Natural Language Processing Workshop

Amy Hemmeter, MSA Class of ‘18
Manager/Data Scientist/Machine Learning Engineer

“We are seeing a surge in demand for NLP applications across sectors and use cases. Our clients are implementing NLP solutions for deeper customer insights, risk mitigation, and operational efficiency.”

- Nathan Peifer, formerly of EY, now at Elicit, personal communication, 2020

“We’re looking for better skills for dealing with unstructured data, NLP is becoming increasingly important as much of the data available is text.”

- Garima Sharma, Manager, Enterprise Data Intelligence at Domino’s (Women in Data Science Conference 2020, Detroit)

“NLP is full of fraudsters”

- Amy

Natural Language Processing

- NLP is a subfield of artificial intelligence that deals with understanding and in some cases producing, human (“natural”) language
- We’re going to cover the following NLP topics:
    - Word vectors
    - Deep learning solutions in NLP
    - Transformers
        - BERT
        - GPT
    - High-level descriptions of NLP tasks

Rule-Based NLP

- A series of rules in code that very explicitly spelled out how a computer is supposed to understand human language
- Often based on keywords and collocations (words that occur together)
- Regular expressions play a big role
- Very time-consuming and expensive for the company
- Fairly old-fashioned, no one’s first choice and doesn’t show up on job postings that request NLP skills very often
- However, many systems are still partially rule-based -- including those of many MSA employers (from my experience of talking to them in 2018)

Expert AI Revealed To Be 1,000,000 If-Else Statements Stacked in a Trenchcoat

“I thought the movie was terribly well done, bravo to all the actors.”

“Garbage”.

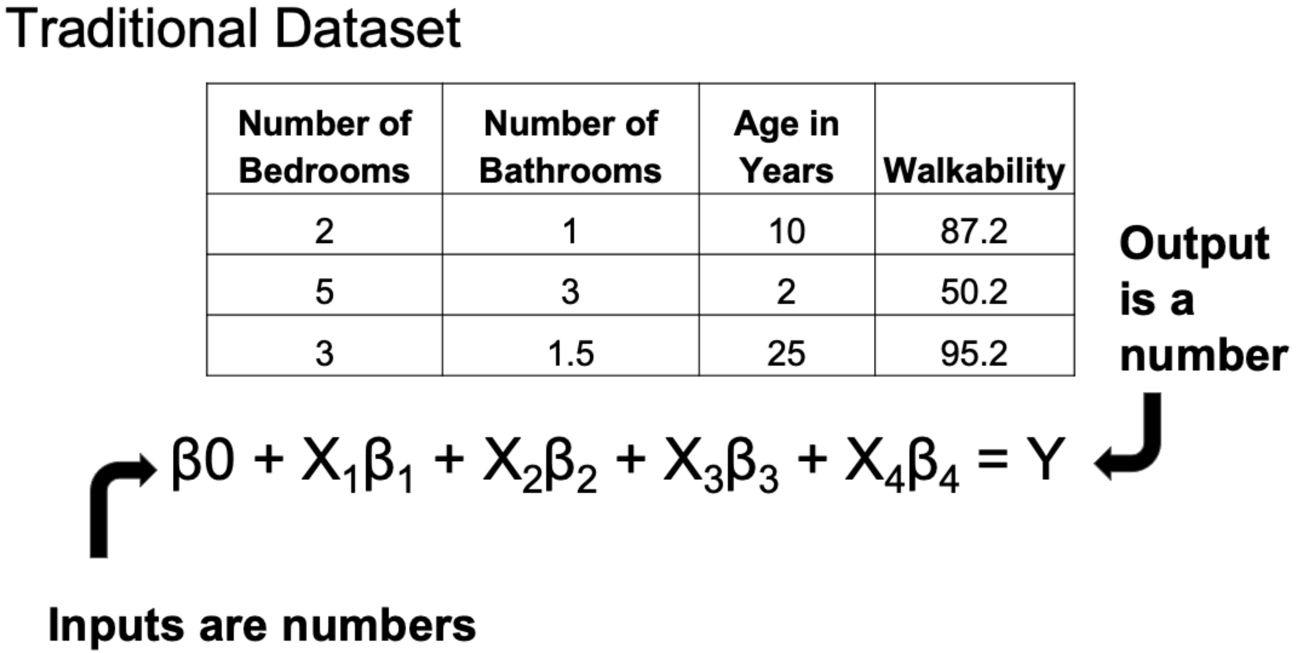
1. Words are not numbers

1. Words are not numbers
2. Input can be different lengths

1. Words are not numbers
2. Input can be different lengths
3. Words can mean different things in different contexts

1. **Words are not numbers**
2. Input can be different lengths
3. Words can mean different things in different contexts

Word Embeddings



Traditional Dataset

Number of Bedrooms	Number of Bathrooms	Age in Years	Walkability
2	1	10	87.2
5	3	2	50.2
3	1.5	25	95.2

Output is a number

$$\beta0 + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 = Y$$

Inputs are numbers

Word Embeddings

Traditional Dataset

Number of Bedrooms	Number of Bathrooms	Age in Years	Walkability
2	1	10	87.2
5	3	2	50.2
3	1.5	25	95.2

Output is a number

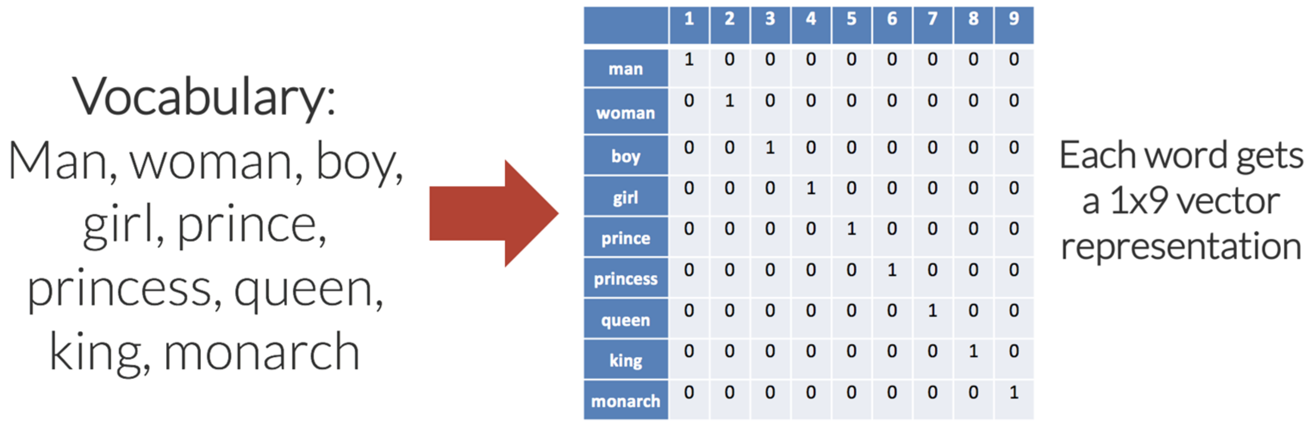
$$\beta 0 + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 = Y$$

Inputs are numbers

Input to ML model is a vector: [2 1 10 87.2]

Word Vectors

- How do we turn a word into a number?
- We could use “one-hot” vectors - each vector is the length of your vocabulary, and a 1 denotes your word in that vocabulary



Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

Source

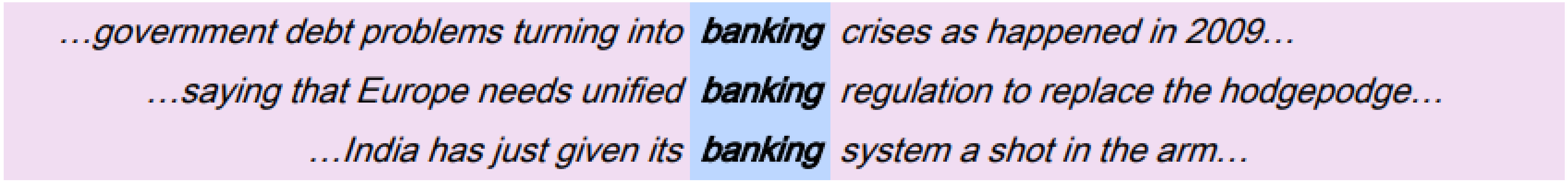
Problems with One-Hot

- Real vocabularies can be millions of words, meaning your vectors could be incredibly high-dimensional
- Very clunky to work with
- No information encoded in the word vector
- All vectors are orthogonal, no one is more similar to the other than any of the others
- What happens if a word occurs that wasn’t in your training set?

Distributional Semantics

"You shall know a word by the company it keeps" - J.R. Firth

- We draw on a linguistic notion called "distributional semantics"
- We can get a large number of contexts for a word and use it to build a vector representation of that word



...government debt problems turning into ***banking*** *crises as happened in 2009...*

...saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge...*

...India has just given its ***banking*** *system a shot in the arm...*

Source: CS224n lecture notes at Stanford

Quick quiz:

Does this sound like anything you’ve learned about in your text analytics class?

Defining word vectors by their context?

Word Embeddings

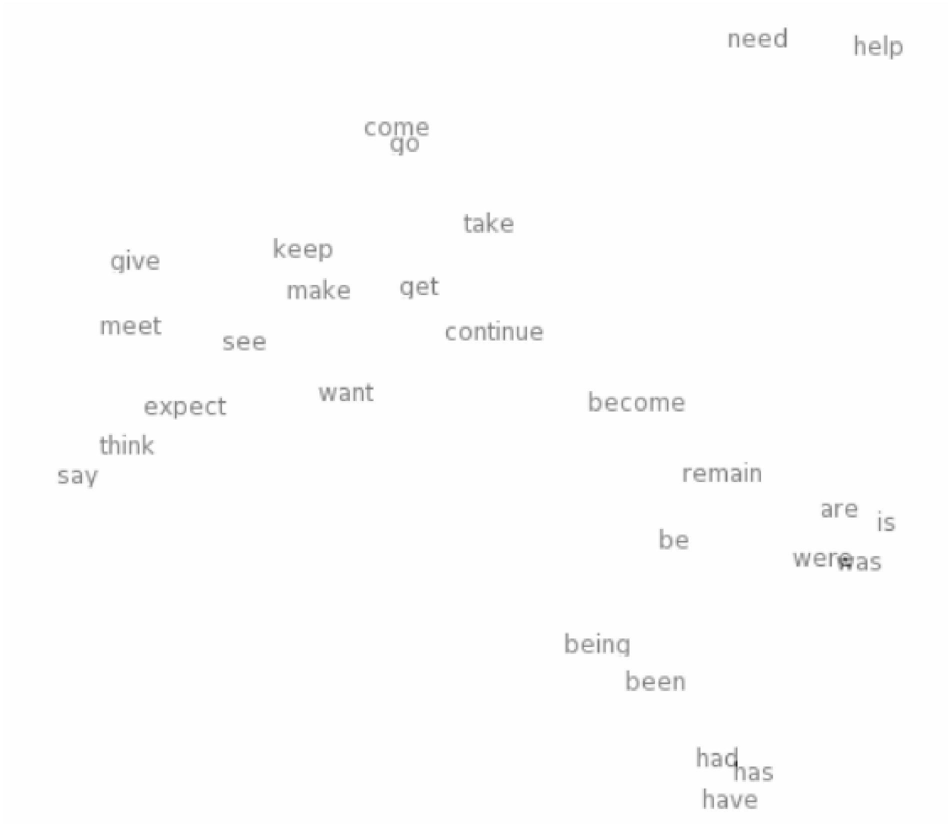
	X_1	X_2	X_3	X_4	X_5
pasta	0.02182935	0.72359903	0.35498939	0.79730453	0.76512217
hurtled	0.53065205	0.27409522	0.95427668	0.75399903	0.4420345
the	0.47452104	0.18728204	0.26258075	0.90970631	0.62042977
said	0.559265	0.39511139	0.02820151	0.03384123	0.30565618
with	0.48774497	0.63185696	0.34809497	0.01427407	0.47724118

Word Embeddings

	X_1	X_2	X_3	X_4	X_5
pasta	0.02182935	0.72359903	0.35498939	0.79730453	0.76512217
hurtled	0.53065205	0.27409522	0.95427668	0.75399903	0.4420345
the	0.47452104	0.18728204	0.26258075	0.90970631	0.62042977
said	0.559265	0.39511139	0.02820151	0.03384123	0.30565618
with	0.48774497	0.63185696	0.34809497	0.01427407	0.47724118

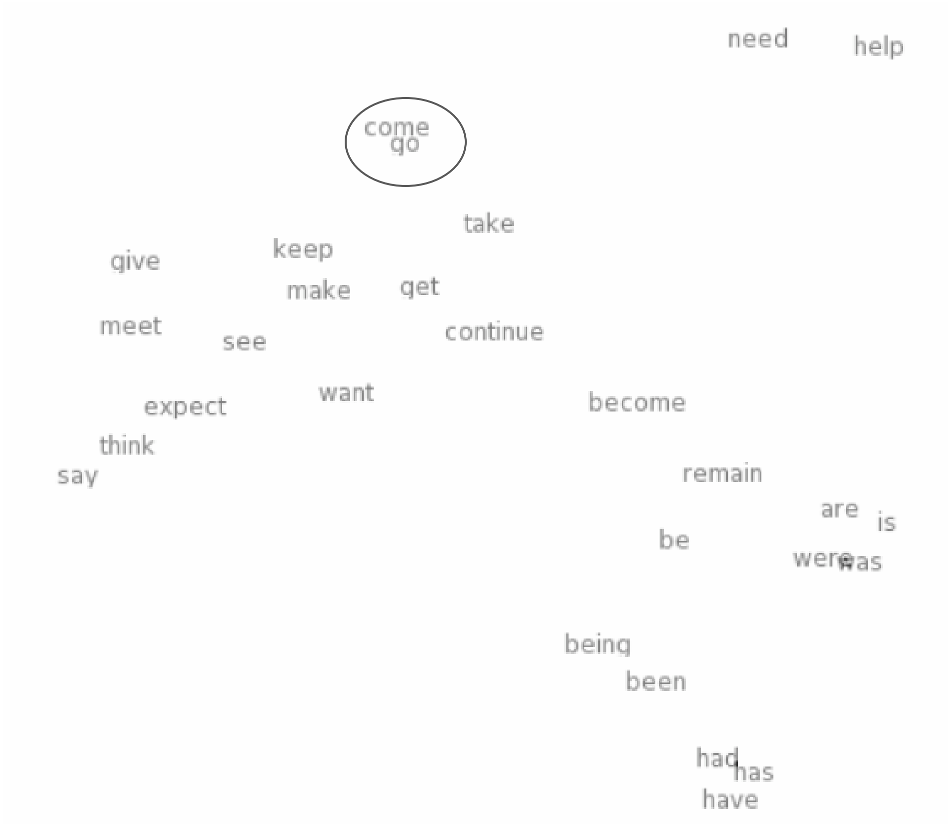
Input to ML model is a vector: [0.02182935 0.72359903 0.35498939 0.79730453 0.76512217]

Word meaning visualized



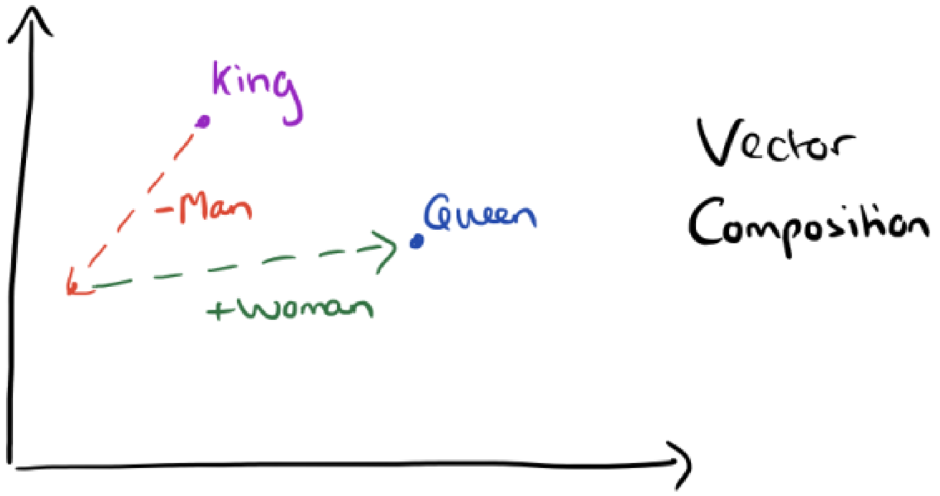
Source: Stanford cs224 notes

Word meaning visualized



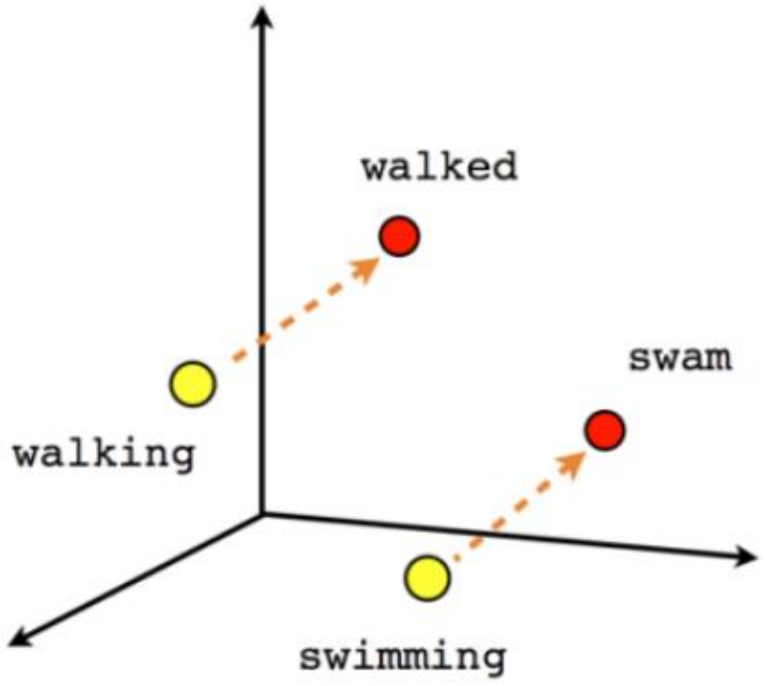
Source: Stanford cs224 notes

Word Vector Arithmetic

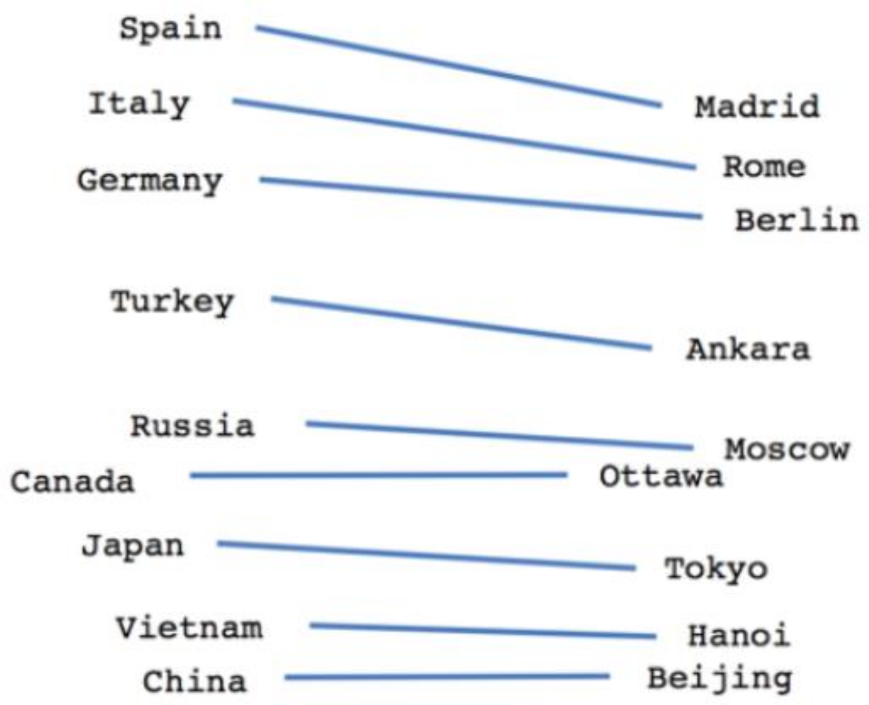


Source: [the morning paper]

Word Vector Arithmetic



Word Vector Arithmetic



How do we get these vectors?

- Training a neural network with one-hot vector encodings as inputs to do a “fake” Language Modeling task
- We then pop out the hidden layer of that neural network and use that as our embedding

What is language modeling?

- The task of predicting the probability of a sentence

“I’ll text you when I get ______”

Other Uses for Language Modeling

- Speech recognition
  - We want to know that the sounds that are very similar in “wreck a nice beach” are more likely “recognize speech” -- we want to get a probability of the sentence
- Machine Translation
  - Change idiomatic “large winds tonight” from another language to more naturally English “High winds tonight”
- Spelling correction
  - “The office is about 15 minuets from my house” → “the office is about 15 minutes from my house”
- Anytime you need to know the probability of a sentence!

Quick quiz:

In some forms of text analytics, you remove so-called “stop words” - function words like “a”, “the”, “of”, “in”, “and”, etc.

Do you think you should remove stop words in a language modeling task? Why or why not?

Answer: NO.

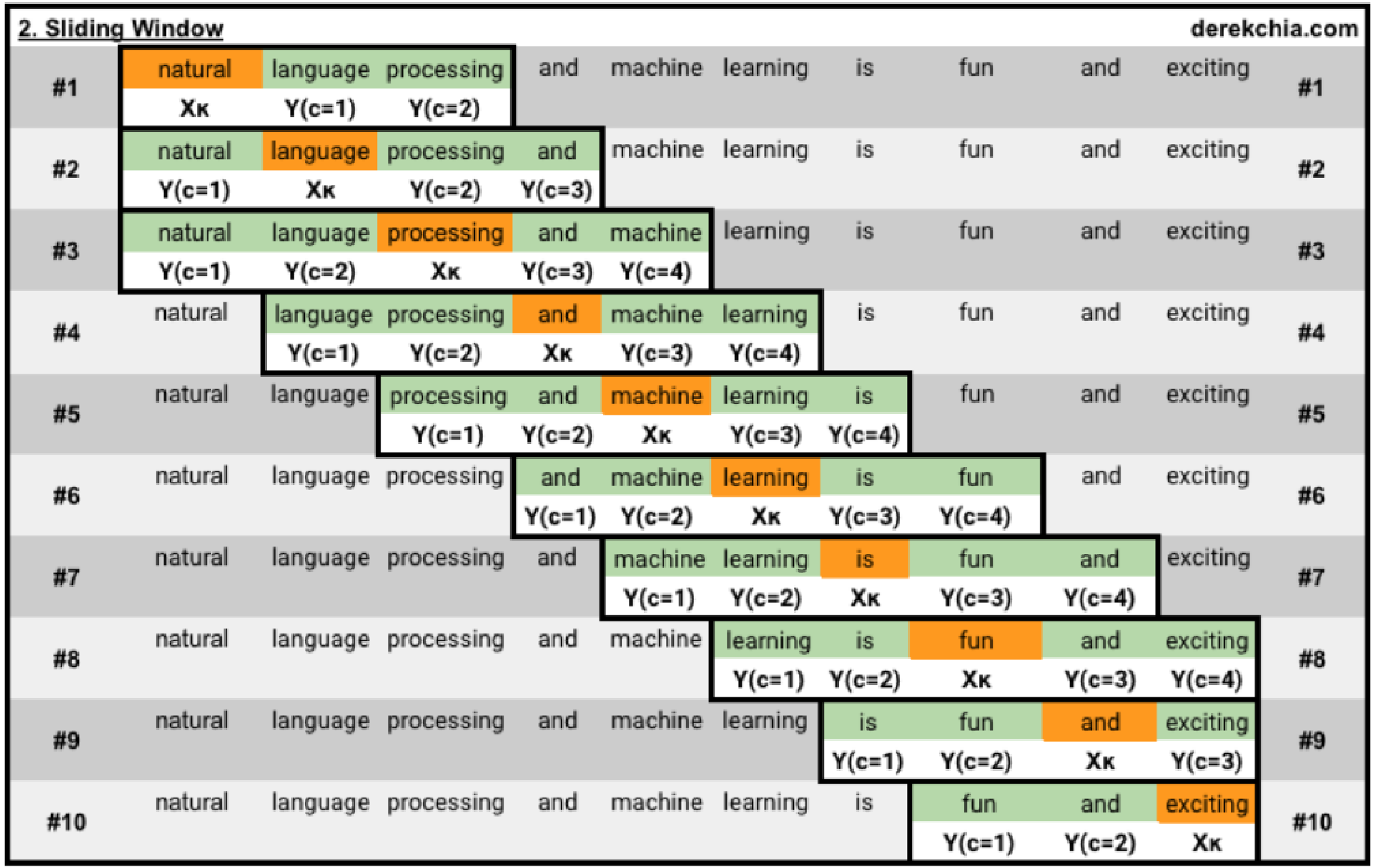
“I want go ______”

“I want to go to the _____”

“I want to go _____”

word2vec

- Word2vec (Mikolov et al. 2013) is a framework for learning word vectors
- Inputs are vectors the same length as your vocabulary - so if you have 10,000 words you have a vector of dimension 10,000
- There are two versions of word2vec
    - SkipGram - predict context words from center word
    - Continuous Bag of Words - predict center word from (bag of) context words



2. Sliding Window derekchia.com

#1	natural	language	processing	and	machine	learning	is	fun	and	exciting	#1
	Xκ	Y(c=1)	Y(c=2)								
#2	natural	language	processing	and	machine	learning	is	fun	and	exciting	#2
	Y(c=1)	Xκ	Y(c=2)	Y(c=3)							
#3	natural	language	processing	and	machine	learning	is	fun	and	exciting	#3
	Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)						
#4	natural	language	processing	and	machine	learning	is	fun	and	exciting	#4
		Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)					
#5	natural	language	processing	and	machine	learning	is	fun	and	exciting	#5
			Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)				
#6	natural	language	processing	and	machine	learning	is	fun	and	exciting	#6
				Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)			
#7	natural	language	processing	and	machine	learning	is	fun	and	exciting	#7
					Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)		
#8	natural	language	processing	and	machine	learning	is	fun	and	exciting	#8
						Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)	
#9	natural	language	processing	and	machine	learning	is	fun	and	exciting	#9
							Y(c=1)	Y(c=2)	Xκ	Y(c=3)	
#10	natural	language	processing	and	machine	learning	is	fun	and	exciting	#10
								Y(c=1)	Y(c=2)	Xκ	

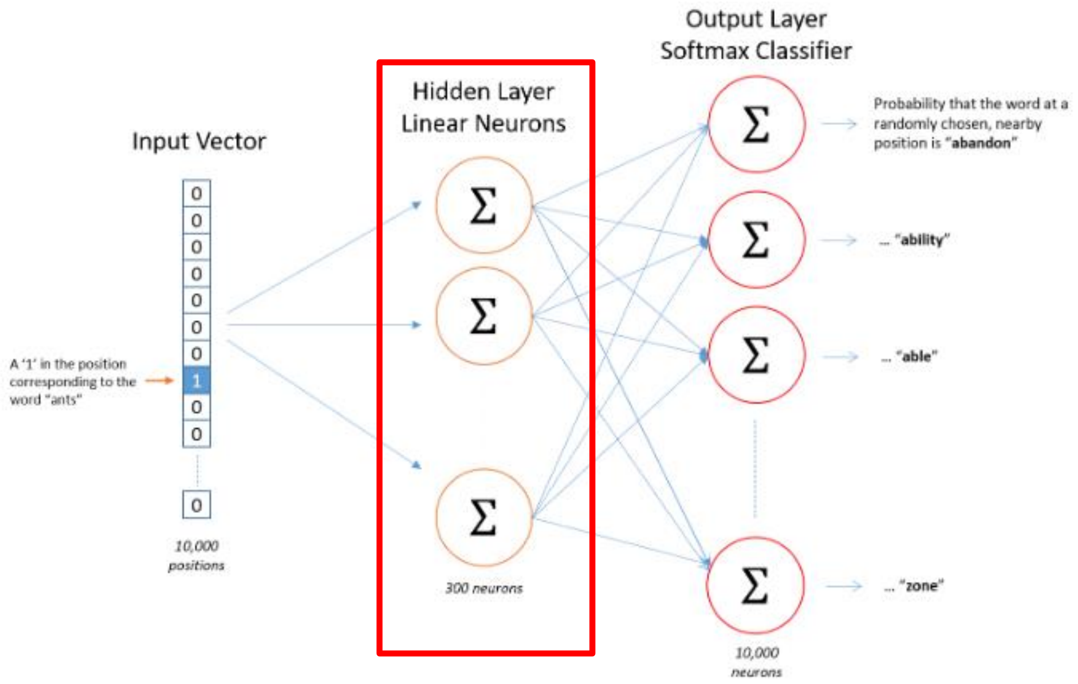
Quick quiz: are language models (large or otherwise), supervised models?

Quick quiz: are language models (large or otherwise), supervised models?

UNSUPERVISED!

This allows for big AI companies with lots of compute resources to scale almost indefinitely, because no human labor (with its time, cost, and mistakes) is required!

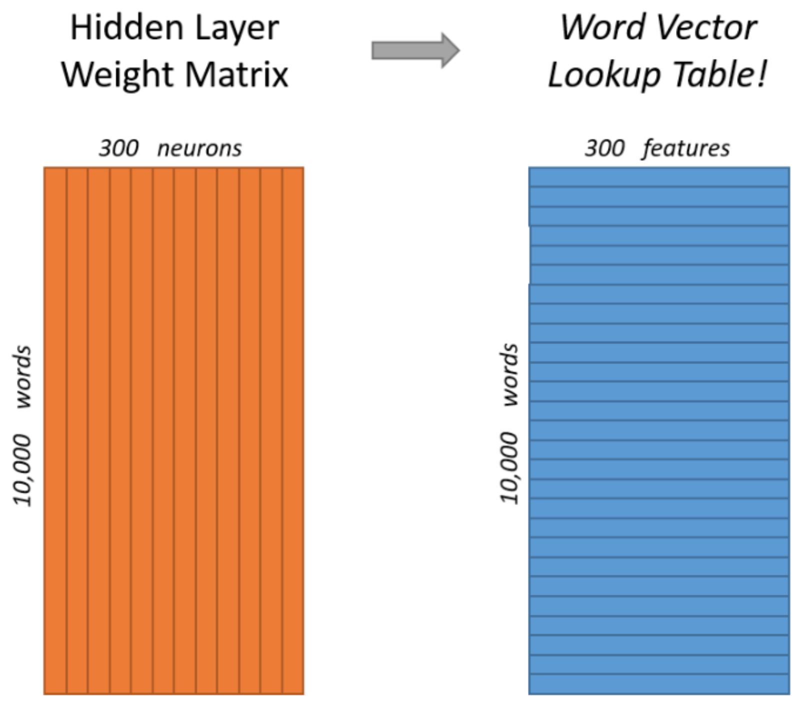
How do we get these vectors?



Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
Hidden Layer Linear Neurons
Σ
300 neurons
Output Layer Softmax Classifier
Σ
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
10,000 neurons

Source: McCormick ML

What do we do when we pop out the layer?



Let’s play with word embeddings!

Deep Learning Solutions for NLP

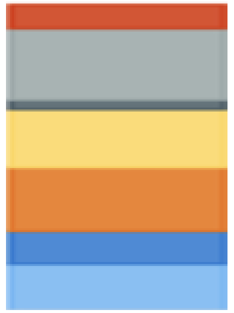
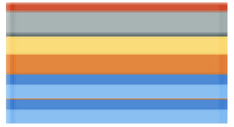
Classification Review

- You have a dataset of samples consisting of:
  - x_i inputs, e.g. words (indices to an embedding lookup table or vectors), sentences, documents, etc.
  - y_i outputs which are *labels* of a certain number of classes (binary or multi-class)
- Example:
  - Sentiment analysis
  - Intents in conversational AI
- Architecture
  - Various flavors of deep learning (we'll go over several in this class) topped off by a softmax layer
  - Generally we use cross-entropy loss as our objective function

Classification Review

Datasets

- We split our models into, minimally, training and test sets
- You train your model on the training set (normally 70%ish of your data) and then test that same model on your test set
- Preferably train, dev, and test sets -- you train on the training set, then test on dev set iteratively until either you reach your number of epochs or performance ceases to improve, then test is set aside for the "final" numbers on your model
- Ideal situation is thousands of examples with a good amount of variance
- Preferably we want a huge training set and large test set that are "balanced" – that is, we have the same proportion of classes in both the training and the dev and test set

Softmax

- You can think of it like an activation function (a la the sigmoid or the ReLU) for the final (output) layer of your network
- Normalizes the output of classifier to be some probability for each class, where they all add up to 1:

$$\boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}}$$

Sample output: $\begin{bmatrix} 0.25 \\ 0.21 \\ 0.9 \\ 0.45 \end{bmatrix}$

- This makes it easier to choose the y with the maximum probability

Softmax Example

Original output:

$$\sigma(x_j) = \frac{e^{x_j}}{\boxed{\sum_i e^{x_i}}}$$

$$\begin{bmatrix} 3.0 \\ 5.0 \\ 2.0 \end{bmatrix}$$

$$\begin{aligned} & e^3 + e^5 + e^2 \\ &= 20.0855 + 148.4132 + 7.3891 \\ &= \boxed{175.8878} \end{aligned}$$

Softmax Example

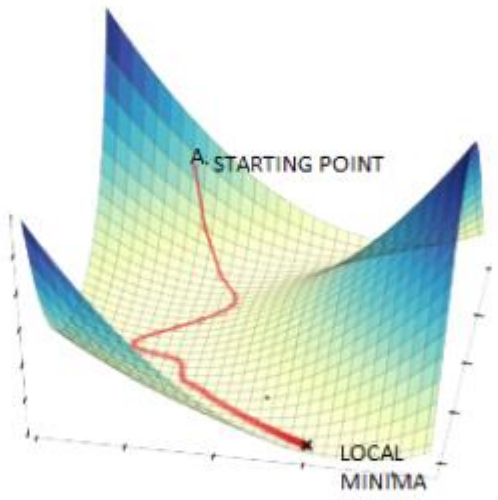
Original output:

$$\sigma(x_j) = \boxed{\frac{e^{x_j}}{\sum_i e^{x_i}}}$$

$$\begin{bmatrix} 3.0 \\ 5.0 \\ 2.0 \end{bmatrix} \qquad \begin{matrix} 20.0855/175.8878 = \\ 148.4132/175.8878 = \\ 7.3891/175.8878 = \end{matrix} \qquad \begin{bmatrix} 0.12 \\ 0.84 \\ 0.04 \end{bmatrix}$$

Gradient Descent Review (Hopefully)

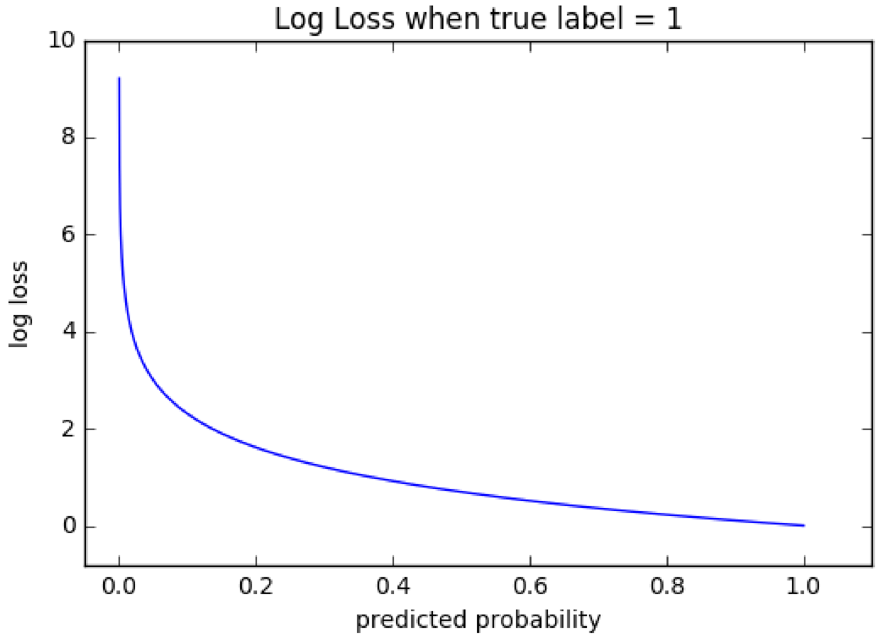
- How to find the minimum of your loss function
- Uses the derivative of the loss function with respect to your parameters to take a step in a direction towards the minimum
- How most machine learning algorithms work
- The rate at which you move down this slope is called the learning rate



Picture from Data Science Central, https://www.datasciencecentral.com/profiles/blogs/alternatives-to-the-gradient-descent-algorithm

Cross Entropy Loss

- Cross-Entropy Loss is the loss function we're trying to minimize in classification
- We're attempting to maximize the probability of the correct class y
- Cross entropy loss measures the deviation of the predicted probability from the actual probability



Source: ML cheatsheet on readthedocs.io

Cross-Entropy Loss

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Cross-Entropy Loss

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Output of softmax:

$$\begin{bmatrix} 0.12 \\ 0.84 \\ 0.04 \end{bmatrix}$$

Correct label:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$-(1 \times log(0.84))$$

$$= 0.174353387$$

Cross-Entropy Loss

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Output of softmax:

$$\begin{bmatrix} 0.84 \\ 0.12 \\ 0.04 \end{bmatrix}$$

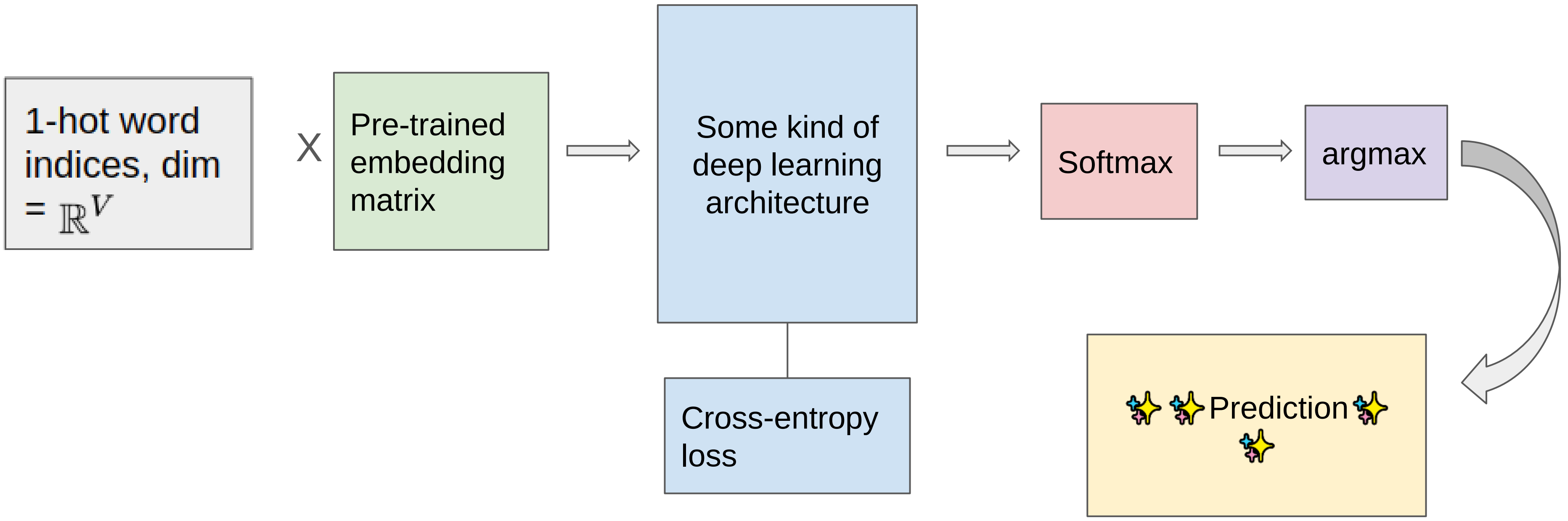
Correct label:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

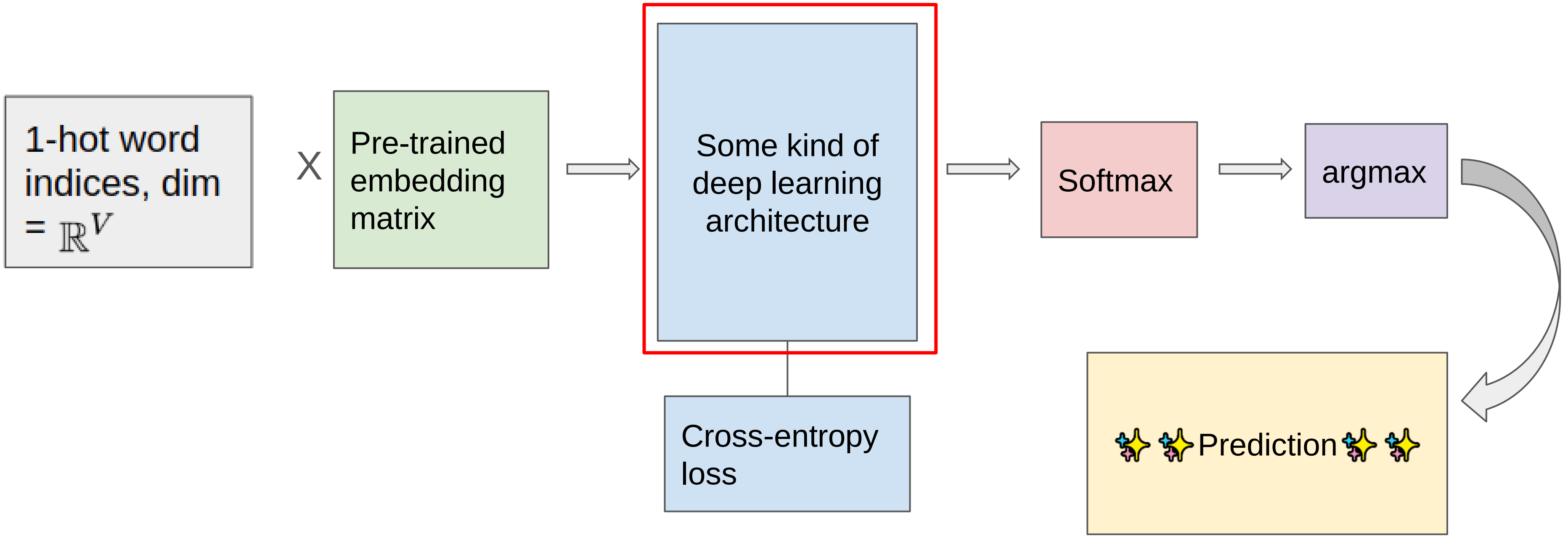
$$-(1 \times log(0.12))$$

$$= \mathbf{2.12026353}$$

Putting it together


1-hot word indices, dim = $\mathbb{R}^V$
X
Pre-trained embedding matrix
Some kind of deep learning architecture
Softmax
argmax
Cross-entropy loss
✨✨Prediction✨✨

Putting it together


1-hot word indices, dim = $\mathbb{R}^V$
X
Pre-trained embedding matrix
Some kind of deep learning architecture
Cross-entropy loss
Softmax
argmax
✨✨Prediction✨✨

Practical Tips for Classification in the Workplace

- Labelers for supervised tasks are an incredibly precious resource
- Many companies outsource to countries like the Phillippines to make the labeling less expensive
- If you have the good fortune of working with internal labelers, domain knowledge is more important than labeling experience (in my experience)
- **Make their lives easier whenever you can with unsupervised methods**

How to Make Your Labelers’ Lives Easier (Yours Too)

- Cluster the data to help them make sense of it and break it into chunks (very helpful for higher-dimensional classifiers like intent classifiers)
- Determine what else you can use to filter - e.g. sentiment for something like hate speech or bias detection
- Use simple anomaly detection methods to find likely errors to send back to them
- Use even simpler similarity metrics to find less-frequent examples in a huge haystack
- Be creative! Keep the data representative and pay attention to lower-resource samples

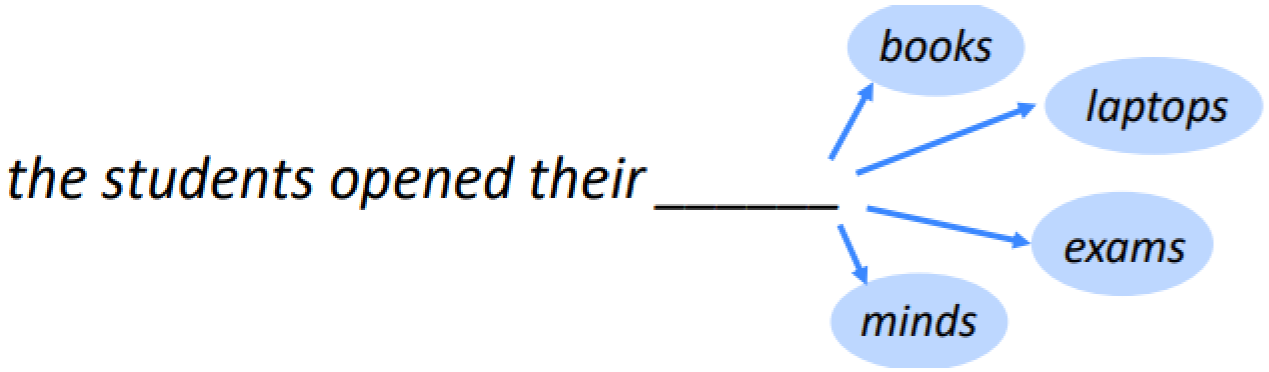
Let’s do some clustering!

Deep Learning Architectures for NLP

- (Vanilla) Recurrent Neural Networks (RNNs)
- Long Short-Term Memory (LSTMs)
- CNNs - they’re not just for image processing anymore!
- Transformers
    - BERT, GPT

Language Modeling with Recurrent Neural Networks (RNNs)

- Words and characters are *not* independent in language, therefore they should not be treated as independent in your neural net
- We'll be revisiting language modeling



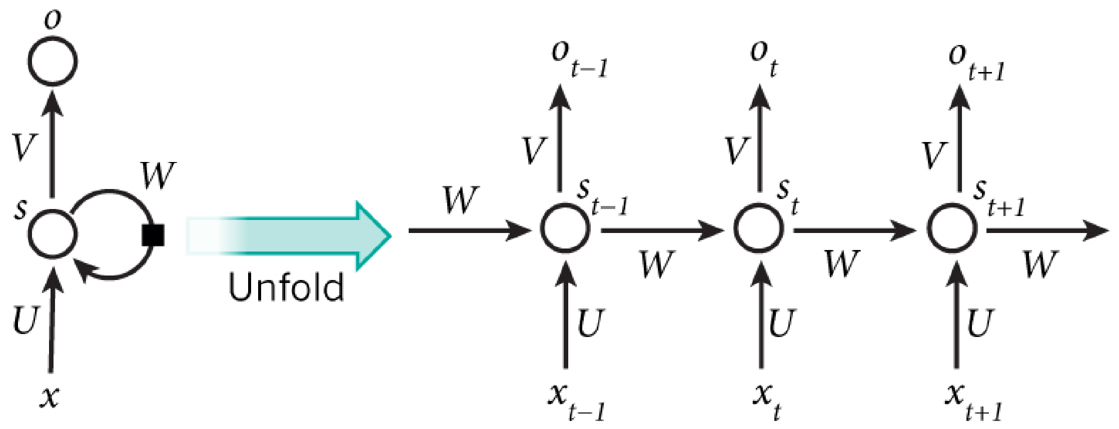
Language Modeling with Recurrent Neural Networks (RNNs)

- Previous methods (n-gram language modeling, see further reading) created incoherent sentences
- Consider the following sentence:

“today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .”

Language Modeling with Recurrent Neural Networks (RNNs)

- RNNs allow us to use a “memory” for the information that came before



$$s_t = f(Ux_t + Ws_{t-1})$$

Language Modeling with Recurrent Neural Networks (RNNs)

- To train
  - Get large corpus of text data with a sequence of words x1, …. xT
  - Predict probability of distribution for each word, given words so far (classification with large number of classes)
  - Loss function is the cross entropy between the predicted next word and the true next word (one-hot vector of class index)
  - Average all of these values to get the overall loss for the training set

Language Modeling with Recurrent Neural Networks (RNNs)

- RNNs can process any length of input, the model size doesn’t increase for longer input
- However, RNNs are slow to compute -- they are difficult to parallelize because the inputs are dependent on one another
- Although they hold some memory, in practice it is difficult to access information from many steps back (5-10 steps)
- Vanishing gradient problem (more on this in a second)

Language Modeling with Recurrent Neural Networks (RNNs)

PANDARUS:

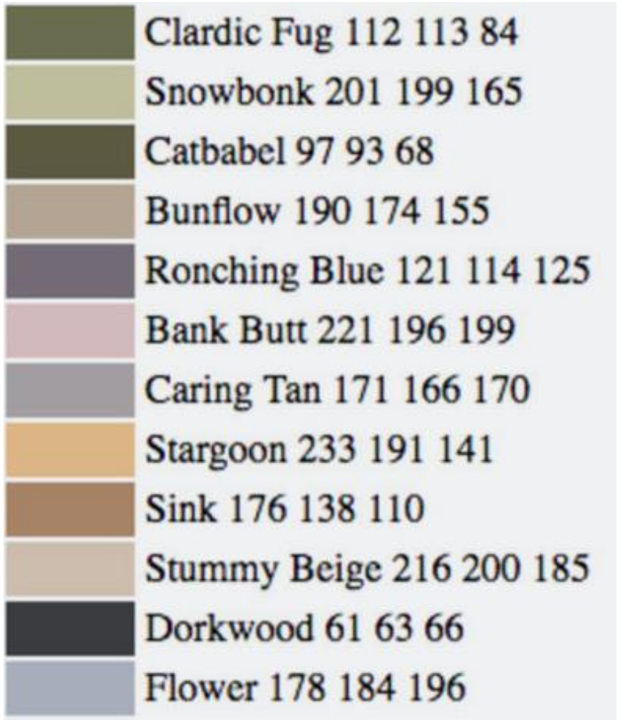
Alas, I think he shall be come approached and the day

When little srain would be attain’d into being never fed,

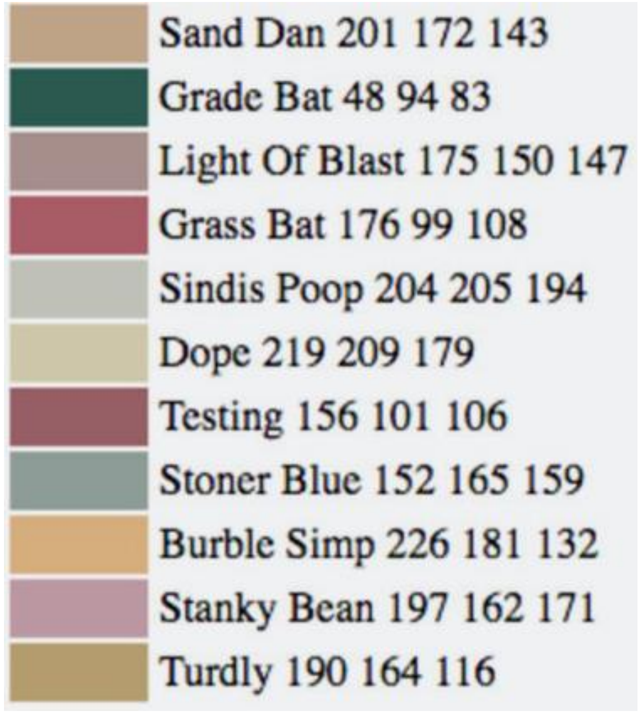
And who is but a chain and subjects of his death,

I should not sleep

What are RNNs capable of?



■	Clardic Fug 112 113 84
■	Snowbonk 201 199 165
■	Catbabel 97 93 68
■	Bunflow 190 174 155
■	Ronching Blue 121 114 125
■	Bank Butt 221 196 199
■	Caring Tan 171 166 170
■	Stargoon 233 191 141
■	Sink 176 138 110
■	Stummy Beige 216 200 185
■	Dorkwood 61 63 66
■	Flower 178 184 196



■	Sand Dan 201 172 143
■	Grade Bat 48 94 83
■	Light Of Blast 175 150 147
■	Grass Bat 176 99 108
■	Sindis Poop 204 205 194
■	Dope 219 209 179
■	Testing 156 101 106
■	Stoner Blue 152 165 159
■	Burble Simp 226 181 132
■	Stanky Bean 197 162 171
■	Turdly 190 164 116

Source: AI Weirdness

Drawbacks of RNNs

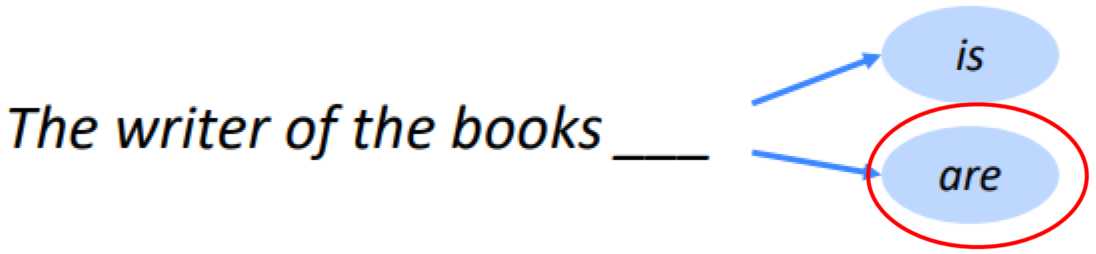
Consider the following word prediction task:

“When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her ________”

Drawbacks of RNNs

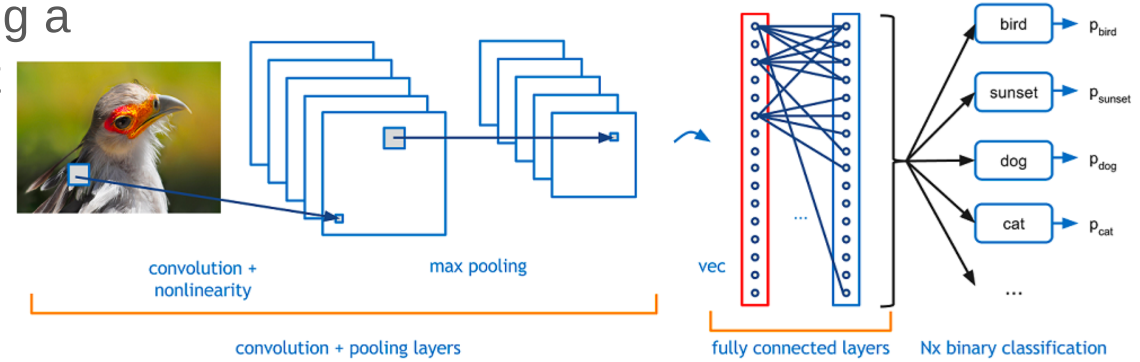
- We get something called the Vanishing Gradient Problem
- The gradient can be viewed as a measure of the effect of the past on the future
- If the gradient becomes too small, it’s impossible to tell whether there’s no dependency between two distant time steps or we have the wrong parameters to capture the dependency between two distant time steps
- This is an artifact of using the chain rule for backpropagation

Drawbacks of RNNs

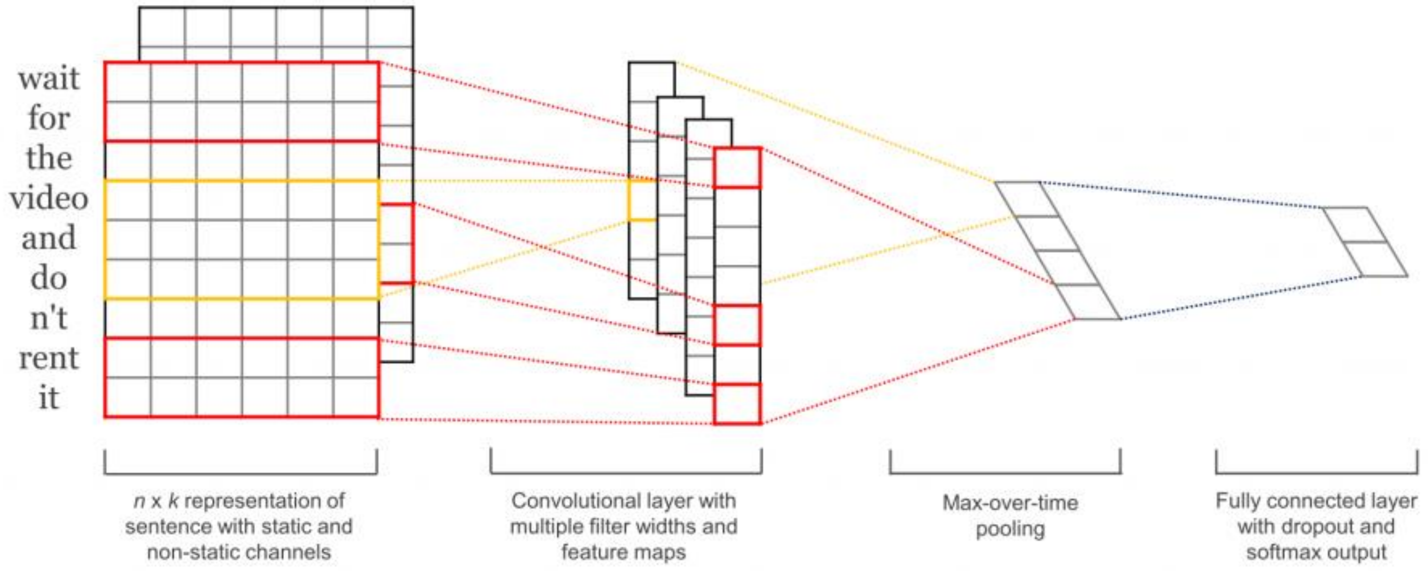


CNN Review

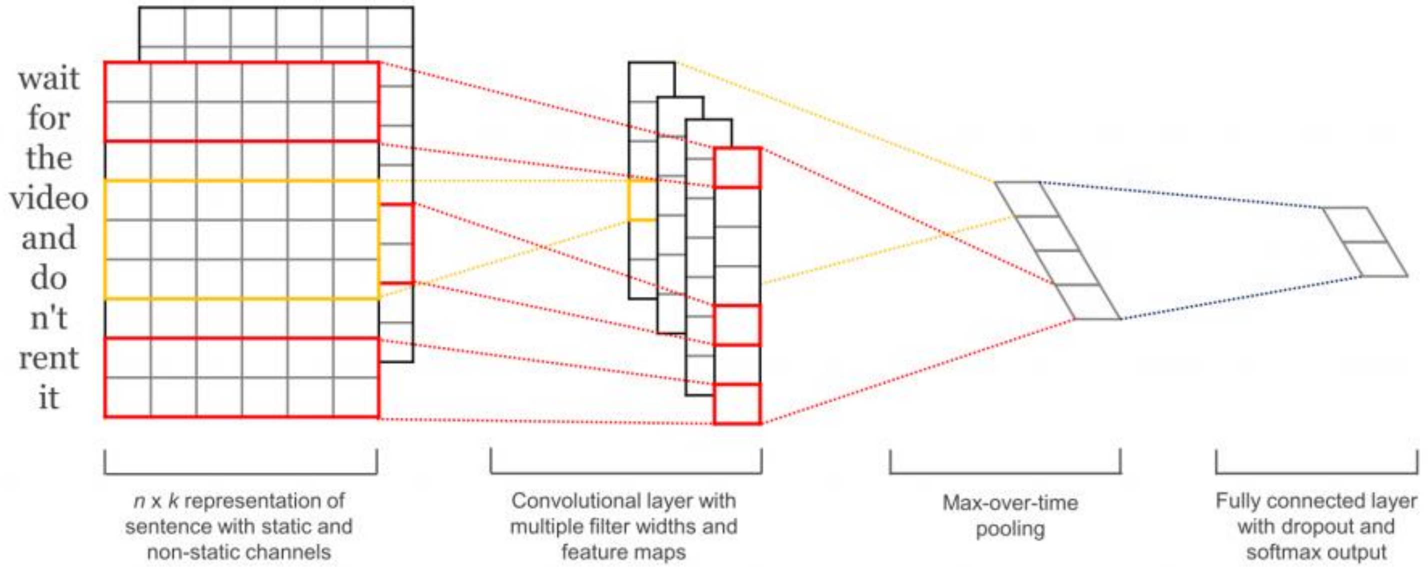
- 2-dim image with sliding window
- At each point you create a convolved value by applying a kernel or filter to the output
- You apply some kind of pooling to the outputs to squash the dimensions
- You then use the smaller-dimensional layers to feed as input to your fully-connected layers



CNNs for NLP


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

CNNs for NLP


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Now for one more thing…

What things did you pay attention to when you were trying to be a language model? What was intuitive? What was challenging?