ACCT 420: ML/AI for numbers and text

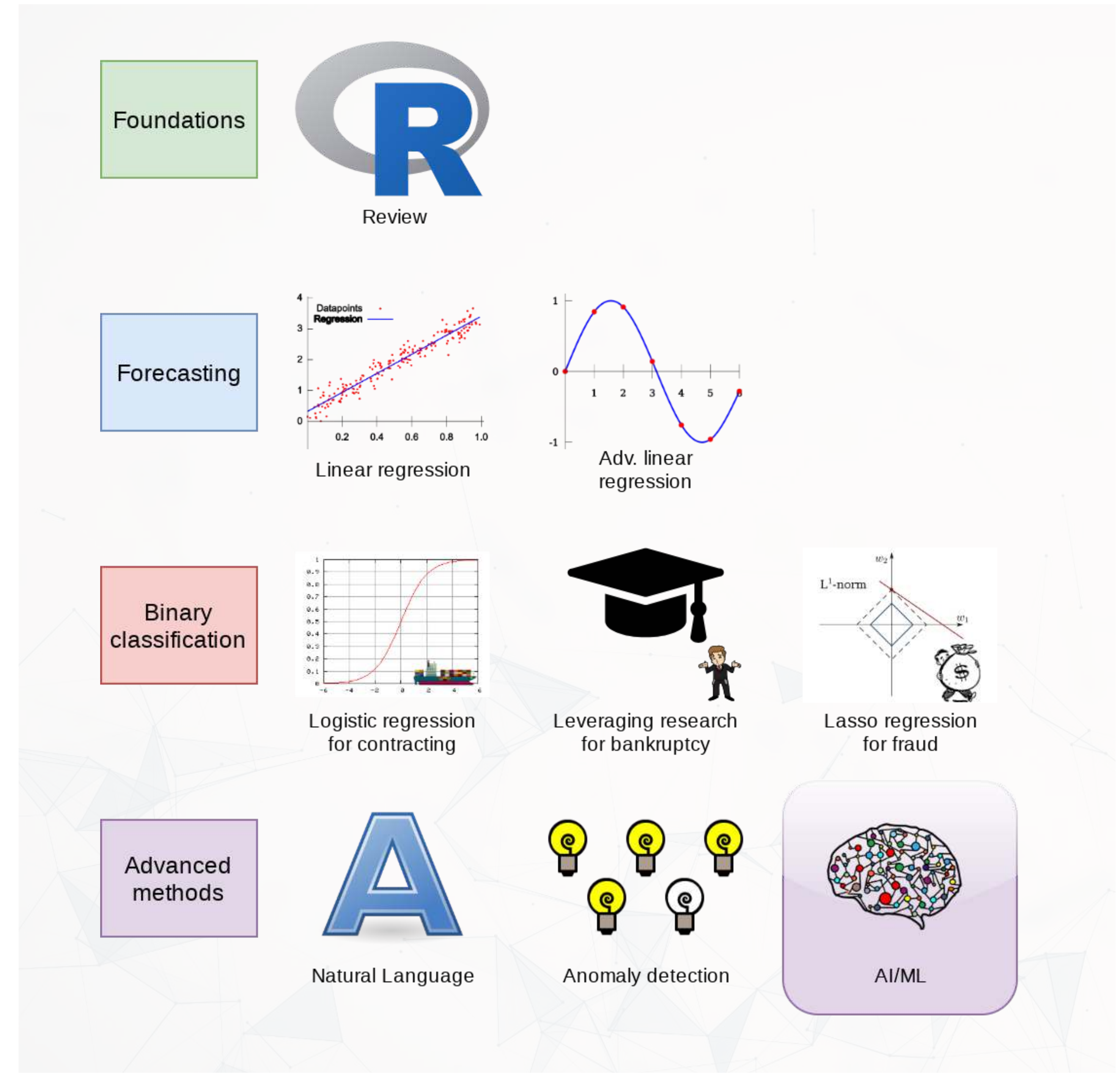
Dr. Richard M. Crowley

rcrowley@smu.edu.sg

https://rmc.link/

Front Matter

Learning objectives



Foundations

Review

Forecasting

Linear regression

Adv. linear regression

Binary classification

Logistic regression for contracting

Leveraging research for bankruptcy

Lasso regression for fraud

Advanced methods

Natural Language

Anomaly detection

AI/ML

- **Theory:**
  - Neural Networks (broad overview)
  - Vector space methods
- **Application:**
  - Neural networks for understanding textual data
  - Annual report sentiment
  - Building our own *mini ChatGPT*
- **Methodology:**
  - Vector methods
  - Neural networks

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
- `{mlverse/torch}`: Plugs in to Torch

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, but often cumbersome
- pytorch – python specific Torch port that is currently very popular
- 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 focus on TensorFlow?

- It can run almost ANY ML/AI/NN algorithm
- It has good community support:
    - 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

TensorFlow Lite

magenta

TensorFlow Hub

> Note: Google appears to be sunsetting this, but many *older* algorithms are based on it. They are shifting toward PyTorch and JAX

About PyTorch

- Based on Torch (for Lua)
- All the companies on the right are on the governing board
- Underpins models by fast.ai, ELF, and AllenNLP
- Easier to use than Tensorflow
- Most *new* off-the-shelf models use it
- Pytorch Hub
    - Has a variety of pretrained models available
    - A bit easier to work with than TensorFlow Hub

AMD
aws
Google Cloud
IBM
intel
Meta
Microsoft Azure
NVIDIA

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
- H2O
  - Python based
  - Integration with R, Scala…

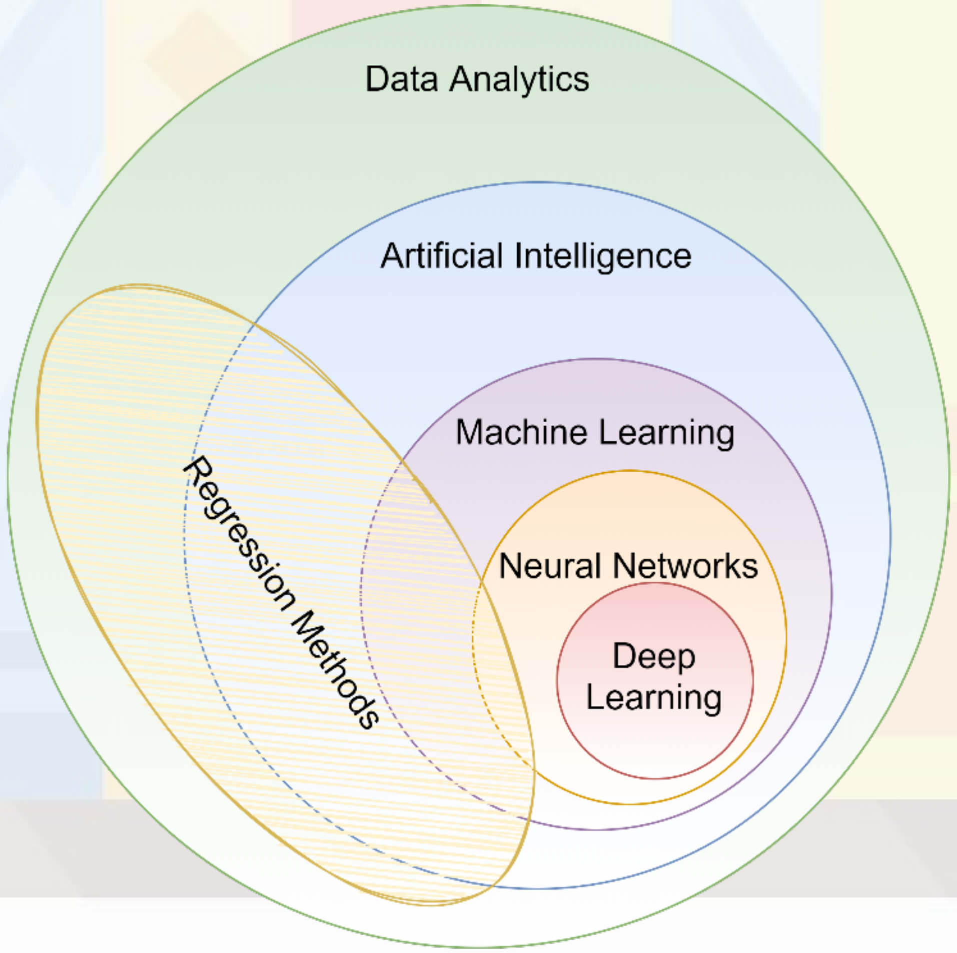
Microsoft
Cognitive
Toolkit
aka.ms/CognitiveToolkit
Microsoft

H_2O.ai

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?

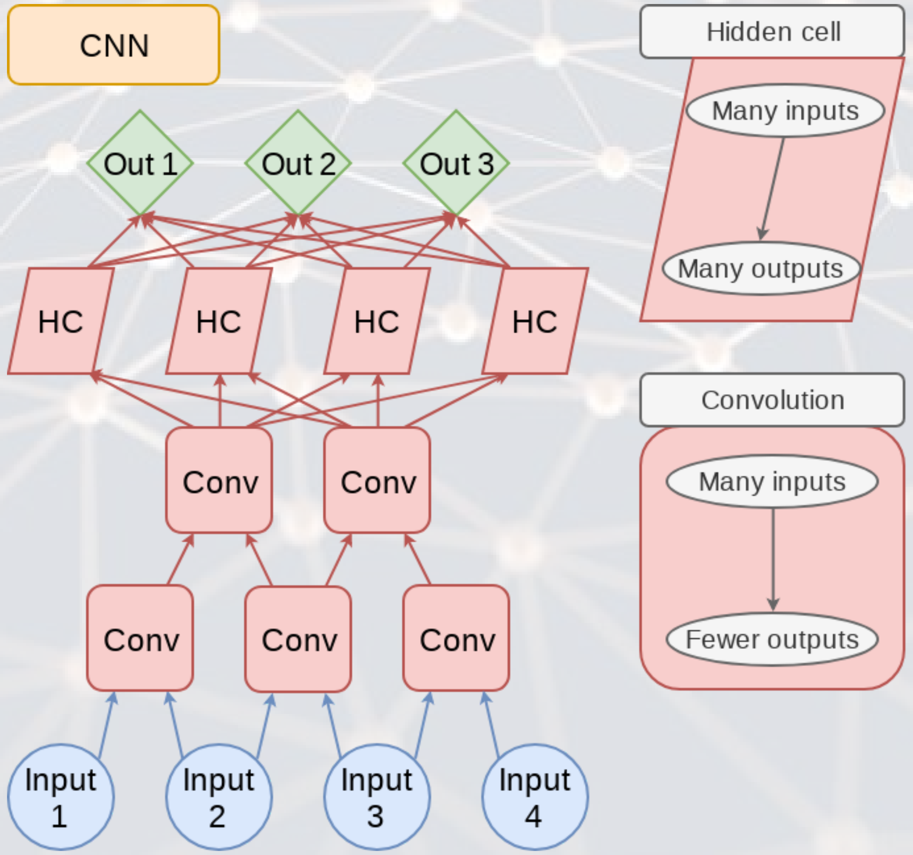
- Neural networks are a method by which a computer can learn from observational data
- In practice:
    - They were not computationally worthwhile until the mid 2000s
    - They have been known since the 1950s (perceptrons)
    - They can be used to construct algorithms that, at times, perform better 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

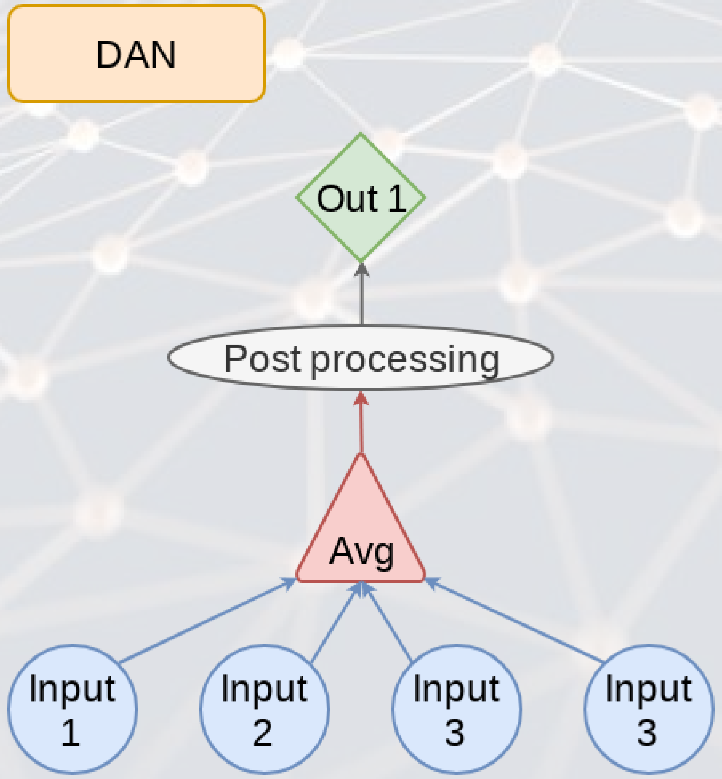
CNN: Convolutional NN

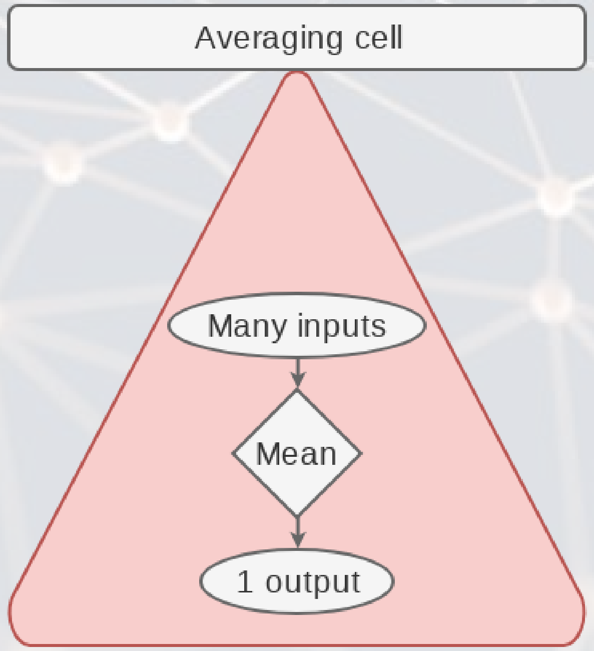
- Networks that excel at object detection (in images)
- Can be applied to other data as well
- Ex.: Inception-v3



DAN: Deep Averaging Network

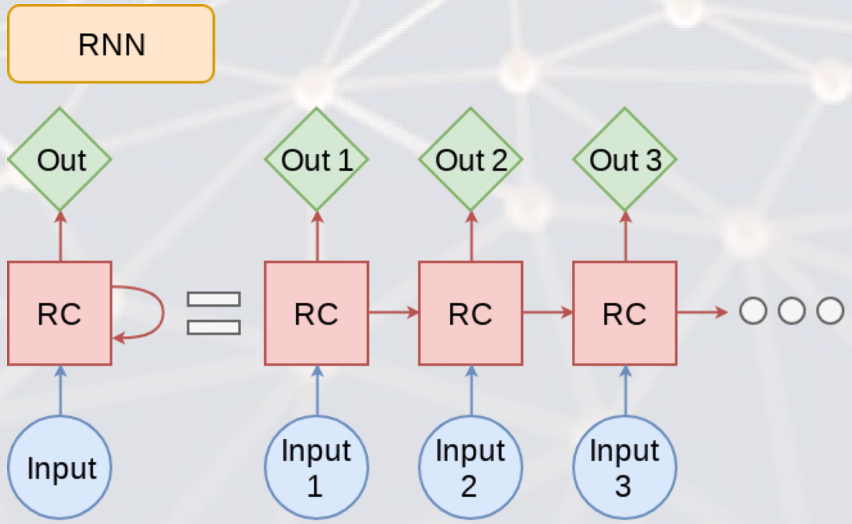
- DANs are simple networks that simply average their inputs
- Averaged inputs are then processed a few times
- These networks have found a home in NLP
  - Ex.: Universal Sentence Encoder

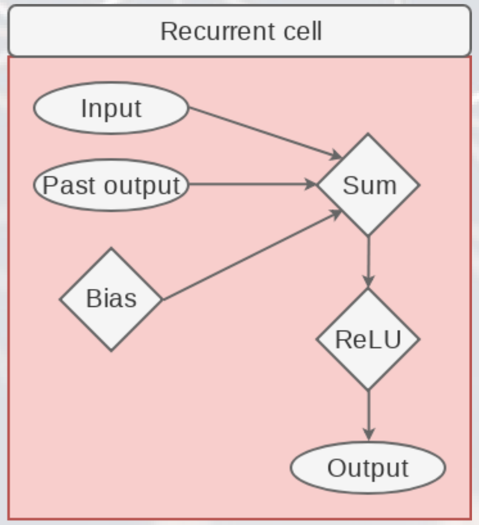




RNN: Recurrent NN

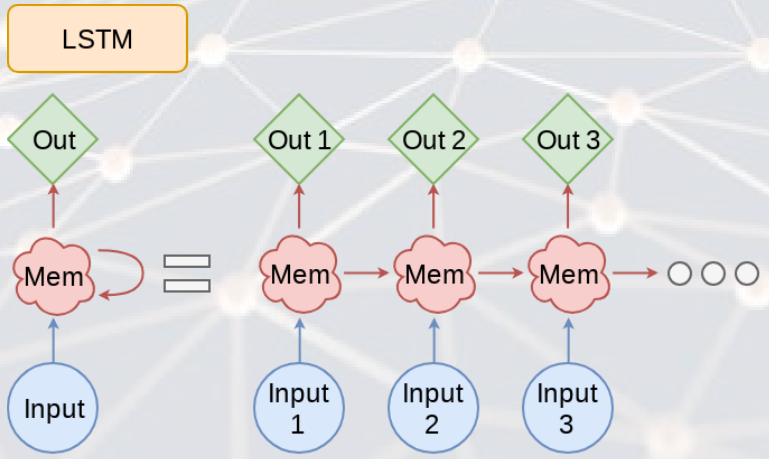
- Recurrent neural networks embed a history of information in the network
  - The previous computation affects the next one
  - Leads to a *short term memory*
- Used for speech recognition, image captioning, anomaly detection, and many others
  - Also the foundation of LSTM
  - SketchRNN (live demo)

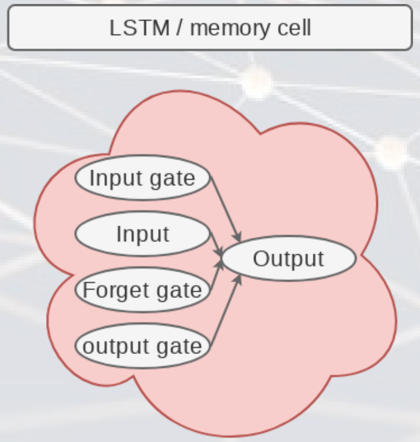




LSTM: Long Short Term Memory

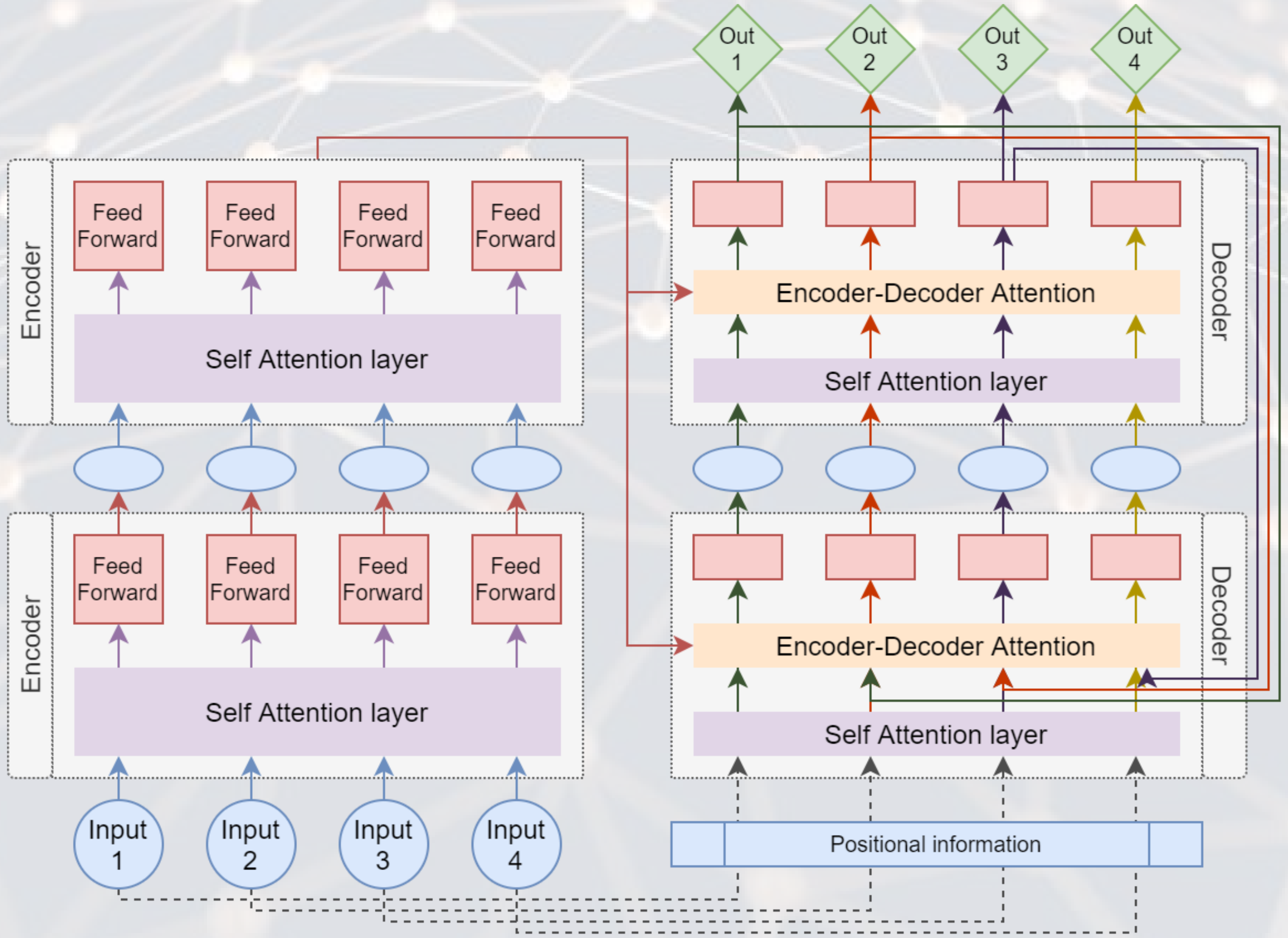
- LSTM improves the *long term memory* of the network while explicitly modeling a *short term memory*
- Used wherever RNNs are used, and then some
  - Ex.: Seq2seq (machine translation)





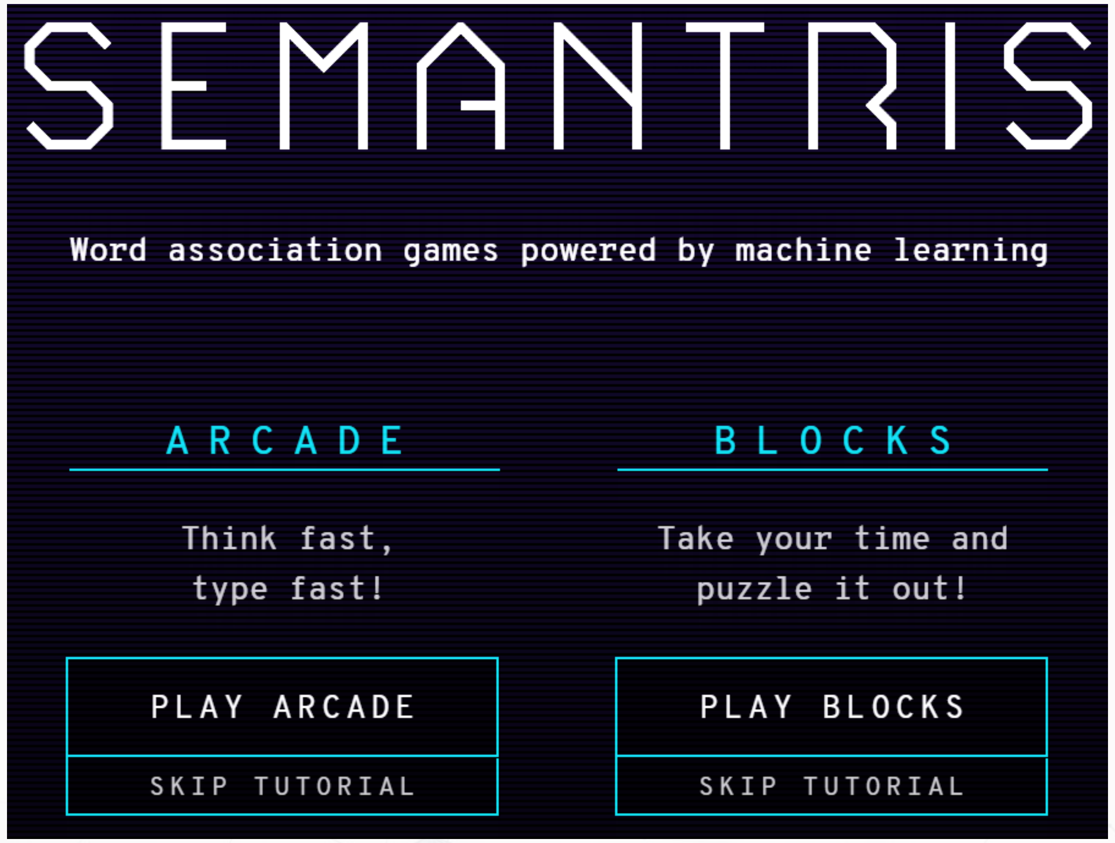
Transformer

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



Vector space models

Motivating examples



SEMANTRIS

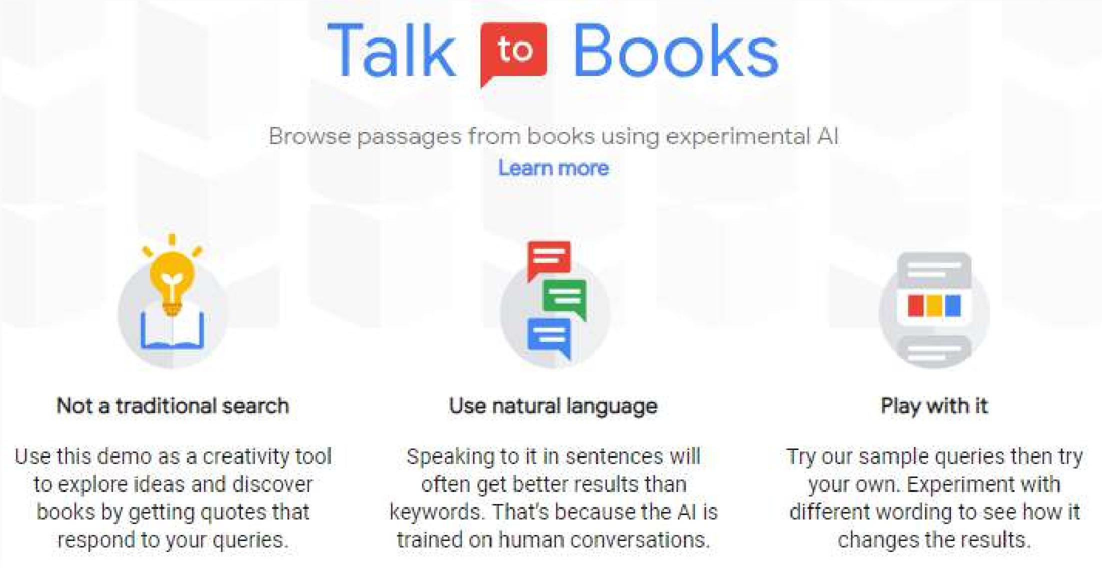
Word association games powered by machine learning

ARCADE

Think fast, type fast!

BLOCKS

Take your time and puzzle it out!



Talk to Books

Browse passages from books using experimental AI

Learn more

Not a traditional search

Use this demo as a creativity tool to explore ideas and discover books by getting quotes that respond to your queries.

Use natural language

Speaking to it in sentences will often get better results than keywords. That's because the AI is trained on human conversations.

Play with it

Try our sample queries then try your own. Experiment with different wording to see how it changes the results.

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
  - Sentence vectors:
    - Universal Sentence Encoder

Word vectors

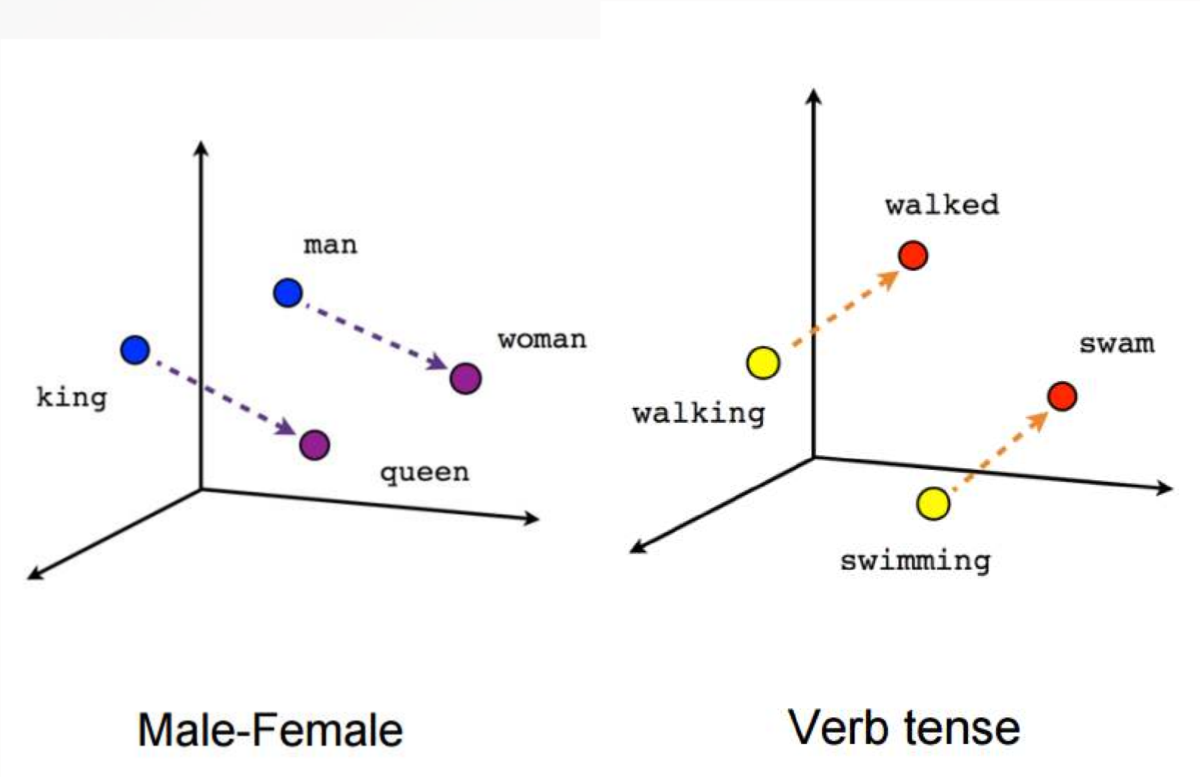
- Instead of coding individual words, encode word meaning
- The idea:
  - Our old way (encode words as IDs from 1 to N) doesn't understand relationships such as:
    - Spatial
    - Categorical
    - Grammatical (weakly when using stemming)
    - Social
    - etc.
  - Word vectors try to encapsulate 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

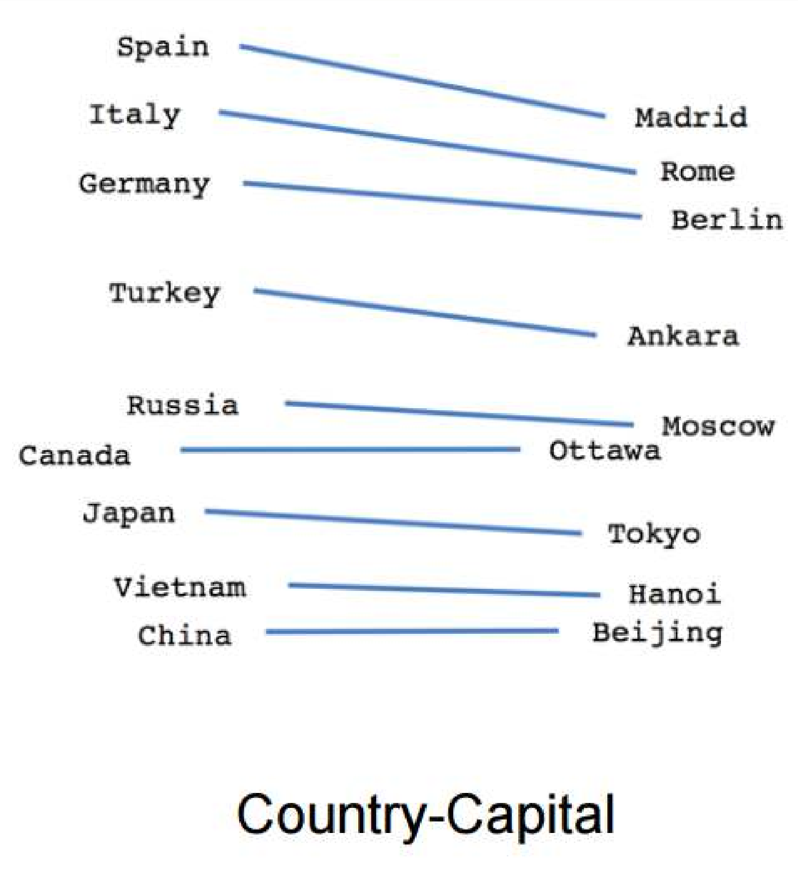
- The above is an idealized example
- Notice how we can tell apart different animals based on their relationship with people
- Notice how we can distinguish turkey (the animal) from Turkey (the country) as well

What it retains: word2vec



Male-Female

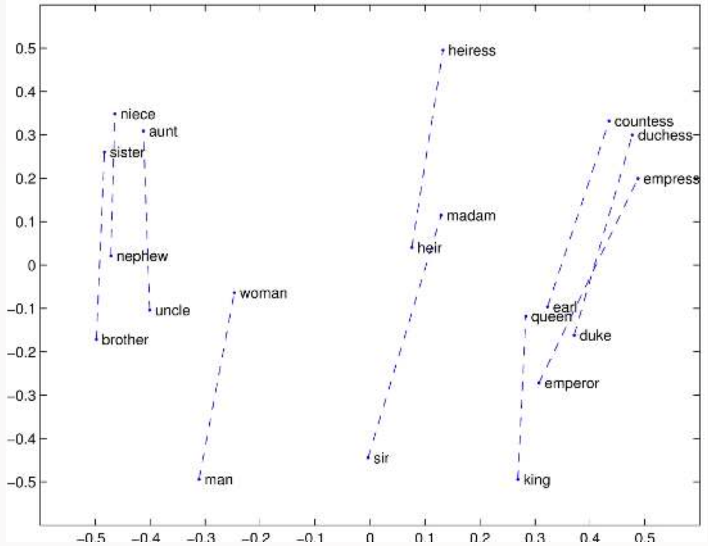
Verb tense

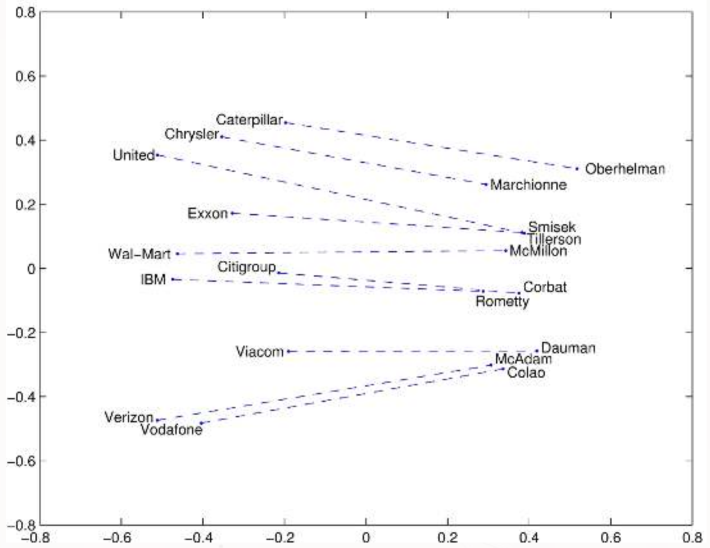


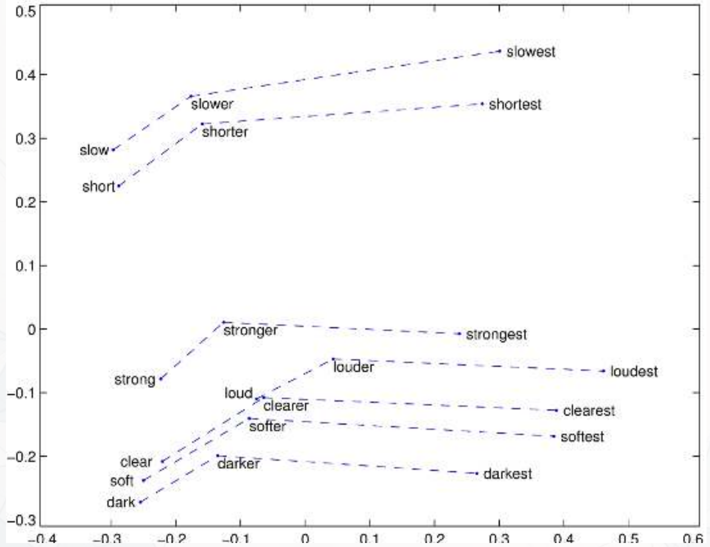
Country-Capital

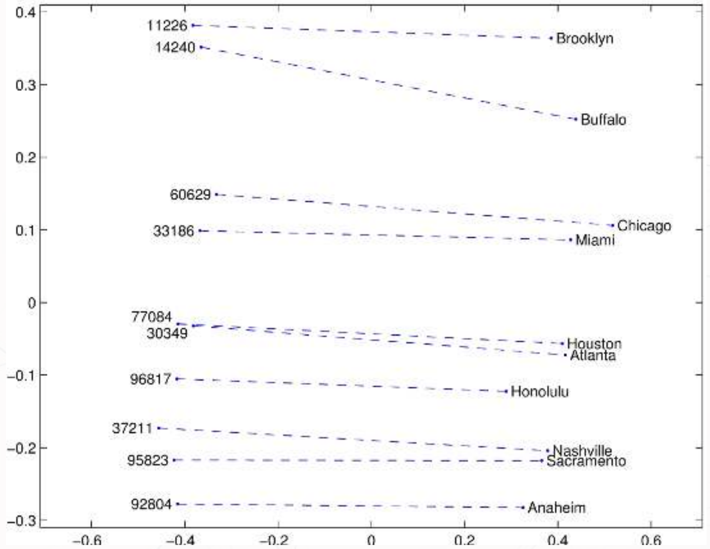
> Relations are retained as vectors between points (distance + direction)

What it retains: GloVe







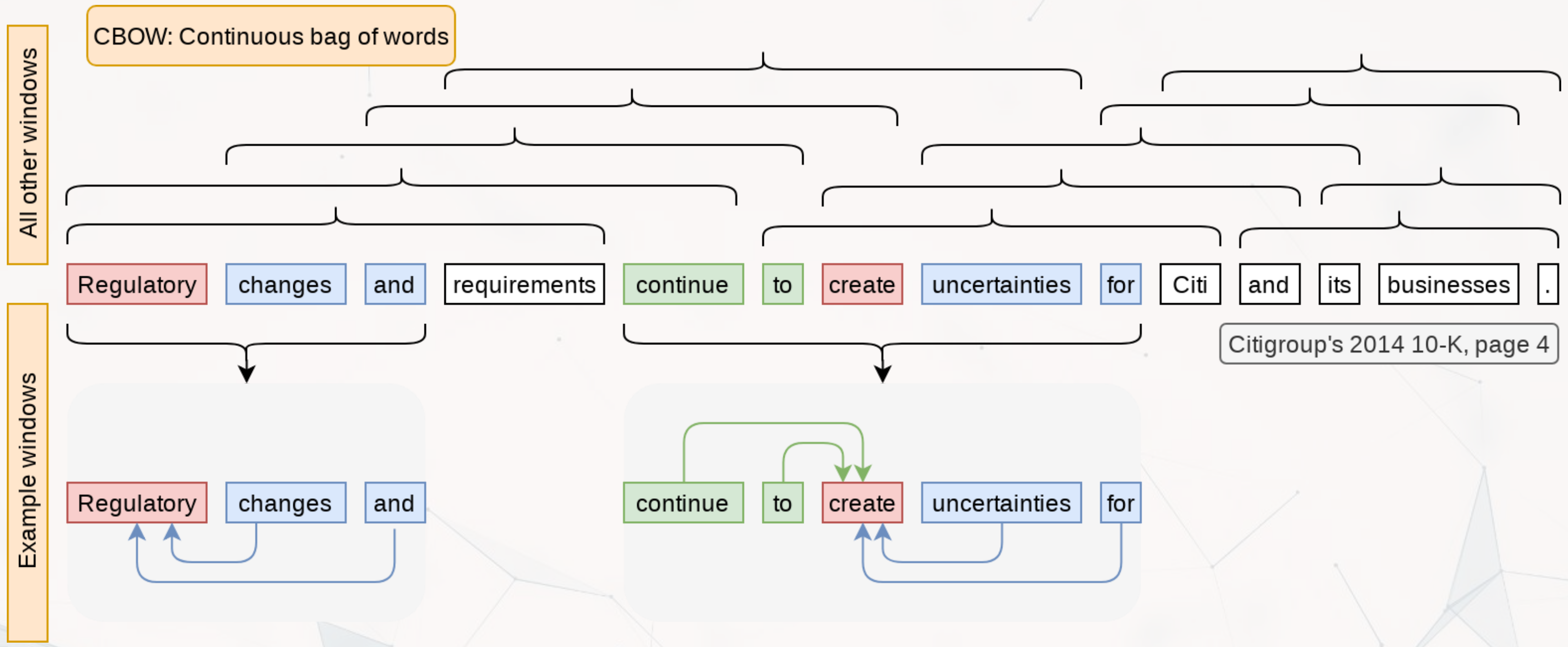


How to build word vectors

- 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?

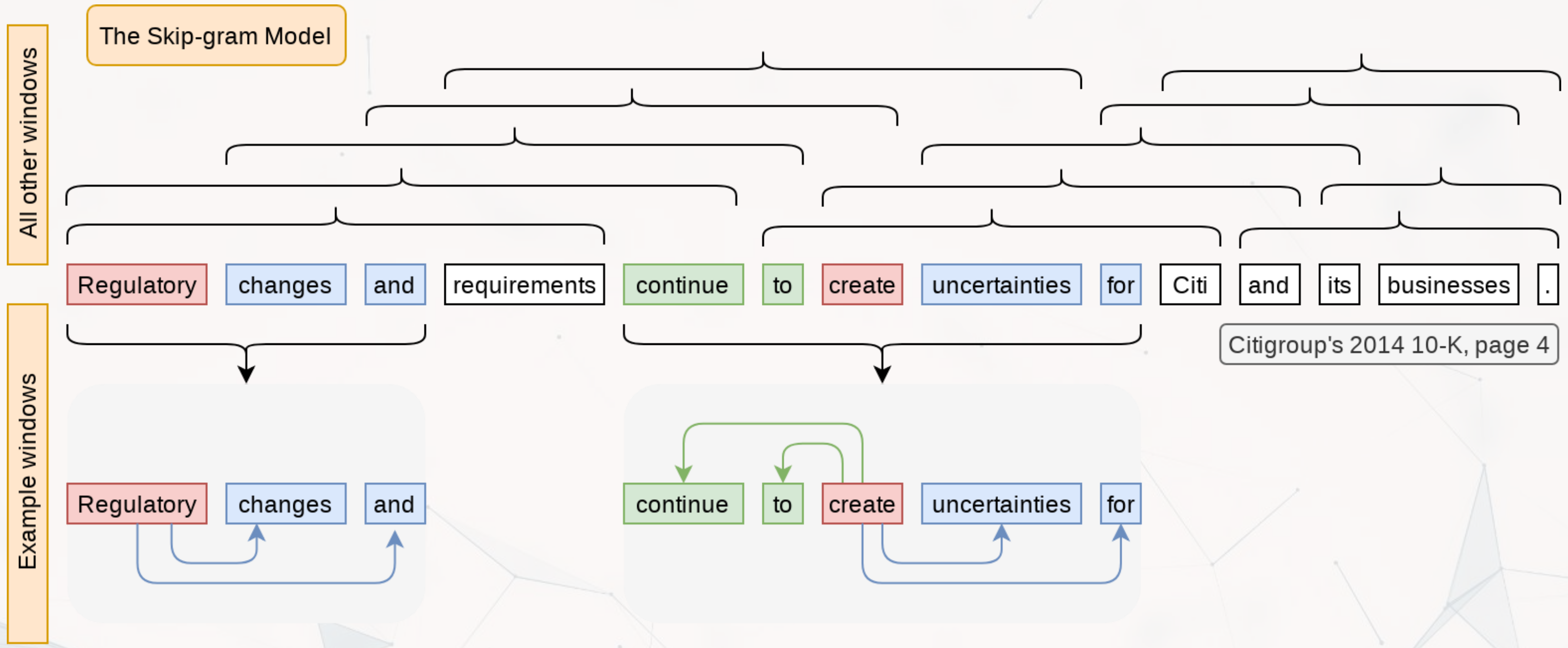
Infer a word’s meaning from the words around it



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

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.

> Demo: rmc.link/colab_w2v

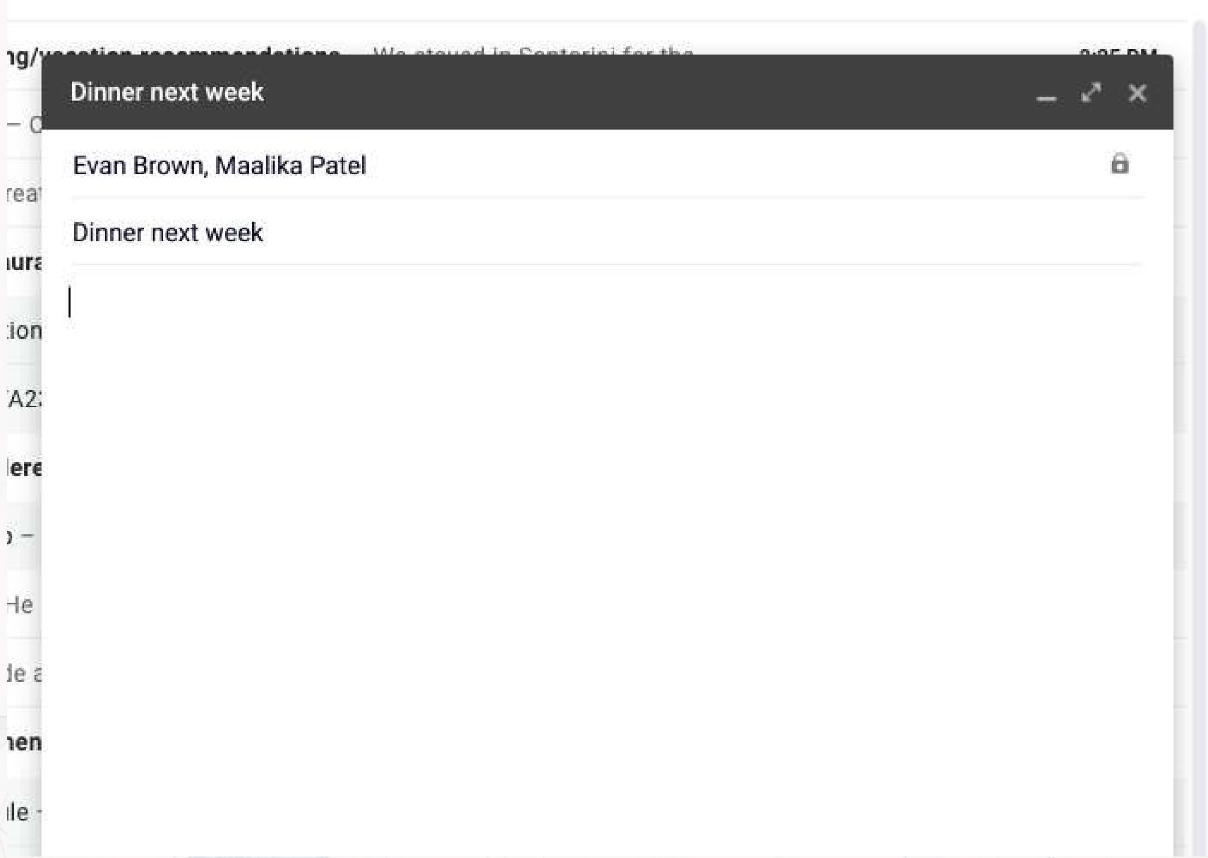
Understanding phrases (or larger)

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
    - Generally learned simultaneously with the word vectors
- This is quite related to what we learned with LDA as well!
  - Both can tell us the topics discussed

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

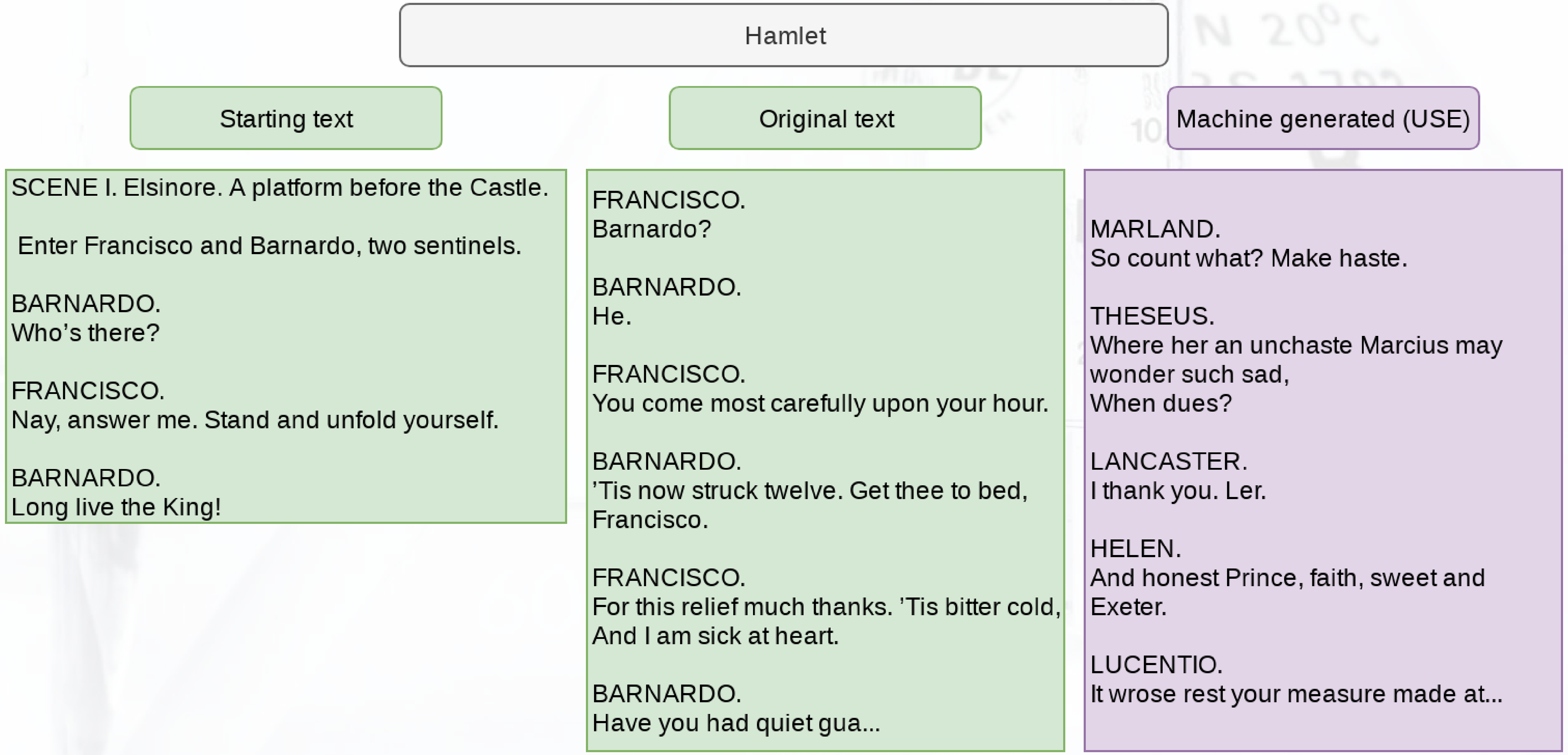


Evan Brown, Maalika Patel

Dinner next week

A fun example of with USE

- Predict Shakespeare with Cloud TPUs and Keras



Hamlet

Starting text

SCENE I. Elsinore. A platform before the Castle.

Enter Francisco and Barnardo, two sentinels.

BARNARDO.
Who’s there?

FRANCISCO.
Nay, answer me. Stand and unfold yourself.

BARNARDO.
Long live the King!

Original text

FRANCISCO.
Barnardo?

BARNARDO.
He.

FRANCISCO.
You come most carefully upon your hour.

BARNARDO.
’Tis now struck twelve. Get thee to bed,
Francisco.

FRANCISCO.
For this relief much thanks. ’Tis bitter cold,
And I am sick at heart.

BARNARDO.
Have you had quiet gua...

Machine generated (USE)

MARLAND.
So count what? Make haste.

THESEUS.
Where her an unchaste Marcius may
wonder such sad,
When dues?

LANCASTER.
I thank you. Ler.

HELEN.
And honest Prince, faith, sweet and
Exeter.

LUCENTIO.
It wrose rest your measure made at...

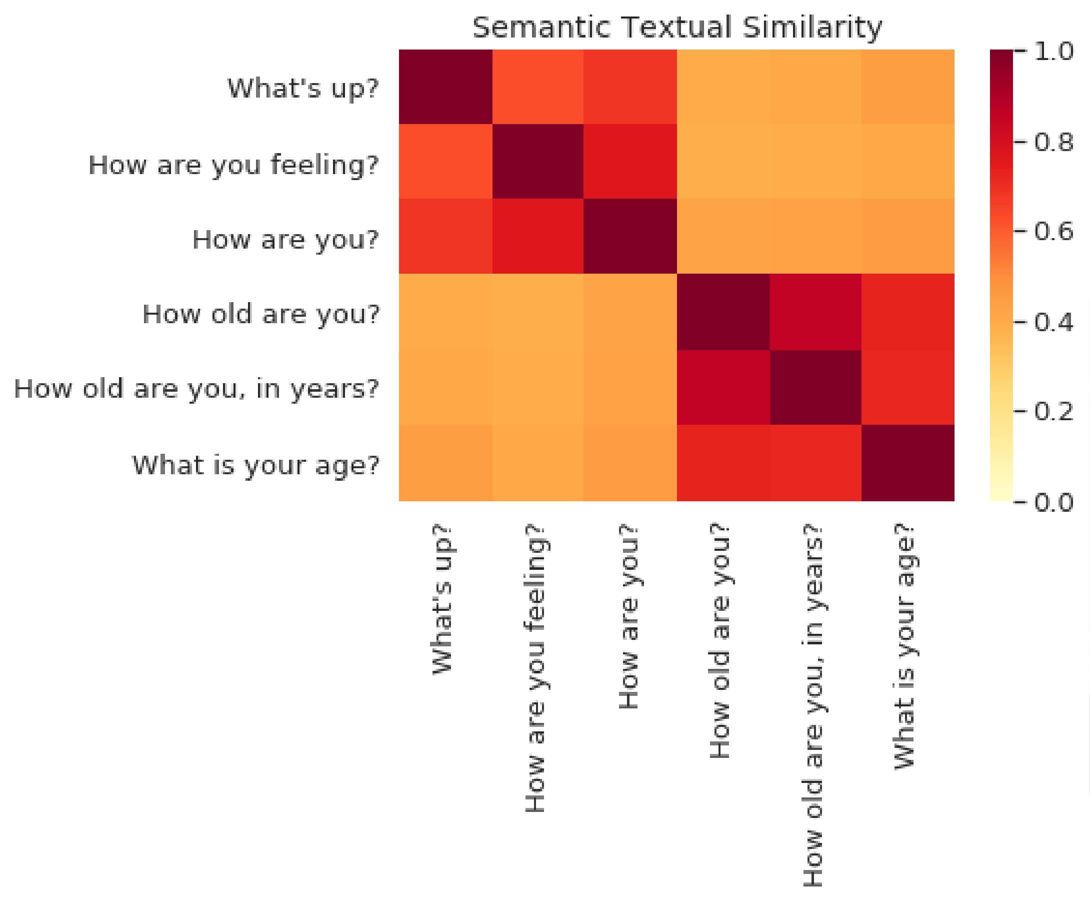
Cavaet on using USE

- 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
  - Learns the meaning of sentences via words' meanings (which it also learns)
- Learn more: Original paper and TensorFlow page
- 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

> Look for the comment `# Add your own messages here`. You can add custom messages to customize the figure and learn what USE understands

Bringing this into accounting

> Understanding how markets react to sentiment-language laden discussion in annual reports

- Crowley and Wong (2023), “Understanding Sentiment through Context”
- Data: Annual report MD&A sections
- Premise: Understand whether *positive* and *negative* discussion are reacted to differently by markets conditional on what is being discussed
- Why USE?: We use it to abstract away from word choice and cluster text by its meaning

> Result: Yes – positive discussion is not always positive, and negative discussion is not always negative

Other Transformer models

- Various **GPT** models (OpenAI)
  - Such as ChatGPT and GPT-4
- **LLAMA 2** (Facebook) – This one is open source
- **PaLM 2** (Google)
  - The model underlying Google’s BARD chat AI
- **Claude v1** (Anthropic)
- **BERT** (Google)
  - Now used for Google Search in at least 70 languages

What is a GPT model?

> A GPT model is a type of *Large Language Model* (**LLM**)

- Large: many parameters in the model (usually >1 billion)
- Language: the models are trained by seeing a large amount of written text
  - They infer everything from language
- Model: It’s just an algorithm like everything else

> What does GPT mean? Generative Pre-trained Transformers

- Generative: It provides answers by generating an answer based on some latent space, as opposed to selecting answers it has previously seen
- Pre-trained: It’s seen a lot of data already. That does not preclude it from seeing more.
- Transformer: It’s based on a *transformer* neural network

What can ____-GPT do?

What can they do

- Classify data based on a small number of examples
  - “Few shot learning”
- Provide answers in flexible/trainable formats
- Encode and decode language
- Pattern matching
- Images as language

What can they not do

- Unless you train it yourself, it won’t have much domain-specific knowledge
- Beat single-purpose SOTA algorithms on most tasks

How do different GPT models vary?

> Context length – the amount of text it can handle at once

- GPT-2: 2,048 tokens
  - ~2 single-spaced pages each for a question and response
- GPT-3: 4,096 tokens
- GPT-3.5: 4,096
- Chat-GPT: 4,096 tokens
- GPT-4: 8,096 or 32,384 tokens
  - The larger model can handle ~30 pages single spaced, each, for a question and response

Let’s build one!

> To build it, go to: rmc.link/colab_gpt2

- This is a simple GPT
  - 12,656 parameters
  - 2 possible tokens
  - A context length of 3

- As a comparison, GPT-2 has:
  - 1.5 billion parameters
  - 50,257 possible tokens
  - a context length of 2,048

How to interpret the network

The arrows show transition from a set of 3 characters (0 or 1) to the next. In this process, the left-most character is dropped, the remaining two characters shift left, and a new character is added to the right side.

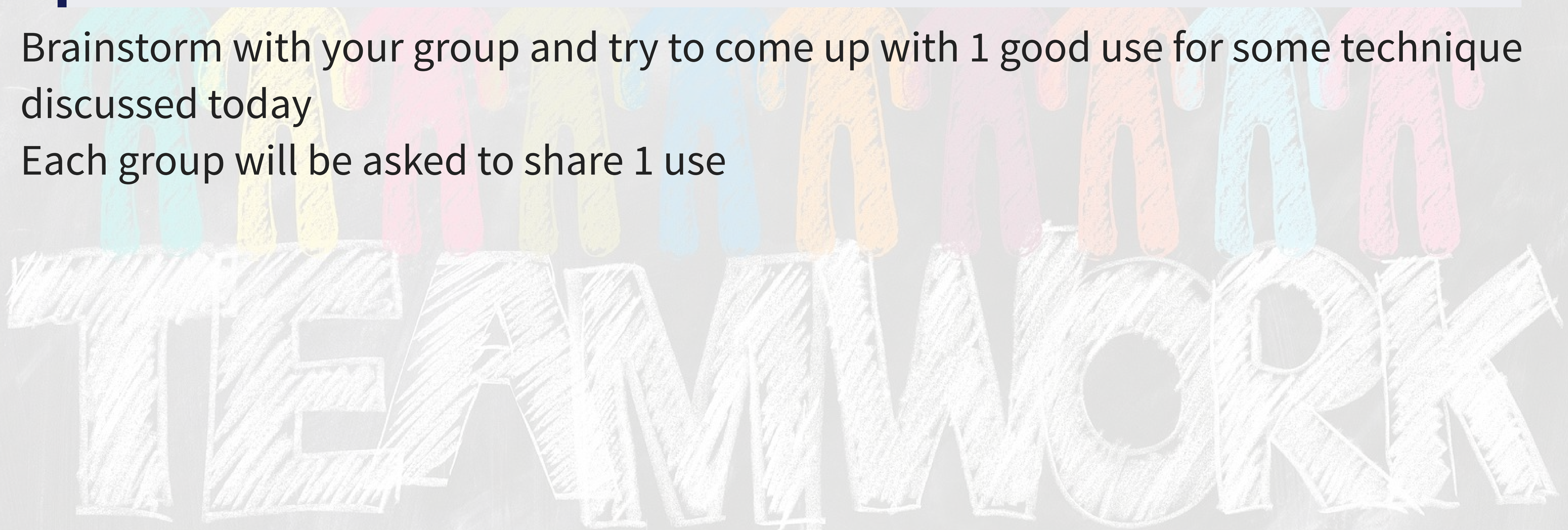
- Look for the following:
  1. That it encodes simple patterns in the data well
  2. That answers are effectively probabilistic
  3. Why hallucination occurs

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?

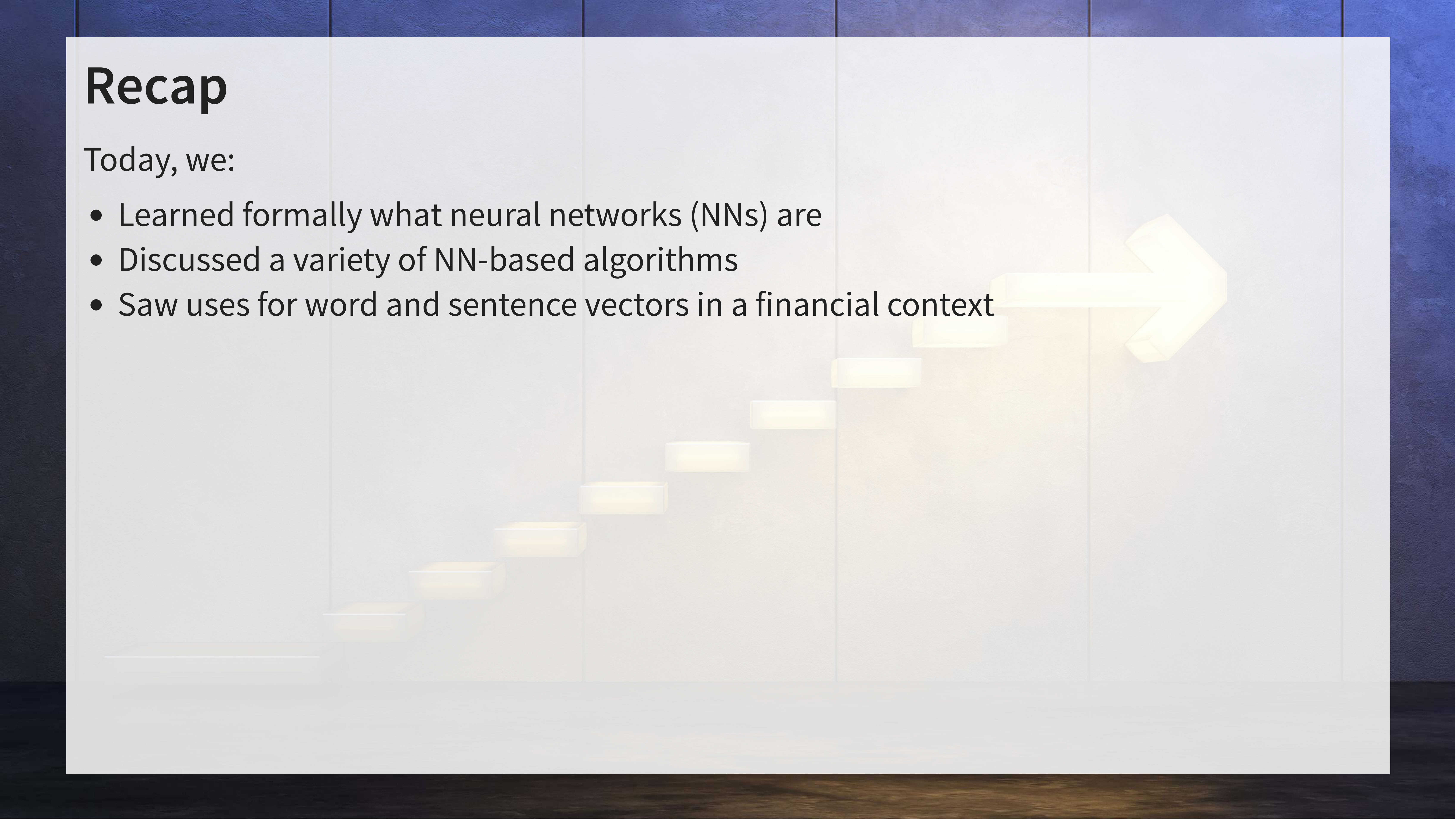
- Brainstorm with your group and try to come up with 1 good use for some technique discussed today
- Each group will be asked to share 1 use



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



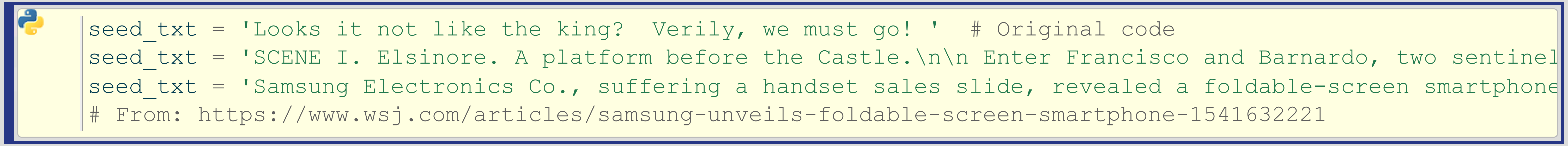
Wrap up

- 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**
- Survey on the class session at this QR code:

Packages used for these slides

- DT
- downlit
- kableExtra
- knitr
- plotly
- quarto
- revealjs
- 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 sentinel
seed_txt = 'Samsung Electronics Co., suffering a handset sales slide, revealed a foldable-screen smartphone
# From: https://www.wsj.com/articles/samsung-unveils-foldable-screen-smartphone-1541632221
```