

# Natural Language Processing Workshop

**Amy Hemmeter, MSA Class of '18**  
**Artificial Intelligence Engineer, Interactions Digital Roots**

“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

“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, Detroit)

# 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:
  - Summarization
  - Language Modeling
  - Classification
  - Practical tips for text data exploration

# **NLP in the pre-Machine Learning Era**

# 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 resumes 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



What's missing from this picture?



What's missing from this picture?

Math!!

“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

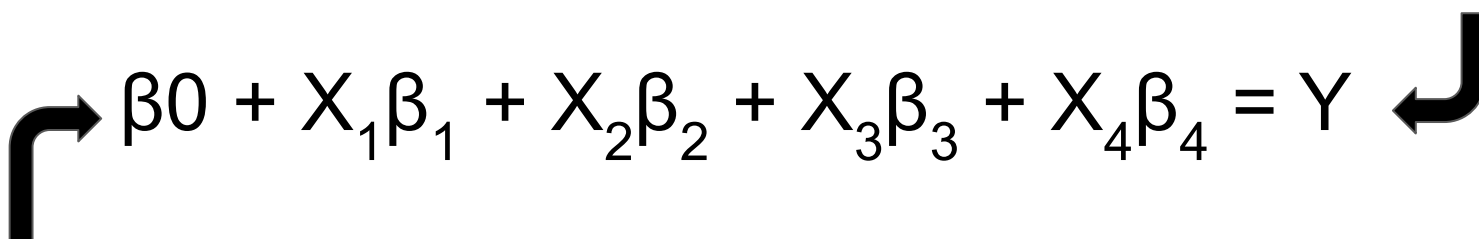
**1. Words are not numbers**

2. Input can be different lengths

# 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**

# NLP Example - Sentiment Analysis

Output - Valence

# NLP Example - Sentiment Analysis

Output - Valence

...



# NLP Example - Sentiment Analysis

Output - Valence

...

A NUMBER!

# NLP Example - Sentiment Analysis

Output - Valence

...

A NUMBER!

What's our input?

# Early ML Solutions to NLP

# Early ML Solutions for NLP

- TextRank for summarization
- Simple n-gram models for language modeling
- Naive Bayes, Logistic Regression for classification
- Various flavors of Markov Models for tagging
- Now, however, almost all solutions are based on deep learning

# Early ML Solutions for NLP

- **TextRank for summarization**
- Simple n-gram models for language modeling
- Naive Bayes, Logistic Regression for classification
- Various flavors of Markov Models for tagging
- Now, however, almost all solutions are based on deep learning

# Problem:

- You want to follow a keyword and get news updates about that keyword
- You don't want to read the 20 articles that pop up, but you want to get a sense for what's happening in the articles pertaining to your keyword

# Summarization

- The solution is summarization!

# Summarization

- The solution is summarization!
- The task: to get 2-3 sentences from each article that sum up the content of the article



# Summarization

- The solution is summarization!
- The task: to get 2-3 sentences from each article that sum up the content of the article
- How do we determine which of the sentences are “important”?

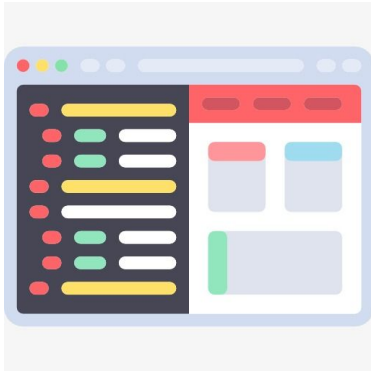
# Summarization

- The solution is summarization!
- The task: to get 2-3 sentences from each article that sum up the content of the article
- How do we determine which of the sentences are “important”?
- We use a tool called [TextRank](#) (Mihalcea and Tarau 2004)

# What is TextRank?

- Determines the top n most relevant sentences via a ranking system
- Draws its inspiration from Google's PageRank Algorithm

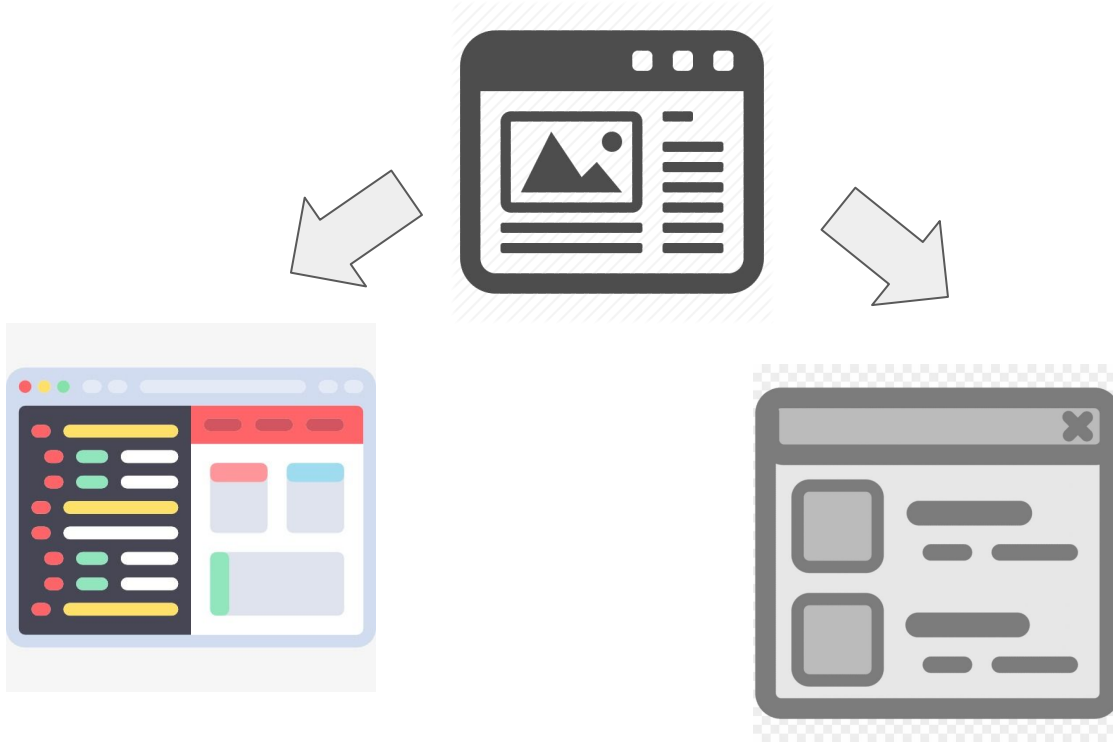
# What is PageRank?



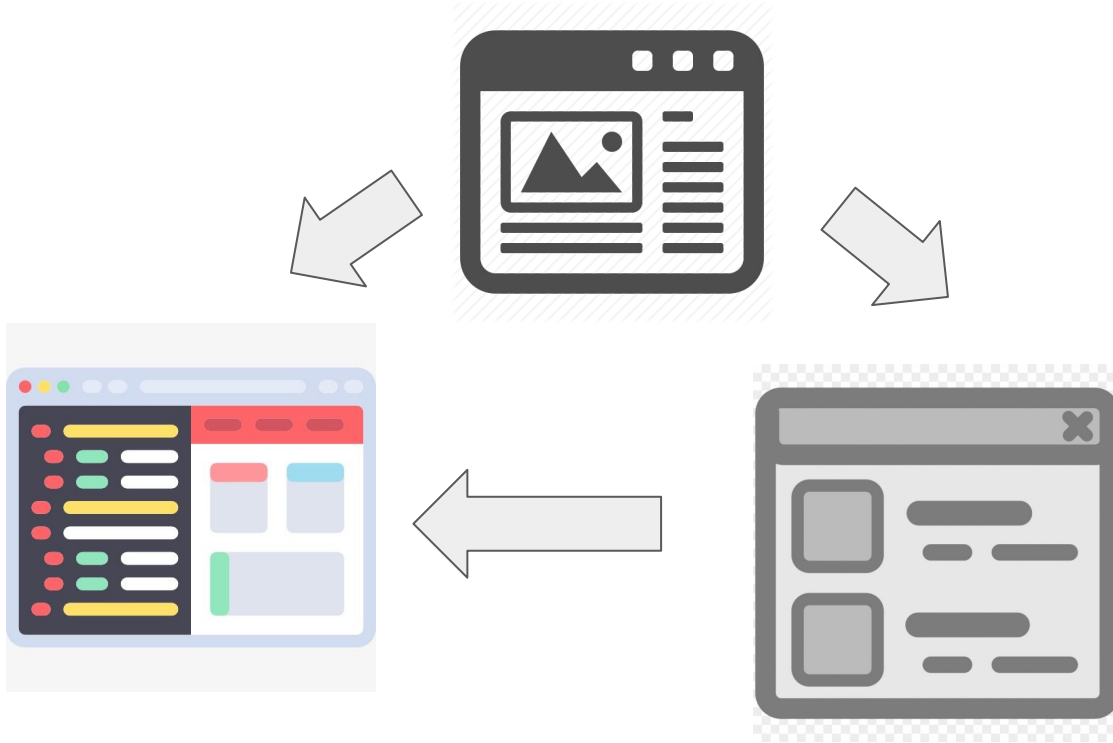
# What is PageRank?



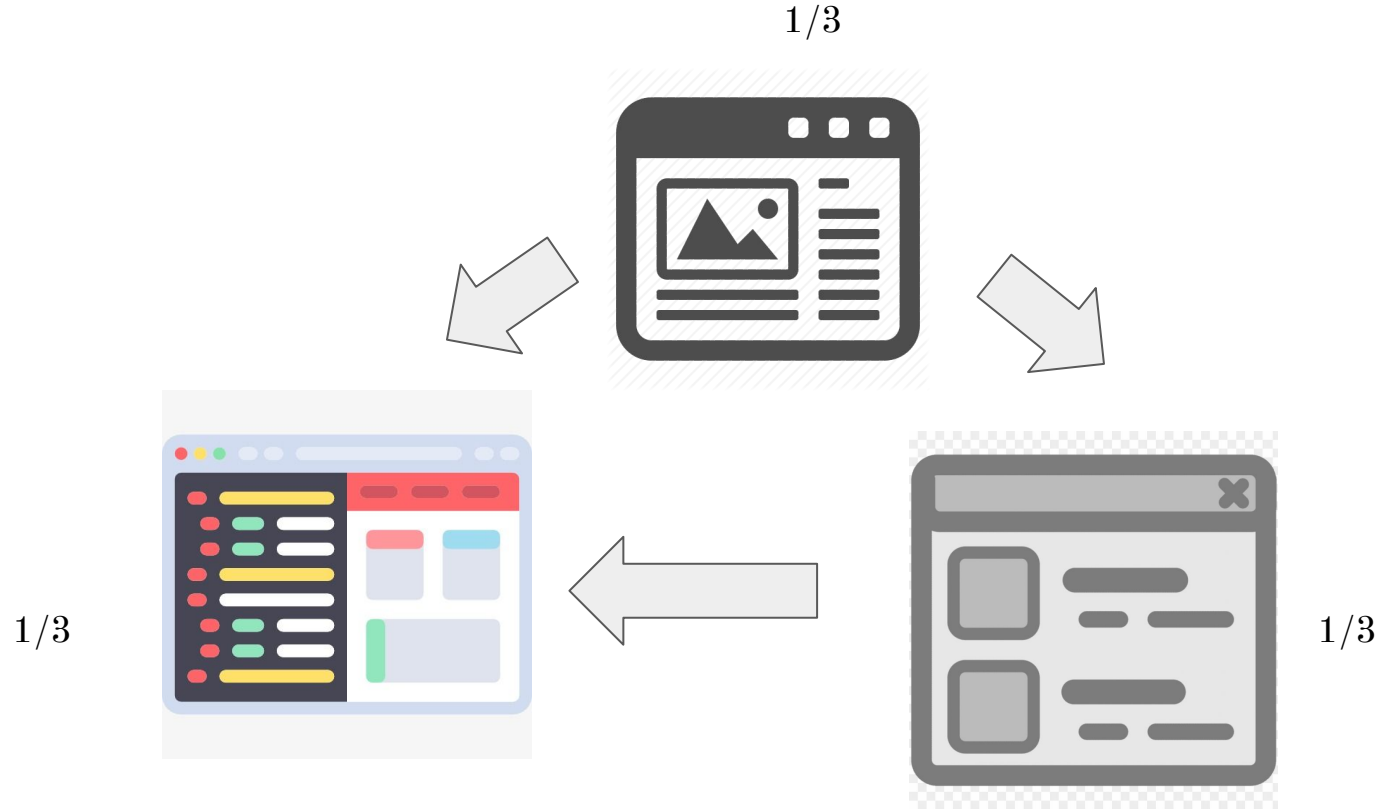
# What is PageRank?



# What is PageRank?

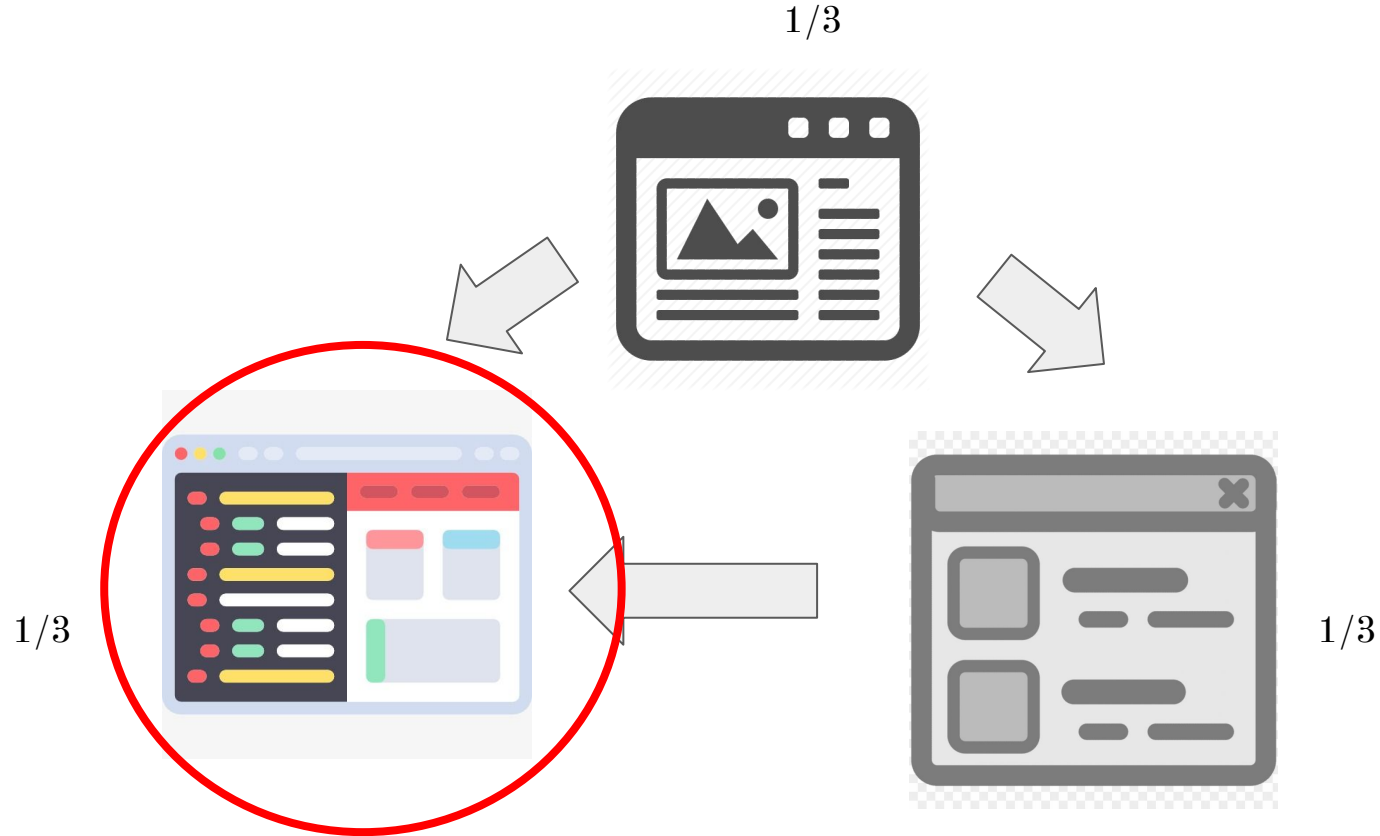


# What is PageRank?

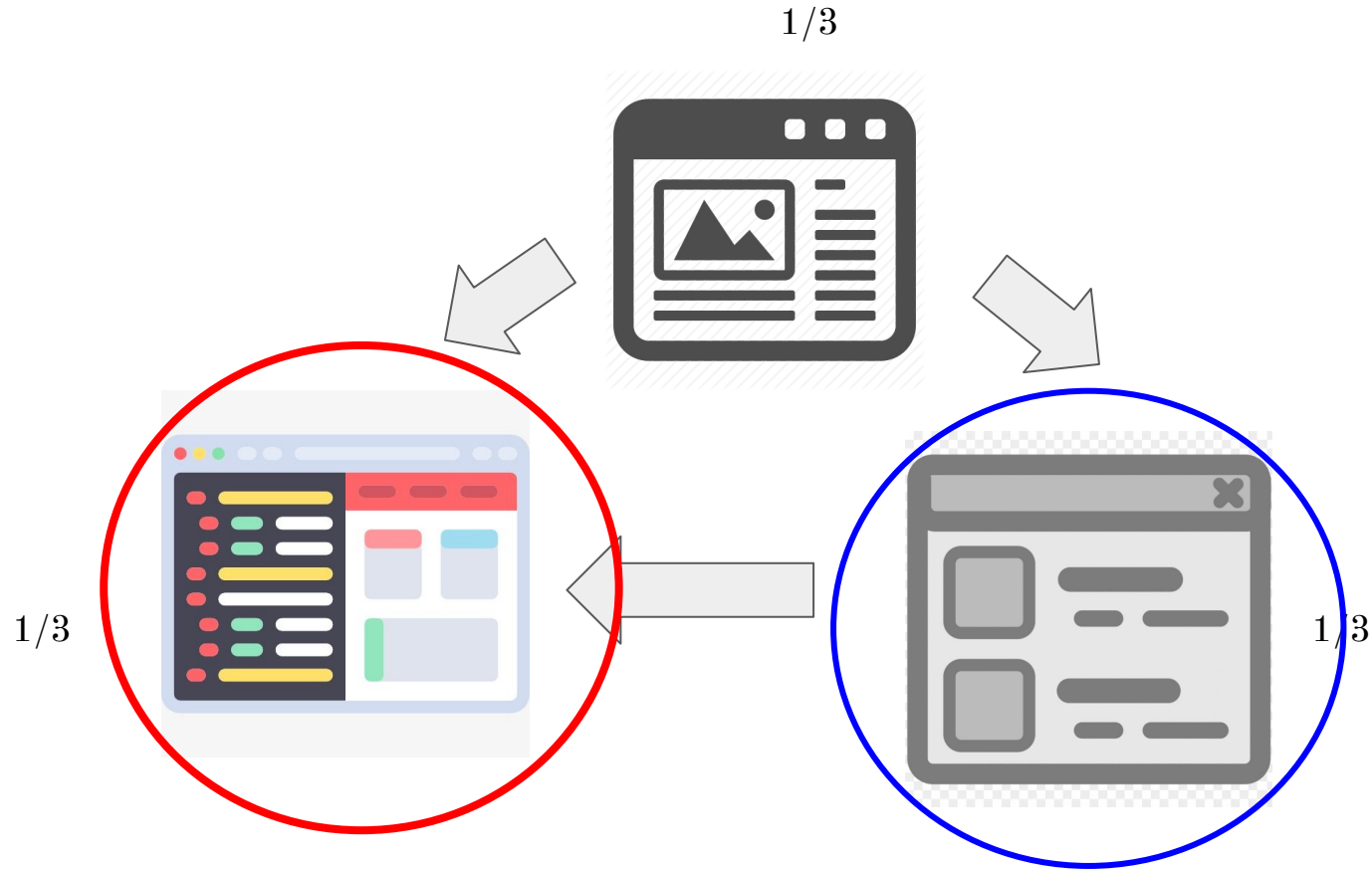




# What is PageRank?



# What is PageRank?



# PageRank Update Formula

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

# PageRank Update Formula

$V_i$  is our  
target  
node



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

# PageRank Update Formula

$V_i$  is our  
target  
node



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$

# PageRank Update Formula

$V_i$  is our  
target  
node



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$

# PageRank Update Formula

$V_i$  is our  
target  
node



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$



This is the number of  
edges that are going  
out of  $V_j$

# PageRank Update Formula

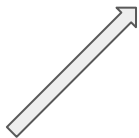
$V_i$  is our  
target  
node



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$d$  is a damping factor  
(normally  $d=0.85$ )

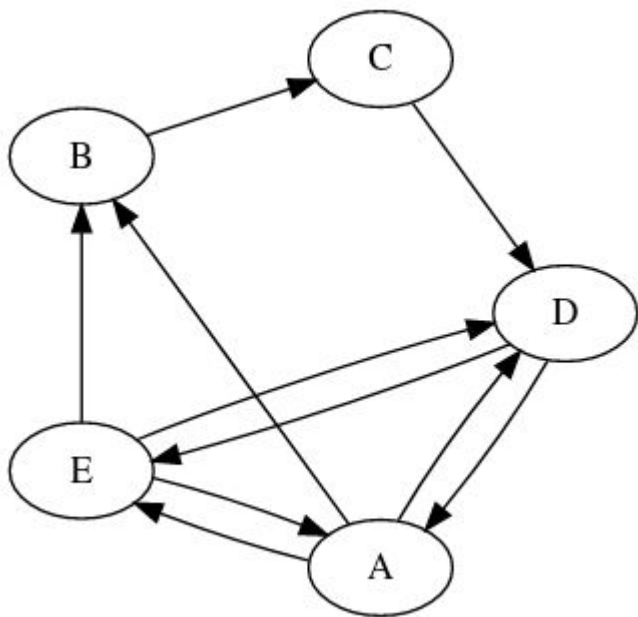


$In(V_i)$  is the set  
of vertices that  
point to  $V_i$

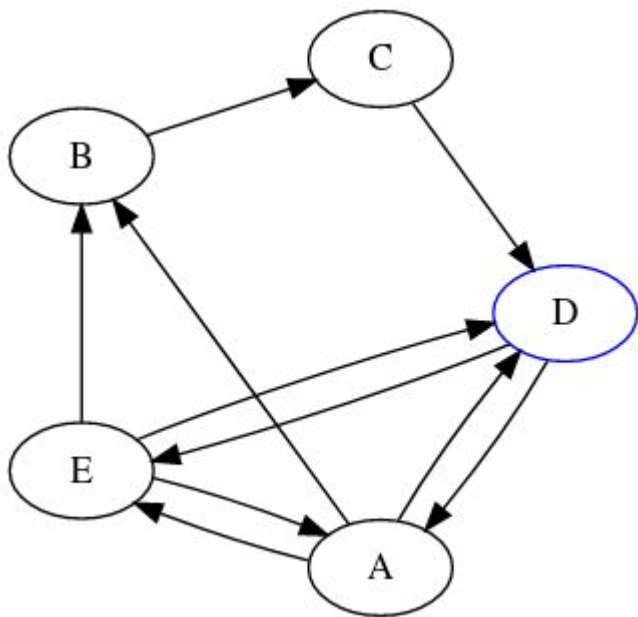


This is the number of  
edges that are going  
out of  $V_j$





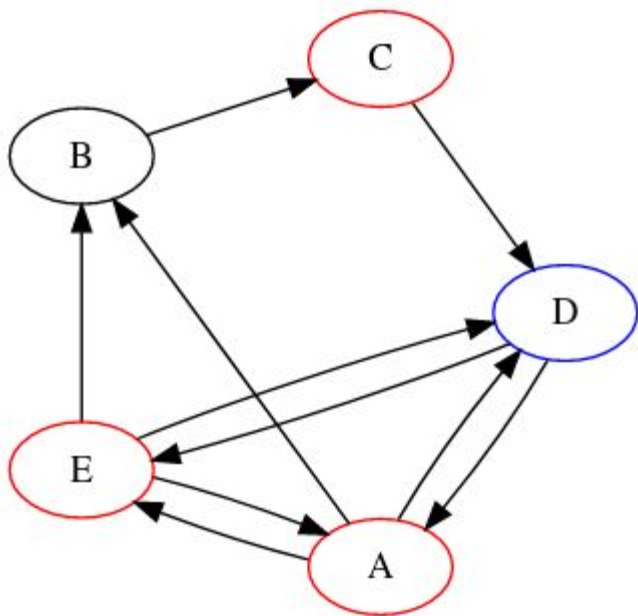
$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$V_i$  is our  
target  
node



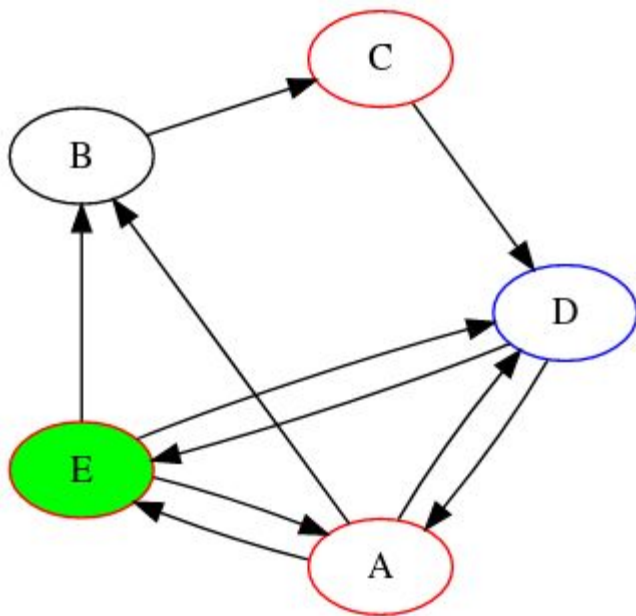
$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



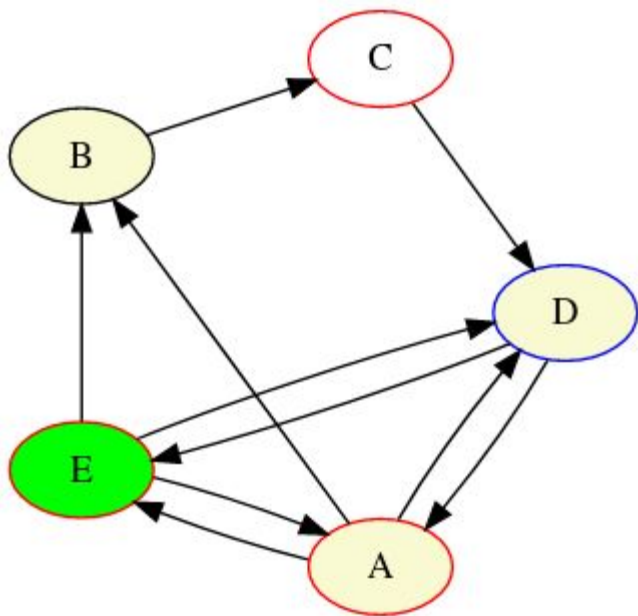
$In(V_i)$  is the set  
of vertices that  
point to  $V_i$



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$



This is the number of  
edges that are going  
out of  $V_j$

# Back to TextRank

- Also uses a graph representation where the edges “vote” for the various vertices

# Back to TextRank

- Also uses a graph representation where the edges “vote” for the various vertices
- But there are no links in sentences -- what do we use for the sentences to recommend other sentences?

# Back to TextRank

- Also uses a graph representation where the edges “vote” for the various vertices
- But there are no links in sentences -- what do we use for the sentences to recommend other sentences?
- We use similarity between sentences!



# Back to TextRank

- Also uses a graph representation where the edges “vote” for the various vertices
- But there are no links in sentences -- what do we use for the sentences to recommend other sentences?
- We use similarity between sentences!
- Instead of a directed unweighted graph we use a weighted undirected graph

# Similarity Metric

$$\textit{Similarity}(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}$$

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Cras ut est a ipsum posuere auctor ac eget ex.

Morbi lacinia justo sit amet consectetur commodo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

0.2282

Cras ut est a ipsum posuere auctor ac eget ex.

Morbi lacinia justo sit amet consectetur commodo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

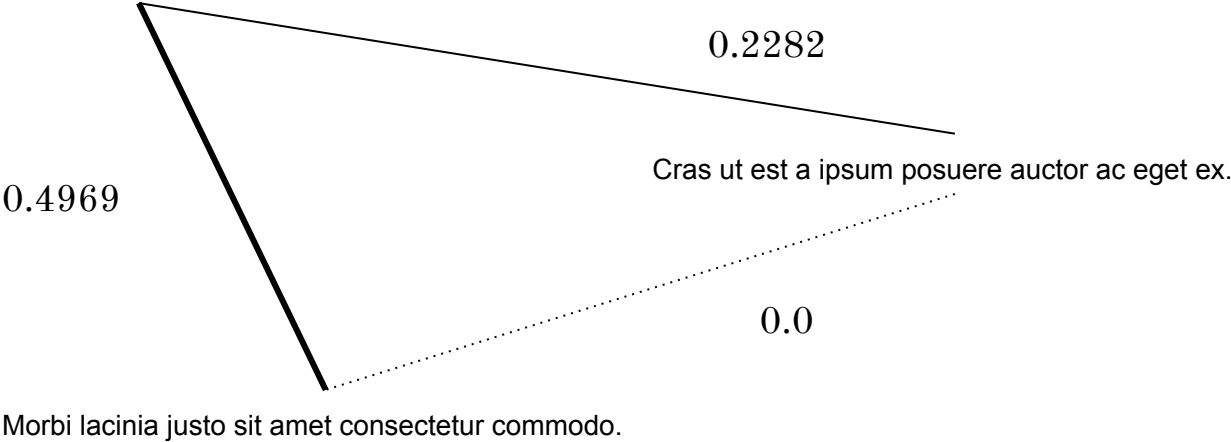
0.2282

Cras ut est a ipsum posuere auctor ac eget ex.

0.0

Morbi lacinia justo sit amet consectetur commodo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.



# What is convergence in this context?

- When the error rate for any vertex falls below a certain threshold
- Because this is an unsupervised task, that is calculated as the difference between the scores in successive iterations:

$$S^{k+1}(V_i) - S^k(V_i)$$

# TextRank Update Formula

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$



# TextRank Update Formula

$V_i$  is our  
target  
node



$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

# TextRank Update Formula

$V_i$  is our  
target  
node



$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$

# TextRank Update Formula

$V_i$  is our  
target  
node



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$

# TextRank Update Formula

$V_i$  is our  
target  
node



$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$



$In(V_i)$  is the set  
of vertices that  
point to  $V_i$



This is the sum of  
weights for all edges  
that are going out of  
 $V_j$

# TextRank Update Formula

$V_i$  is our  
target  
node

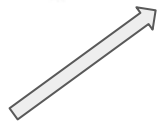


$V_j$  is one of the  
nodes in  $In(V_i)$ ,  
our source node  
for this part of the  
update



$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

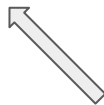
$d$  is a damping factor  
(normally  $d=0.85$ )



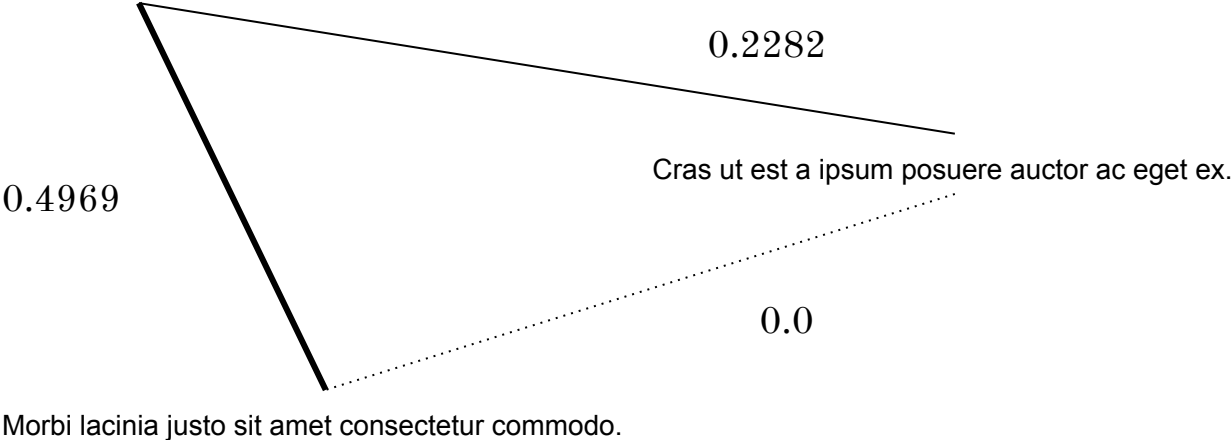
$In(V_i)$  is the set  
of vertices that  
point to  $V_i$



This is the sum of  
weights for all edges  
that are going out of  
 $V_j$



Lorem ipsum dolor sit amet, consectetur adipiscing elit.



**1.45**

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

0.2282

**0.54**

Cras ut est a ipsum posuere auctor ac eget ex.

0.4969

0.0

Morbi lacinia justo sit amet consectetur commodo.

**1.00**

Now that we have some of the basic concepts,  
let's look at some code!\*

\* Adapted from Brian Lester's implementation of TextRank: <https://github.com/blester125/text-rank>



# Summary #1

Volkswagen is partnering with the University of Tennessee and the Oak Ridge National Laboratory to create the company's first innovation hub for developing new technology in North America, officials said Friday in a statement. "Working with the University of Tennessee and Oak Ridge National Laboratory is a great opportunity to continue growing Volkswagen's engineering footprint in the North American region," said Wolfgang Demmelbauer-Ebner, VW's executive vice president and chief engineering officer for the region.

## Summary #2

“4 Min Read MANILA (Reuters) - Schools and businesses shut across the Philippine capital on Monday as a volcano belched clouds of ash across the city and seismologists warned an eruption could happen at any time, potentially triggering a tsunami. Thousands of people were forced to evacuate their homes around Taal, one of the world's smallest active volcanoes, which spewed ash for a second day from its crater in the middle of a lake about 70 km (45 miles) south of central Manila.”

# Early ML Solutions for NLP

- TextRank for summarization
- **Simple n-gram models for language modeling**
- Naive Bayes, Logistic Regression for classification
- Various flavors of Markov Models for tagging
- Now, however, almost all solutions are based on deep learning

# 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!

# How to do n-gram language modeling

We're trying to find  $P(w|h)$

# How to do n-gram language modeling

We're trying to find  $P(w|h)$

$P(\textit{the}|\textit{its water is so transparent that})$

# How to do n-gram language modeling

We're trying to find  $P(w|h)$

$P(\textit{the}|\textit{its water is so transparent that})$

=

$$\frac{C(\textit{its water is so transparent that the})}{C(\textit{its water is so transparent that})}$$



## How to do n-gram language modeling

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

therefore

$$P(A, B) = P(A)P(B|A)$$

If you add more variables:

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

How to do n-gram language modeling

$$P(w_1, w_2, \dots, w_{n-1}) =$$

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^n P(w_k|w_1^{k-1}) \end{aligned}$$

# How to do n-gram language modeling

We can apply a simplifying assumption:

$$P(\textit{the}|\textit{its water is so transparent that}) \approx P(\textit{the}|\textit{that})$$

Or perhaps:

$$P(\textit{the}|\textit{its water is so transparent that}) \approx P(\textit{the}|\textit{transparent that})$$

# Unigram Language Model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

# Unigram Language Model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars,  
quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

# Bigram Language Model

$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2|w_1)P(w_3|w_2)P(w_4|w_3)$$

# Bigram Language Model

$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2|w_1)P(w_3|w_2)P(w_4|w_3)$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said,  
mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five,  
hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

# How many Ns in your n-gram?

- N-gram models can also work for trigrams, 4-grams, and 5-grams
- Note that this is not a full model of language because language contains long-range dependencies

“The *people* who have worked the longest on this project *are* the most industrious.”



# How to do n-gram language modeling

## Maximum Likelihood Estimate

- Counts from a corpus (a large body of text that serves as training data) and normalizes the counts so that they end up between 0 and 1

# How to do n-gram language modeling

## Maximum Likelihood Estimate

- Counts from a corpus (a large body of text that serves as training data) and normalizes the counts so that they end up between 0 and 1

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

# How to do n-gram language modeling

An example\*:

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>  
<s> Sam I am </s>  
<s> I do not like green eggs and ham </s>

$$\begin{array}{lll} P(\text{I} | \text{<s>}) = \frac{2}{3} = .67 & P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33 & P(\text{am} | \text{I}) = \frac{2}{3} = .67 \\ P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5 & P(\text{Sam} | \text{am}) = \frac{1}{2} = .5 & P(\text{do} | \text{I}) = \frac{1}{3} = .33 \end{array}$$

\*From Dan Jurafsky's lecture slides

# How to do language modeling

- In order to avoid “floating point underflow”, in practice we do all of our calculations in log space rather than multiplying

$$\log(p_1 p_2 p_3 p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

# How to evaluate your n-gram language model

- Perplexity

# How to evaluate your n-gram language model

- Perplexity



# How to evaluate your n-gram language model

- Perplexity

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}}$$

# How to evaluate your n-gram language model

- Pretend you have a sentence that has random digits
- The perplexity of this sentence according to a model that assigns  $p=1/10$  to each digit:

$$\begin{aligned}\text{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10\end{aligned}$$



# How to evaluate your n-gram language model

A lower perplexity = a better model

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

# Time to look at some code

Let's see how an implementation of an n-gram language model could work.

# **Preparing Linguistic Data for Deep Learning NLP Solutions**

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

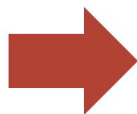
**1. Words are not numbers**

2. Input can be different lengths

# 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...*

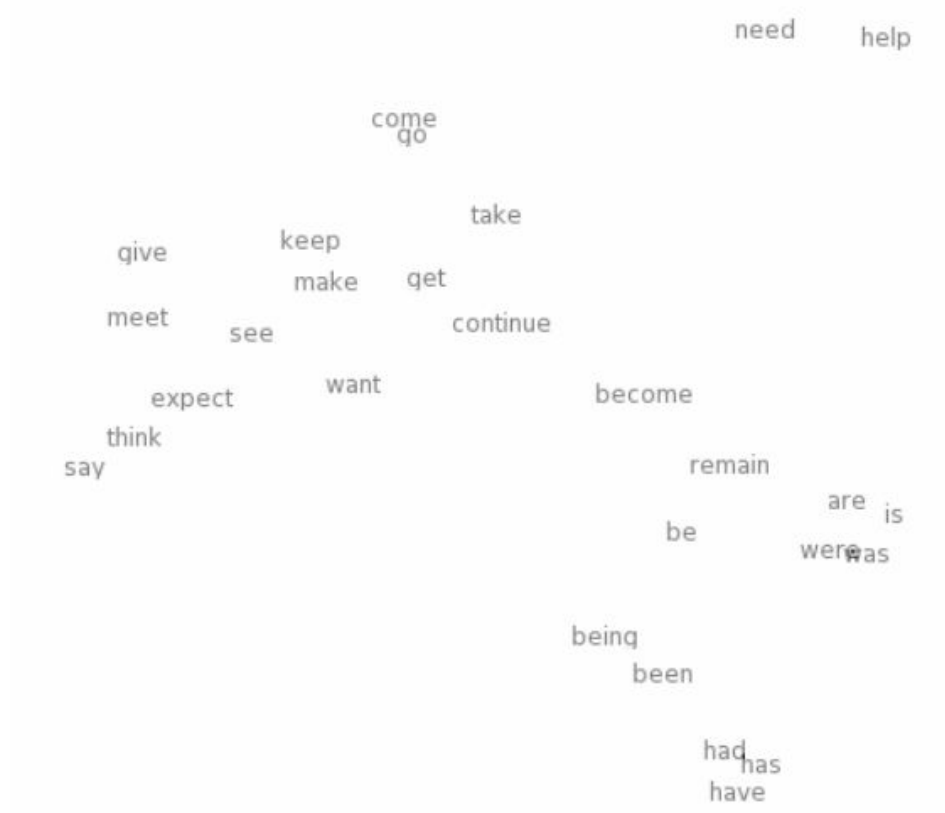
*...India has just given its **banking** system a shot in the arm...*



## Quick quiz:

Does this sound like anything you've learned about in your text analytics class?  
Defining word vectors by their context?

# Word meaning visualized



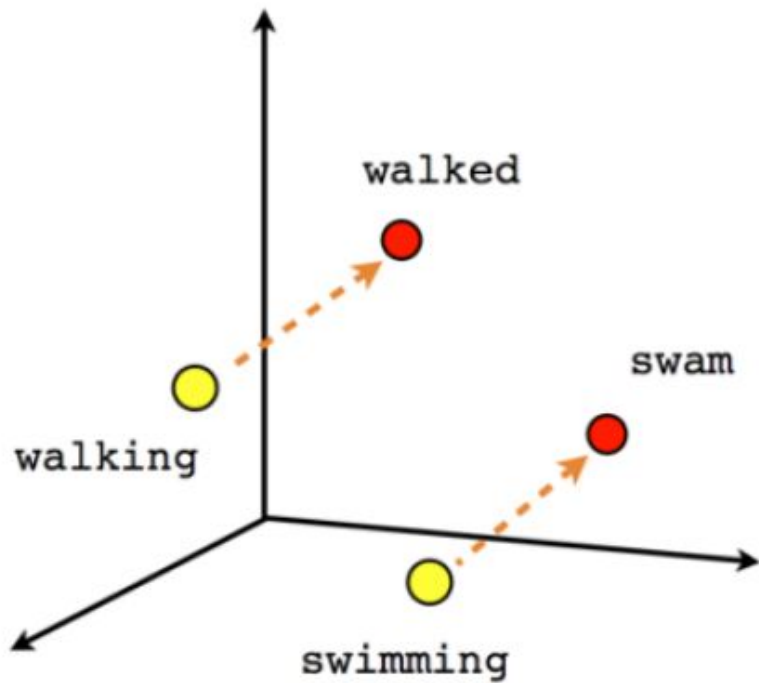
Source: Stanford cs224 notes

# Word Vector Arithmetic

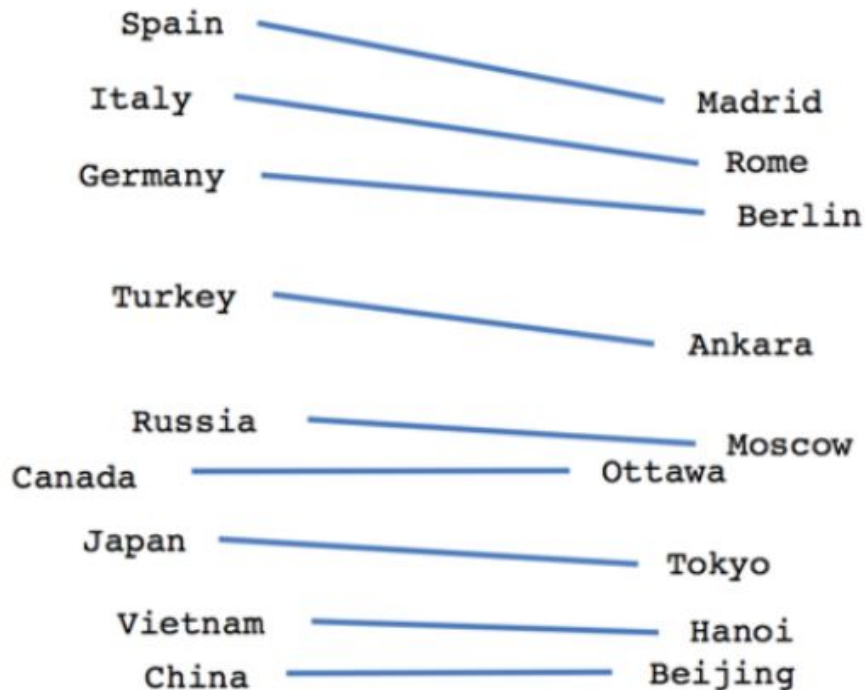


Source: [the morning paper](#)

# 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” task
- We then pop out the hidden layer of that neural network and use that as our embedding

## Quick quiz:

What NLP task have we already talked about (hint: one that takes into account the context of words) might be a good fake task for this algorithm to use?

# 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 vectors 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

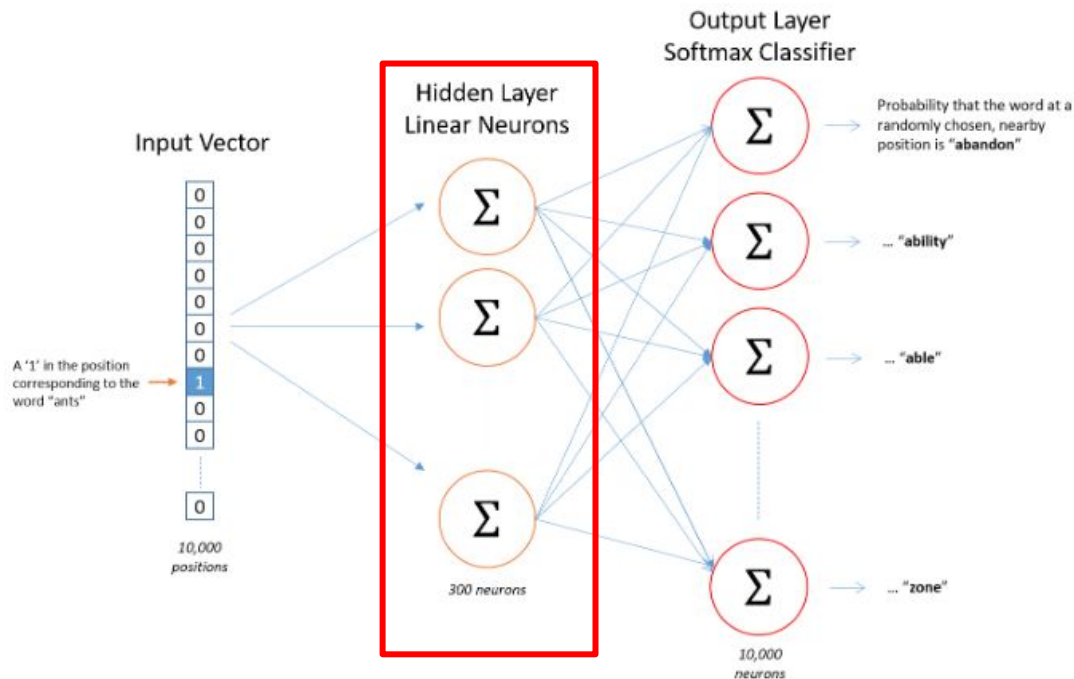


## 2. Sliding Window

derekchia.com

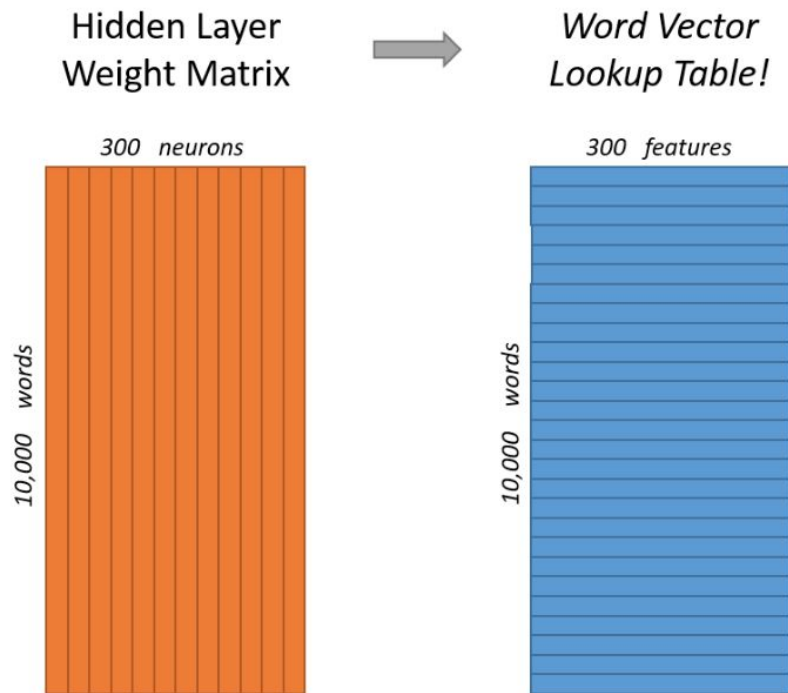
#1	natural	language processing	and machine learning is fun and exciting	#1
	X <sub>k</sub>	Y(c=1) Y(c=2)		
#2	natural	language processing and	machine learning is fun and exciting	#2
	Y(c=1)	X <sub>k</sub> Y(c=2) Y(c=3)		
#3	natural	language processing and machine	learning is fun and exciting	#3
	Y(c=1)	Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#4	natural	language processing and machine learning	is fun and exciting	#4
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#5	natural	language processing and machine learning is	fun and exciting	#5
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#6	natural	language processing and machine learning is fun	and exciting	#6
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#7	natural	language processing and machine learning is fun and	exciting	#7
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#8	natural	language processing and machine learning is fun and exciting		#8
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3) Y(c=4)		
#9	natural	language processing and machine learning is fun and exciting		#9
		Y(c=1) Y(c=2) X <sub>k</sub> Y(c=3)		
#10	natural	language processing and machine learning is fun and exciting		#10
		Y(c=1) Y(c=2) X <sub>k</sub>		

# How do we get these vectors?



Source: McCormick ML

# What do we do when we pop out the layer?



# How do we evaluate word vectors?

- There are a couple different kinds of word vectors - word2vec and GloVe, as well as contextualized word embeddings like BERT or ELMo (see further reading)
- Intrinsic evaluation
  - Evaluate on an intermediate task -- such as word vector analogies (like man:woman, king:queen and visualizations)
  - Allows you to understand more about the word embeddings but not downstream tasks
- Extrinsic evaluation
  - How do these vectors perform in the task you need them for? Does your classification accuracy go up when you use these vectors for the same data vs. another kind of word vector?

# Let's play around with word vectors!

If you want to follow along, open your “exploring-word-vectors.ipynb” file in the code folder I gave you.

# **Deep Learning Solutions for NLP**

# Classification Review

- You have a training set 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





# 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:

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

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:

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

$$\begin{bmatrix} 3.0 \\ 5.0 \\ 2.0 \end{bmatrix} \quad \begin{aligned} & e^3 + e^5 + e^2 \\ &= 20.0855 + 148.4132 + 7.3891 \\ &= 175.8878 \end{aligned}$$

# Softmax Example

Original output:

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

$$20.0855/175.8878 =$$

$$148.4132/175.8878 =$$

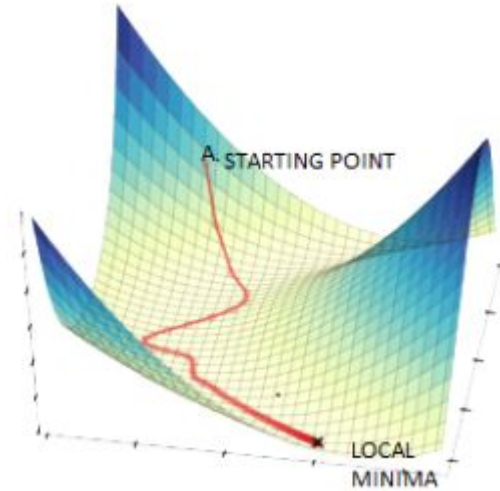
$$7.3891/175.8878 =$$

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

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

# 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

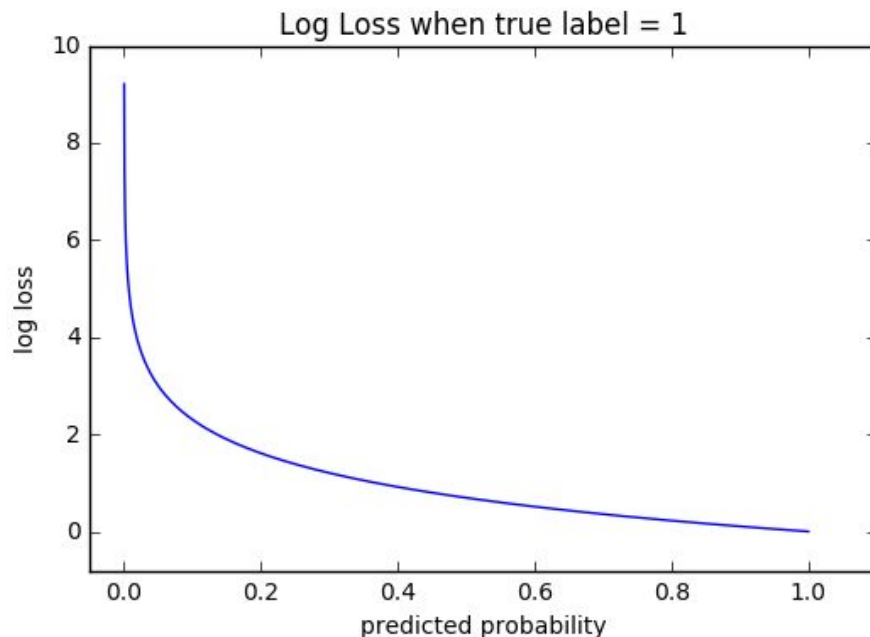


Picture from Data Science Central,

<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



## 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:

Correct label:

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

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

$$\begin{aligned} & -(1 \times \log(0.84)) \\ & = 0.174353387 \end{aligned}$$

# Cross-Entropy Loss

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

Output of softmax:

Correct label:

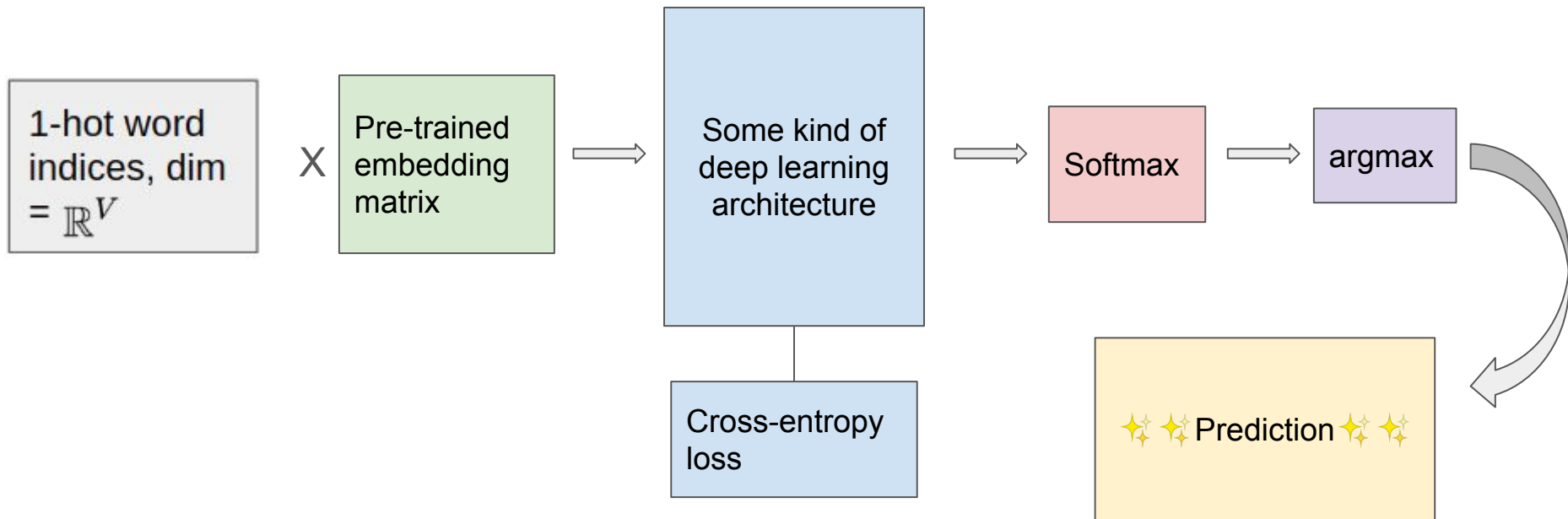
$$\begin{bmatrix} 0.84 \\ 0.12 \\ 0.04 \end{bmatrix}$$

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

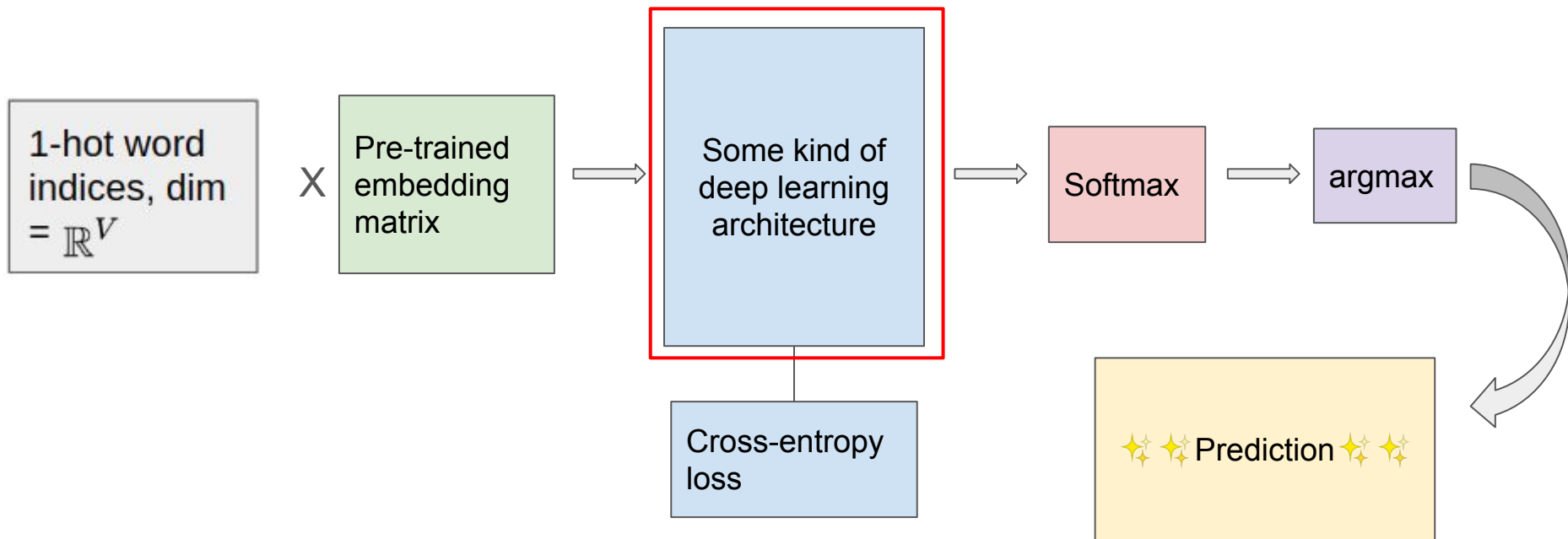
$$\begin{aligned} & -(1 \times \log(0.12)) \\ &= 2.12026353 \end{aligned}$$



# Putting it together



# Putting it together

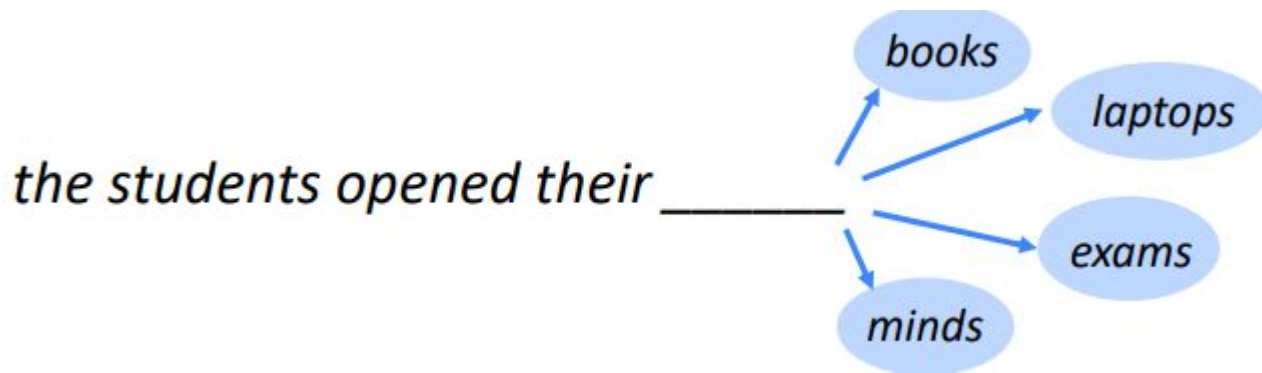


# Deep Learning Architectures for NLP

- (Vanilla) Recurrent Neural Networks (RNNs)
- Long Short-Term Memory (LSTMs)
- CNNs - they're not just for image processing anymore!

# 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



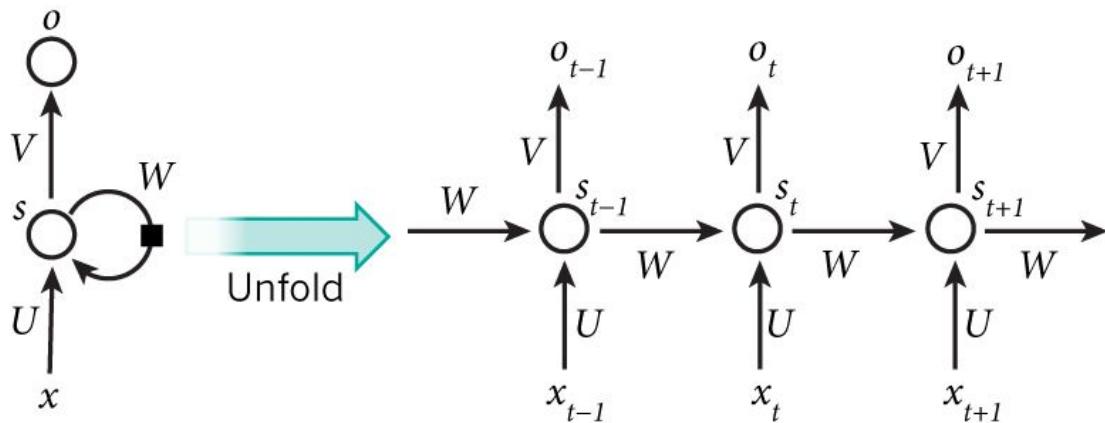
# Language Modeling with Recurrent Neural Networks (RNNs)

- Increasing the size of your n-gram in an n-gram language model causes many problems with computation and sparsity
- However, in order to create output that is more coherent, you need bigger n-grams
- 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  $x_1, \dots, x_T$
  - 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

## Quick quiz:

What metric do we use to evaluate language models again?



# 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






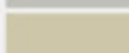





When little strain would be attain'd into being never fed,

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
	Burple Simp 226 181 132
	Stanky Bean 197 162 171
	Turdly 190 164 116

Source: [AI Weirdness](#)

# What is machine learning not capable of?

DINERS, DRIVE-INS, AND DIVES

INT. PARKING LOT

GUY FIERI sits in a convertible. He looks like America.

GUY FIERI  
I'm Guy Fieri and there's nothing  
you can do about it. Today I'm  
eating it all.

Guy takes a bite out of his hair.

INT. DINER'S KITCHEN

Guy and a CHEF stand in a kitchen. Guy has 3 pairs of  
sunglasses on. The sun can't get him.

GUY FIERI  
Prove to me you can panini!

The chef starts boiling a pot of milk. He's scared.

CHEF  
Flavortown is near.

# Drawbacks of RNNs

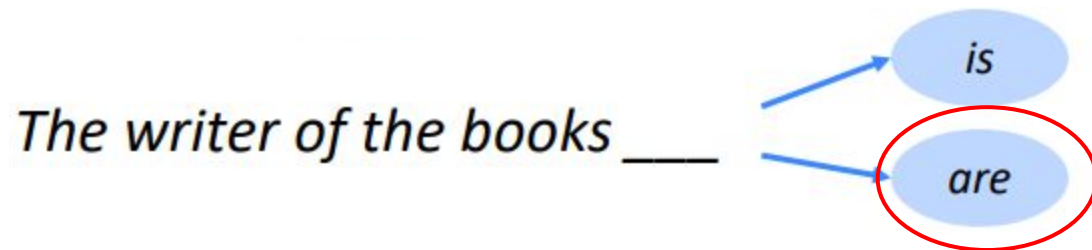
Consider the following word prediction task:

“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

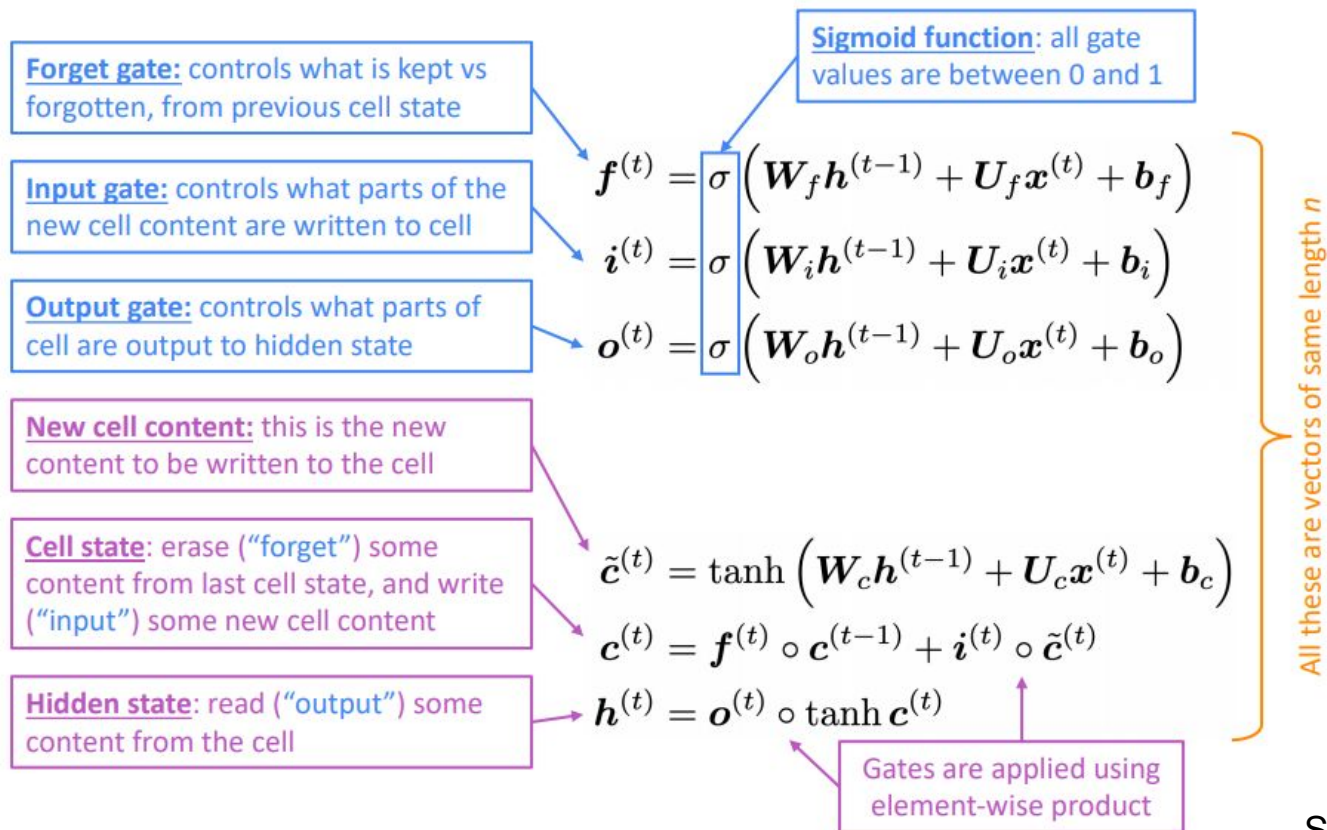


# Long Short-Term Memory (LSTMs)

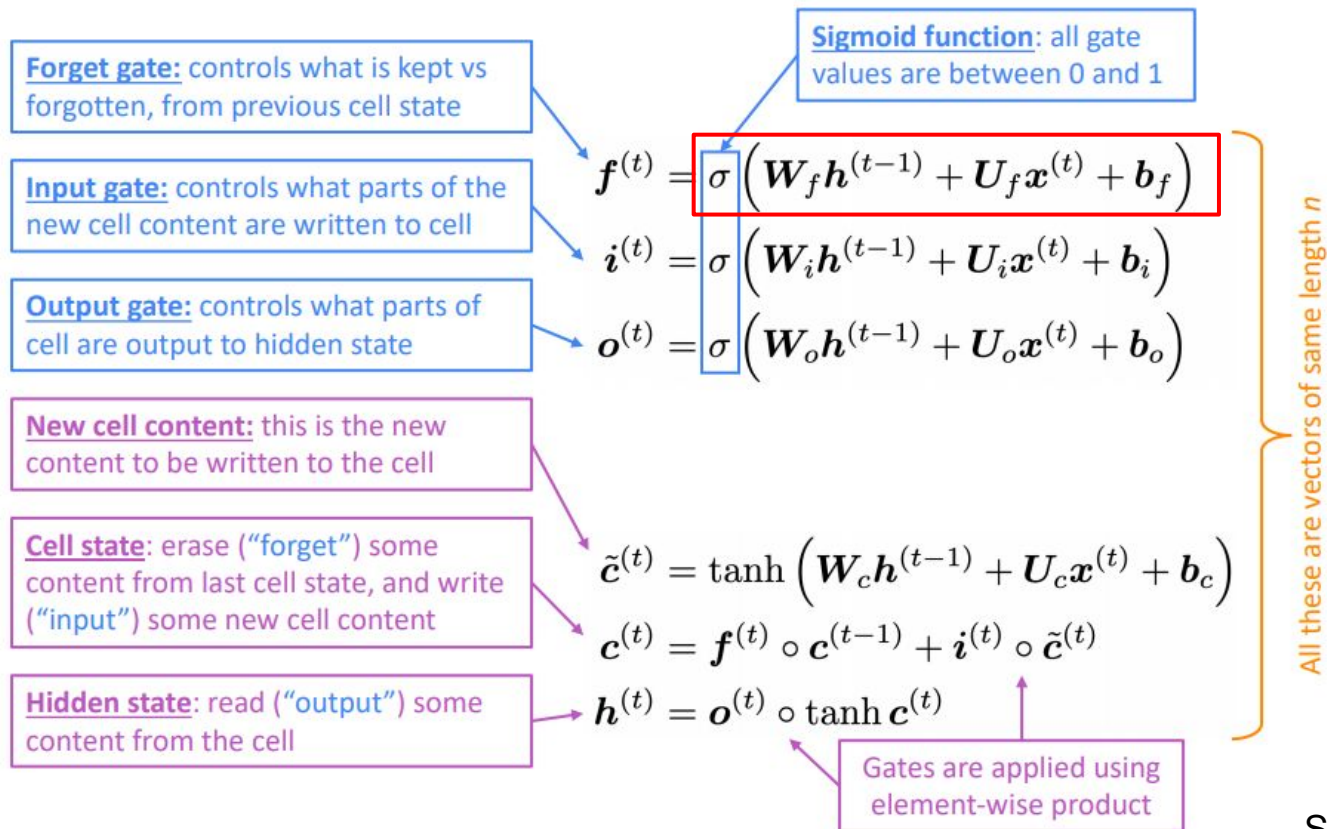
- Like an RNN, but includes a cell state in addition to a hidden state
- The cell stores long-term information
- The LSTM can erase, write and read information from the cell with gates, which are vectors of length  $n$  - the same length as the hidden state and the cell state



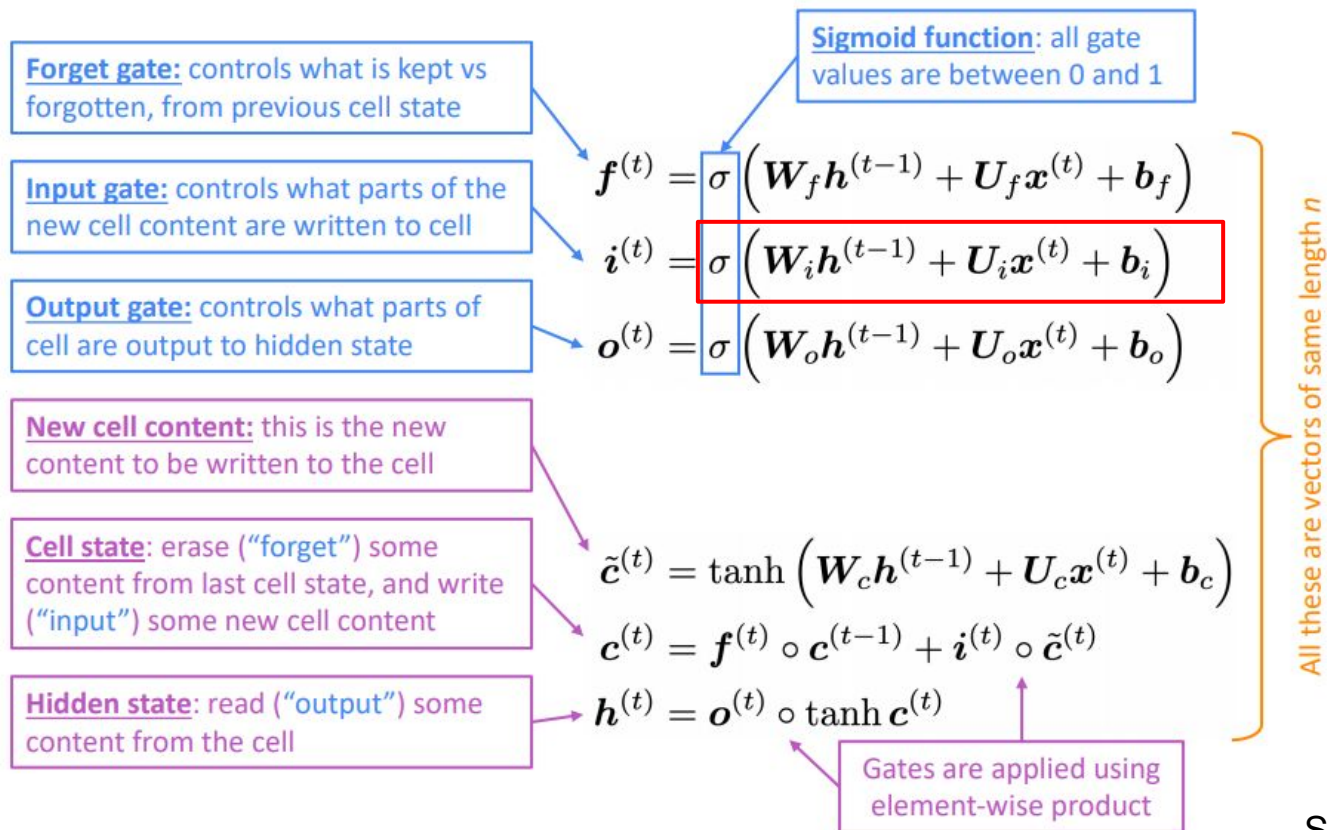
# Long Short-Term Memory (LSTMs)



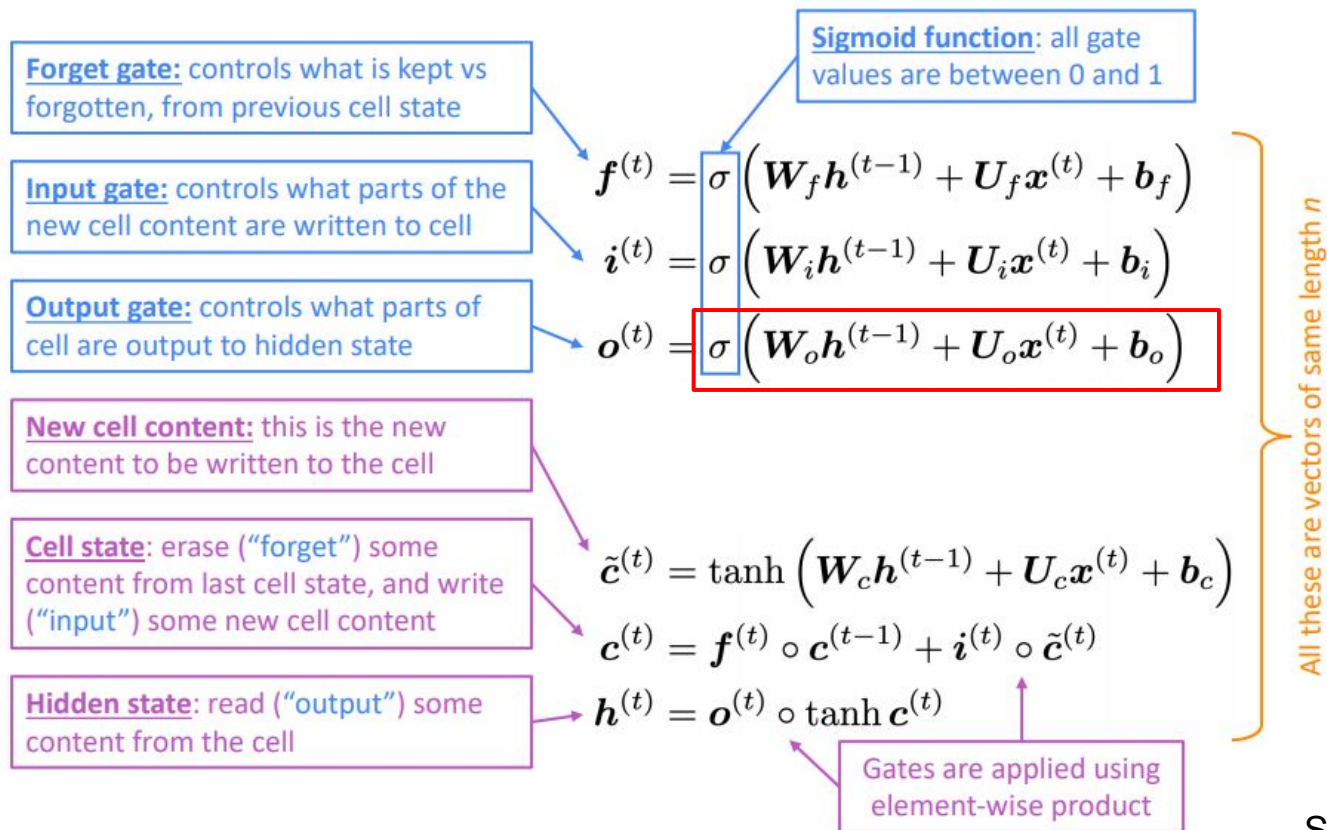
# Long Short-Term Memory (LSTMs)



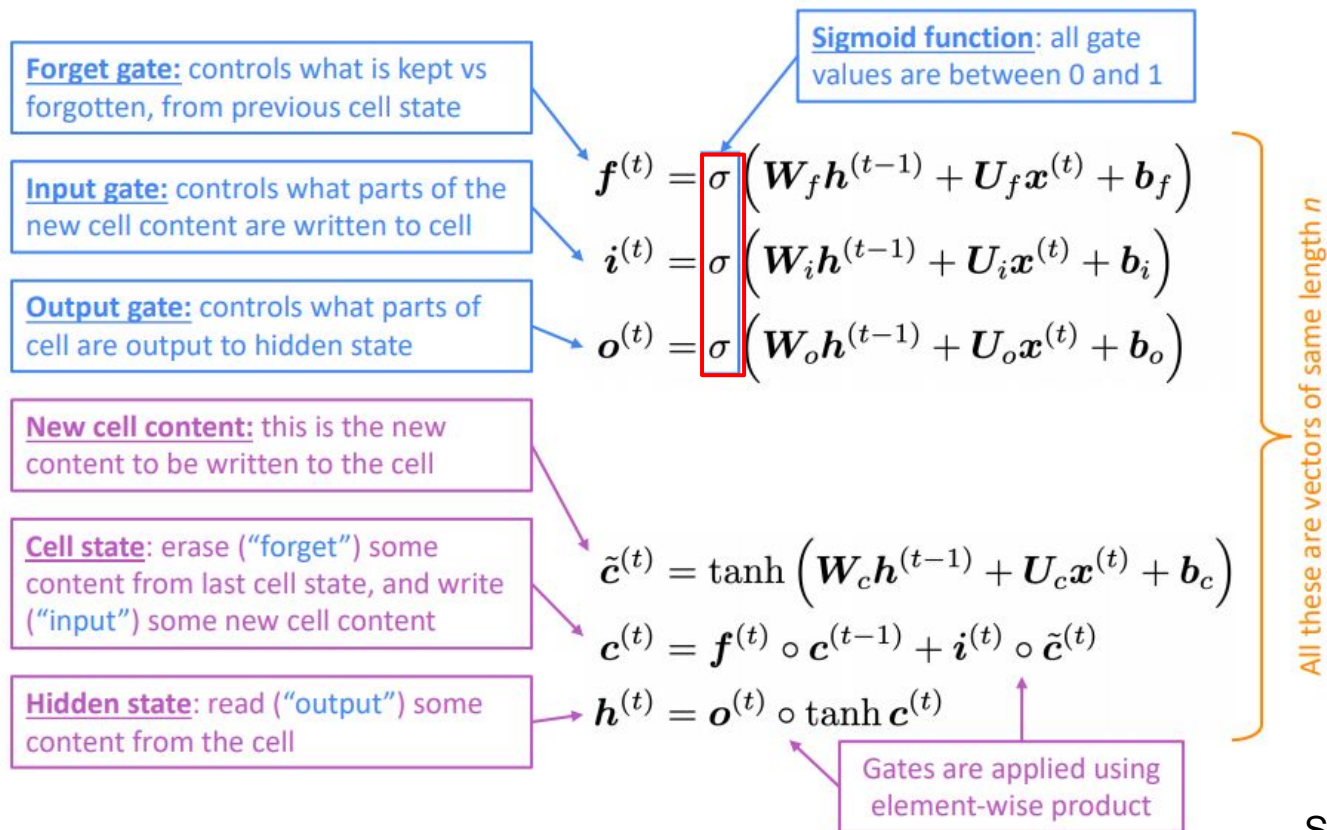
# Long Short-Term Memory (LSTMs)



# Long Short-Term Memory (LSTMs)

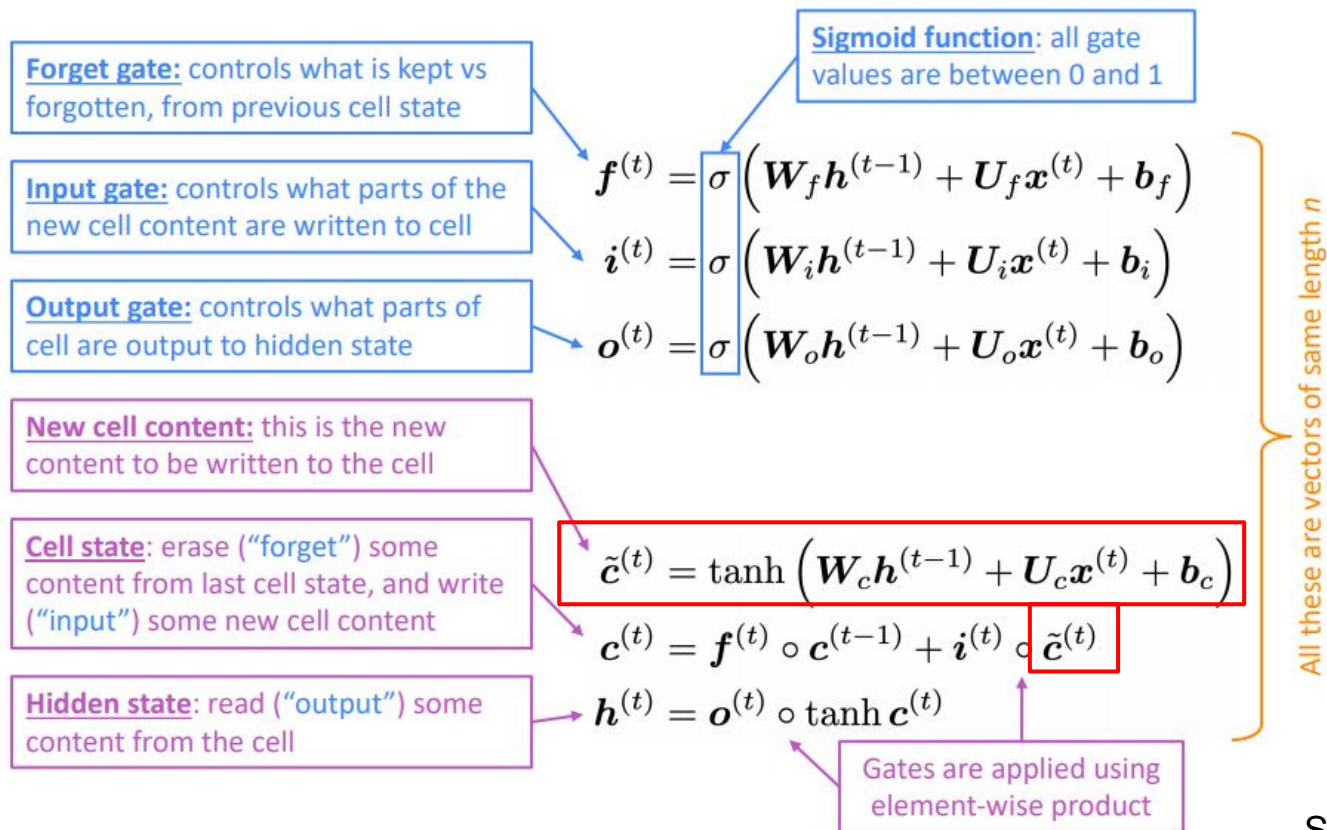


# Long Short-Term Memory (LSTMs)

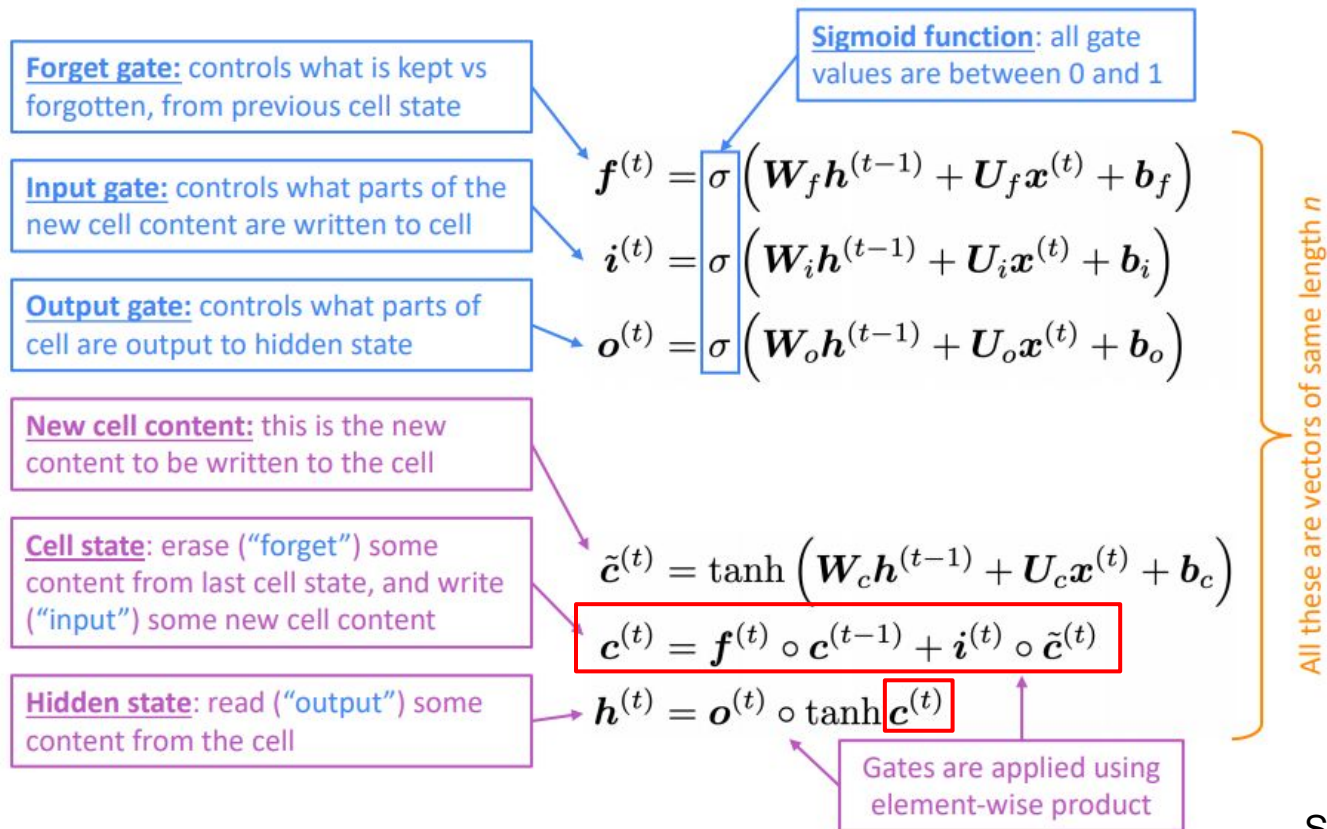




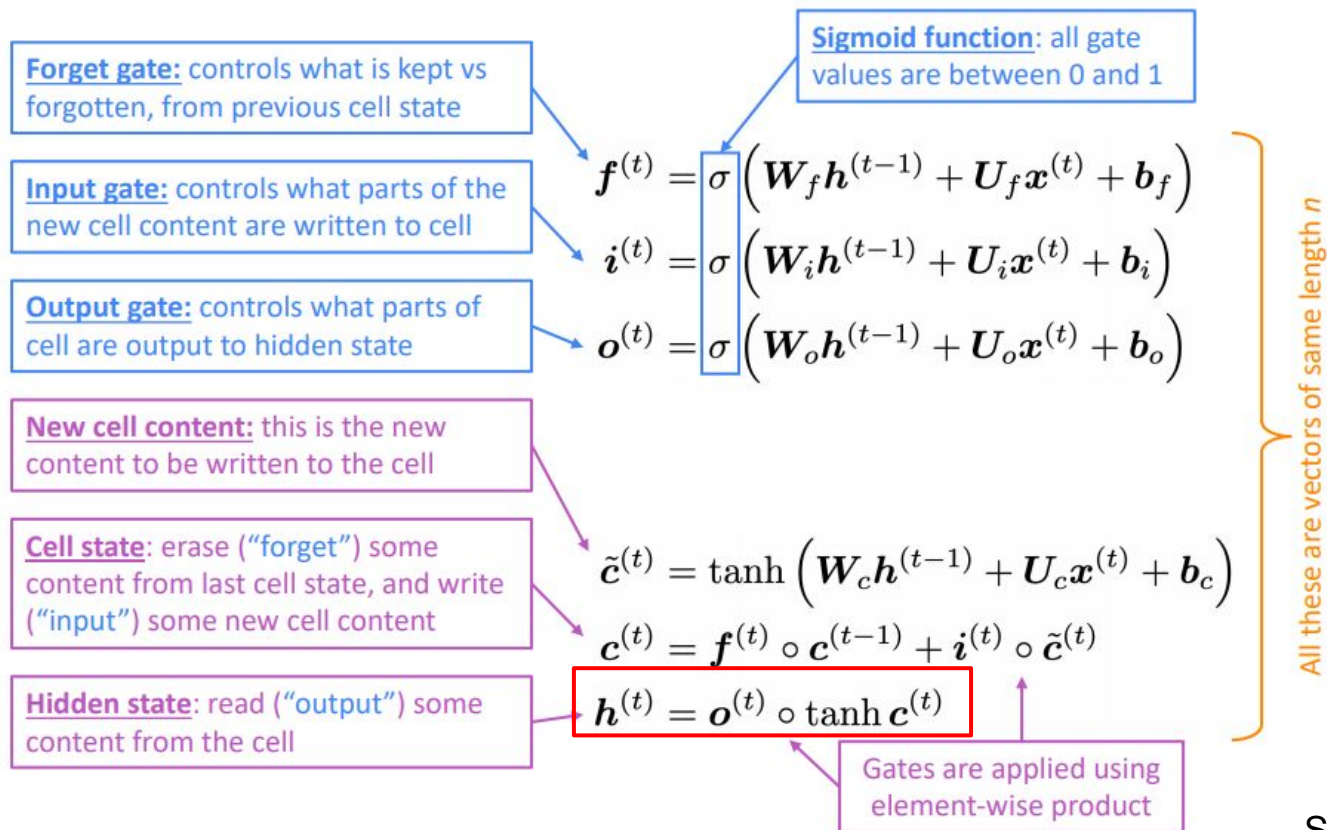
# Long Short-Term Memory (LSTMs)



# Long Short-Term Memory (LSTMs)

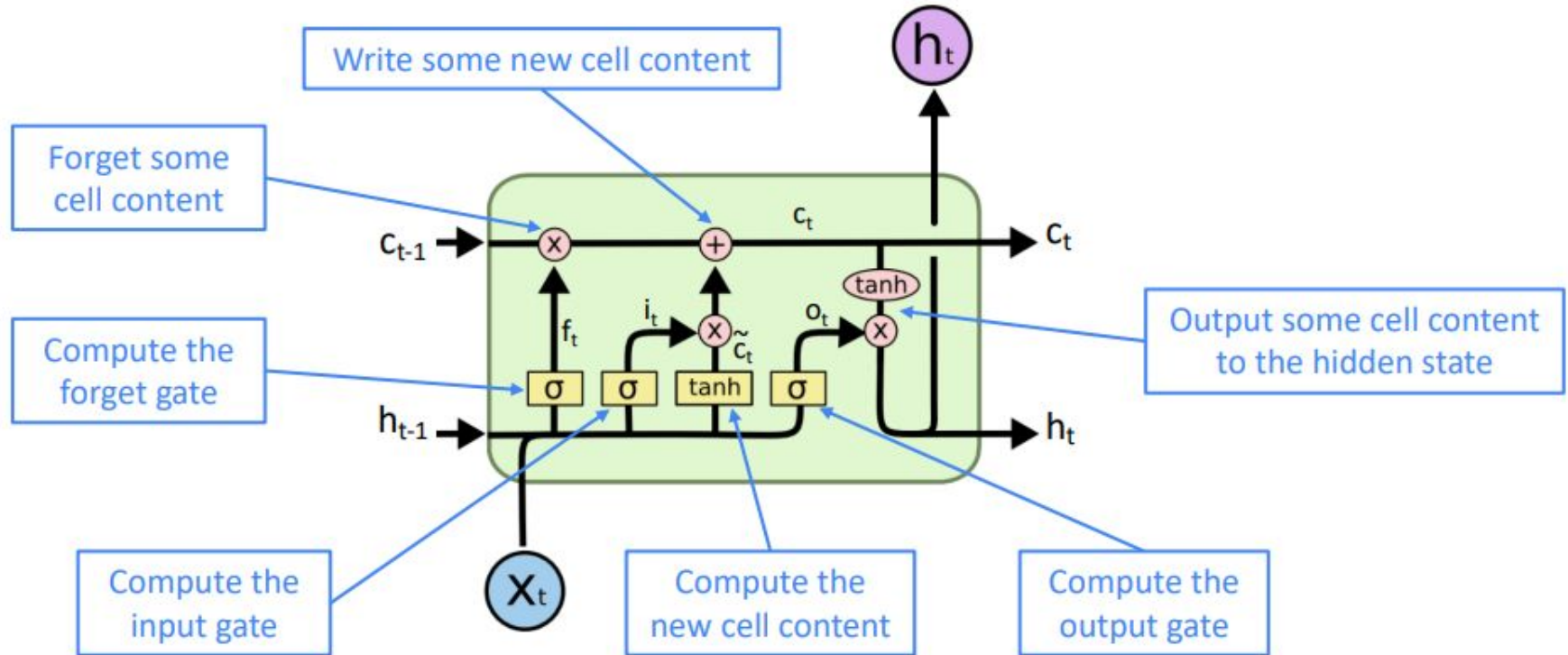


# Long Short-Term Memory (LSTMs)

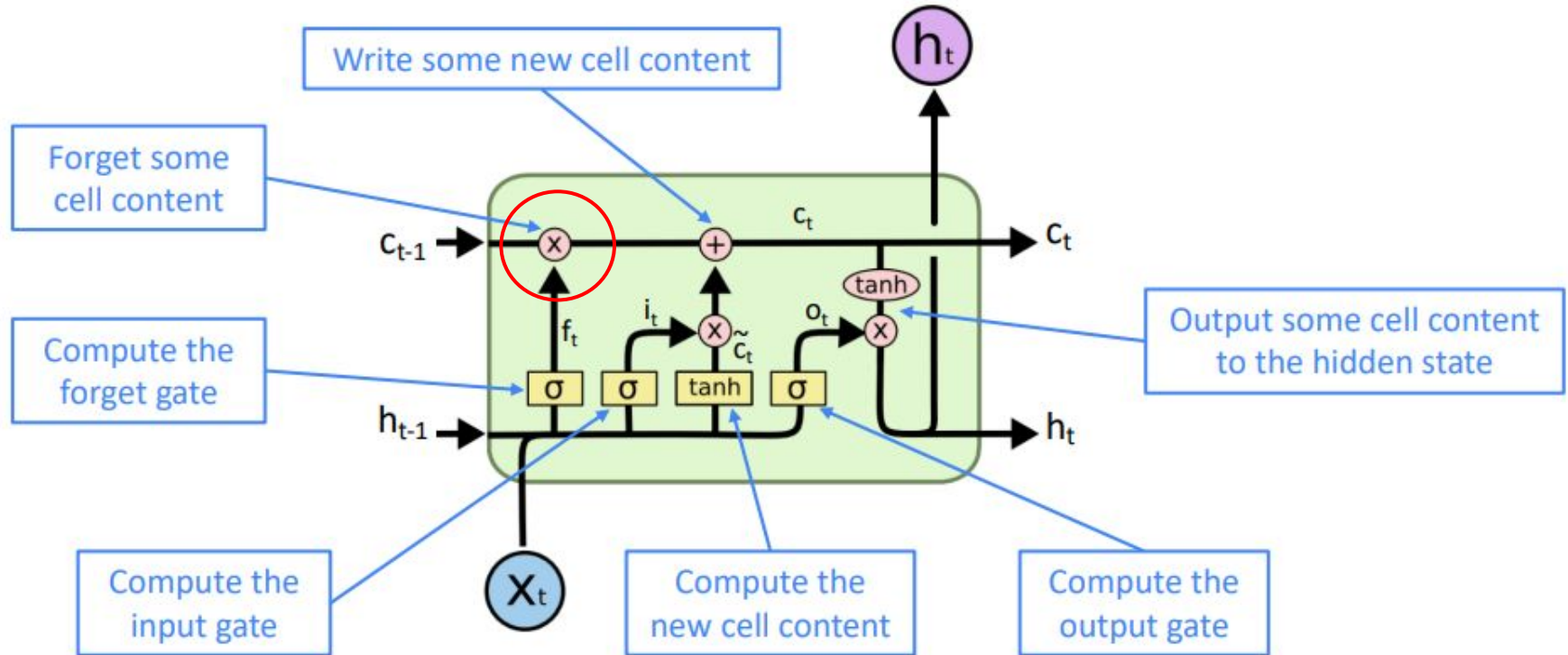




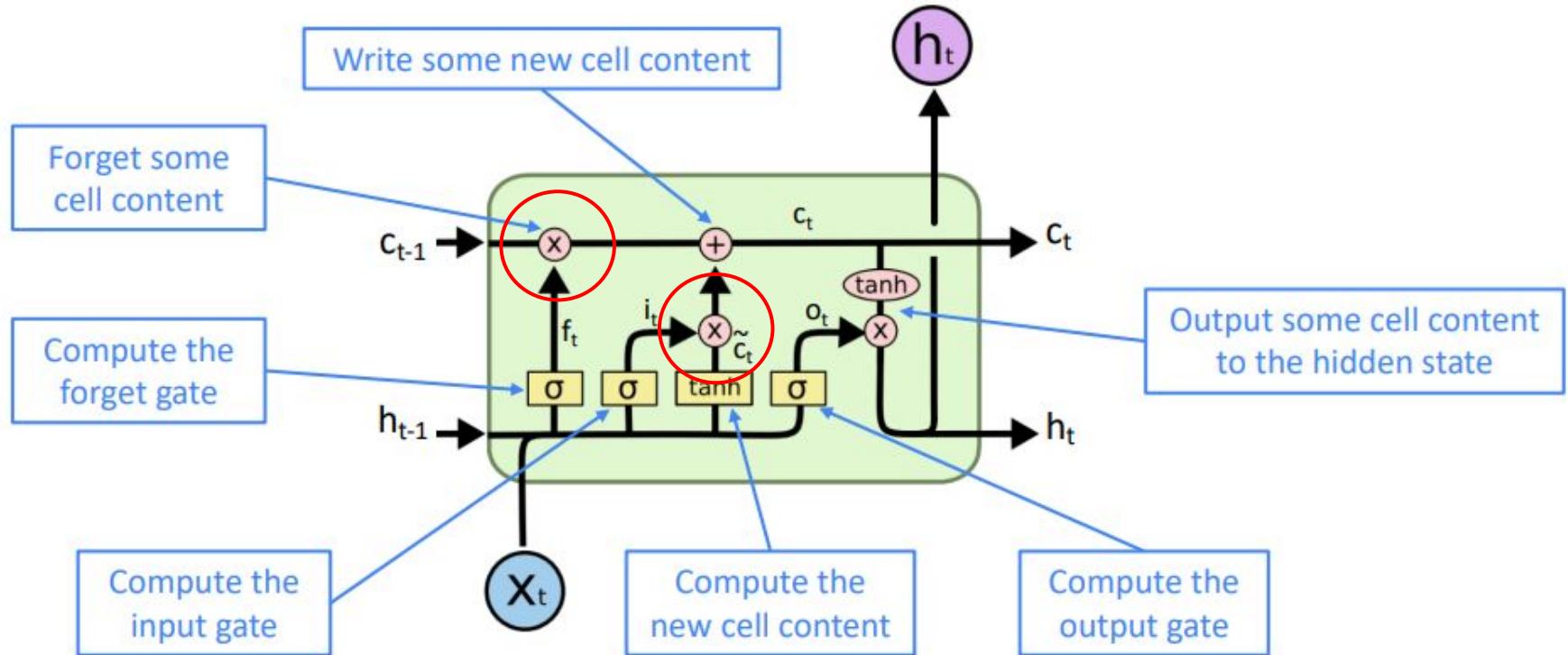
# Long Short-Term Memory (LSTMs)



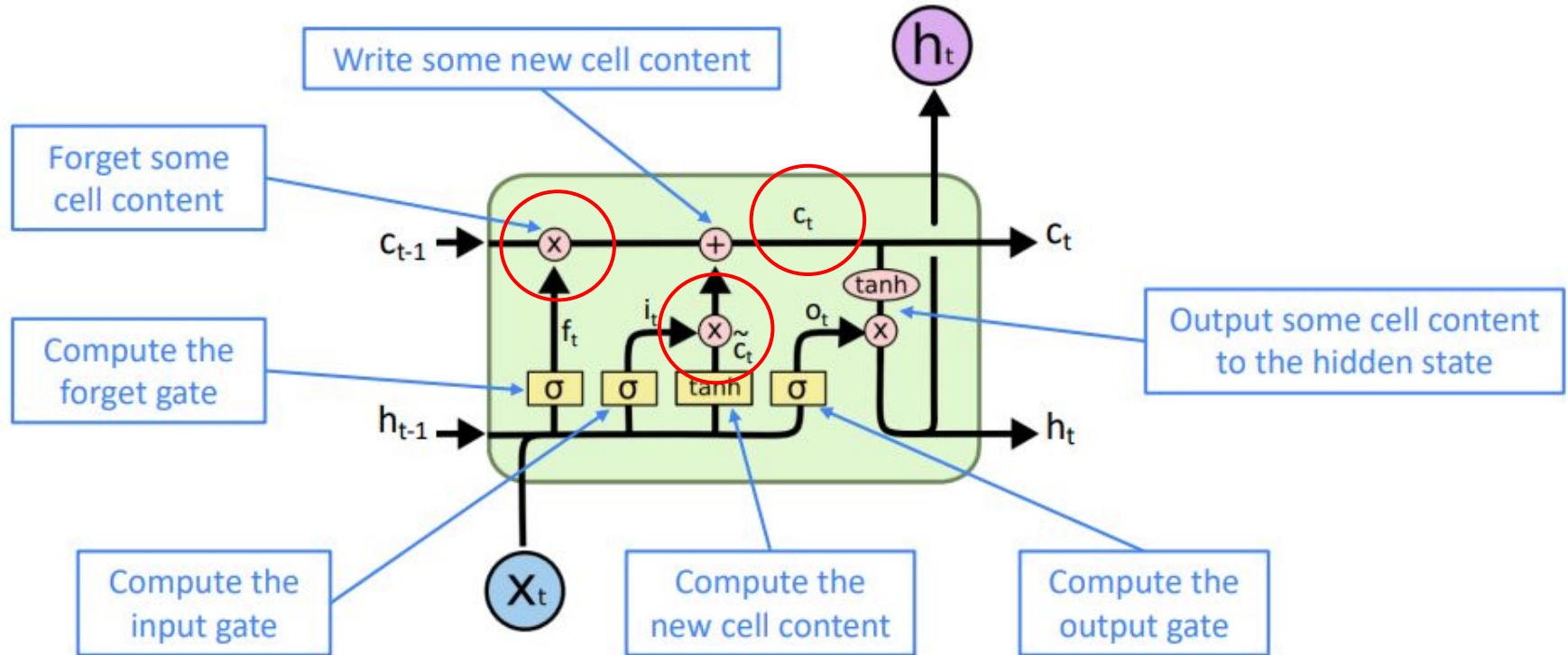
# Long Short-Term Memory (LSTMs)



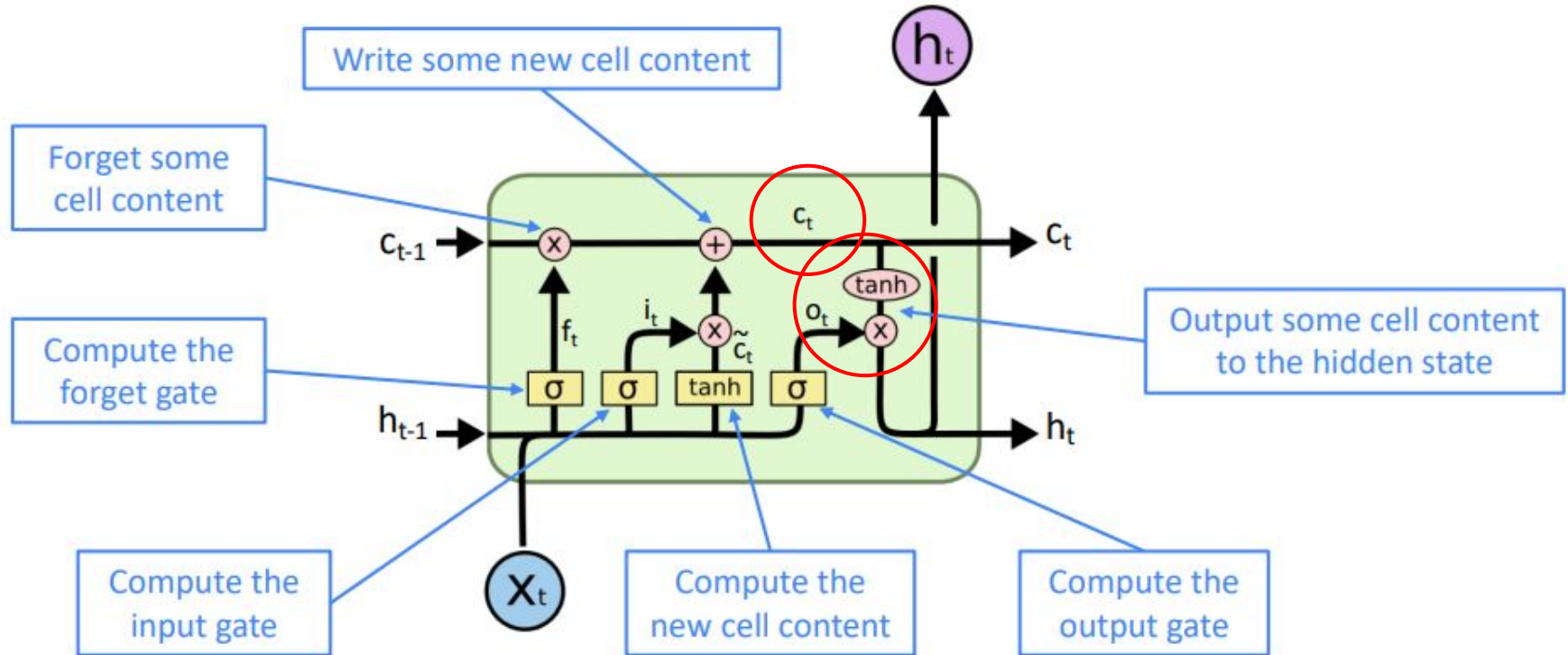
# Long Short-Term Memory (LSTMs)



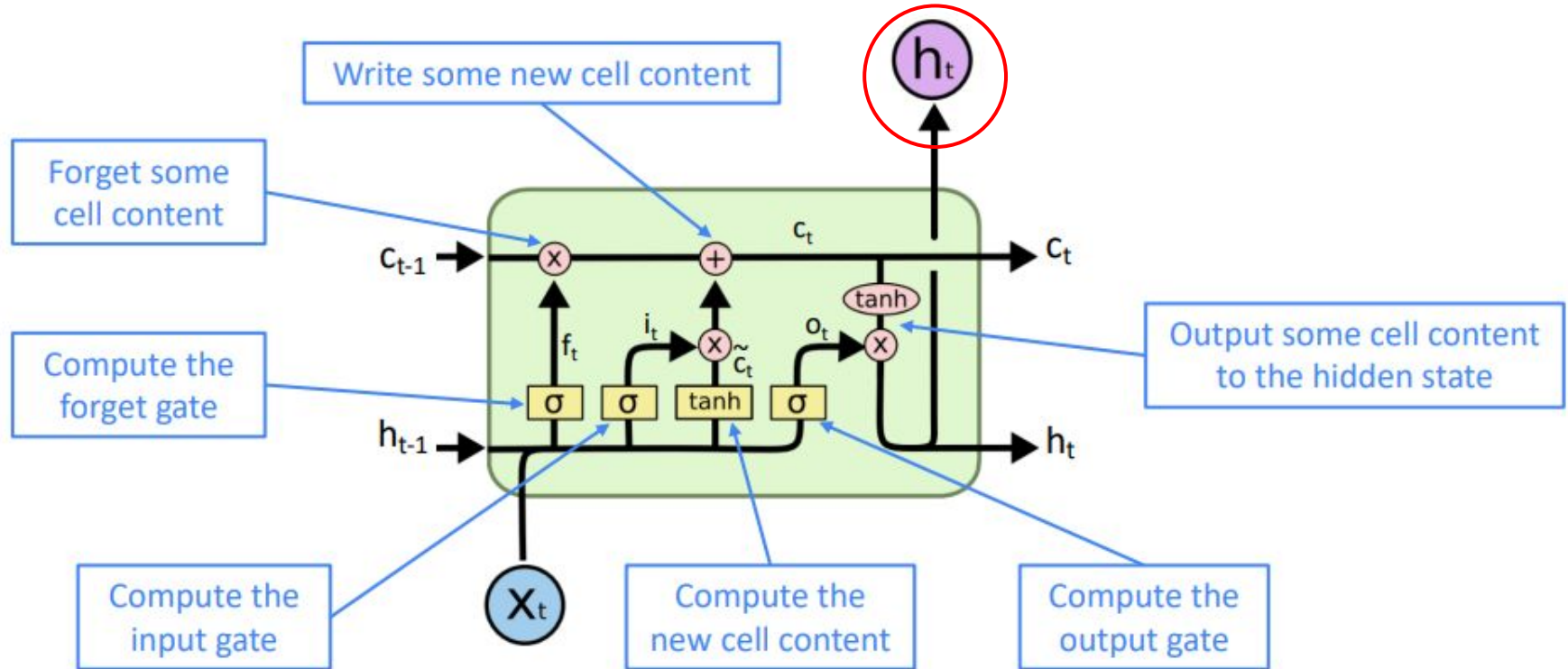
# Long Short-Term Memory (LSTMs)



# Long Short-Term Memory (LSTMs)



# Long Short-Term Memory (LSTMs)



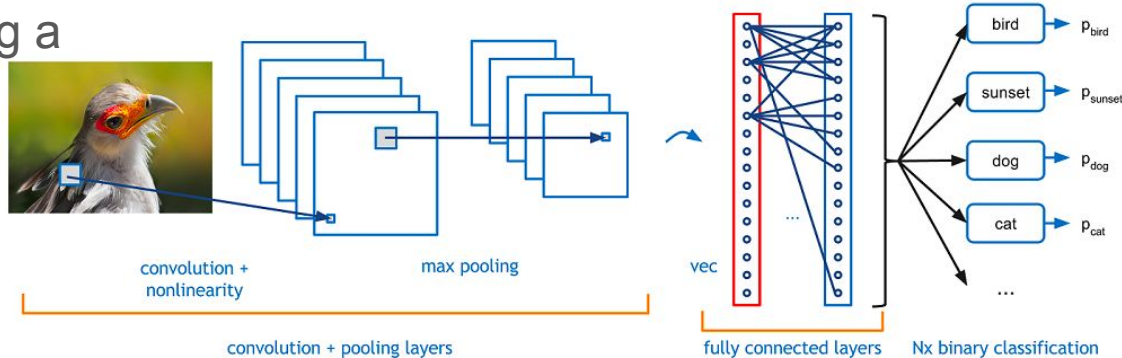
# Long Short-Term Memory (LSTMs)

- Because we have the forgetting gate, it's easier to keep things around for a long time, mostly solving our vanishing gradient problem
- Perform better on a large number of tasks
- Makes sense for language because of the time dependencies and long-range dependencies



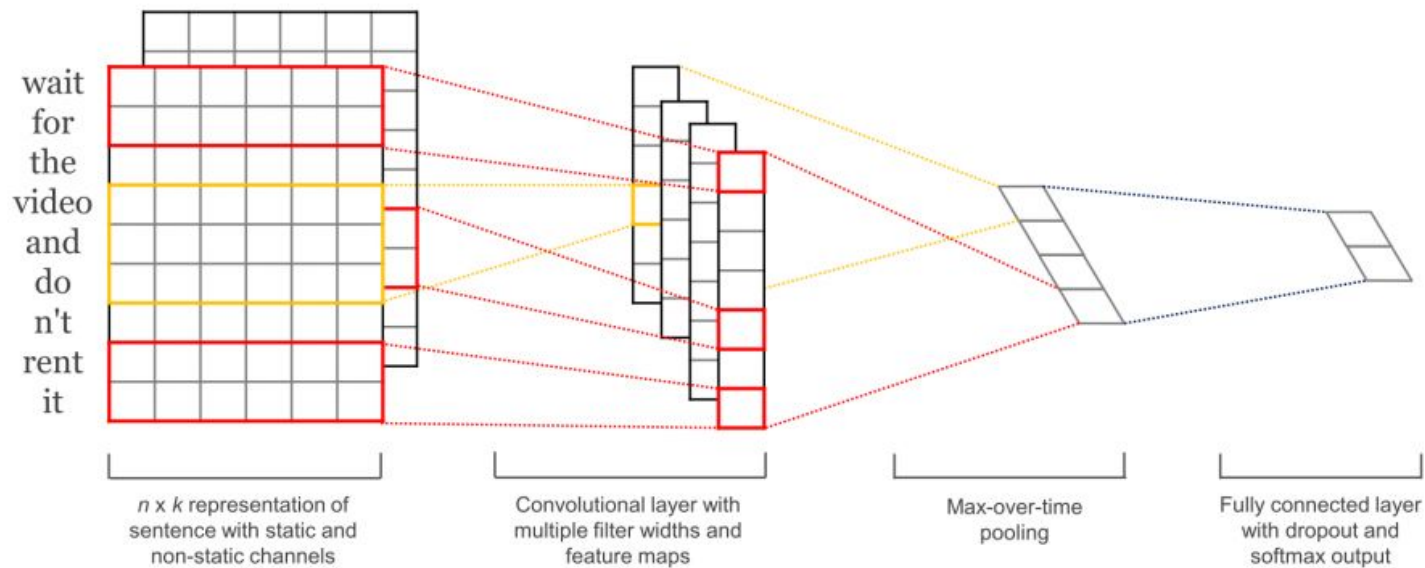
# CNN Review

- 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





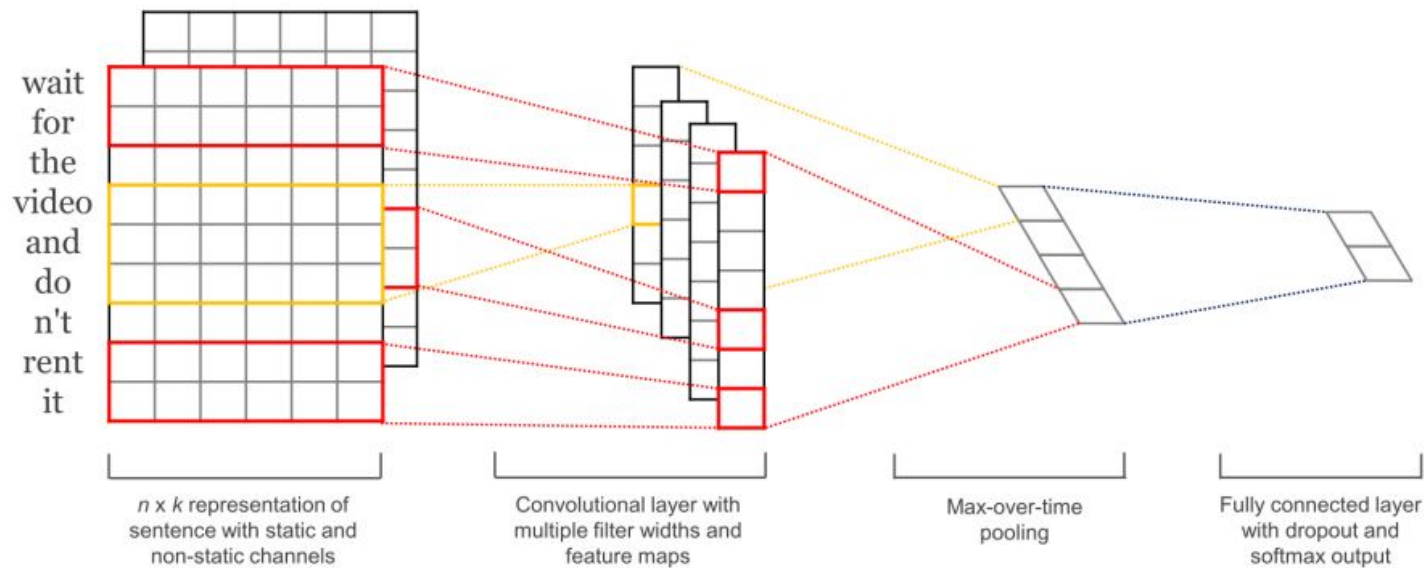
# CNNs for NLP



# Quick quiz

The filter mechanism in this CNN architecture is equivalent to what, from our first lecture?

# CNNs for NLP



# Next Time

- We'll talk more preprocessing and cut our teeth on some code!
- And we'll work on some more practical tips and tricks for any analytics professional dealing with text data