NLP: A Super-Simple Introduction

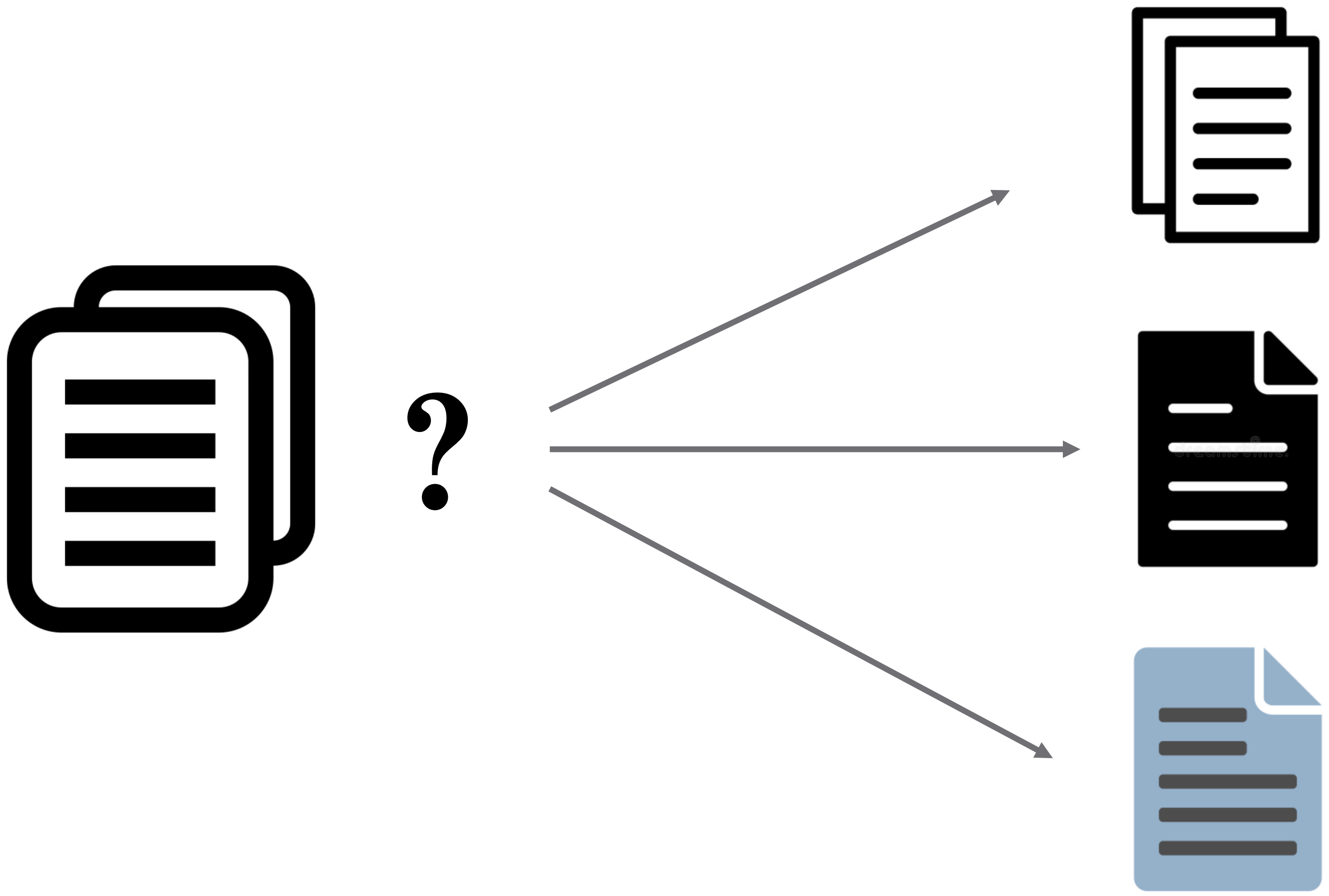
December 5, 2022

Goals for today

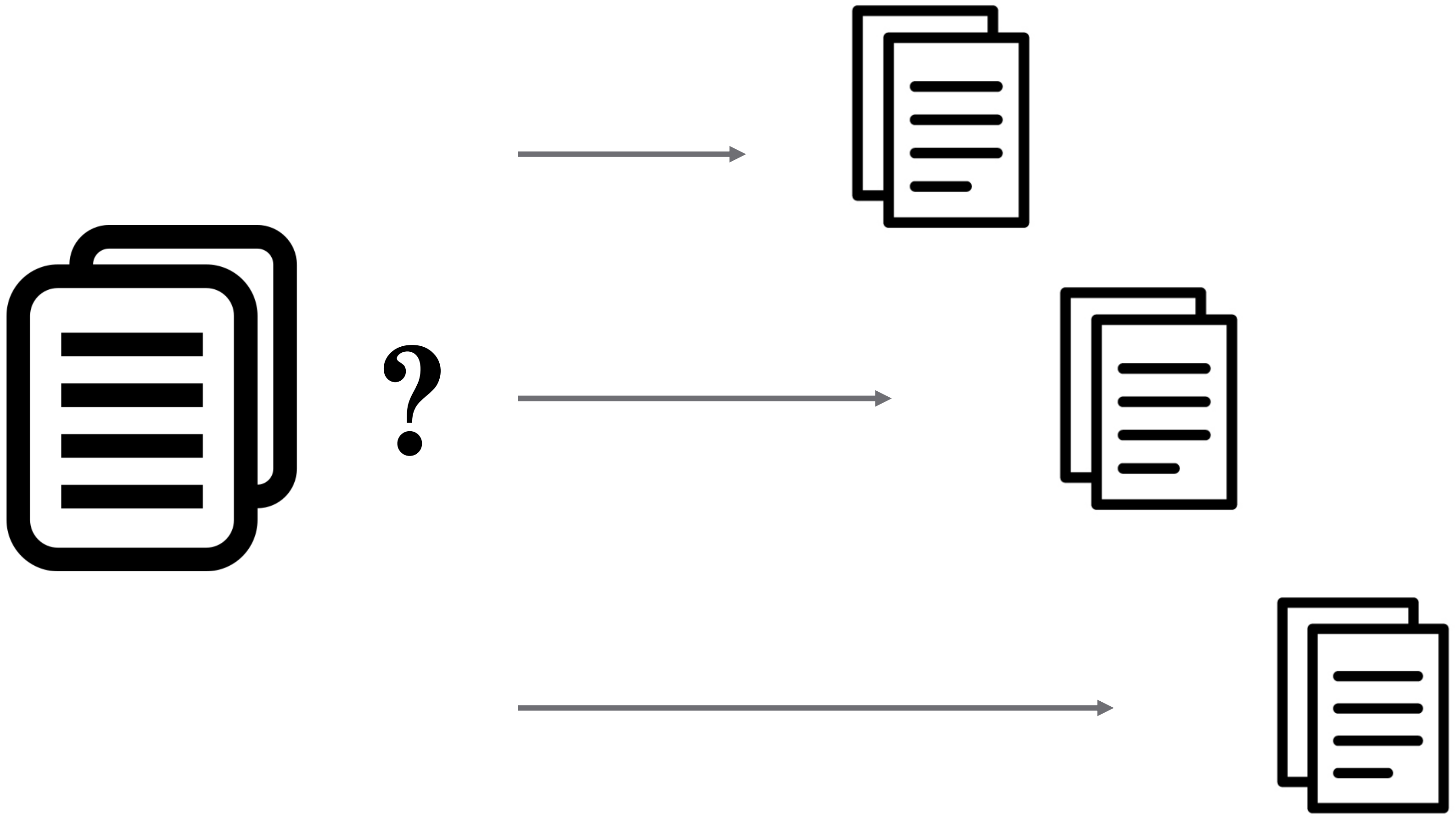
1. Understand the two primary models of language, their assumptions, and when to use them
2. Conceptual understanding of how word embeddings work
3. Conceptual understanding of how embedding models are applied
4. Practical guidance for application and demonstration

Objective

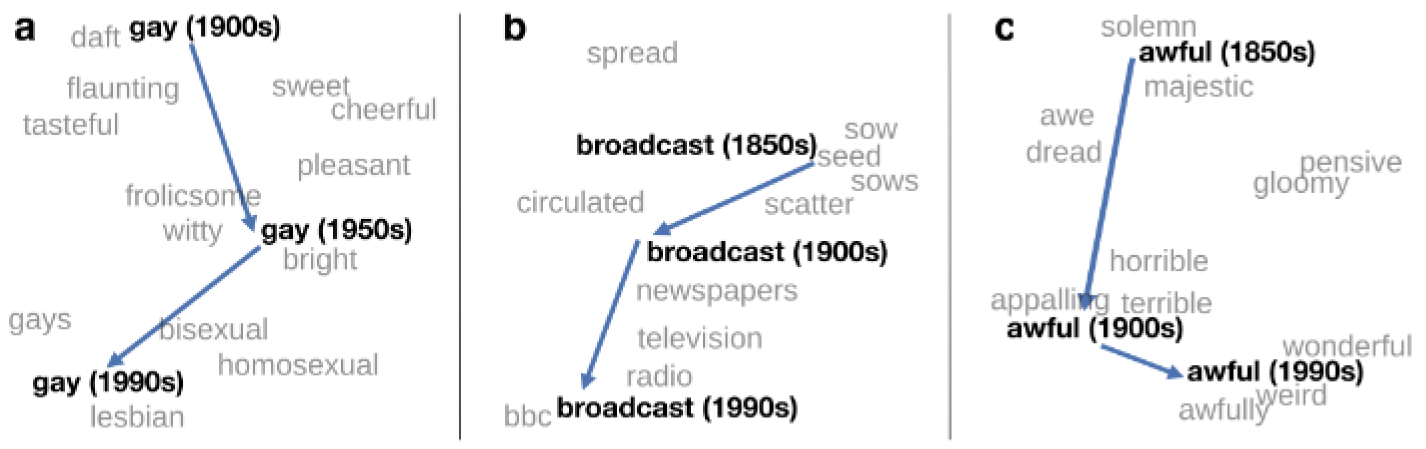
Objective: Word Math



Objective: Word Math



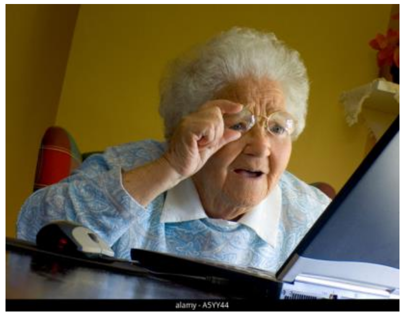
Objective: Word Math



Approaches

“Text as Data”

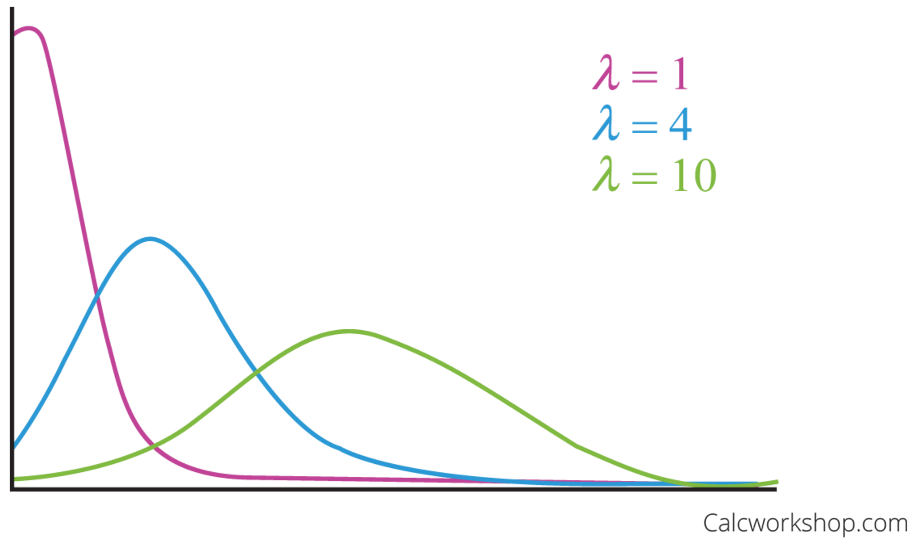
“NLP”



HER: I'M AN NLP PRACTITIONER
ME: (TRYING TO IMPRESS HER) I DON'T KNOW MATH EITHER

Bag-of-Words

- AKA Word counts
- Assumes different types of documents, people, etc. will have different distributions of word use frequency.
- Mostly document or corpus level



Bag-of-Words: Advantages

- Very fast
- Descriptive analysis
- Very large documents
- Teaches manual text processing and document cleaning

Bag-of-Words: Disadvantages

- Mostly ignores context
- Generally poor model of semantics
- Generally bad for short documents

$$y_{ijt} \sim Poisson(\lambda_{ijt})$$
$$\lambda_{ijt} = \exp(\alpha_{it} + \psi_j + \beta_j * \omega_{it})$$

wordfish

Bag-of-Words: Disadvantages

Embeddings

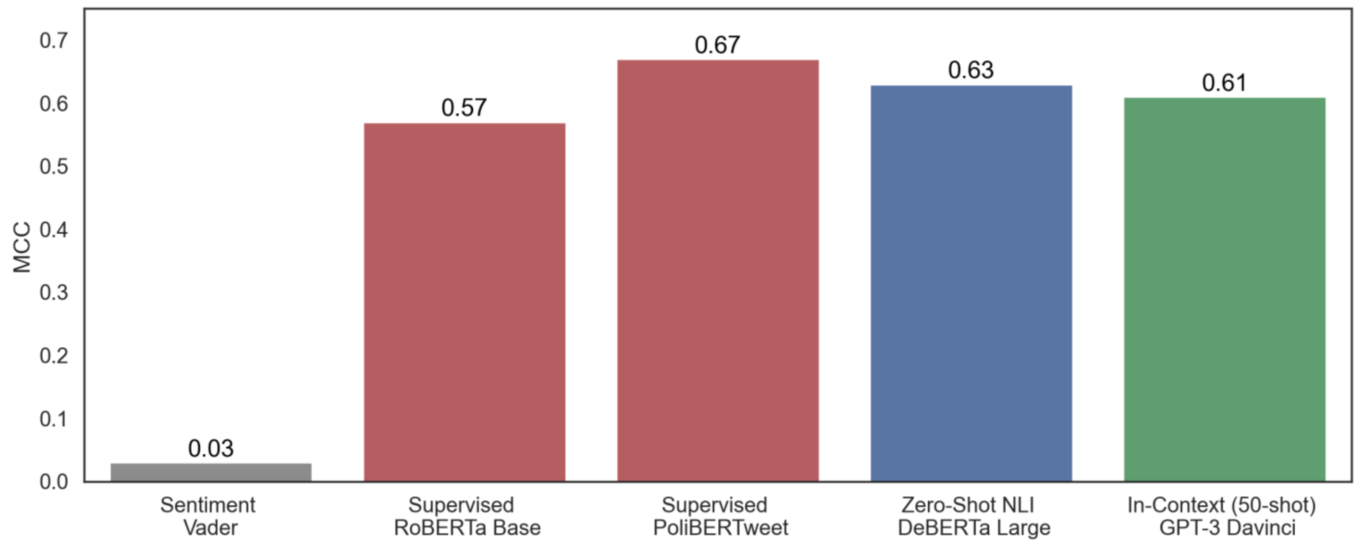
- AKA Distributional semantics, distributional methods, word embeddings, word vectors
- Each word (token) is represented by an n dimensional vector.
- The value for each dimension is determined by which words are commonly found next to the word in a k sized window.
- Word level (generally)

Embeddings

	Animal	Fluffy	Pet	Dangerous	Food
Cat	.98	.94	.87	.30	.15
Dog	.98	.87	.95	.35	.15
Pig	.95	.20	.45	.26	.60
Carrot	.01	.02	.01	.01	.97
Rock	.01	.02	.43	.25	.01

Embeddings: Advantages

- More theoretically robust model of language
- Far better at semantic representation
- Excellent with small texts
- Almost entirely eliminate pre-processing



Embeddings: Disadvantages

- Much more computationally intensive
- Document size limits
- Black box
- Neural networks can be hard

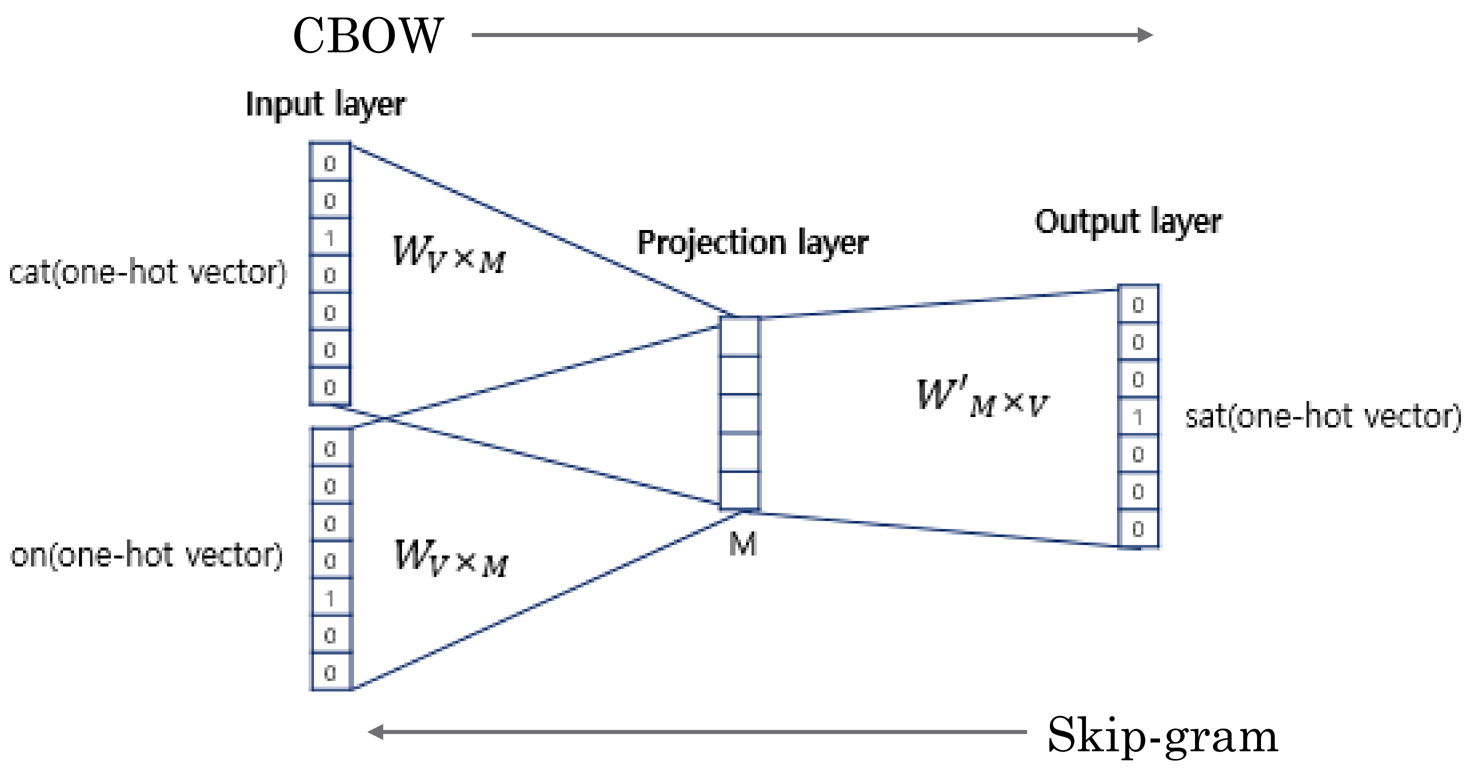
Practical advice

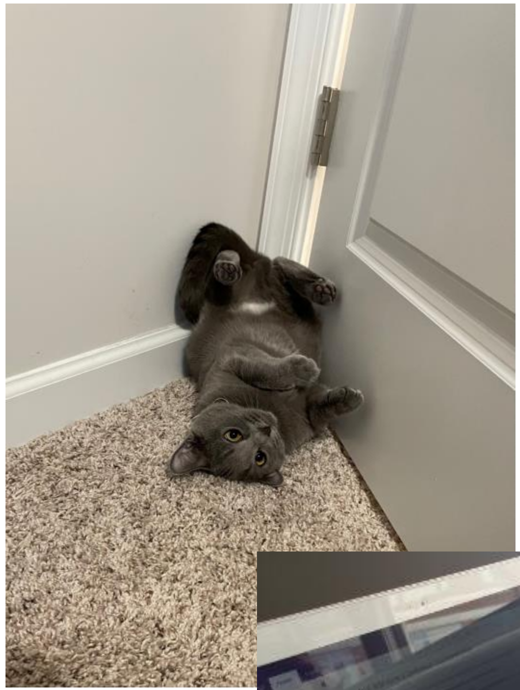
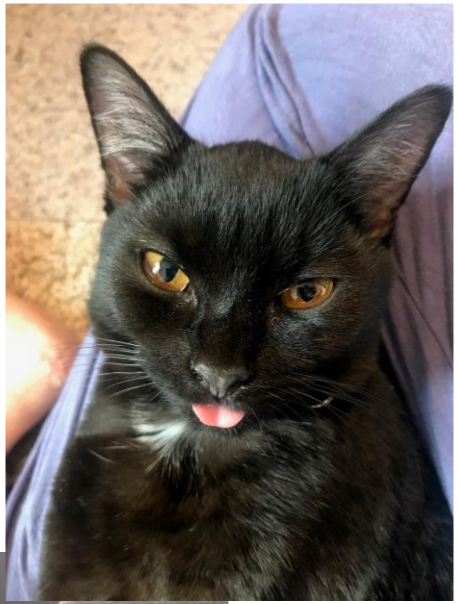
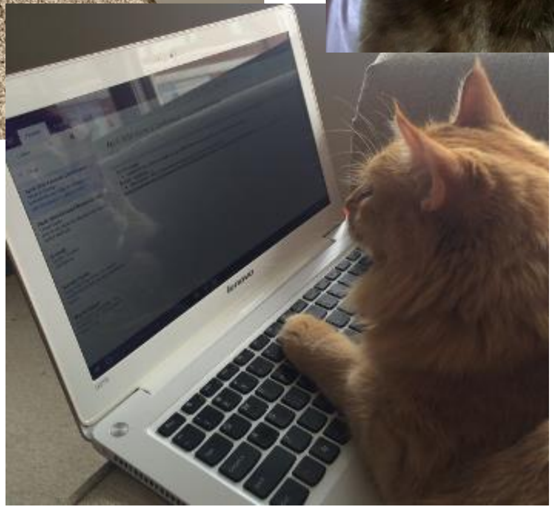
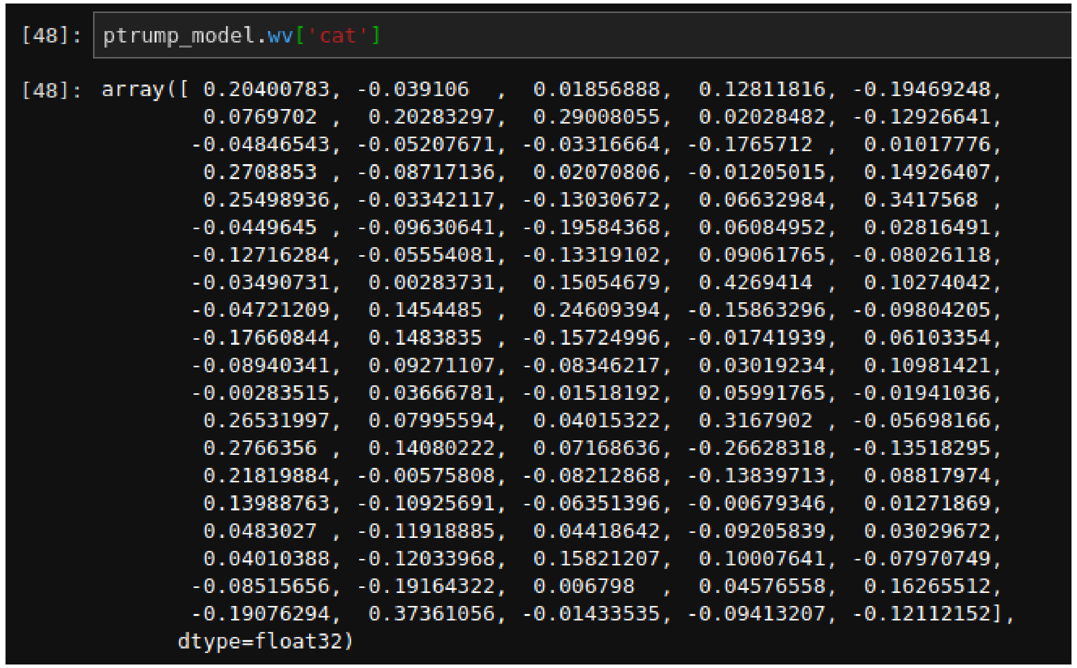
- Use embedding methods as a default
- Bag of Words is still very useful for descriptive analysis
- Bag of Words is still useful for particularly long documents
- Refer to the computer science literature
- Co-author!

Embeddings

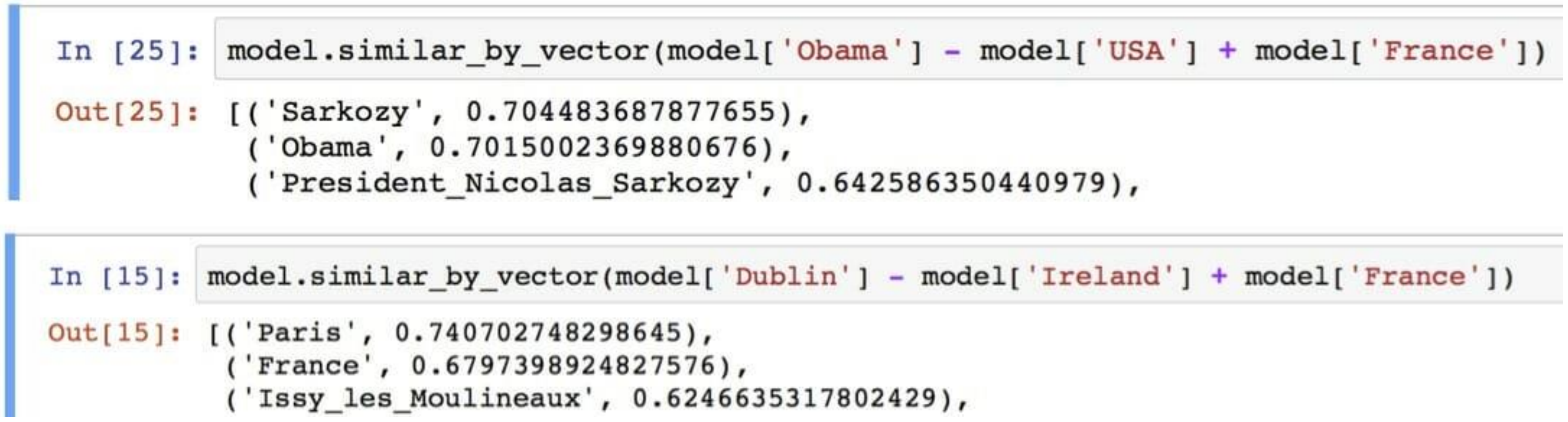
Word2Vec

“The cat sat on my lap”



```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
             0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
            -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
             0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
             0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
            -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
            -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
            -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
            -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
            -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
            -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
            -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
             0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
             0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
             0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
             0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
             0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
             0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
            -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
            -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
           dtype=float32)
```

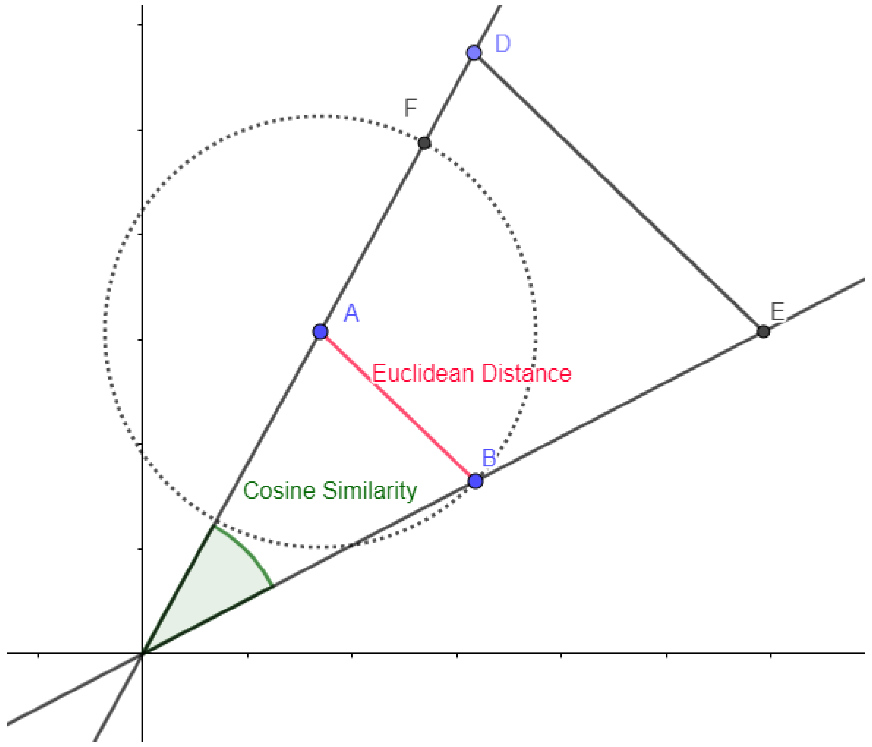


```
In [25]: model.similar_by_vector(model['Obama'] - model['USA'] + model['France'])

Out[25]: [('Sarkozy', 0.704483687877655),
          ('Obama', 0.7015002369880676),
          ('President_Nicolas_Sarkozy', 0.642586350440979),
```

```
In [15]: model.similar_by_vector(model['Dublin'] - model['Ireland'] + model['France'])

Out[15]: [('Paris', 0.740702748298645),
          ('France', 0.6797398924827576),
          ('Issy_les_Moulineaux', 0.6246635317802429),
```

Cosine similarity

- Computing the similarity between words and documents is foundational to everything we do in text analysis
  - Classification
  - Clustering
  - Description
- Cosine (orientation)
- Euclidean (magnitude)
- We use cosine because it is more robust to differences in document size

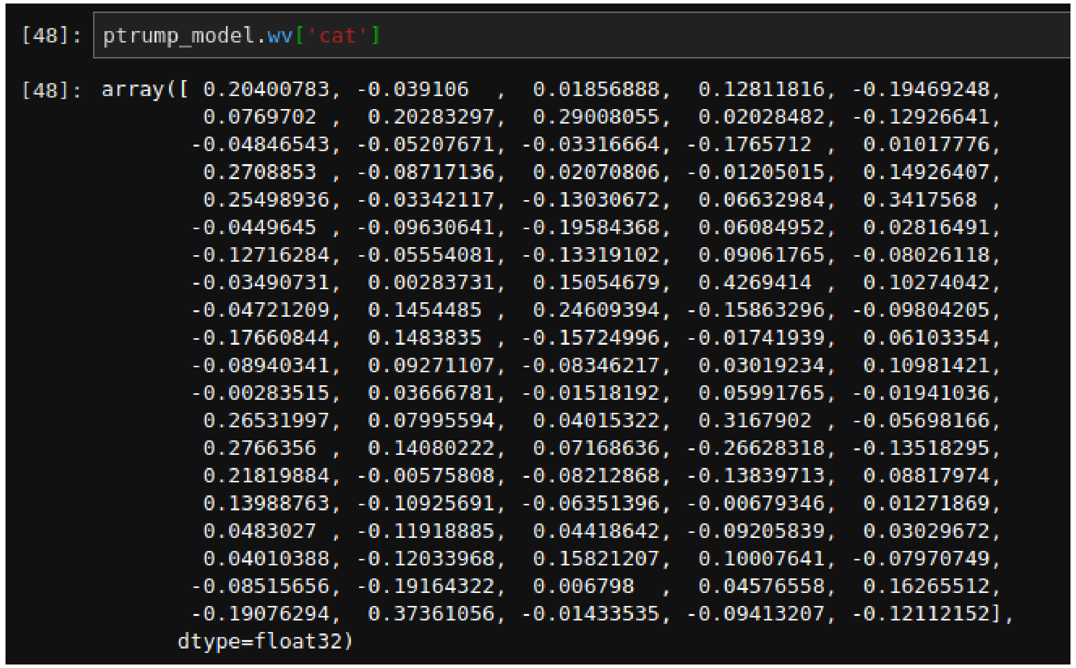


The problem with cats

- “This is my cat Popkins”
- “Who let the cat out of the bag?”
- “Joe is one cool cat”
- “Cats are powerful machines”
- “It’s raining cats and dogs”

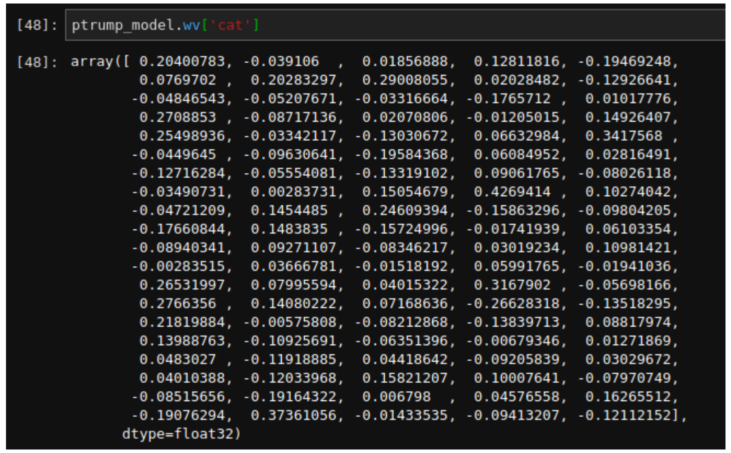
Global embeddings

=



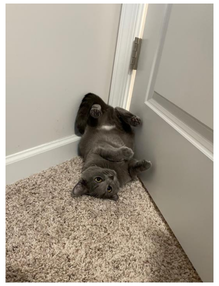
```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
         0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
        -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
         0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
         0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
        -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
        -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
        -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
        -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
        -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
        -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
        -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
         0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
         0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
         0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
         0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
         0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
         0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
        -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
        -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
      dtype=float32)
```

Contextual embeddings

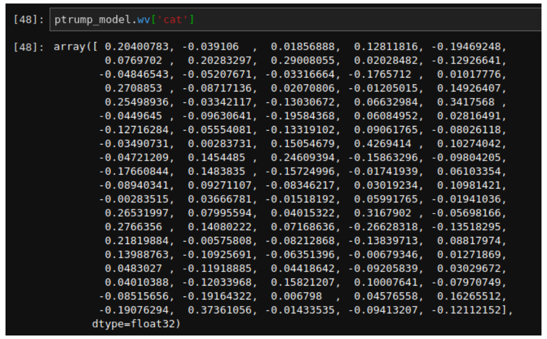


```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
         0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
        -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
         0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
         0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
        -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
        -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
        -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
        -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
        -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
        -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
        -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
         0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
         0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
         0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
         0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
         0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
         0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
        -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
        -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
       dtype=float32)
```

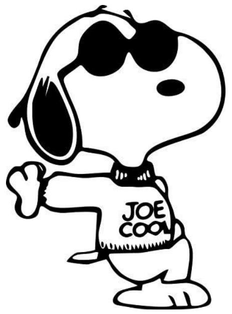
+



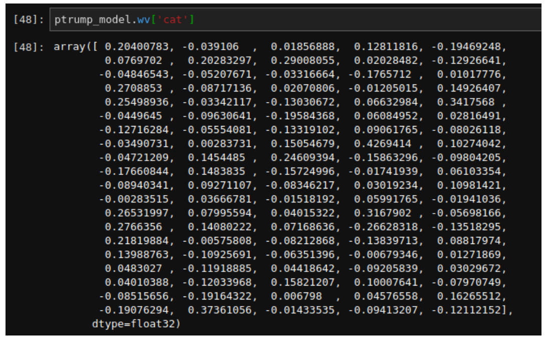
=



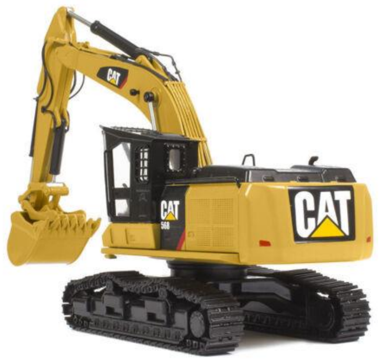
```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
         0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
        -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
         0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
         0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
        -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
        -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
        -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
        -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
        -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
        -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
        -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
         0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
         0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
         0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
         0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
         0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
         0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
        -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
        -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
       dtype=float32)
```



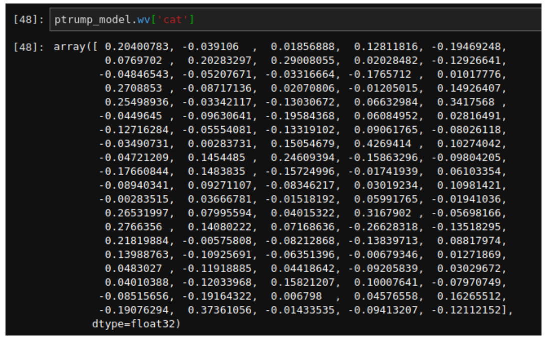
=



```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
         0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
        -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
         0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
         0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
        -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
        -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
        -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
        -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
        -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
        -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
        -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
         0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
         0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
         0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
         0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
         0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
         0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
        -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
        -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
       dtype=float32)
```



=



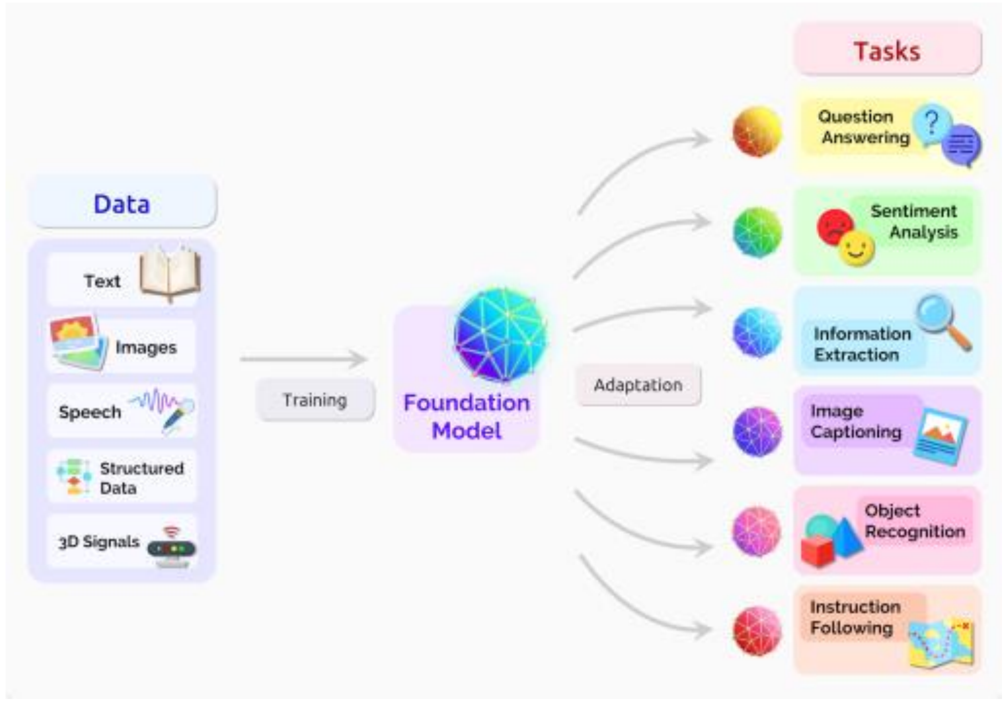
```
[48]: ptrump_model.wv['cat']

[48]: array([ 0.20400783, -0.039106  ,  0.01856888,  0.12811816, -0.19469248,
         0.0769702 ,  0.20283297,  0.29008055,  0.02028482, -0.12926641,
        -0.04846543, -0.05207671, -0.03316664, -0.1765712 ,  0.01017776,
         0.2708853 , -0.08717136,  0.02070806, -0.01205015,  0.14926407,
         0.25498936, -0.03342117, -0.13030672,  0.06632984,  0.3417568 ,
        -0.0449645 , -0.09630641, -0.19584368,  0.06084952,  0.02816491,
        -0.12716284, -0.05554081, -0.13319102,  0.09061765, -0.08026118,
        -0.03490731,  0.00283731,  0.15054679,  0.4269414 ,  0.10274042,
        -0.04721209,  0.1454485 ,  0.24609394, -0.15863296, -0.09804205,
        -0.17660844,  0.1483835 , -0.15724996, -0.01741939,  0.06103354,
        -0.08940341,  0.09271107, -0.08346217,  0.03019234,  0.10981421,
        -0.00283515,  0.03666781, -0.01518192,  0.05991765, -0.01941036,
         0.26531997,  0.07995594,  0.04015322,  0.3167902 , -0.05698166,
         0.2766356 ,  0.14080222,  0.07168636, -0.26628318, -0.13518295,
         0.21819884, -0.00575808, -0.08212868, -0.13839713,  0.08817974,
         0.13988763, -0.10925691, -0.06351396, -0.00679346,  0.01271869,
         0.0483027 , -0.11918885,  0.04418642, -0.09205839,  0.03029672,
         0.04010388, -0.12033968,  0.15821207,  0.10007641, -0.07970749,
        -0.08515656, -0.19164322,  0.006798  ,  0.04576558,  0.16265512,
        -0.19076294,  0.37361056, -0.01433535, -0.09413207, -0.12112152],
       dtype=float32)
```

Transfer Learning and Foundation Models

Definitions

Transfer learning: Taking knowledge gained from one data set and applying it to another.

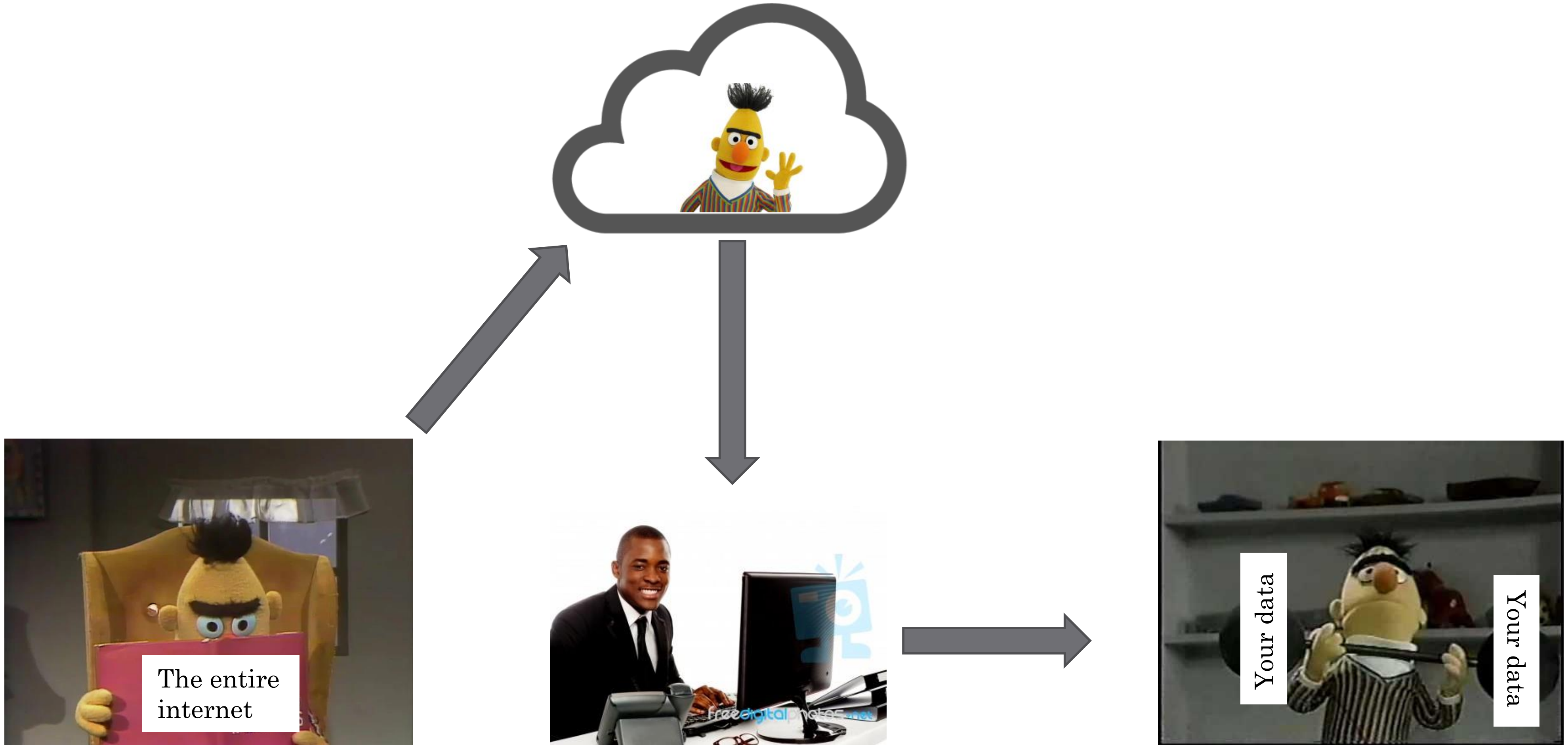
Foundation Models: Large models trained on massive data sets intended to be adapted for downstream tasks.



Bidirectional Encoder Representations from Transformers (BERT)

- Neural network known as a transformer
- Uses contextualized embeddings
- Pre-trained on massive corpora and then adapted to hundreds of thousands of tasks.
- Lots of evolutions:
  - RoBERTa
  - Electra
  - DeBERTa
  - BigBird
  - DistilBERT
  - Etc.

BERT Supervised Learning Pipeline

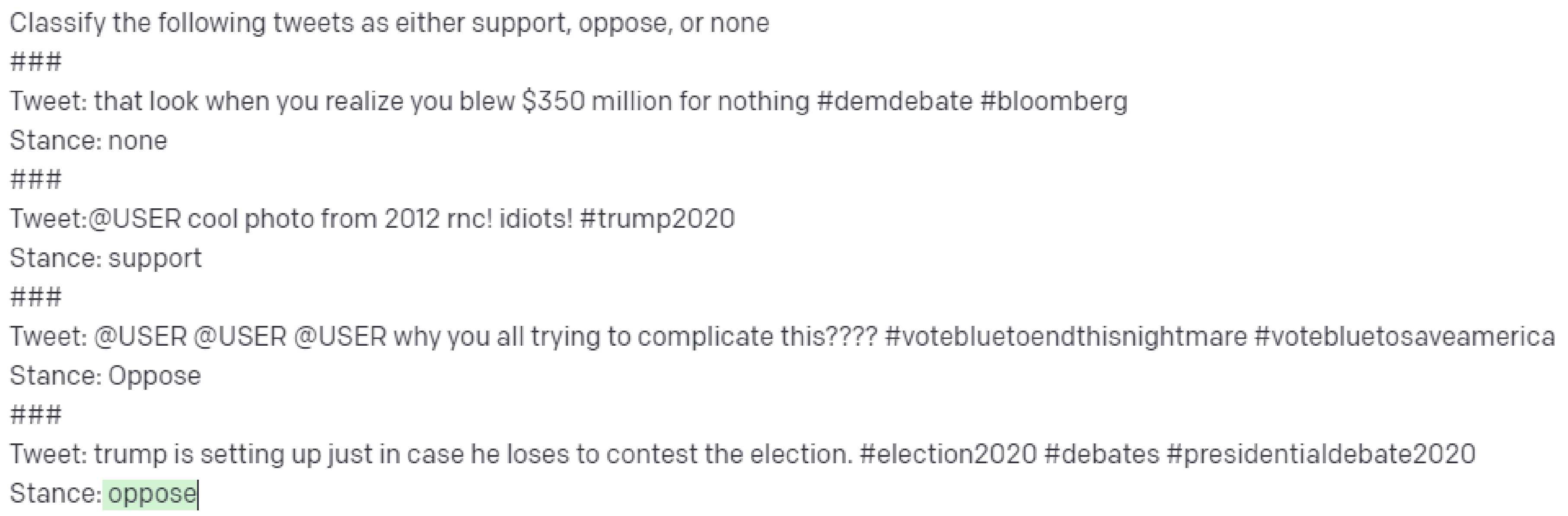


BERT tasks

- Supervised classification
- Zero-shot classification
- Topic models
- Sentiment analysis
- Translation
- Summarization
- Document/sentence similarity
- Question answering

Generative Pre-trained Transformer (GPT)

- Transformer similar to BERT, but much larger and trained differently
- Primary application is text generation tasks.
- Attractive because of their ease of use
- Currently limited application for social science research because:
  - BERT-like models already match human performance on most tasks researchers care about.
  - Only the largest GPT models can compete with BERT-like models. Both expensive and inefficient.



```
Classify the following tweets as either support, oppose, or none
###
Tweet: that look when you realize you blew $350 million for nothing #demdebate #bloomberg
Stance: none
###
Tweet:@USER cool photo from 2012 rnc! idiots! #trump2020
Stance: support
###
Tweet: @USER @USER @USER why you all trying to complicate this???? #votebluetoendthisnightmare #votebluetosaveamerica
Stance: Oppose
###
Tweet: trump is setting up just in case he loses to contest the election. #election2020 #debates #presidentialdebate2020
Stance: oppose
```

Hugging Face

Applications

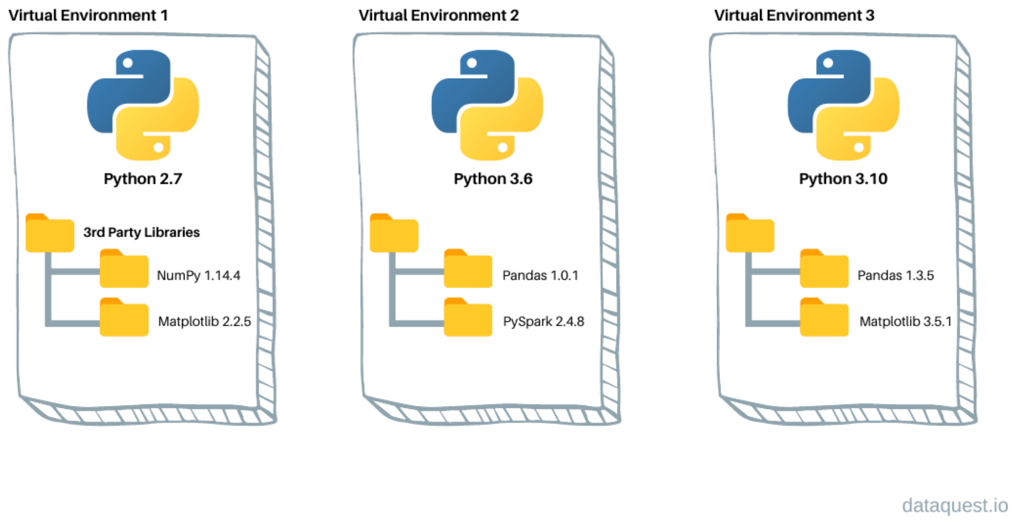
Some notes on tools: Language

ONLY FOR
GAME BOY ADVANCE
POKéMON
Official Nintendo Seal
Nintendo
EVERYONE
E
CONTENT RATED BY ESRB

VS.

ONLY FOR
GAME BOY ADVANCE
POKéMON
Official Nintendo Seal
Nintendo
EVERYONE
E
CONTENT RATED BY ESRB

Some notes on tools: Environments



CONDA

Some notes on tools: Packages

PyTorch vs. TensorFlow

Weights & Biases

Some notes on tools: Hardware

NVIDIA® CUDA®

Or

Colaboratory

Summary:

- Use python if possible
- Always set up a virtual environment. I recommend anaconda.
- Pytorch is generally more user friendly
- Local hardware requires a Nvidia GPU. You can access free/cheap hardware via Google colab

Coding Demonstration: Classification