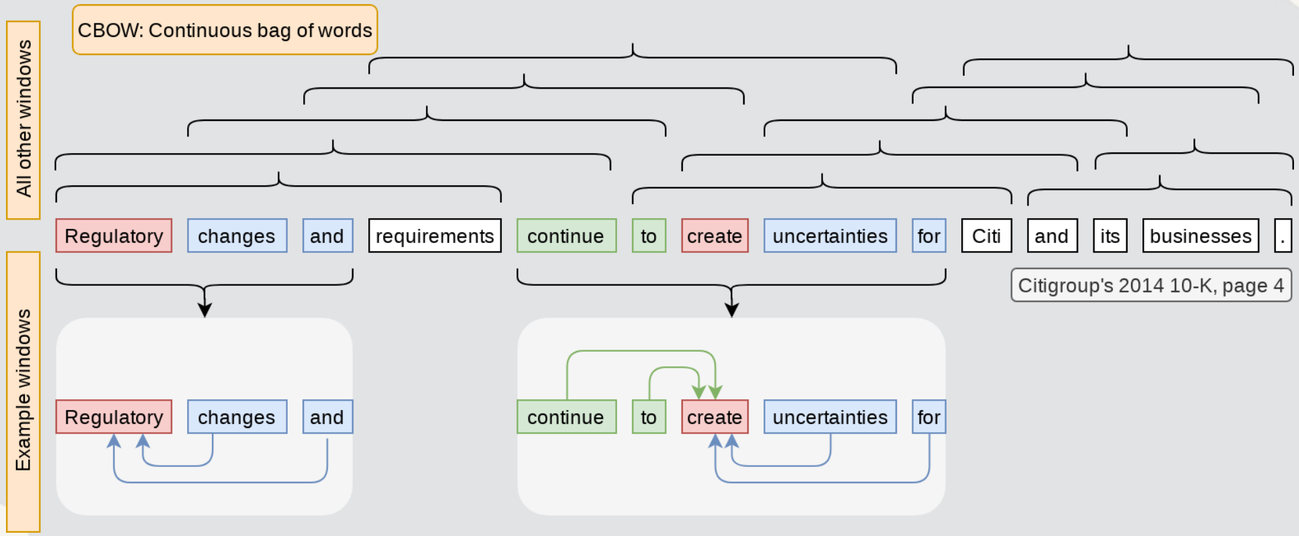


How to build word vectors

- Two ways:
  1. Word co-occurrence (like how LDA worked)
     - Global Vectors (GloVe) works this way
       - Available from the `text2vec` package
  2. Word order (using an NN)
     - word2vec works this way
       - Available from the `rword2vec` package
       - Uses a 2 layer neural network

How does word order work?

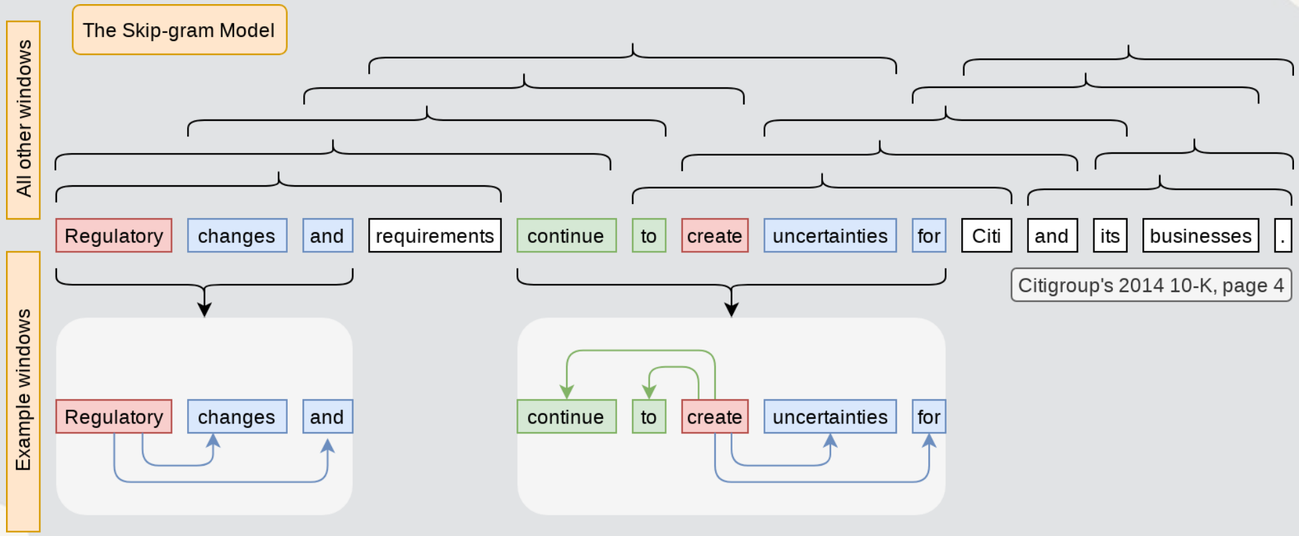
Infer a word's meaning from the words around it



Refered to as CBOW (continuous bag of words)

How else can word order work?

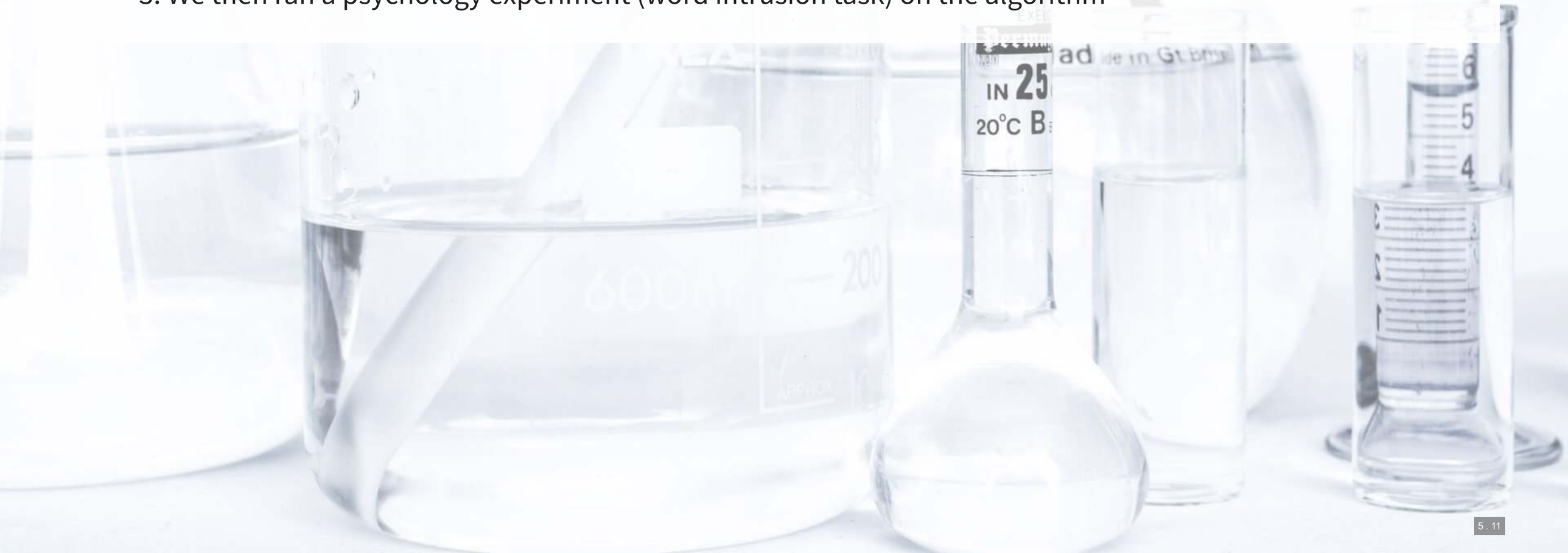
Infer a word's meaning by *generating* words around it



Refered to as the Skip-gram model

An example of using word2vec

- In the BCE paper from Session 6, word2vec was used to provide assurance that the LDA model works reasonably well on annual reports
  1. We trained a word2vec model on random issues of the Wall Street Journal (247.8M words)
  2. The resulting model “understood” words in the context of the WSJ
  3. We then ran a psychology experiment (word intrusion task) on the algorithm



Word intrusion task

- The task is to find which word doesn’t belong
- Each question consisted of 3 words from 1 topic and 1 *intruded* from another random topic
  - Ex.:
    - Laser, Drug, Viral, Therapeutic
    - Supply, Steel, Capacity, Losses
    - Relief, Lousisiana, Cargo, Assisted

Results


60
55
50
45
40
35
30
25
% of questions correct
2000
2002
2004
2006
2008
2010
2012
Year
Web corpus
WSJ corpus
10-K corpus
Random chance

Implementing in R

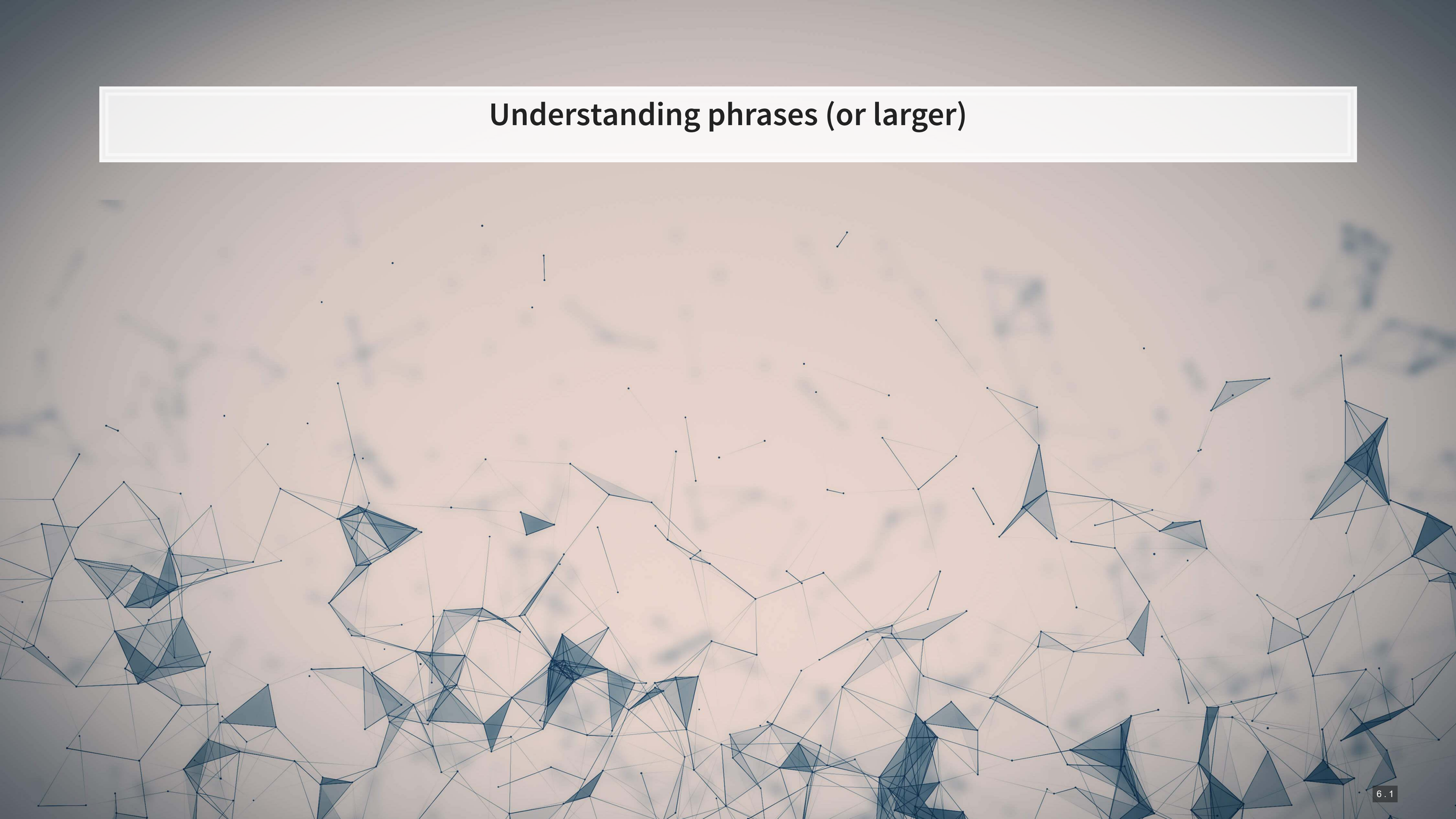
- A few options:
  - The rword2vec package for word2vec
  - The text2vec package for GloVe
  - Rolling your own neural network for word2vec with `keras` (guide here)

When are vector embeddings useful?

1. You care about the words used, by not stylistic choices
2. You want to crunch down a bunch of words into a smaller number of dimensions without running any bigger models (like LDA) on the text.

> An interactive demo of word similarity

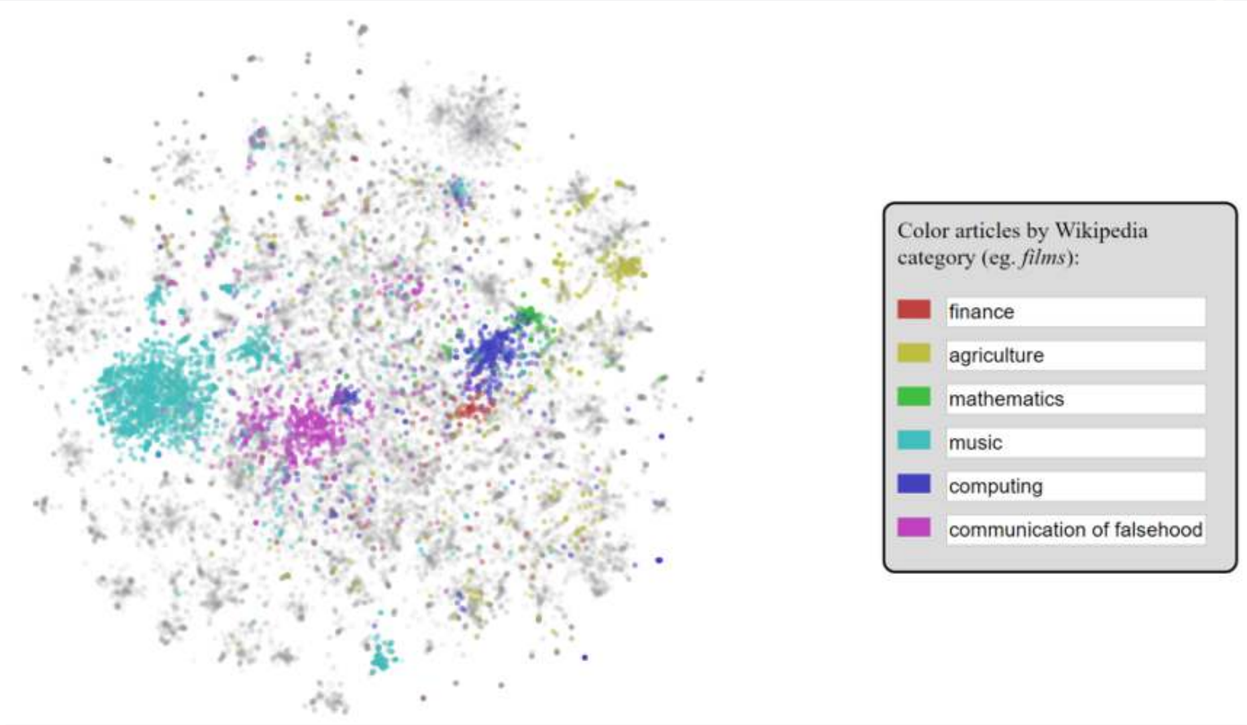
Understanding phrases (or larger)



Document vectors

- Document vectors work very similarly to word vectors
  - 1 added twist: a document/paragraph/sentence level factor variable
    - This is used to learn a vector representation of each text chunk
    - Learned simultaneously with the word vectors
      - Caveat: it can also be learned independently using PV-DBOW
- This is quite related to what we learned with LDA as well!
  - Both can tell us the topics discussed

Wikipedia article categorization



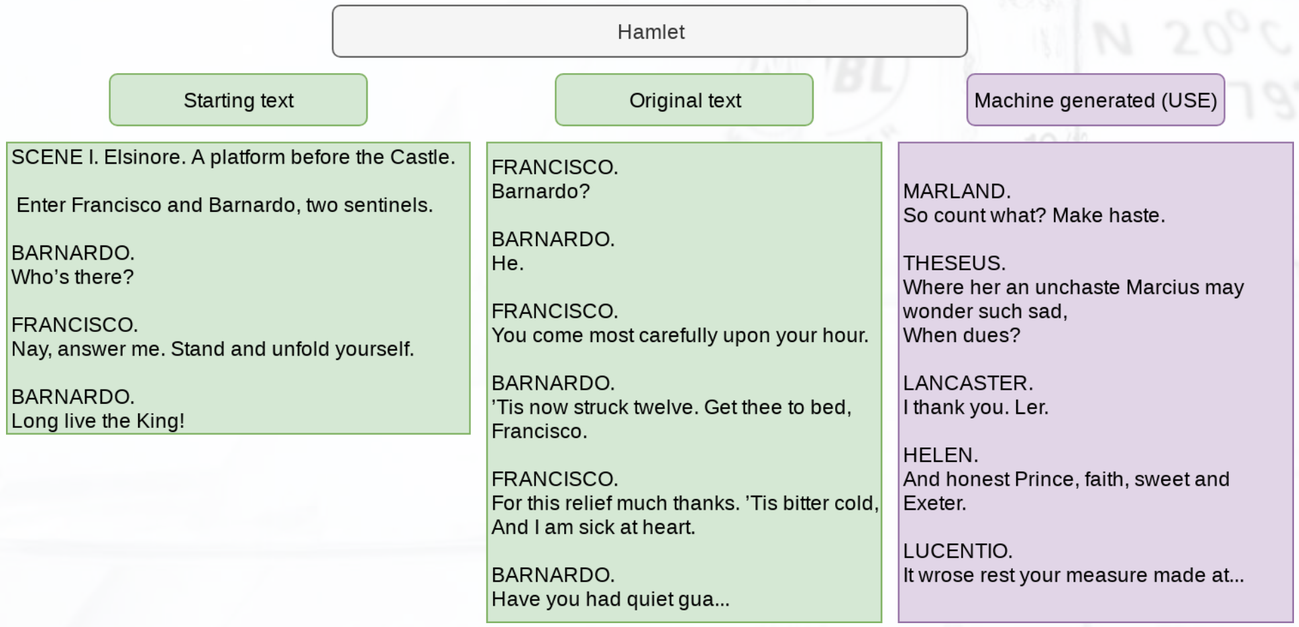
Source article (colah.github.io)

Universal Sentence Encoder (USE)

- We saw this briefly last week
  - This is the algorithm with less bias
- Focused on representing sentence-length chunks of text

A fun example of with USE

- Predict Shakespeare with Cloud TPUs and Keras



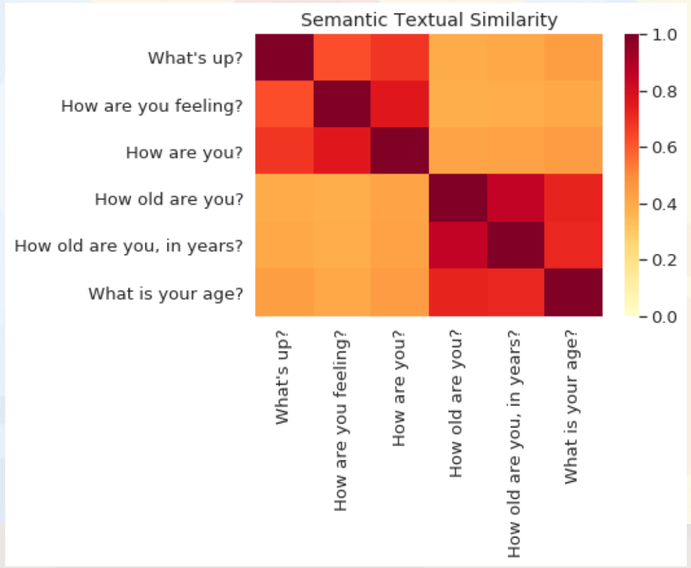
Cavaet on using USE

- One big caveat: USE only knows what it's trained on
  - Ex.: Feeding the same USE algorithm WSJ text

> Samsung Electronics Co., suffering a handset sales slide, revealed a foldable-screen smartphone that folds like a book and opens up to tablet size. Ah, horror? I play Thee to her alone;
> And when we have withdrom him, good all.
> Come, go with no less through.
>
> Enter Don Pedres. A flourish and my money. I will tarry. Well, you do!
>
> LADY CAPULET.
> Farewell; and you are

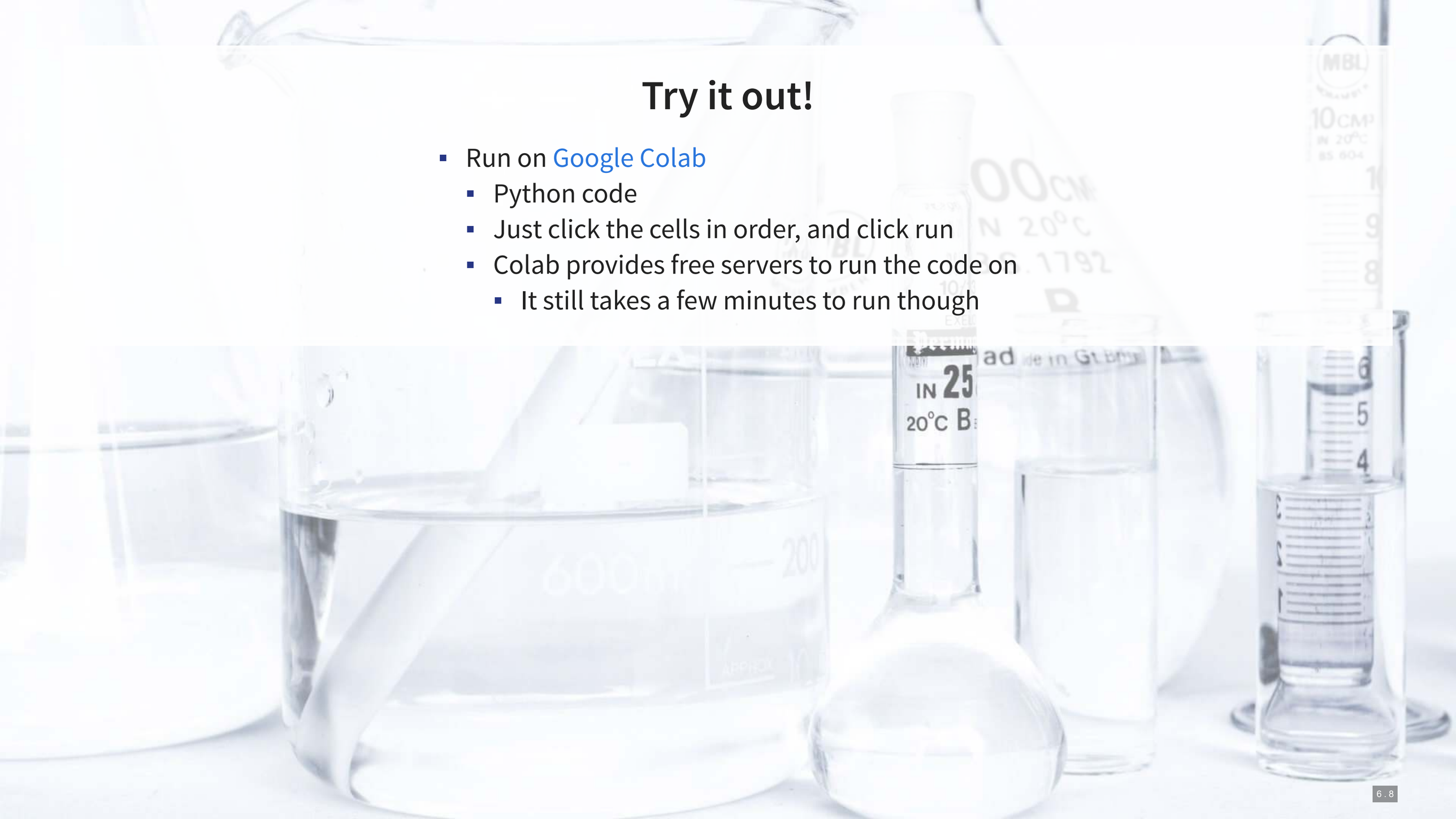
How does USE work?

- USE is based on DAN and Transformer
  - There is another specification as well
  - Learns the meaning of sentences via words' meanings
- Learn more: Original paper and TensorFlow site
- In practice, it works quite well



Try it out!

- Run on Google Colab
  - Python code
  - Just click the cells in order, and click run
  - Colab provides free servers to run the code on
    - It still takes a few minutes to run though



Bringing this into accounting

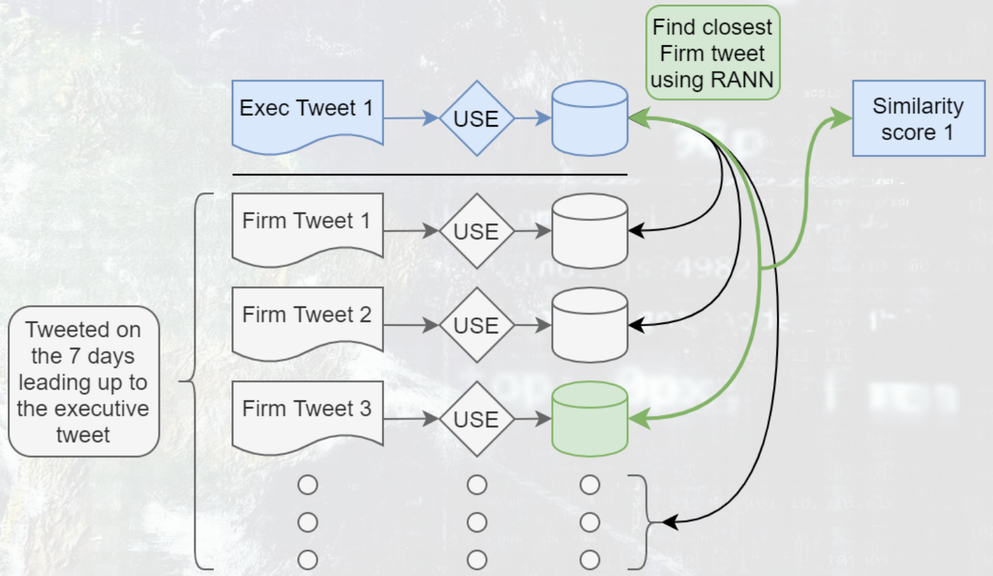
Understanding why stock markets respond strongly to CEOs and CFOs tweets:

- Crowley, Huang, and Lu 2020, “Executive Tweets”
- Data: Tweets for ~100 executives and their firms from 2011 through 2017
- Premise: Markets respond more strongly to executives’ tweets than firms’ tweets
- Idea: Do markets *trust* executives more or do executives post new *useful information*?

How can USE help us to solve this?

> Use USE to determine if there is new content

1. Use USE to extract each tweet's meaning
2. See how similar executives' tweets are to their firms' tweets
   - Using the great RANN library in R to efficiently calculate this
3. See how markets respond conditional on tweet similarity



Mechanism: Market reaction to overnight financial tweets

Variable	$\lvert MMR_t \rvert$	t-value
Exec fin tweets$_{overnight}$	-0.027^{***}	(-3.47)
Similarity $\times$ Exec fin tweets$_{overnight}$	0.059^{***}	(3.49)
Firm tweet controls & interactions	Yes	
Matching controls & interactions	Yes	
Controls	Yes	
Firm FE	Yes	
Exec FE	Yes	
Year FE	Yes	
Month FE	Yes	

Result is consistent with *Trust* driving investor reaction to executive financial tweets.

Other Transformer models

- **BERT**
  - Optimized to mimic question and answer behavior (examples)
  - Now used in Google Search for at least 70 languages
  - Additional reading
  - Available in TensorFlow Hub
- **XLNet**
  - Similar objective to BERT, but with a focus on word order
- **T5**
  - A more extensible transformer model
  - Details

Other Transformer models

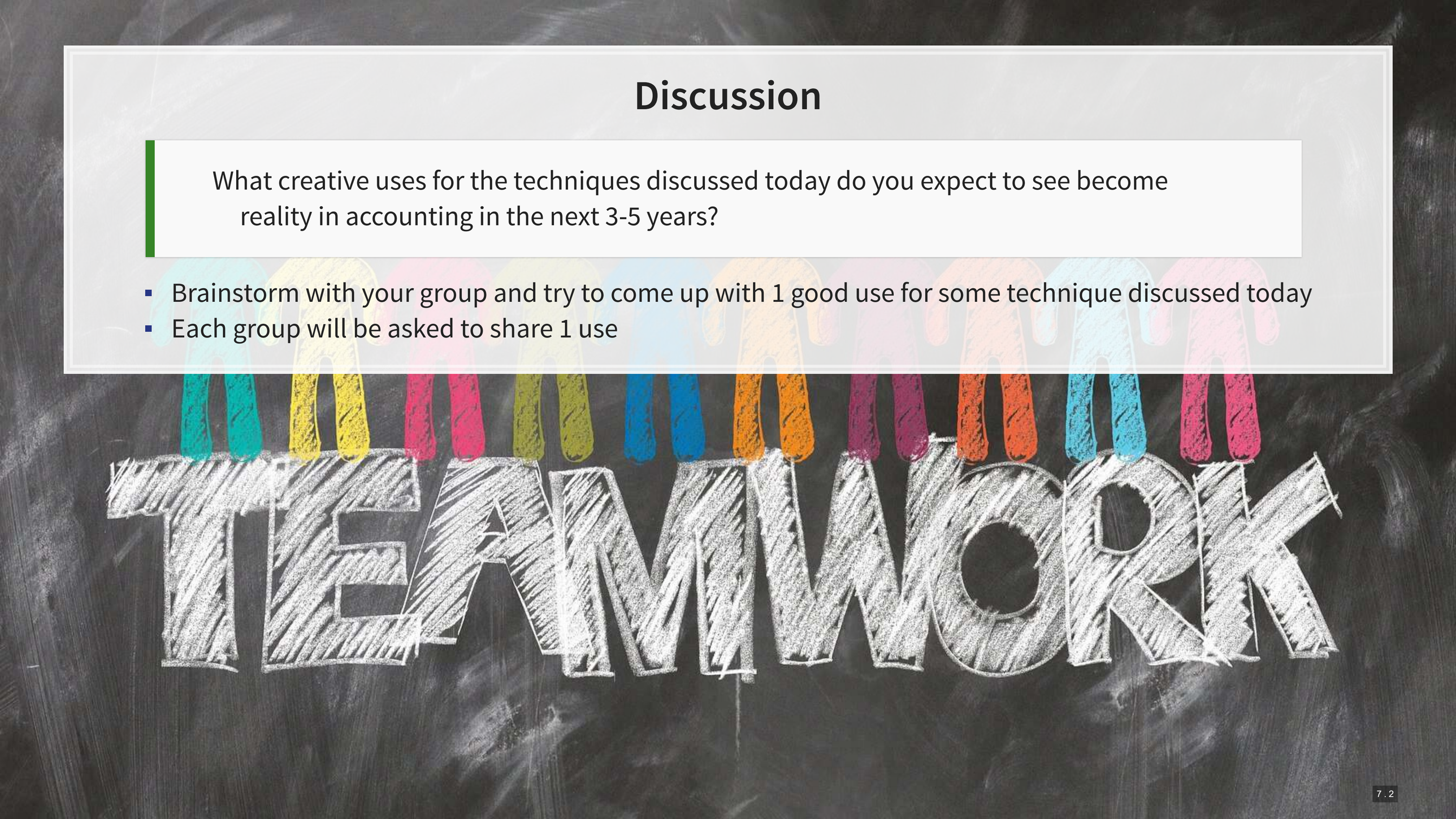
- **GPT-2**
  - A pretty good model for mimicking human speech patterns
  - Considered dangerous enough to not release initially (source)
    - Released 9 months later alongside a model to detect GPT-2 text
  - Demo: Talk to Transformer
- **GPT-3**
  - Follow-up to GPT-2, remarkably good at generating human-like text
  - A massive model containing 175 billion parameters inside
  - Exclusively licensed by Microsoft and available as an API

End matter

Discussion

> What creative uses for the techniques discussed today do you expect to see become reality in accounting in the next 3-5 years?

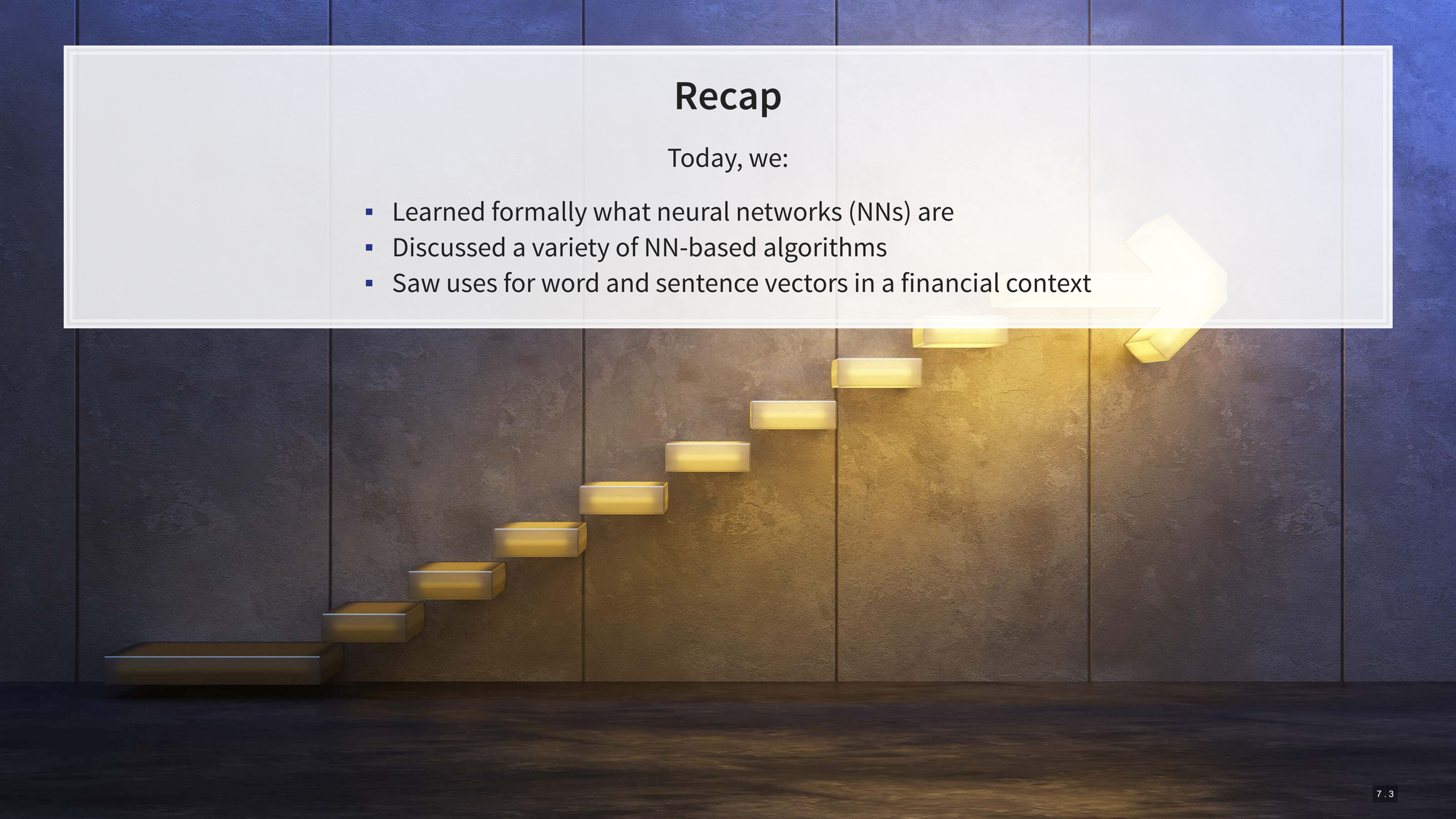
- Brainstorm with your group and try to come up with 1 good use for some technique discussed today
- Each group will be asked to share 1 use



Recap

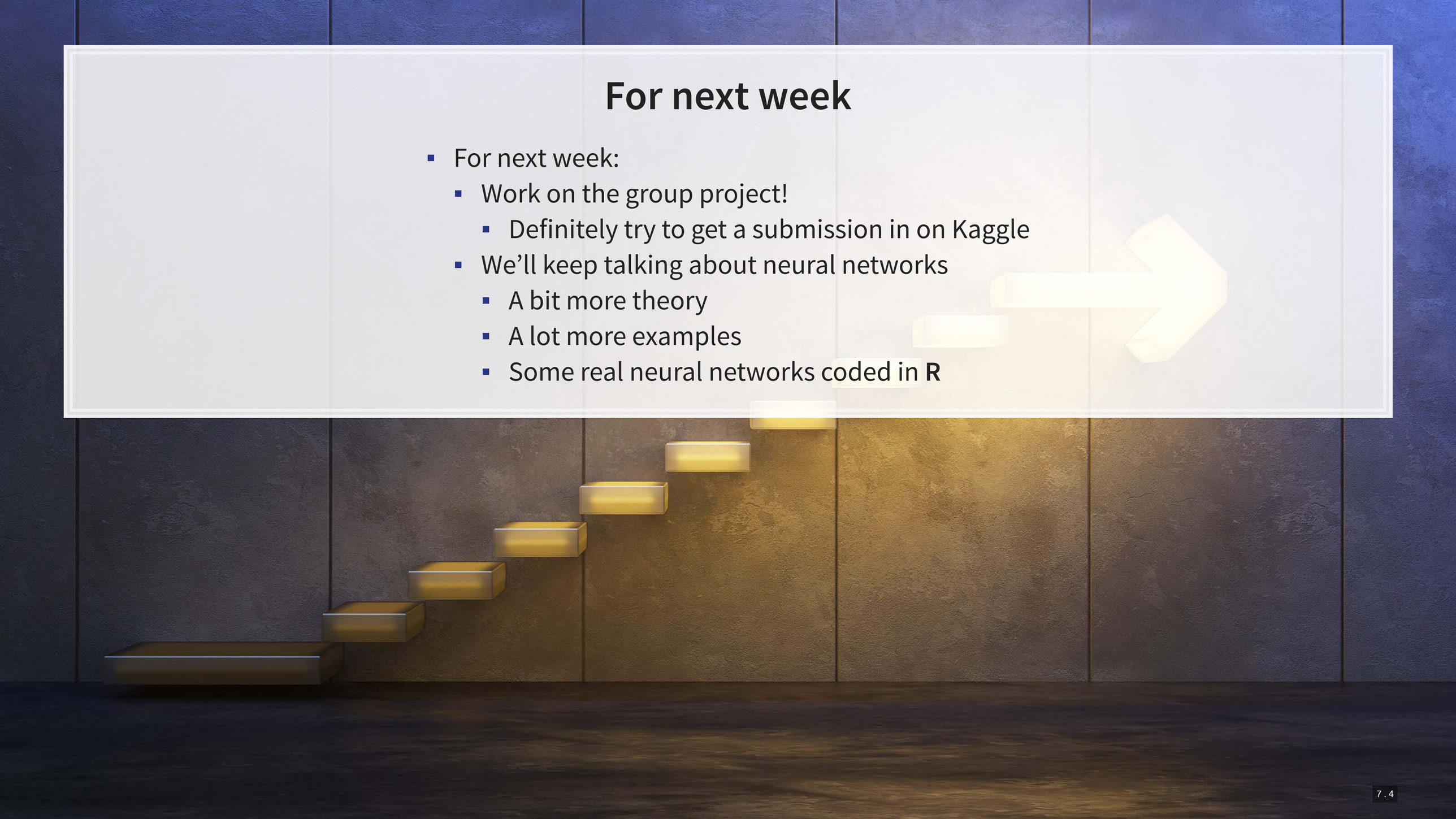
Today, we:

- Learned formally what neural networks (NNs) are
- Discussed a variety of NN-based algorithms
- Saw uses for word and sentence vectors in a financial context



For next week

- For next week:
  - Work on the group project!
    - Definitely try to get a submission in on Kaggle
  - We'll keep talking about neural networks
    - A bit more theory
    - A lot more examples
    - Some real neural networks coded in **R**

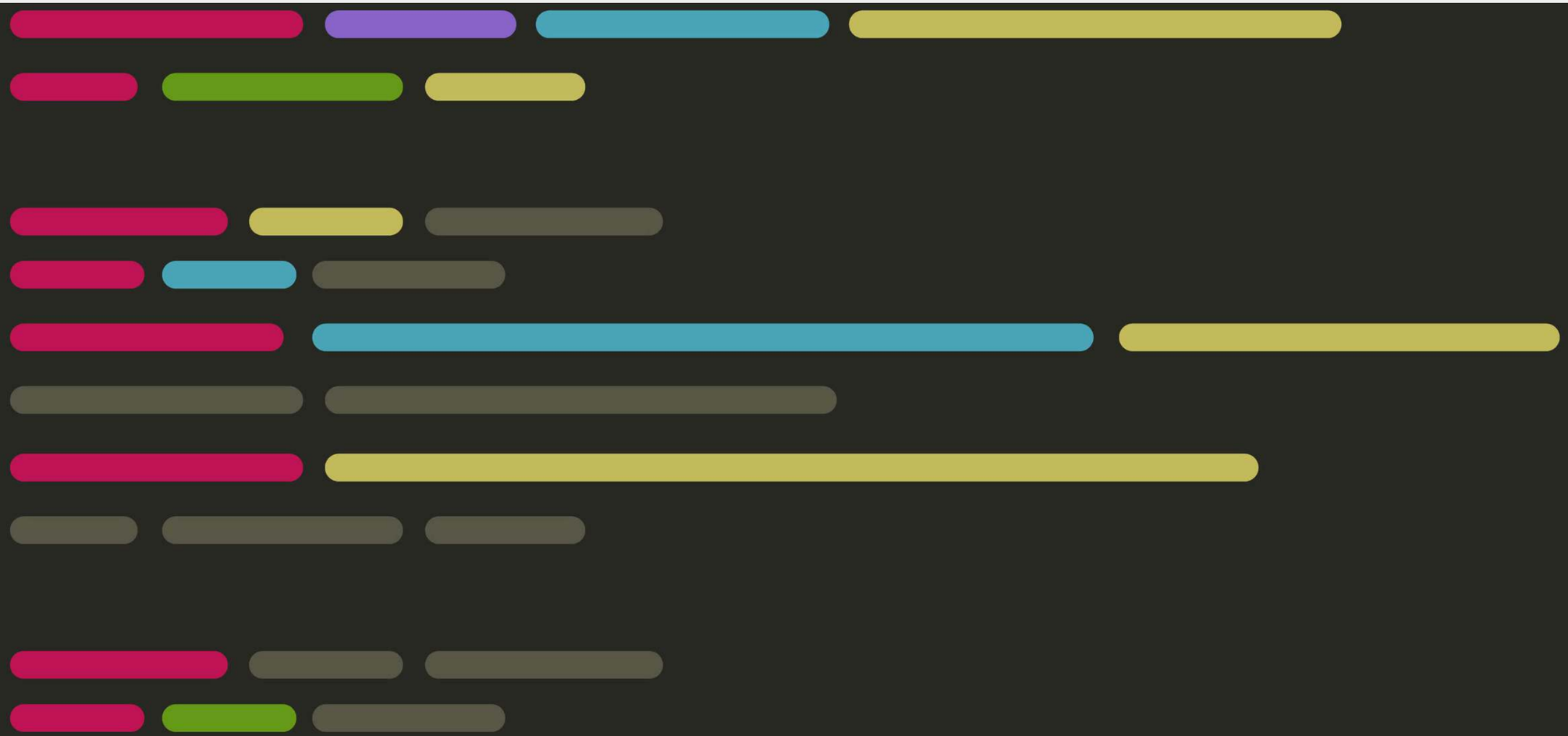


More fun examples

- Interactive:
  - TensorFlow.js examples
  - Semantris
    - A game based on the Universal Sentence Encoder
- Non-interactive
  - Predicting e-sports winners with Machine Learning

Packages used for these slides

- `kableExtra`
- `knitr`
- `tidyverse`



Generating Shakespeare

```
seed_txt = 'Looks it not like the king?  Verily, we must go! '  # Original code
seed_txt = 'SCENE I. Elsinore. A platform before the Castle.\n\n Enter Francisco and Barnardo, two sentinels.\n\nBARNARDO.\nWh
seed_txt = 'Samsung Electronics Co., suffering a handset sales slide, revealed a foldable-screen smartphone that folds like a
# From: https://www.wsj.com/articles/samsung-unveils-foldable-screen-smartphone-1541632221
```