Neural models of language

CSC401/2511 – Natural Language Computing – Spring 2021
Lecture 5
University of Toronto

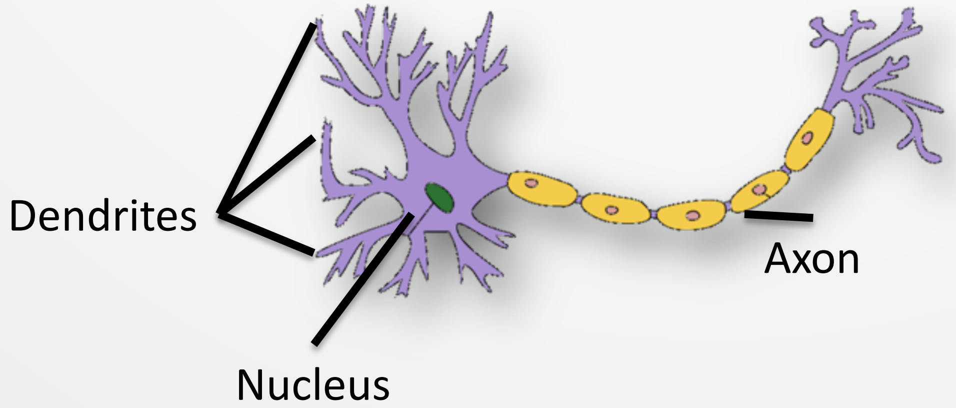
Neural networks

- Introduction
- Word-level representations
- Neural language models
- Recurrent neural networks
- Sequence-to-sequence modelling
- Some recent developments

With material from Phil Blunsom, Piotr Mirowski, Adam Kalai, and James Zou

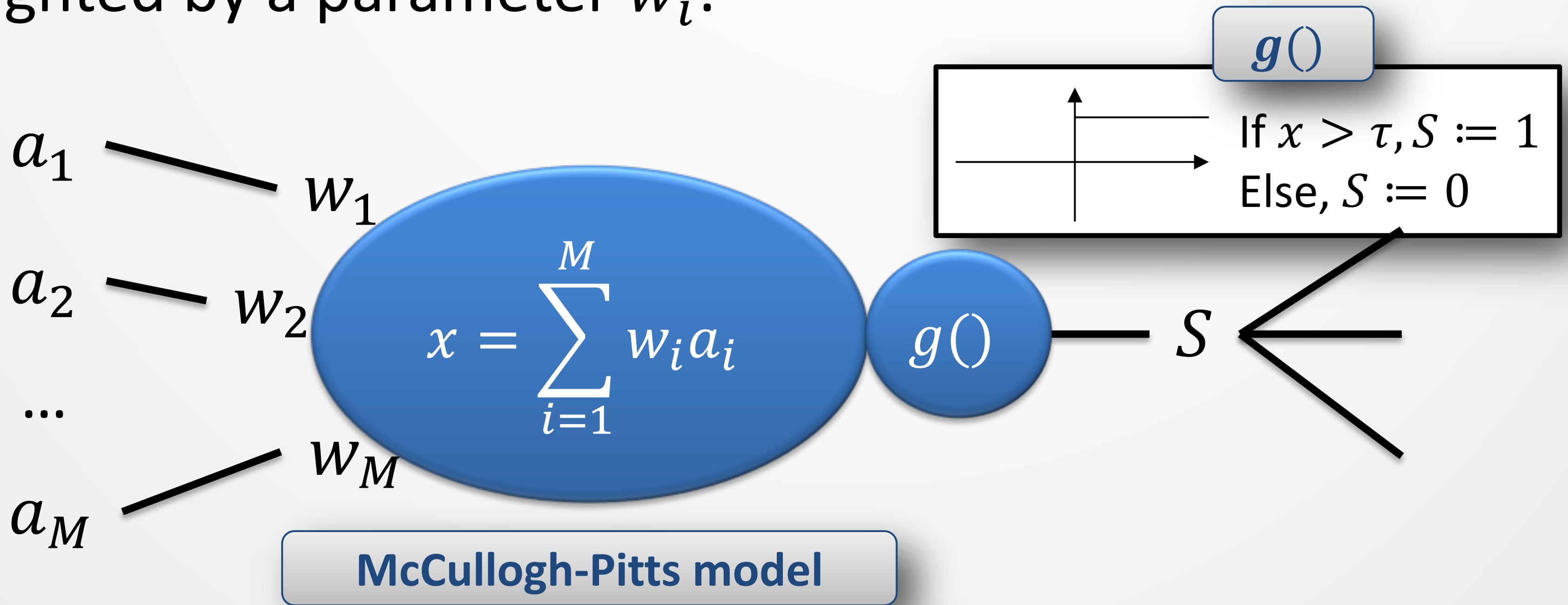
Artificial neural networks

- **Artificial neural networks (ANNs)** were (kind of) inspired from neurobiology (Widrow and Hoff, 1960).
  - Each unit has many inputs (**dendrites**), one output (**axon**).
  - The **nucleus** fires (sends an electric signal along the axon) given input from other neurons.
  - 'Learning' occurs at the **synapses** that connect neurons, either by amplifying or attenuating signals.



Perceptron: an artificial neuron

- Each neuron calculates a **weighted sum** of its inputs and compares this to a threshold, τ. If the sum exceeds the threshold, the neuron fires.
  - Inputs a_i are activations from adjacent neurons, each weighted by a parameter w_i.



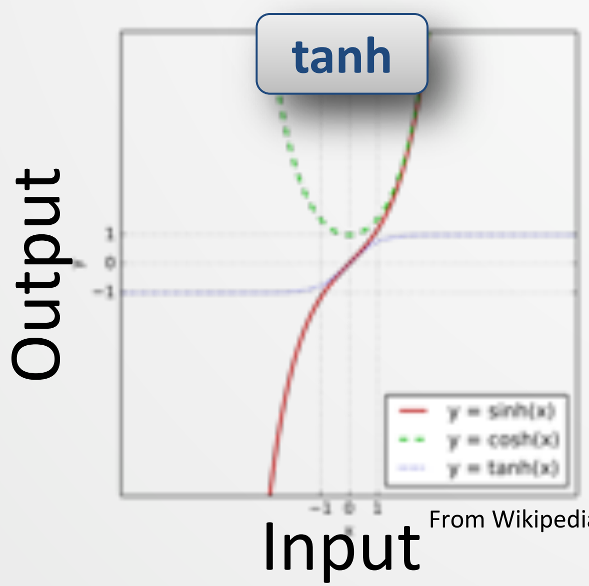
UNIVERSITY OF TORONTO

Perceptron output

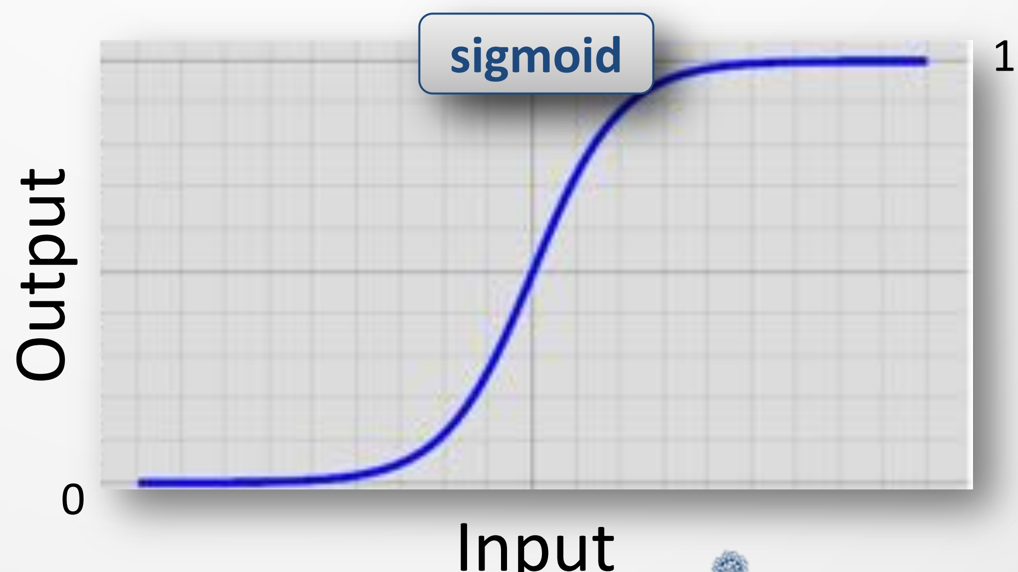
- Perceptron output is determined by **activation functions**, $g()$, which **can be non-linear functions** of weighted input.
- Popular activation functions include **tanh** and the **sigmoid**:

$$g(x) = \sigma(x) = \frac{1}{1 + e^{\rho x}}$$

  - The sigmoid's derivative is the easily computable $\sigma' = \sigma \cdot (1 - \sigma)$



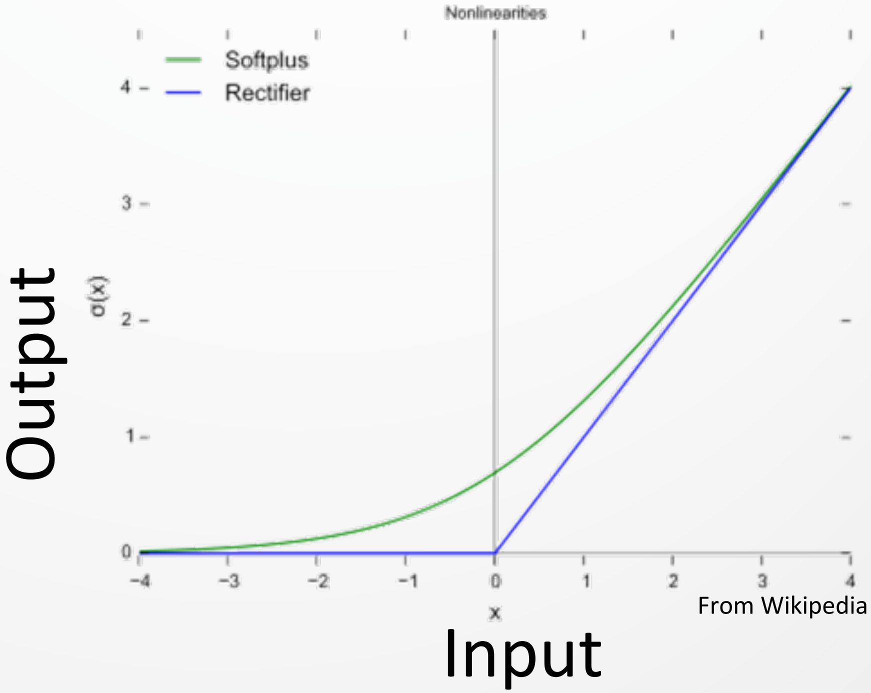
From Wikipedia



UNIVERSITY OF TORONTO

Rectified Linear Units (ReLUs)

- Since 2011, the **ReLU** $S = g(x) = \max(0, x)$ has become more popular.
  - More biologically plausible, sparse activation, limited (vanishing or exploding) gradient problems, efficient computation.
- A smooth approximation is the **softplus** $\log(1 + e^{x})$, which has a simple derivative $1/(1 + e^{-x})$
- *Why do we care about the derivatives?*



X Glorot, A Bordes, Y Bengio (2011). Deep sparse rectifier neural networks. AISTATS.

Perceptron learning

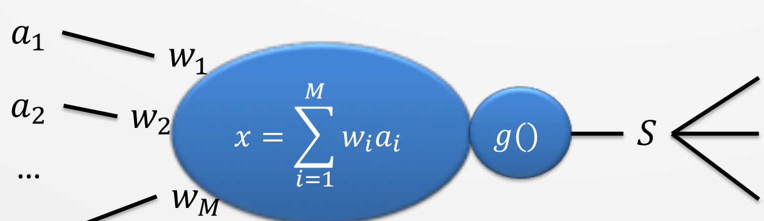
- Weights are adjusted in **proportion to the error** (i.e., the **difference** between the desired, y, and the actual output, S.
- The **derivative** g' allows us to assign blame proportionally.

- Given a small learning rate, α (e.g., 0.05), we can repeatedly adjust each of the weight parameters by

$$w_j := w_j + \alpha \cdot \sum_{i=1}^{R} Err_i \cdot g'(x_i) \cdot a_j[i]$$

Assumes mean-square error objective

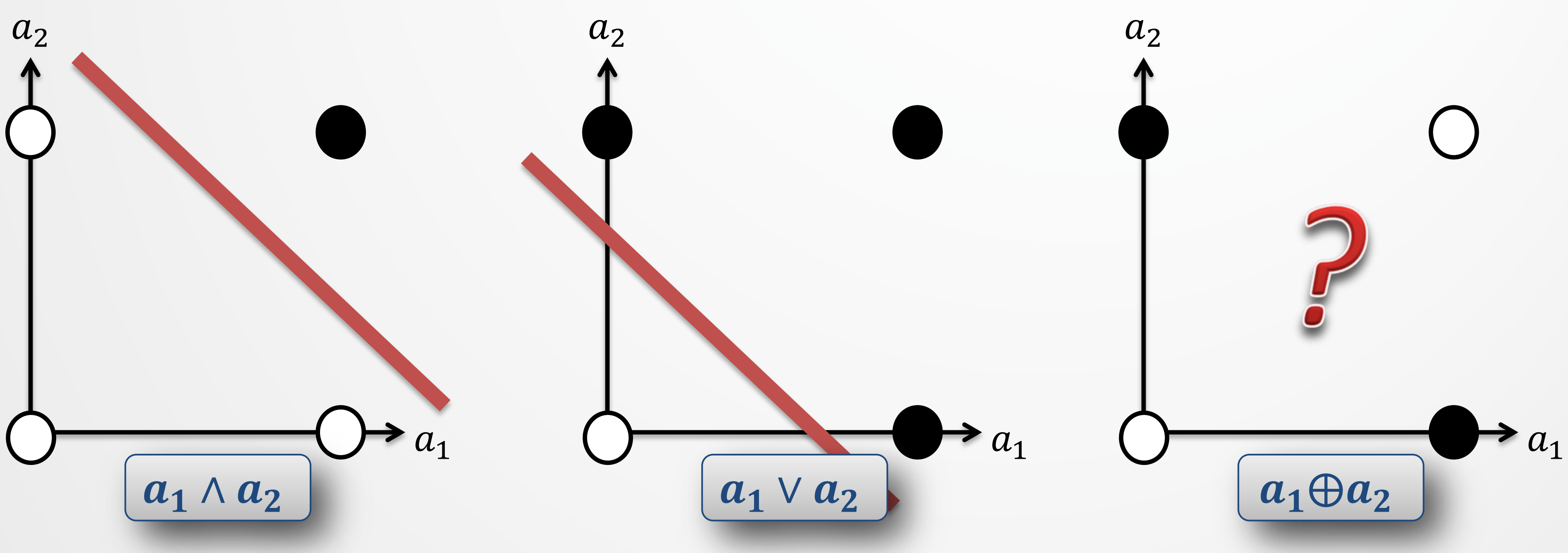
where $Err_i = (y_i - S_i)$, among R training examples.



McCullogh-Pitts model

Threshold perceptra and XOR

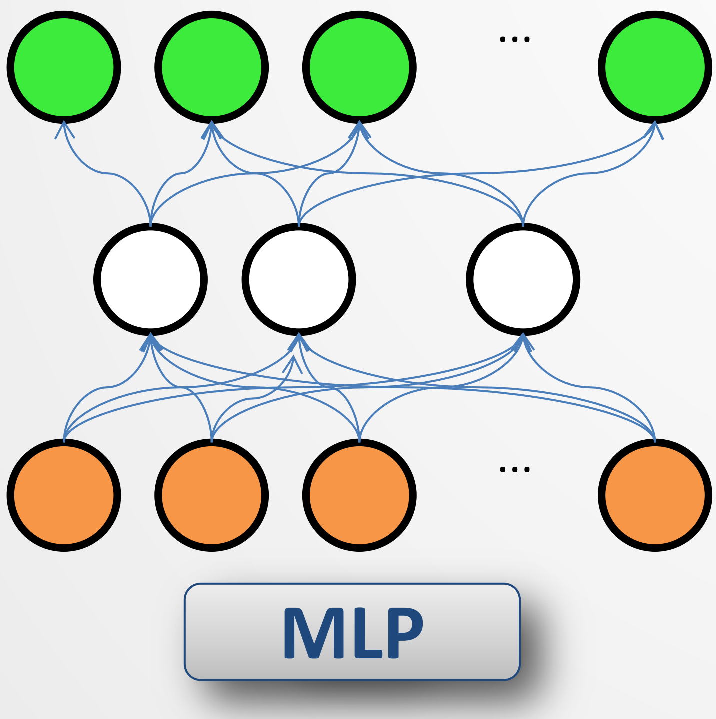
- Some relatively simple logical functions cannot be learned by threshold perceptra (since they are not linearly separable).



UNIVERSITY OF TORONTO

Artificial neural networks

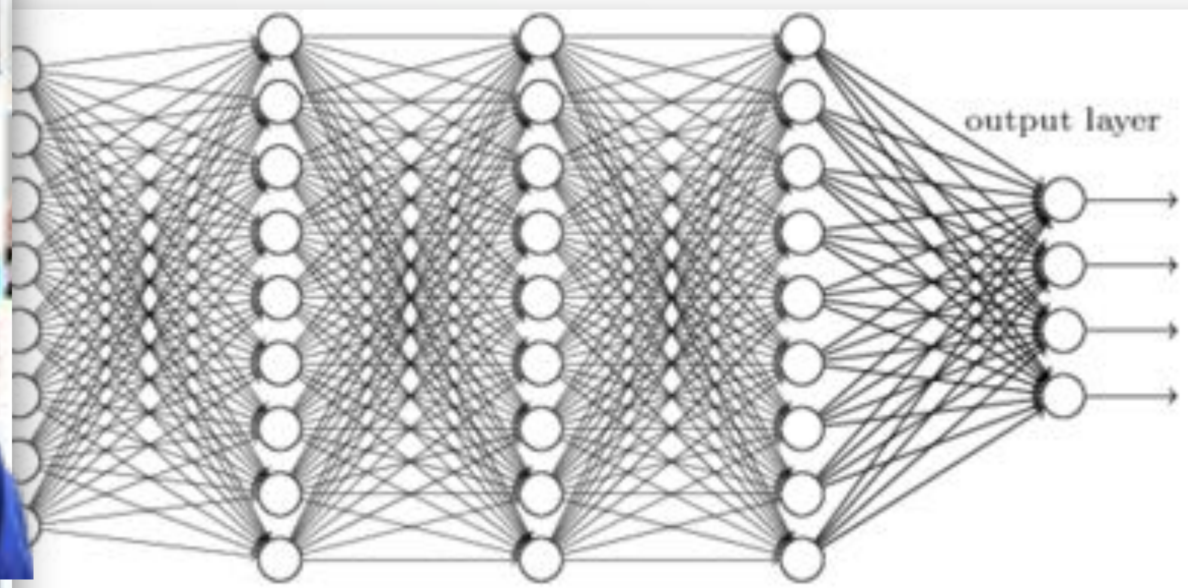
- Complex functions can be represented by layers of perceptra (**multi-layer perceptra, MLPs**).



- Inputs are passed to the **input layer.**
- **Activations** are propagated through **hidden layers** to the **output layer**.

- MLPs are quite **robust to noise**, and are trained specifically to reduce error.

Deep



It's a cat.

Depression.

'hidden' representations are learned here

Can we find hidden patterns in words?

Words

- Given a corpus with D (e.g., $= 100K$) unique words, the **classical approach** is to uniquely assign **each word** with an index in D-dimensional vectors ('one-hot' representation).

lugubrious	**0**	**0**	**0**	**0**	**..**	**0**	**1**	**0**	**...**	**0**

D

- Classic **word-feature representation** assigns **features** to each index in a much denser vector.
  - E.g., 'VBG', 'negative', 'age-of-acquisition'.

1	**0.8**	**2.5**	**0.81**	**...**	**99**

$d \ll D$

- Can we learn a dense representation? What will it give us?

UNIVERSITY OF TORONTO

Learning word semantics

"*You shall know a word by the company it keeps.*"

— J.R. Firth (1957)

$$P(w_t = lugubrious \mid w_{t-1} = feeling, w_{t-2} = been, \ldots)$$

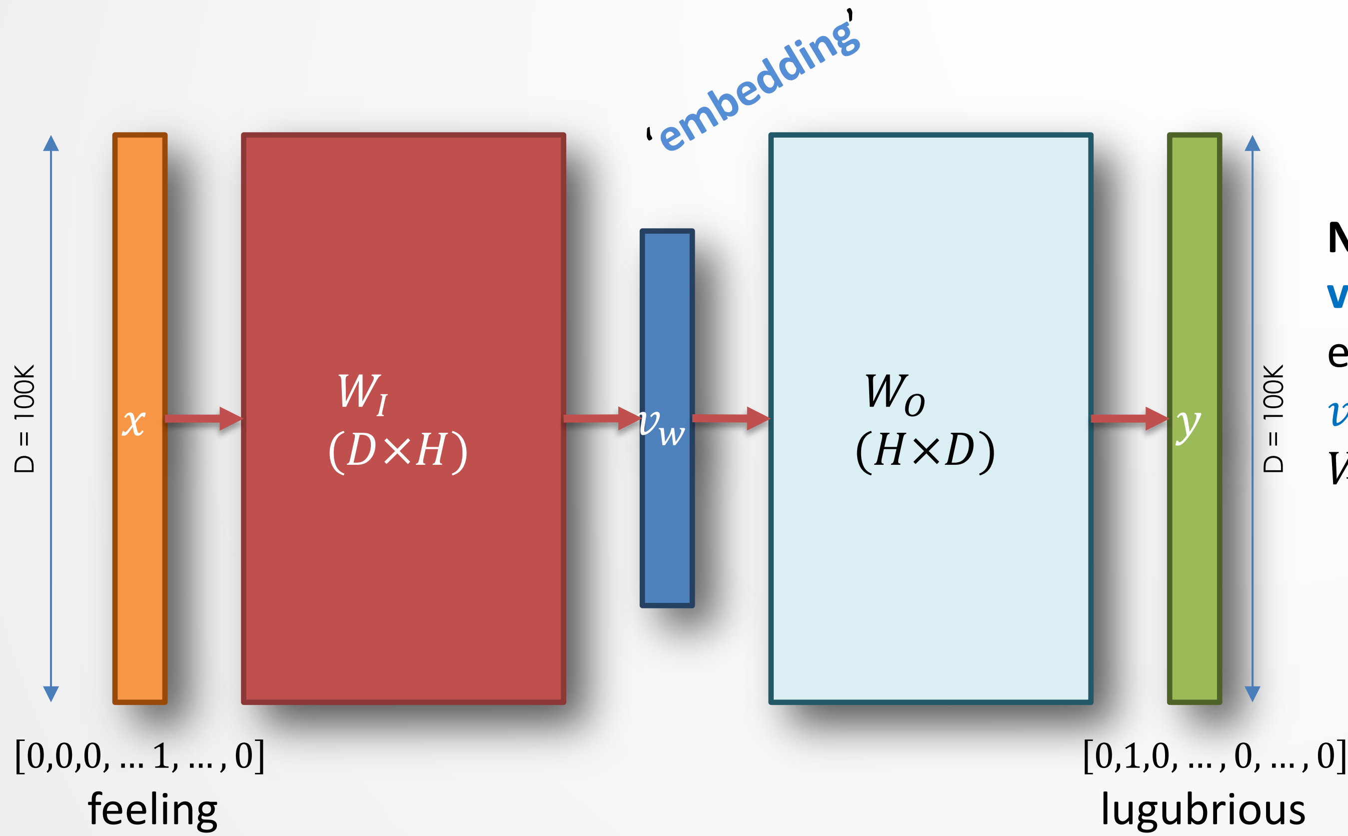
been	feeling	lugubrious	all	day
felt	a	lugubrious	sadness	in

…

Here, we're predicting the *center* word given the context.
This is called the '**continuous bag of words**' (**CBOW**) model.

https://code.google.com/p/word2vec/

Continuous bag of words (1 word context)



Note: we have *two* **vector representations** of each word:

$v_w = x^\intercal W_I$ (w^{th} row of W_I)

$V_w = W_O^\intercal y$ (w^{th} col of W_O)

feeling	lugubrious	all
a	lugubrious	sadness
	…	

'**softmax**': $P(w_o|w_i) = \dfrac{\exp(V_{w_o}^\intercal v_{w_i})}{\sum_{w=1}^{W} \exp(V_w^\intercal v_{w_i})}$

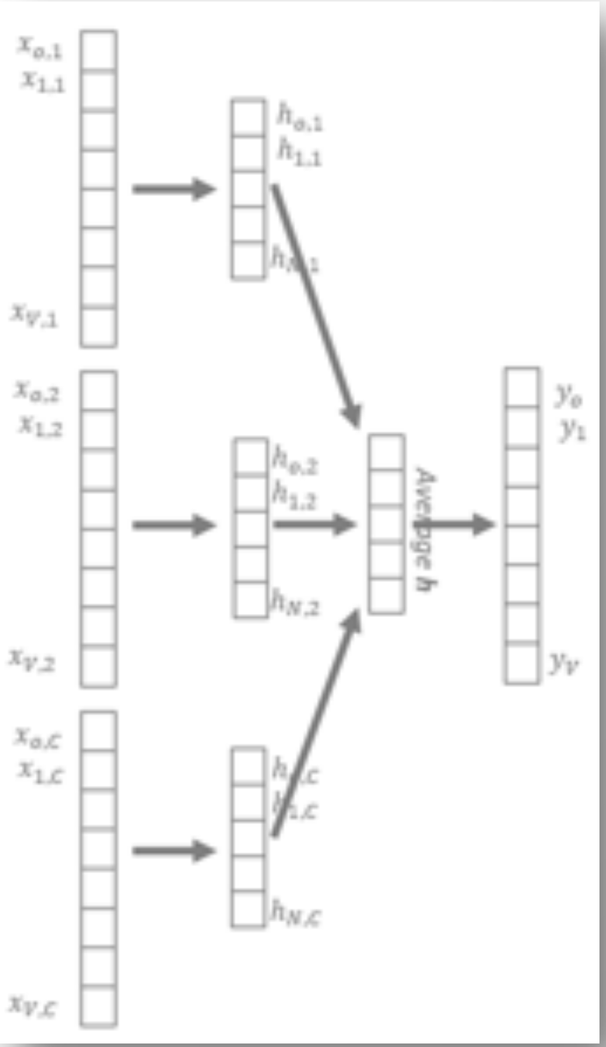
Where

v_w is the 'input' vector for word w,

V_w is the 'output' vector for word w,

Continuous bag of words (C words context)

- If we want to use **more context**, C, we need to change the network architecture somewhat.
  - Each input word will produce one of C embeddings
  - We just need to add an **intermediate layer**, usually this just averages the embeddings.



been	feeling	lugubrious	all
felt	a	lugubrious	sadness
		…	

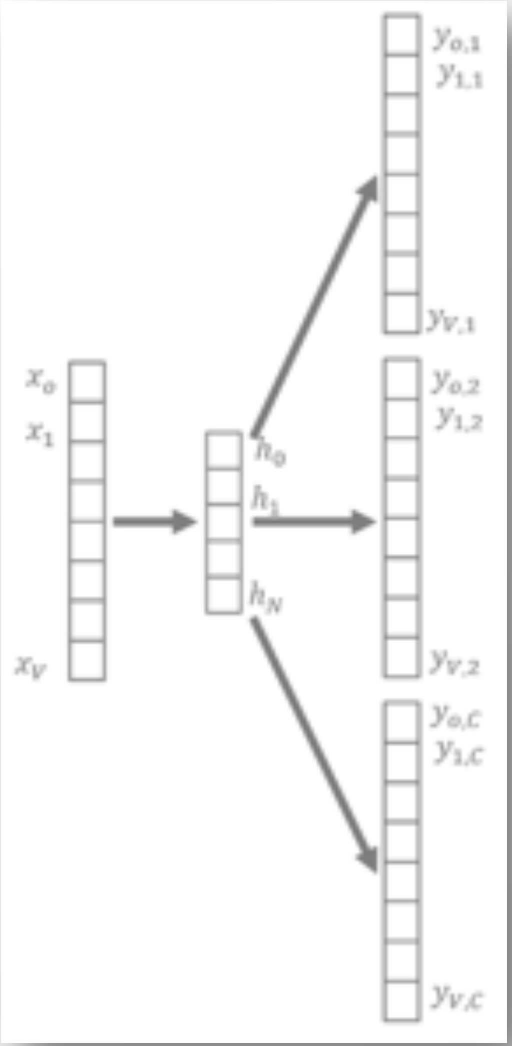
UNIVERSITY OF TORONTO

Skip-grams

- **Skip-grams** invert the task – we predict context words given the current word.

- According to Mikolov,
  Skip-gram: works well with small amounts of training data, represents rare words.

  CBOW: several times faster to train, slightly better accuracy for frequent words



Mikolov T, Corrado G, Chen K, *et al.* Efficient Estimation of Word Representations in Vector Space. *Proc (ICLR 2013)* 2013;:1–12. https://arxiv.org/pdf/1301.3781.pdf

UNIVERSITY OF TORONTO

Actually doing the learning

- Given H-dimensional embeddings, and V word types, our parameters, θ, are:

$$\theta = \begin{bmatrix} v_a \\ v_{aardvark} \\ \vdots \\ v_{zymurgy} \\ V_a \\ V_{aardvark} \\ \vdots \\ V_{zymurgy} \end{bmatrix} \in \mathbb{R}^{2V \times H}$$

Actually doing the learning

We have many options. Gradient descent is popular.

We want to optimize, given T tokens of training data,

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c<j<c,j\neq 0}\log P(w_{t+j}|w_t)$$

And we want to update vectors $V_{w_{t+j}}$ then v_{w_t} within θ

$$\theta^{(new)} = \theta^{(old)} - \alpha\nabla_\theta J(\theta)$$

so we'll need to take the derivative of the (log of the) softmax function:

$$P(w_o|w_i) = \frac{\exp(V_{w_o}^\intercal v_{w_i})}{\sum_{w=1}^{W}\exp(V_w^\intercal v_{w_i})}$$

Where v_w is the 'input' vector for word w,
and V_w is the 'output' vector for word w,

UNIVERSITY OF TORONTO

Actually doing the learning

We need the derivative of the (log of the) softmax function:

$$\frac{\delta}{\delta v_{w_t}} \log P(w_{t+j}|w_t) = \frac{\delta}{\delta v_{w_t}} \log \frac{\exp(V_{w_{t+j}}^\intercal v_{w_t})}{\sum_{w=1}^{W} \exp(V_w^\intercal v_{w_t})}$$

$$= \frac{\delta}{\delta v_{w_t}} \left[\log \exp\left(V_{w_{t+j}}^\intercal v_{w_t}\right) - \log \sum_{w=1}^{W} \exp(V_w^\intercal v_{w_t})\right]$$

$$= V_{w_{t+j}} \qquad - \frac{\delta}{\delta v_{w_t}} \log \sum_{w=1}^{W} \exp(V_w^\intercal v_{w_t})$$

[apply the chain rule $\frac{\delta f}{\delta v_{w_t}} = \frac{\delta f}{\delta z} \frac{\delta z}{\delta v_{w_t}}$]

$$= V_{w_{t+j}} - \sum_{w=1}^{W} p(w|w_t) V_w$$

More details: http://arxiv.org/pdf/1411.2738.pdf

UNIVERSITY OF TORONTO

Using word representations

Without a latent space,

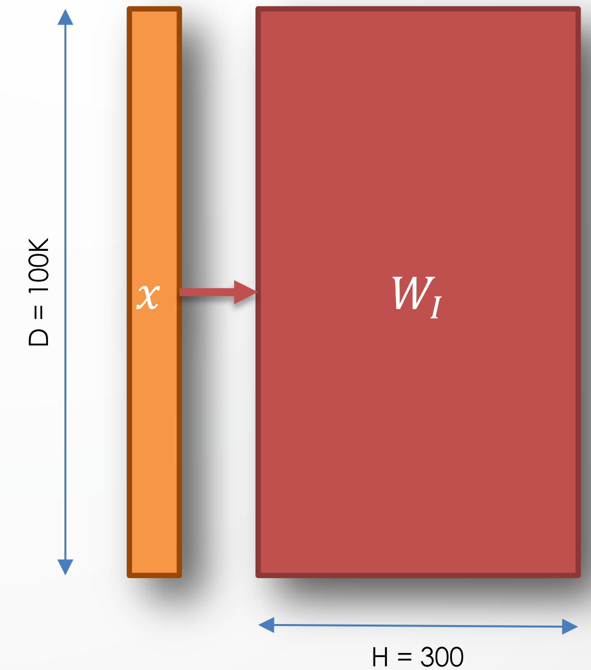
lugubrious = $[0,0,0,\ldots,0,1,0,\ldots,0]$, &

sad = $[0,0,0,\ldots,0,0,1,\ldots,0]$ so

Similarity = $\cos(x,y) = 0.0$

EMBEDDING

$$v_w = x^\intercal W_I$$



In latent space,

lugubrious = $[0.8,0.69,0.4,\ldots,0.05]_H$, &

sad = $[0.9,0.7,0.43,\ldots,0.05]_H$ so

Similarity = $\cos(x,y) = 0.9$

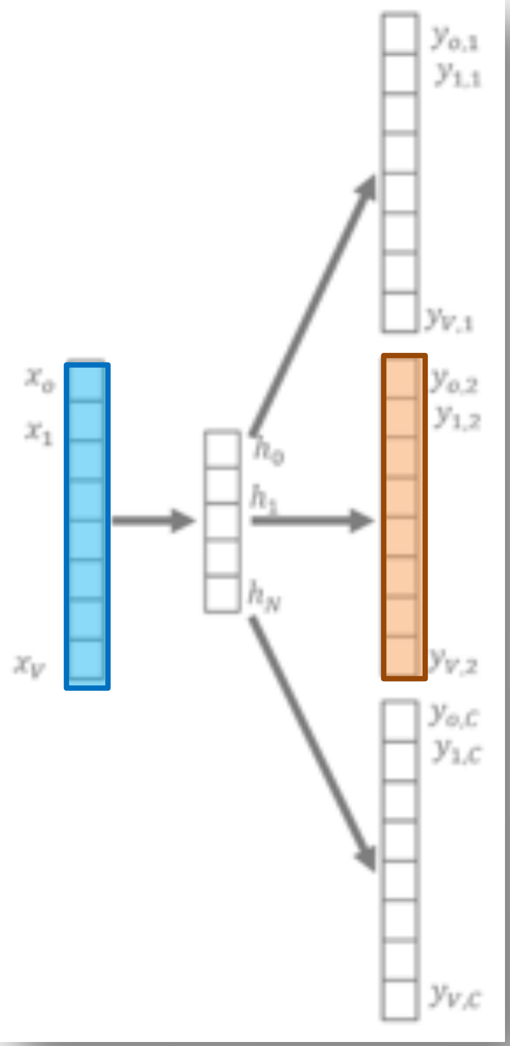
Reminder:

$$\cos(u,v) = \frac{u \cdot v}{||u|| \times ||v||}$$

Skip-grams with negative sampling

- The default process is inefficient.
  - For one – **what a waste of time**!
    We don't want to update $H \times D$ weights!
  - For two – **we want to avoid confusion**!
    'Hallucinated' contexts should be minimized.

- For the observed pair (*lugubrious*, *sadness*), only the output neuron for *sadness* should be 1, and all $D-1$ others should be 0.



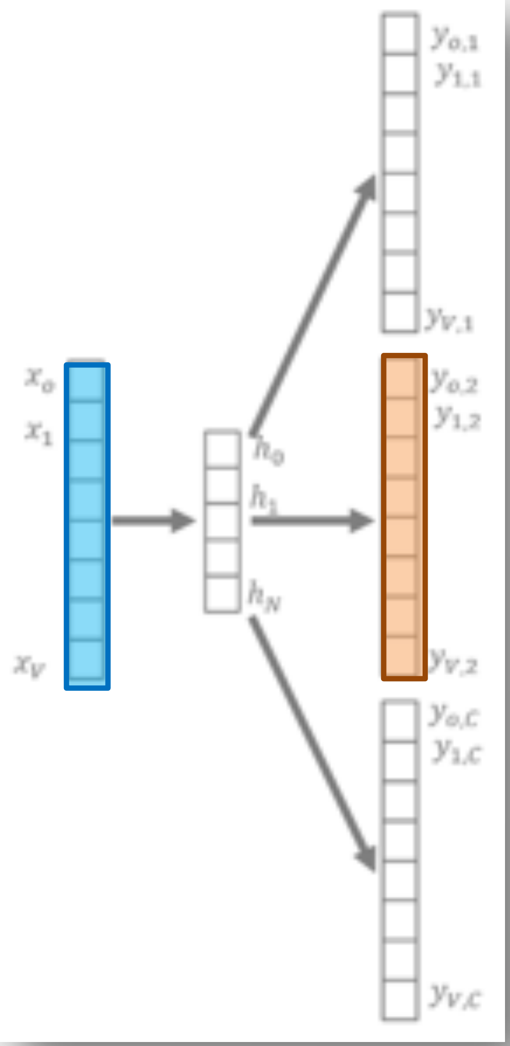
Skip-grams with negative sampling

- We want to **maximize** the association of ***observed*** (positive) contexts:

  lugubrious *sad*
  lugubrious *feeling*
  lugubrious *tired*

- We want to **minimize** the association of ‘***hallucinated***’ (negative) contexts:

  lugubrious *happy*
  lugubrious *roof*
  lugubrious *truth*



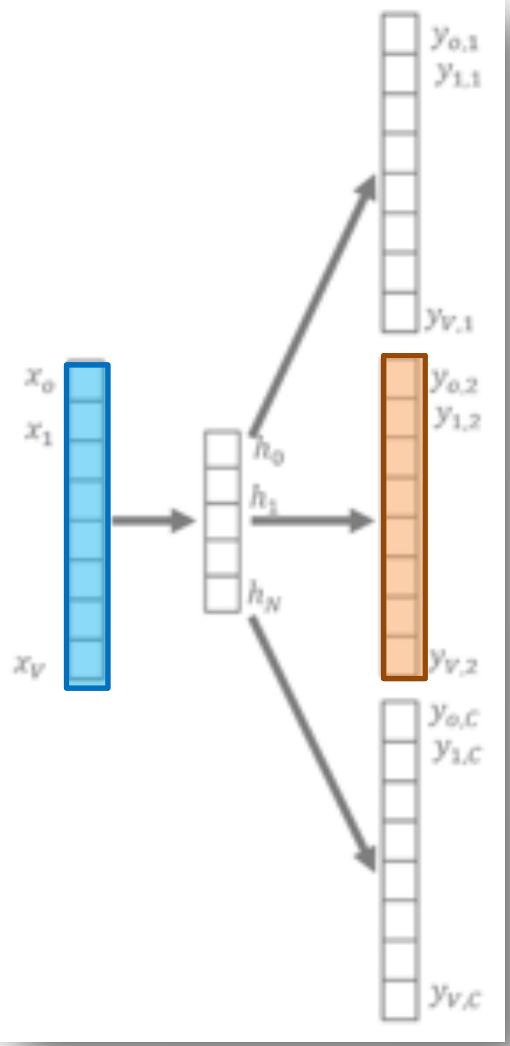
UNIVERSITY OF TORONTO

Skip-grams with negative sampling

- Choose a small number η of 'negative' words, and just update the weights for the 'positive' word plus the η 'negative' words.
  - $5 \leq \eta \leq 20$ can work in practice for fewer data.
  - For $D = 100K$, we only update 0.006% of the weights in the output layer.

- Mimno and Thompson (2017) choose the top η words by modified unigram probability:

$$P^*(w_{t+1}) = \frac{C(w_{t+1})^{\frac{3}{4}}}{\sum_w C(w)^{\frac{3}{4}}}$$



Mimno, D., & Thompson, L. (2017). The strange geometry of skip-gram with negative sampling. *EMNLP 2017* , 2873–2878. https://doi.org/10.18653/v1/d17-1308

UNIVERSITY OF TORONTO

Smell the GloVe

- **GloVe** ('**Glo**bal **Ve**ctors') is an alternative method of obtaining word embeddings.
  - Instead of predicting words at particular positions, look at the **co-occurrence matrix.**

$$X = \begin{array}{cc} & \begin{array}{cccccccc} \textit{I} & \textit{like} & \textit{enjoy} & \textit{deep} & \textit{learning} & \textit{NLP} & \textit{flying} & . \end{array} \\ \begin{array}{c} \textit{I} \\ \textit{like} \\ \textit{enjoy} \\ \textit{deep} \\ \textit{learning} \\ \textit{NLP} \\ \textit{flying} \\ . \end{array} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Word w_i occurs
$X_{i,j} (= X_{j,i})$
times with word w_j,
within some context
window (e.g., 10 words,
a sentence, …).

Pennington J, Socher R, Manning CD. (2014) GloVe: Global Vectors for Word Representation. *Proc EMNLP 2014*:1532–43. doi:10.3115/v1/D14-1162 https://nlp.stanford.edu/projects/glove/

Smell the GloVe

- Populating the co-occurrence matrix requires a complete pass through the corpus, but needs only be done once.
- Let $P_{i,j} = P(w_j | w_i) = X_{i,j} / X_i$,

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Pennington J, Socher R, Manning CD. (2014) GloVe: Global Vectors for Word Representation. *Proc EMNLP 2014*:1532–43. doi:10.3115/v1/D14-1162 https://nlp.stanford.edu/projects/glove/

Aside – smell the GloVe

- Minimize $J = \sum_{i,j=1}^{V} f\left(X_{i,j}\right)\left(v_{w_i} V_{w_j} + b_i + \widetilde{b_j} - \log X_{i,j}\right)^2$

  where b_i and $\widetilde{b_j}$ are input and output bias terms associated with w_i and w_j, respectively

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log^2 x$ is finite.

2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.

3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

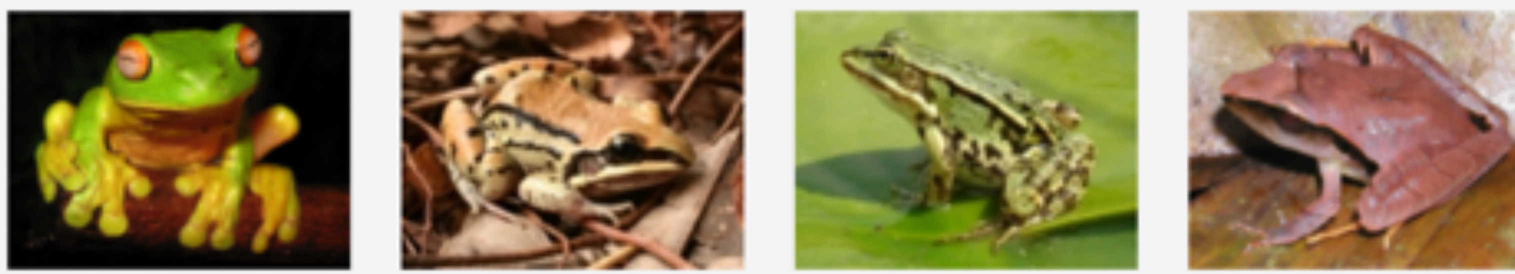
Of course a large number of functions satisfy these properties, but one class of functions that we found to work well can be parameterized as,

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

Aside – smell the GloVe

- **Intrinsic evaluation**: popular Redacted method is to cherry-pick a few *k*-nearest neighbours examples that match expectations.

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

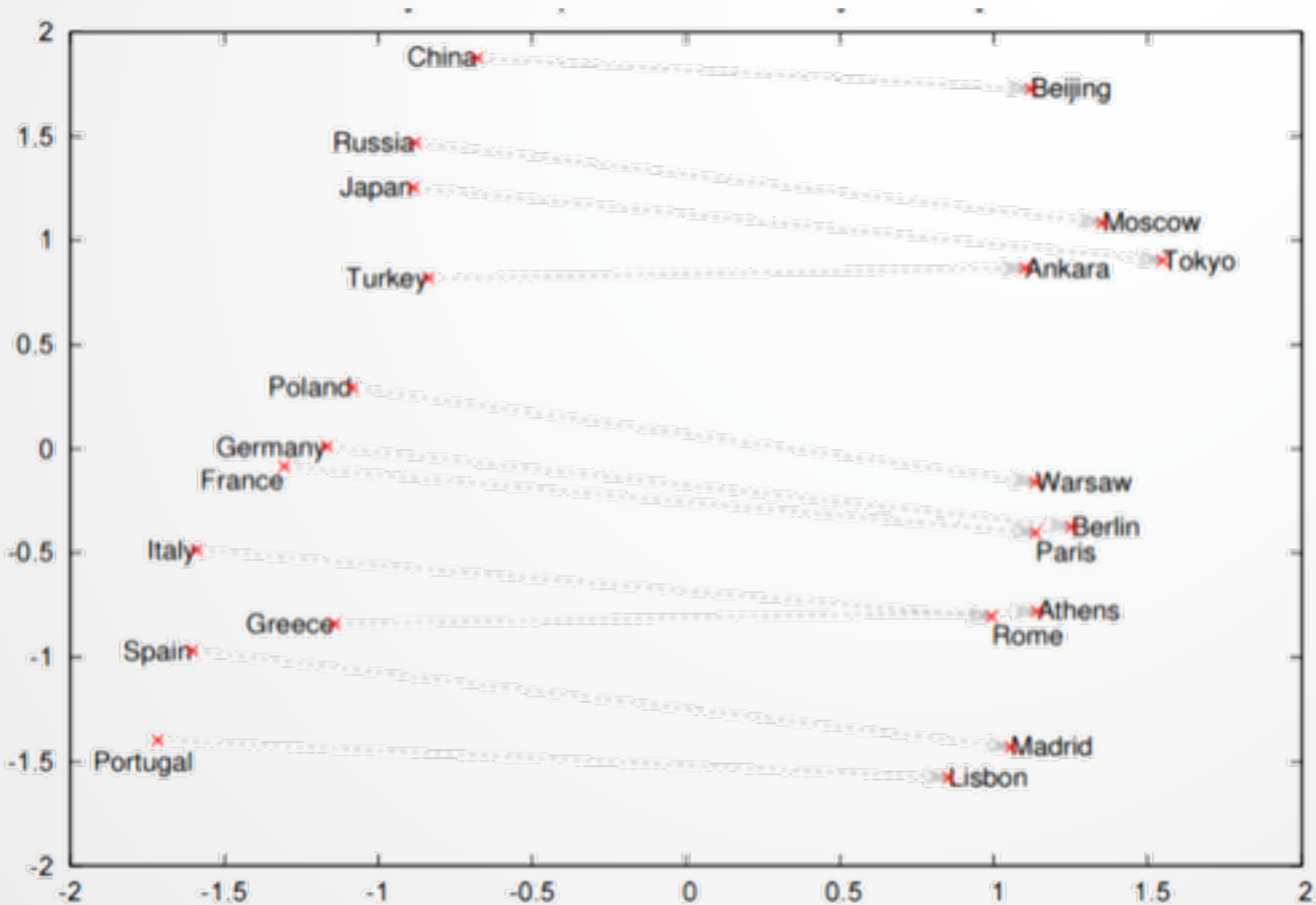


3. litoria   4. leptodactylidae   5. rana   7. eleutherodactylus

- **Extrinsic evaluation**: embed resulting vectors into a variety of tasks.

Redacted. See https://github.com/sebastianruder/NLP-progress

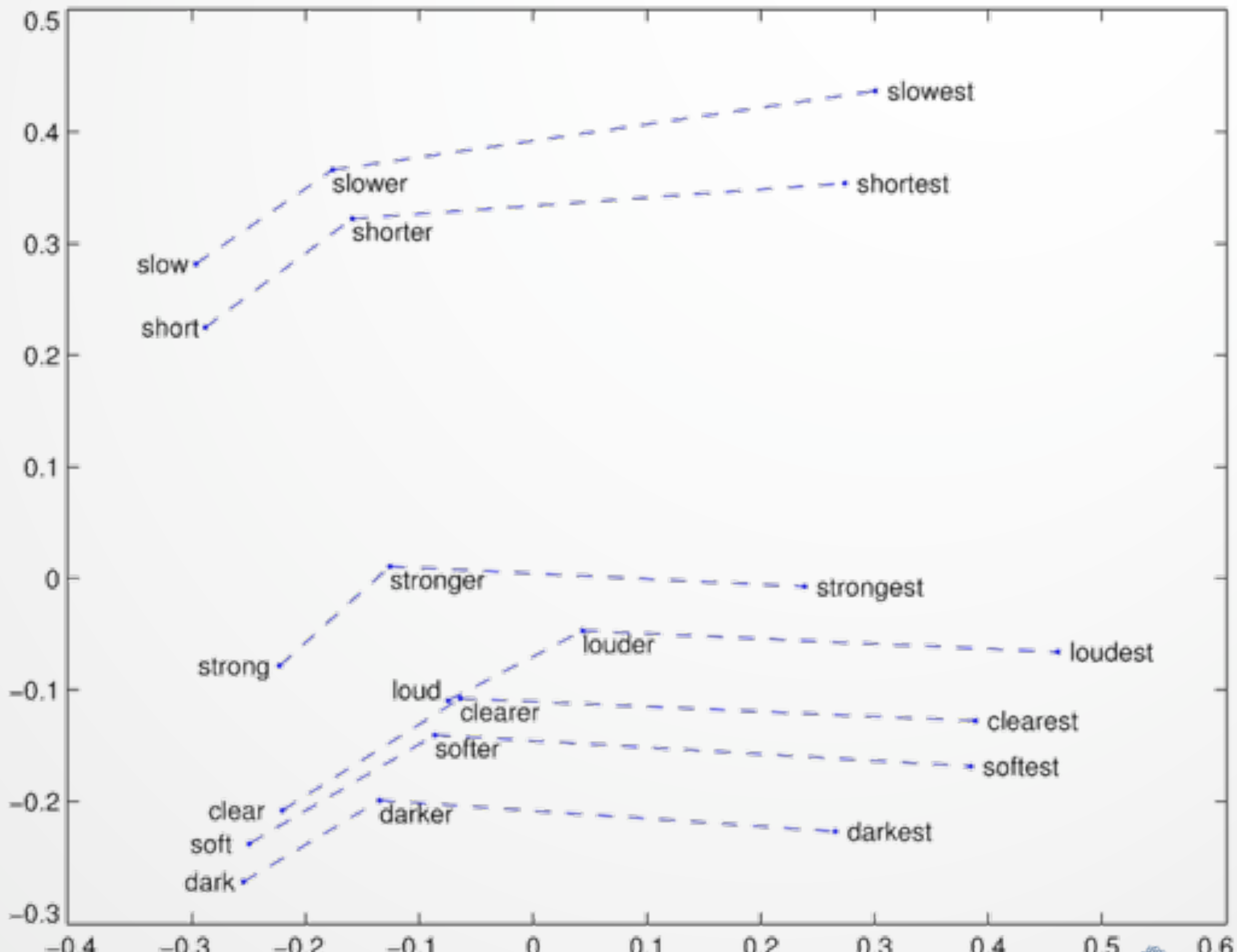
Linguistic regularities in vector space



Trained on the Google news corpus with over 300 billion words.

UNIVERSITY OF TORONTO

Linguistic regularities in vector space



(from GloVe)

Linguistic regularities in vector space

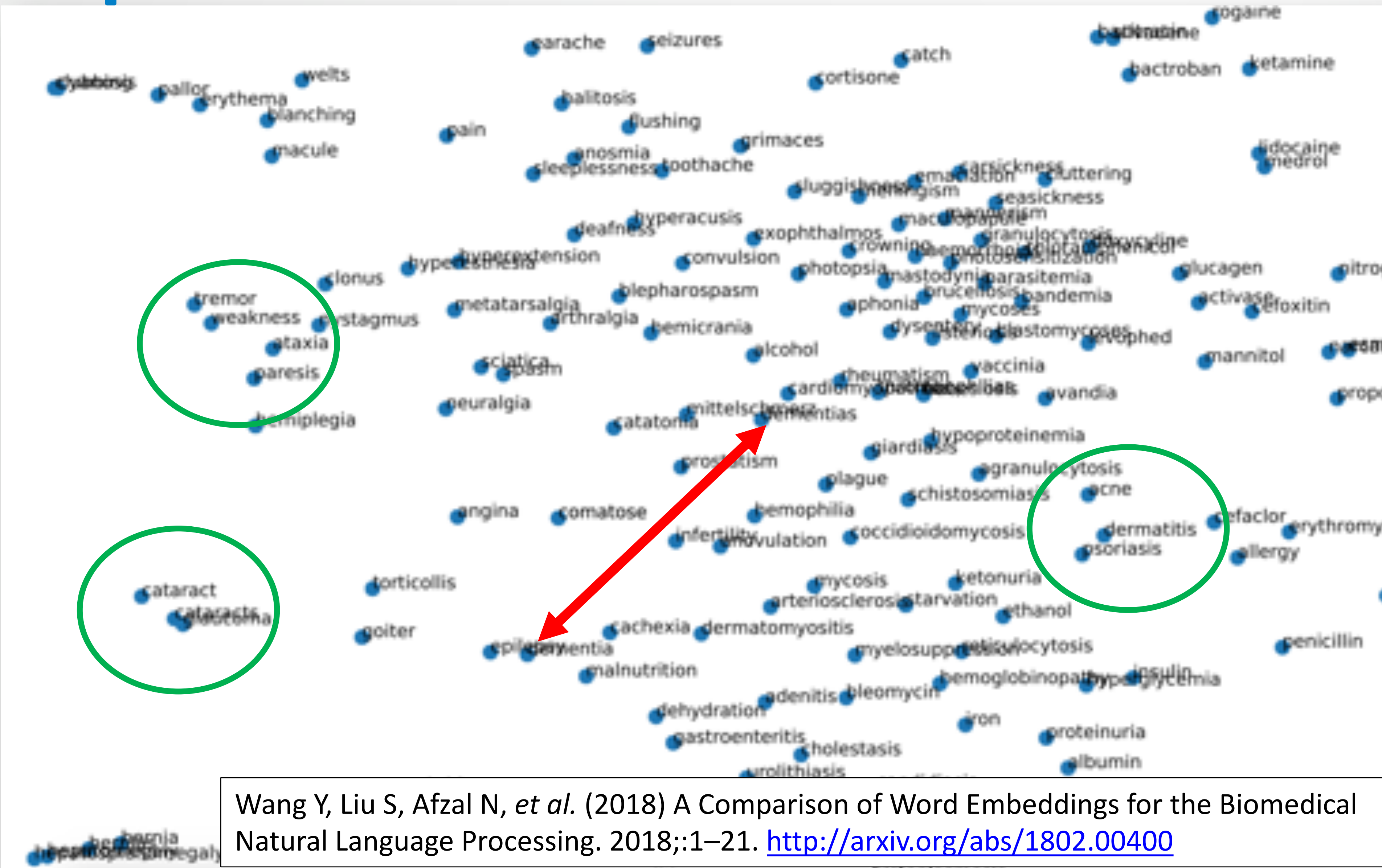
Expression	Nearest token
Paris – France + Italy	Rome
Bigger – big + cold	Colder
Sushi – Japan + Germany	bratwurst
Cu – copper + gold	Au
Windows – Microsoft + Google	Android

Analogies: apple:apples :: octopus:octopodes

Hypernymy: shirt:clothing :: chair:furniture

UNIVERSITY OF TORONTO

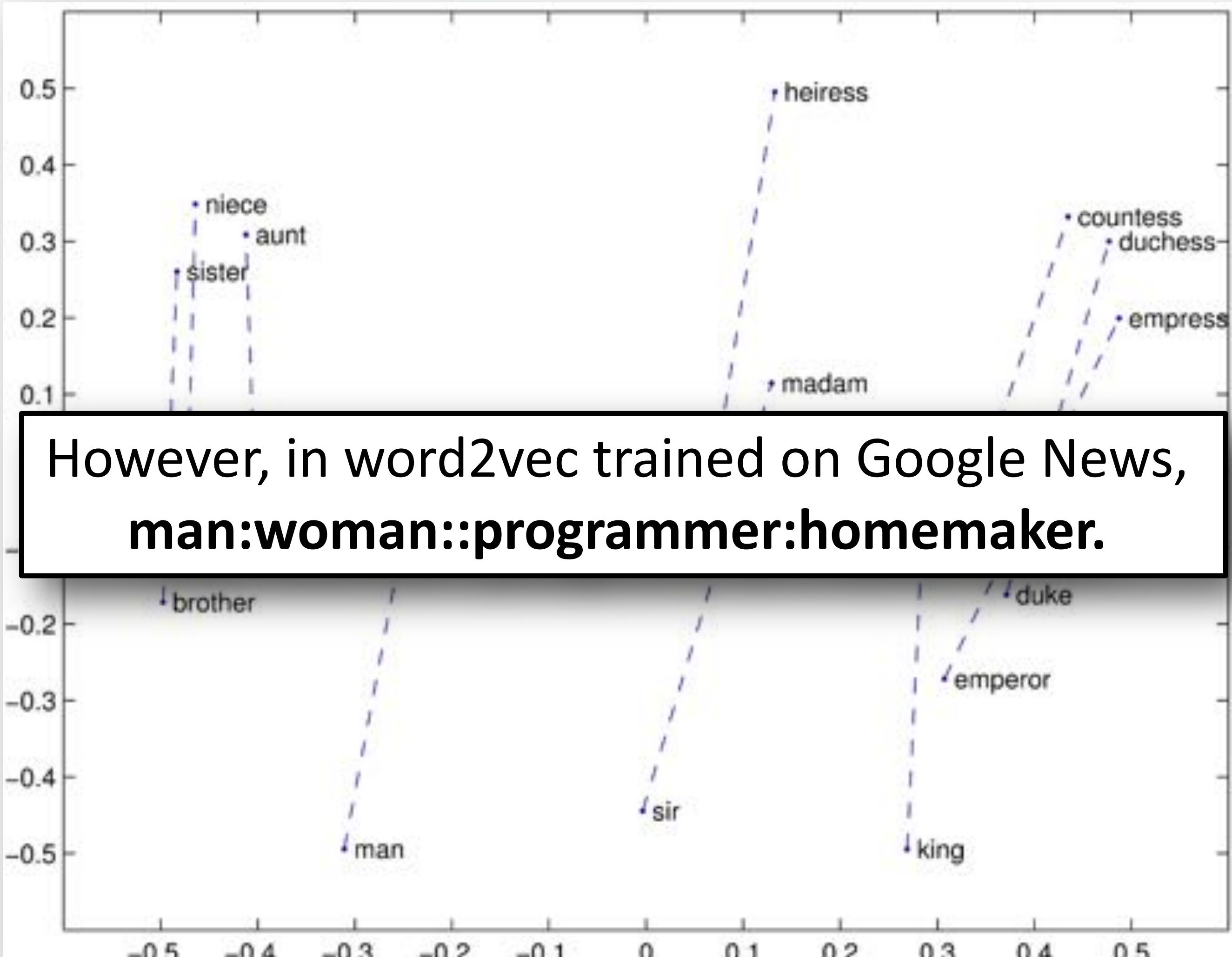
Importance of in-domain data



Wang Y, Liu S, Afzal N, *et al.* (2018) A Comparison of Word Embeddings for the Biomedical Natural Language Processing. 2018;:1–21. http://arxiv.org/abs/1802.00400

UNIVERSITY OF TORONTO

Let's talk about gender at the UofT



However, in word2vec trained on Google News,
man:woman::programmer:homemaker.

Bolukbasi T, Chang K, Zou J, *et al.* Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In: *NIPS*. 2016. 1–9.

UNIVERSITY OF TORONTO

Let’s talk about gender at the UofT

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], Kai-Wei Chang[2], James Zou[2], Venkatesh Saligrama[1,2], Adam Kalai[2]
[1]Boston University, 8 Saint Mary’s Street, Boston, MA
[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and

Extreme *she*	Extreme *he*
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Gender stereotype *she-he* analogies

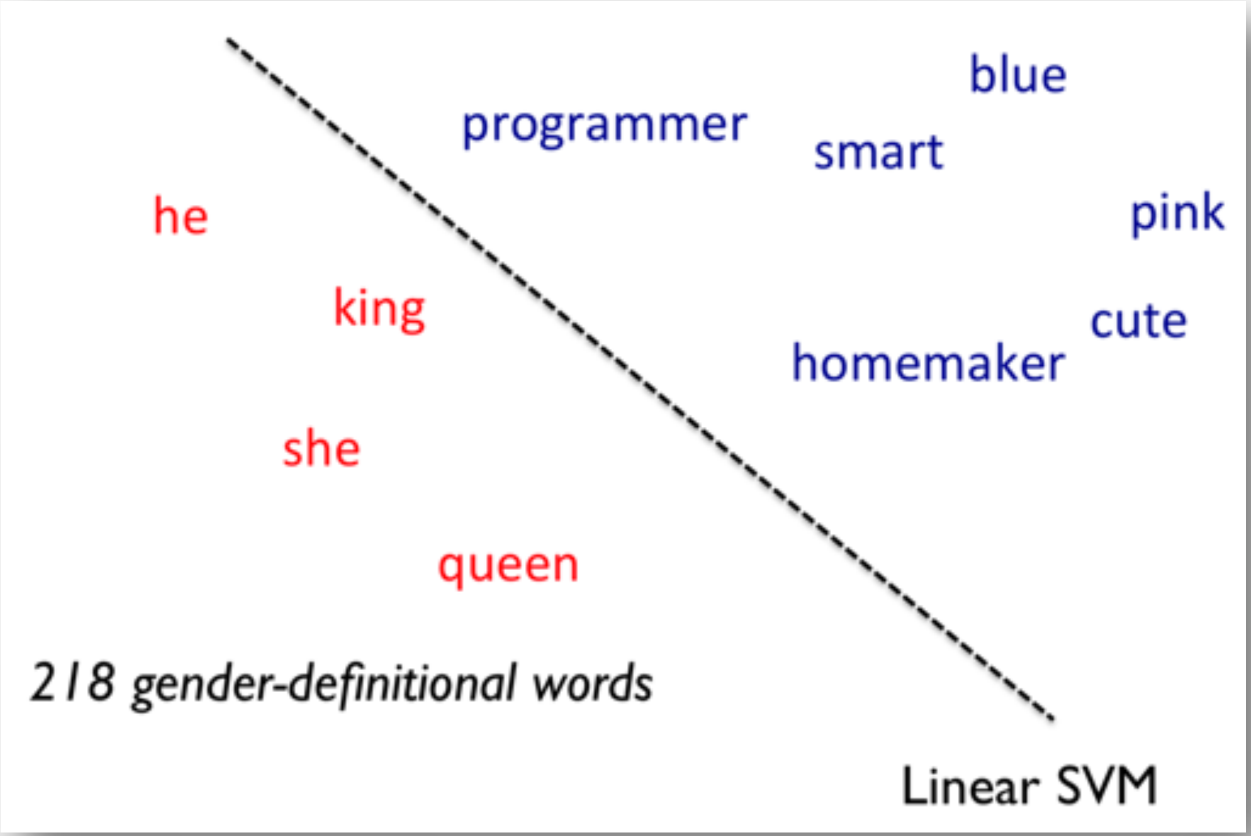
sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	lovely-brilliant

Gender appropriate *she-he* analogies

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

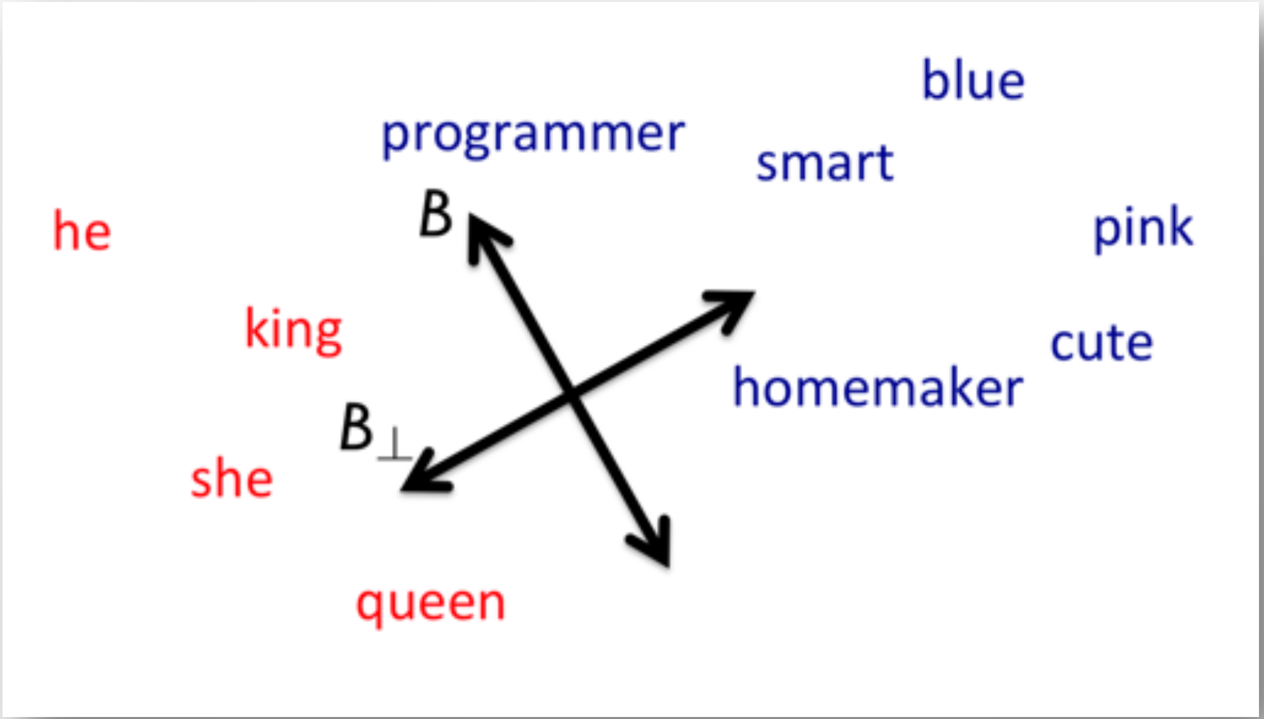
Solution?

1. Hand-pick words S_0 that are ’gender definitional’.
   ‘Neutral’ words are the complement, $N = V \setminus S_0$.



Solution?

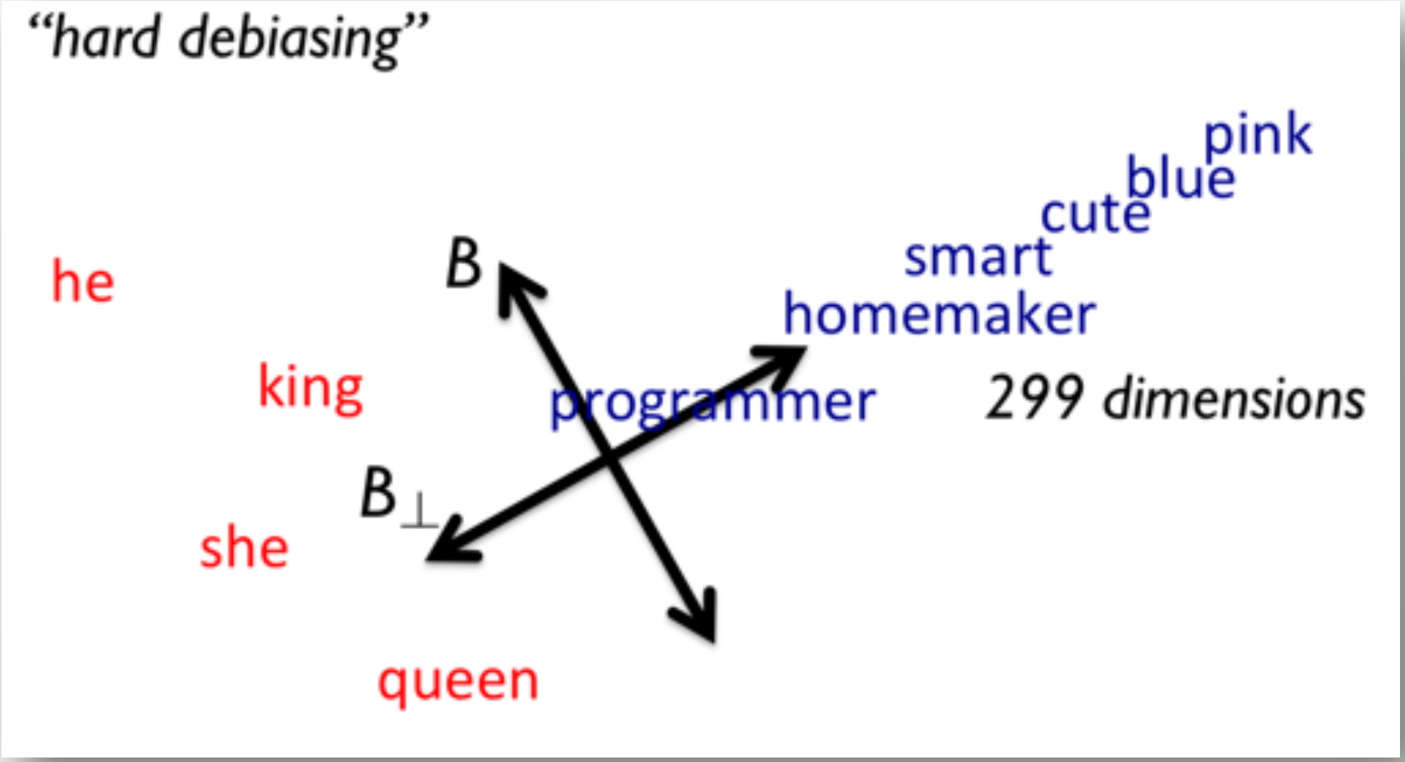
2. Project away gender subspace from gender-neutral words, $w \coloneqq w - w \cdot B$ for $w \in N$, where B is the gender subspace.



UNIVERSITY OF TORONTO

Solution?

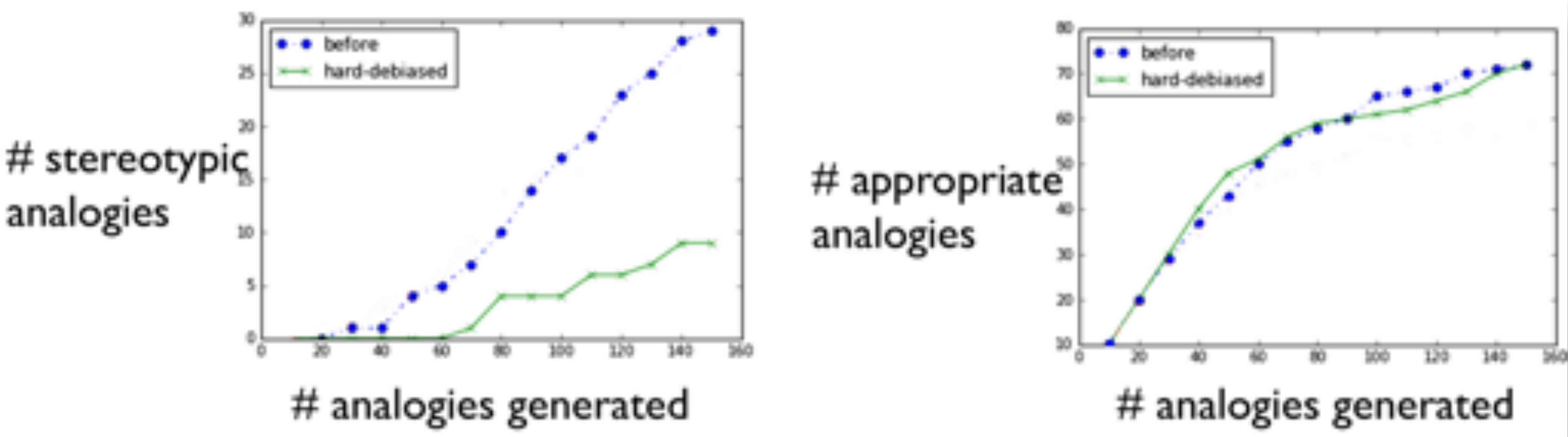
2. Project away gender subspace from gender-neutral words, $w := w - w \cdot B$ for $w \in N$, where B is the gender subspace.



Results

- Generate many analogies, see which ones preserve gender stereotypes.

He:*Blue* :: She: *?*
He:*Doctor* :: She: *?*
He:Brother :: She: *?*

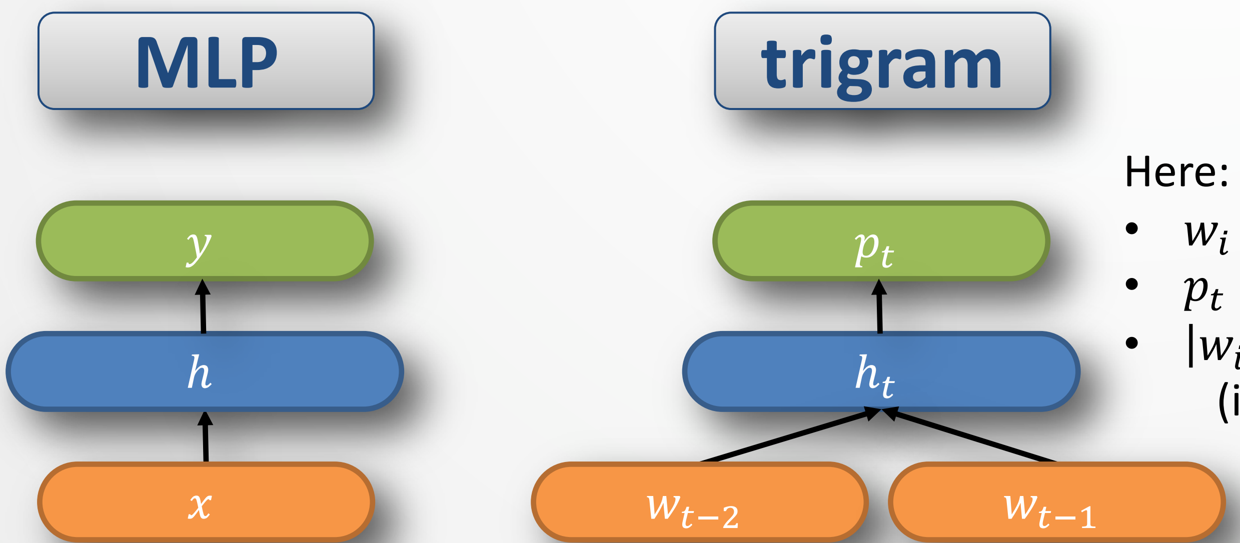


UNIVERSITY OF TORONTO

NEURAL LANGUAGE MODELS

Trigram models

- CBOW: prediction of current word w_t given w_{t-1}.
- Let's reconsider predicting w_t given multiple w_{t-j}?
  - I.e., let's think about **language modelling.**



$$h = g(W_I \boldsymbol{x} + c)$$
$$\boldsymbol{y} = W_O \boldsymbol{h} + b$$

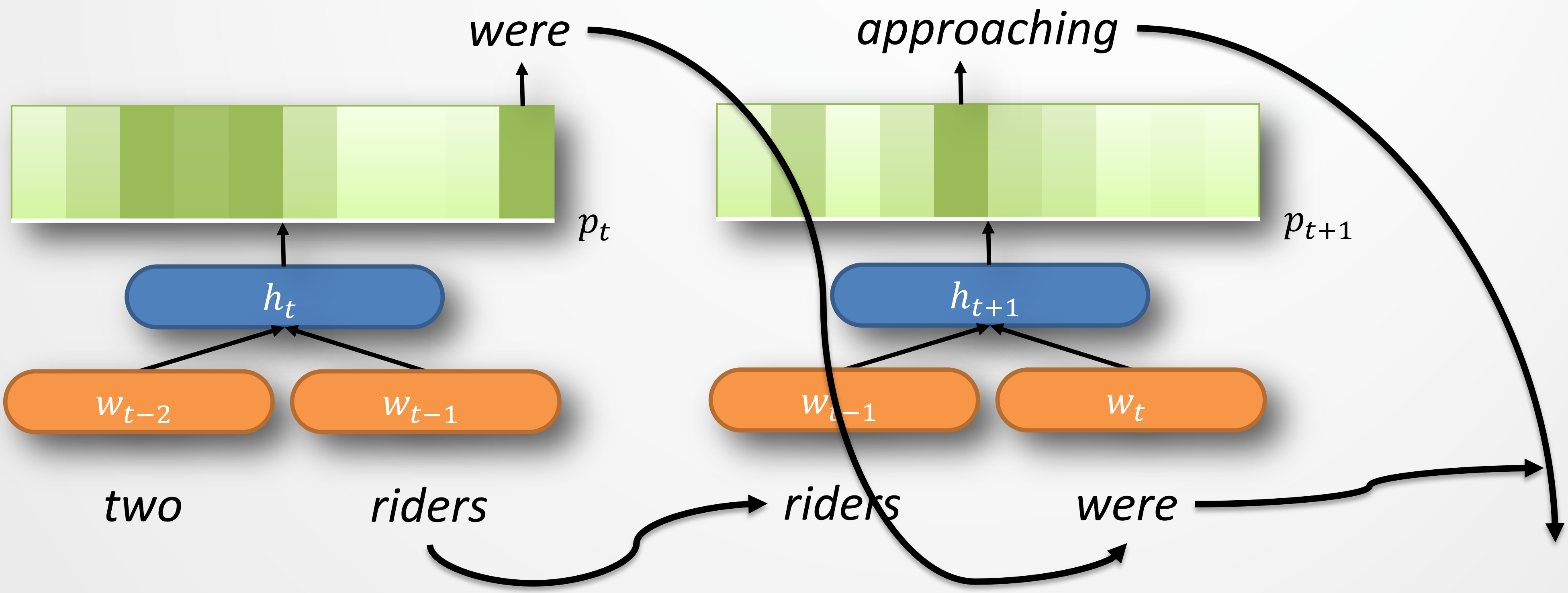
$$h_t = g(W_I[\boldsymbol{w_{t-2}}; \boldsymbol{w_{t-1}}] + c)$$
$$\boldsymbol{p_t} = softmax(W_O \boldsymbol{h_t} + b)$$

Here:

- w_i is a one-hot vector,
- p_t is a distribution, and
- $|w_i| = |p_t| = |V|$
  (i.e., the size of the vocabulary)

Sampling from trigram models

- Since $p_t \sim P(w_t | w_{t-2}\, w_{t-1})$, we just feed forward and sample from the output vector.



Training trigram models

- Here's one approach:

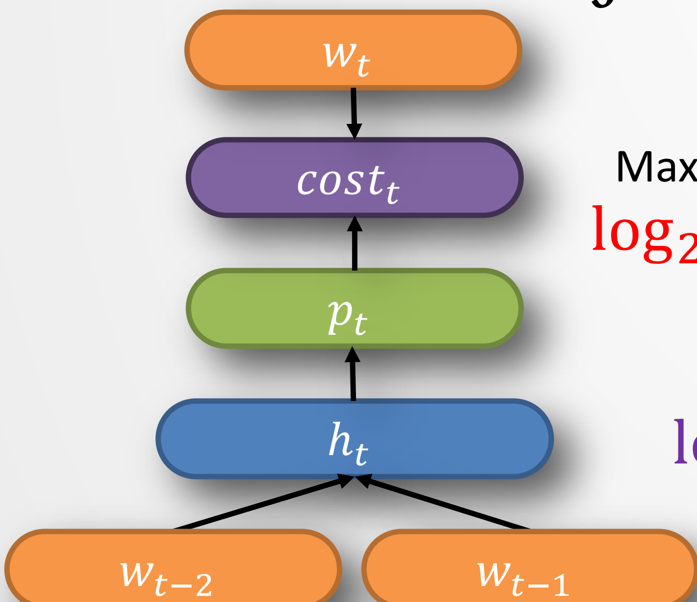
1. Randomly choose a batch (e.g., 10K consecutive words)
2. Propagate words through the current model
3. Obtain word likelihoods (loss)
4. Back-propagate loss
5. Gradient step to update model
6. Go to (1)

Training trigram models

- The typical training objective is the cross entropy (see Lecture 3) of the corpus C given the model M:

$$\mathcal{F} = H(C; M) = -\frac{\log_2 P_M(C)}{\|C\|}$$

Minimize



Maximize

$$\log_2 P_M(C) = \log_2 \prod_{t=0}^{T} P(w_t) = \sum_{t=0}^{T} \log_2 P(w_t)$$

$$\log_2 P(w_t) = w_t^\intercal \log p_t$$

$$h_t = g(W_I[\mathbf{w_{t-2}}; \mathbf{w_{t-1}}] + c)$$
$$\mathbf{p_t} = softmax(W_O \mathbf{h_t} + b)$$

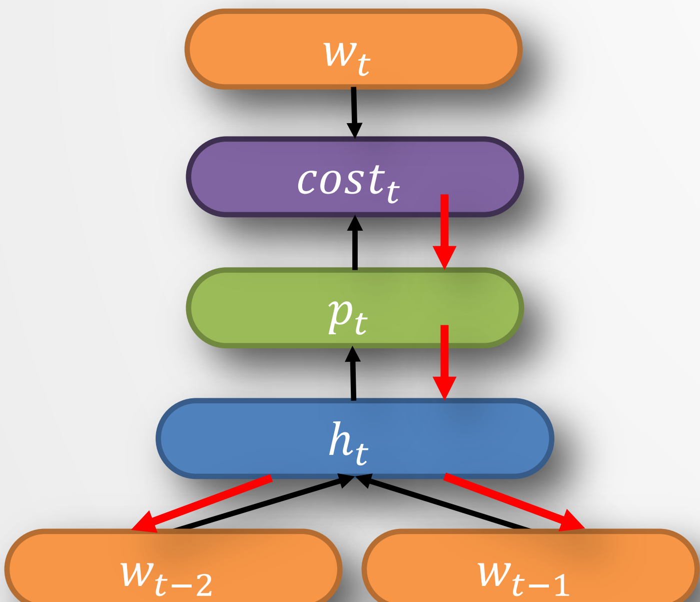
Here:

- w_i is a one-hot vector, and
- p_t is a distribution.

UNIVERSITY OF TORONTO

Training trigram models

- Compute our gradients, using $\mathcal{F} = -\frac{\log_2 P_M(C)}{\|C\|}$ and $\log_2 P(w_t) = w_t^\intercal \log p_t$ and backpropagate.



$$\frac{\delta\mathcal{F}}{\delta W_O} = -\frac{1}{\|C\|}\sum_t \frac{\delta cost_t}{\delta p_t}\frac{\delta p_t}{\delta W_O}$$

$$\frac{\delta\mathcal{F}}{\delta W_I} = -\frac{1}{\|C\|}\sum_t \frac{\delta cost_t}{\delta p_t}\frac{\delta p_t}{\delta h_t}\frac{\delta h_t}{\delta W_I}$$

$h_t = g(W_I[\boldsymbol{w_{t-2}};\boldsymbol{w_{t-1}}] + c)$
$\boldsymbol{p_t} = softmax(W_O\boldsymbol{h_t} + b)$

Here:

- w_i is a one-hot vector, and
- p_t is a distribution.

UNIVERSITY OF TORONTO

So what?

- ☺ Neural language models of this type:
  - Can generalize better than MLE LMs to unseen *n*-grams,
  - Can be modified to use *semantic* information as in word2vec.

$$P(the\ cat\ sat\ on\ the\ \boldsymbol{mat}) \approx P(the\ cat\ sat\ on\ the\ \boldsymbol{rug})$$

- ☹ Neural language models of this type:
  - Can take *relatively* long to train. “GPUs kill the Earth.”
  - Number of parameters scale poorly with increasing context.

Let’s improve both of these issues…

Dealing with that bottleneck

- Traditional datasets for neural language modeling include:
  - AP News (14M tokens, 17K types)
  - HUB-4 (1M tokens, 25K types)
  - Google News (6B tokens, 1M types)
  - Wikipedia (3.2B tokens, 2M types)

- Datasets for **medical/clinical** LM include:
  - EMRALD/ICES (3.5B tokens, 13M types)

- Much of the computational effort is in the initial embedding, and in the softmax.
  - Can we simplify and speed up the process?

UNIVERSITY OF TORONTO

Dealing with that bottleneck

- **Replace** rare words with <out-of-vocabulary> token.
- **Subsample** frequent words.

- Hierarchical softmax.
- Noise-contrastive estimation.
- Negative sampling.

[Morin & Bengio, 2005, Mikolov et al, 2011, 2013b;
Mnih & Teh 2012, Mnih & Kavukcuoglu, 2013]

Hierarchical softmax with grouping

- Group words into distinct classes, c, e.g., by frequency.
  - E.g., c_1 is top 5% of words by frequency, c_2 is the next 5%, …
- Factorize $p(w_o|w_i) = p(c|w_i)p(w_o|w_i, c)$

'**softmax**': $P(w_o|w_i) = \frac{\exp(V_{w_o}^{\intercal} v_{w_i})}{\sum_{w=1}^{W} \exp(V_w^{\intercal} v_{w_i})}$ $\Rightarrow$ $\frac{\exp(c_j v_{w_i})}{\sum_c \exp(c v_{w_i})} \times \frac{\exp(V_{w_o}^{\intercal} v_{w_i})}{\sum_{w \in c} \exp(V_w^{\intercal} v_{w_i})}$

Where

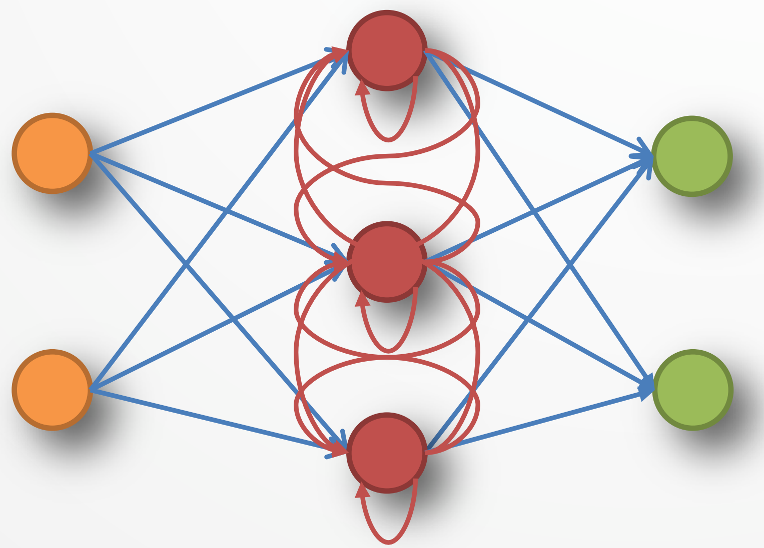
v_w is the 'input' vector for word w,
V_w is the 'output' vector for word w,

[Mikolov et al, 2011, Auli et al, 2013]

RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs)

- An RNN has **feedback** connections in its structure so that it 'remembers' previous states, when reading a sequence.
  - i.e., it passes information from one step to the next.

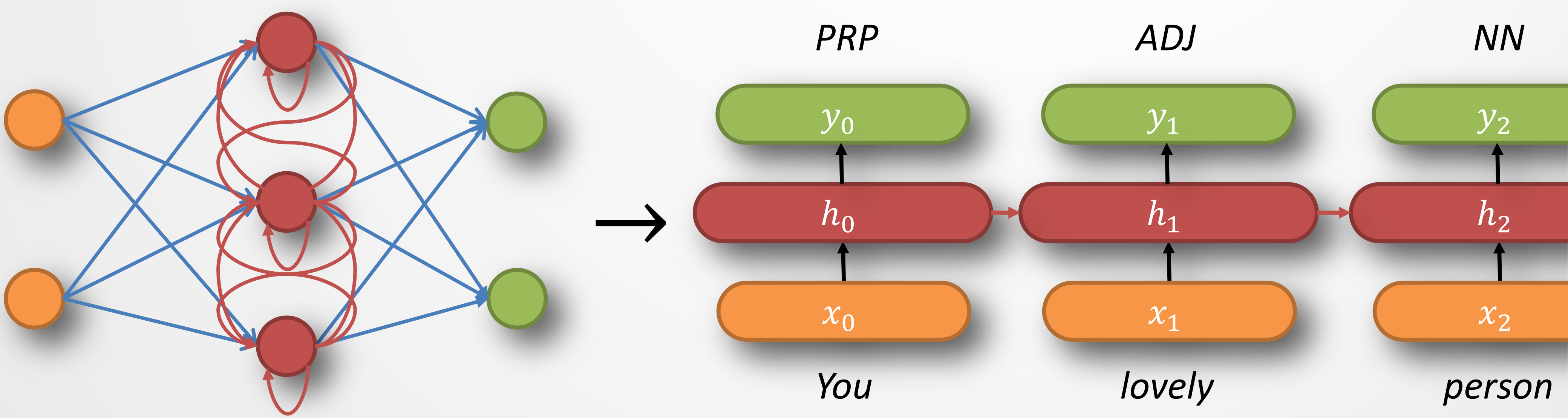


Elman network feed hidden units back

Jordan network (not shown) feed output units back

Unrolling the h_i

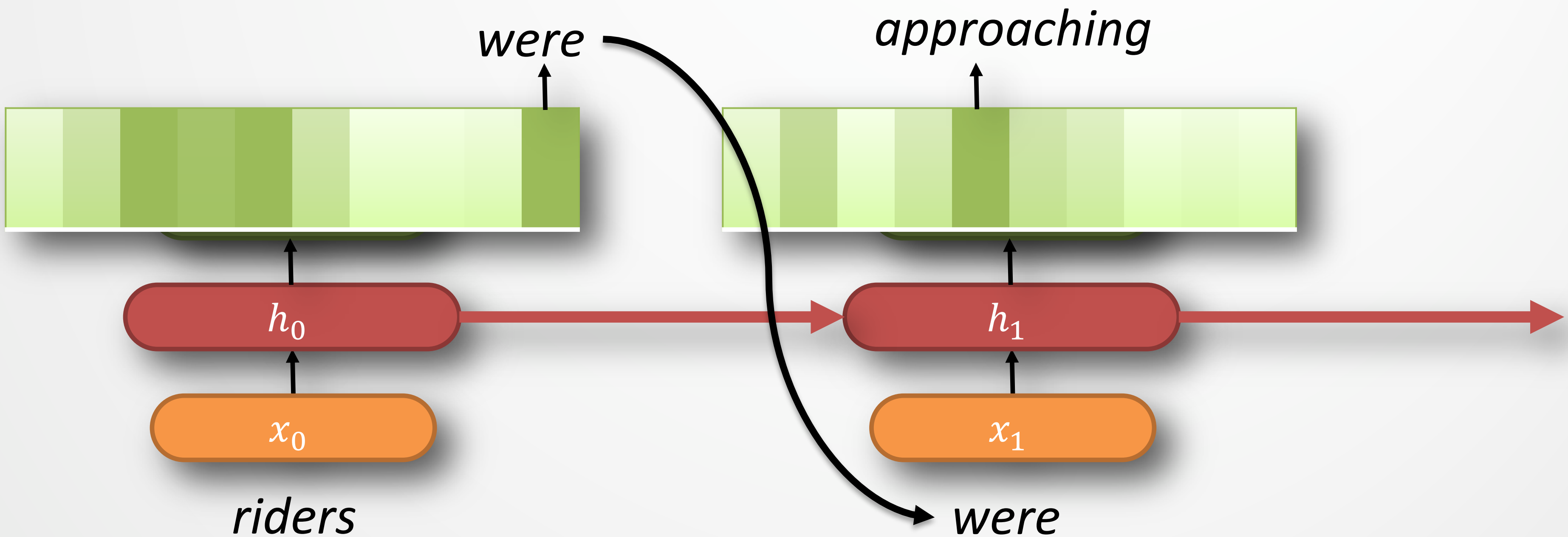
- Copies of the same network can be applied (i.e., **unrolled**) at each point in a time series.
  - These can be applied to various tasks.



$$h_t = g(W_I[\boldsymbol{x}; \boldsymbol{h_{t-1}}] + c)$$
$$\boldsymbol{y_t} = W_O \boldsymbol{h_t} + b$$

Sampling from a RNN LM

- If $|h_i| < |V|$, we've already reduced the number of parameters from the trigram NN.
  - In 'theory', information is maintained in h_i across arbitrary lengths of time…



$$h_t = g(W_I[\boldsymbol{x}; \boldsymbol{h_{t-1}}] + c)$$
$$\boldsymbol{y_t} = W_O \boldsymbol{h_t} + b$$

Karpathy (2015), [The Unreasonable Effectiveness of Recurrent Neural Networks]

University of Toronto

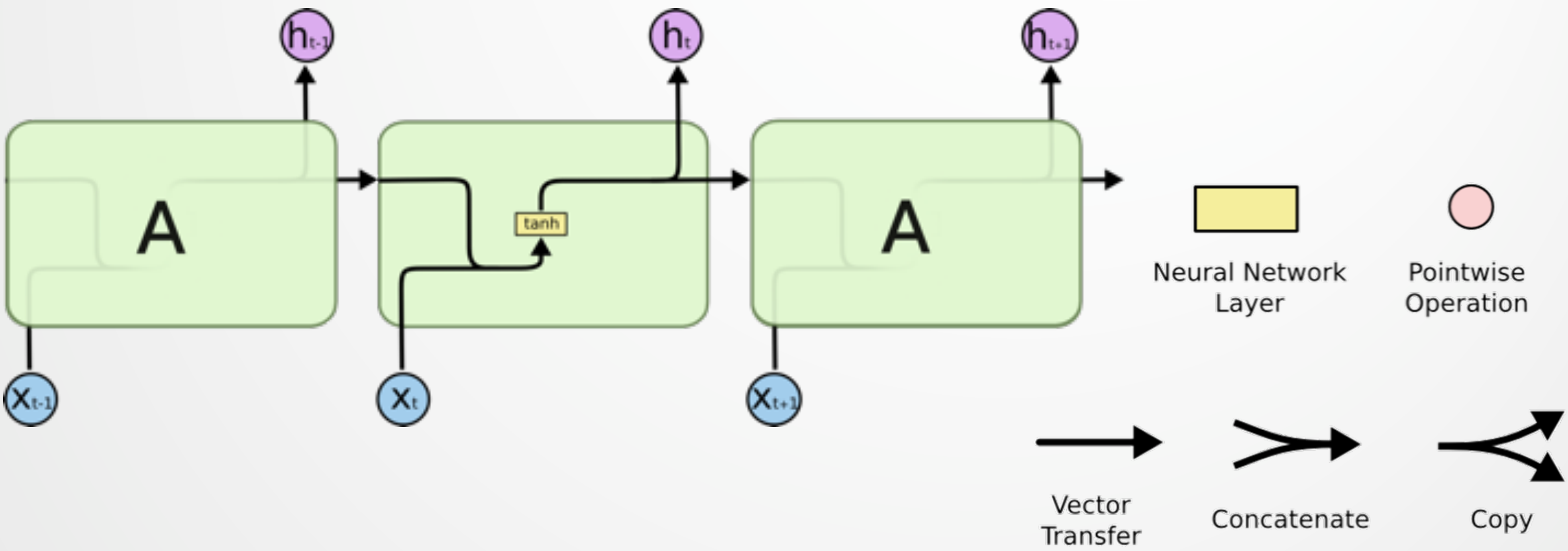
RNNs and retrograde amnesia

- Unfortunately, **catastrophic forgetting** is common.
  - E.g., the **relevant** context in "*The sushi the sister of your friend's programming teacher told you about was…*" has likely been **overwritten** by the time h_{13} is produced.

Bengio Y, Simard P, Frasconi P. (1994) Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Trans. Neural Networks.;**5**:157–66. doi:10.1109/72.279181

RNNs and retrograde amnesia

- The challenge with RNNs is that the **gradient** decays quickly as one pushes it back in time. Can we store relevant information?

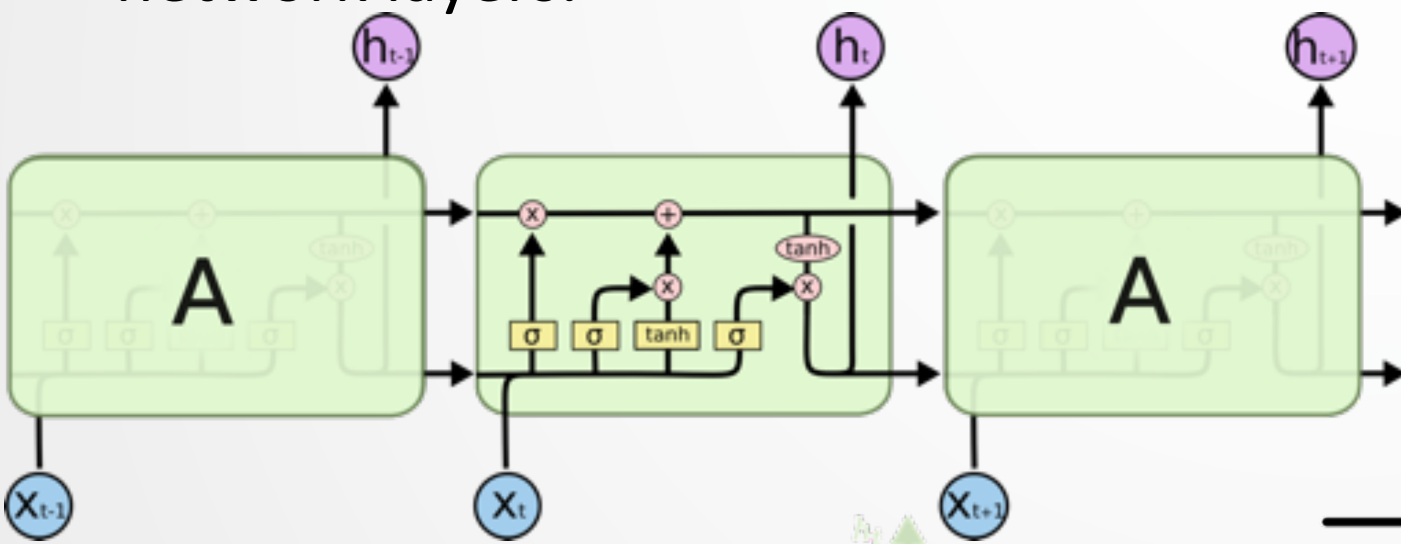


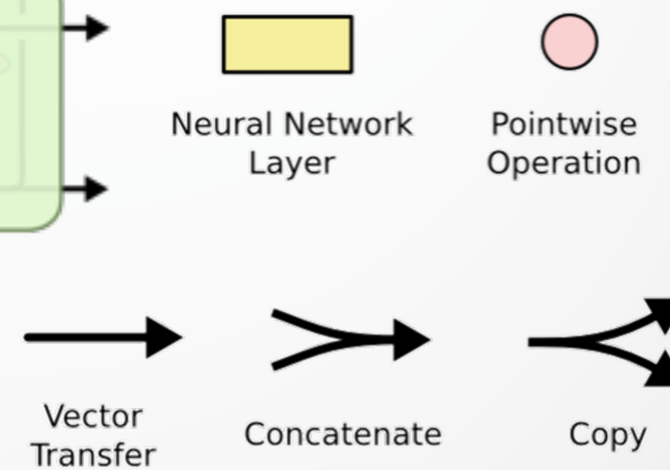
Imagery and sequence from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

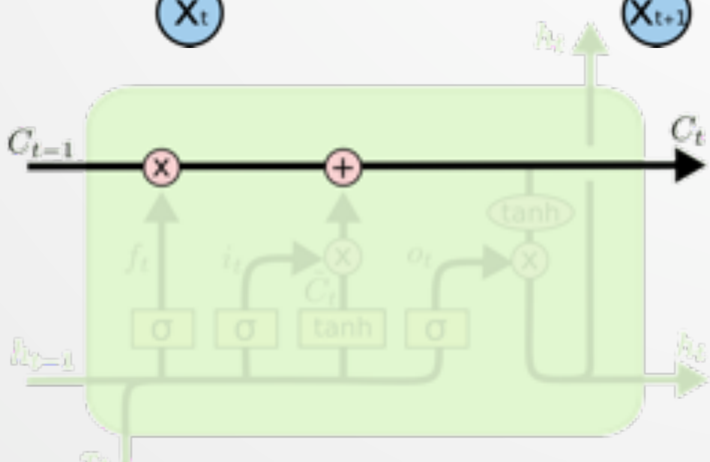
UNIVERSITY OF TORONTO

Long short-term memory (LSTM)

- In each **module**, in an LSTM, there are four interacting neural network layers.





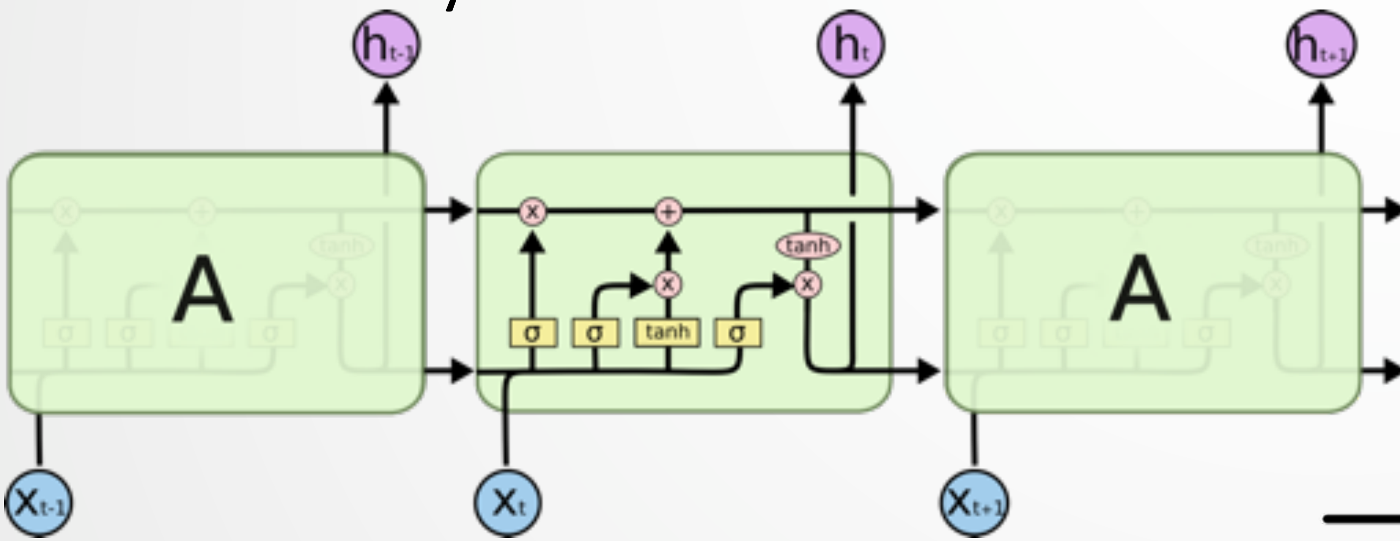


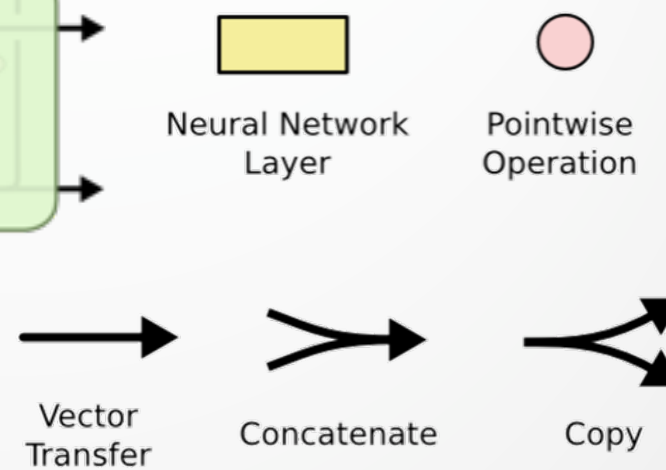
The **cell state** is a special vector stream that runs through the entire chain and **stores the long-term information.**

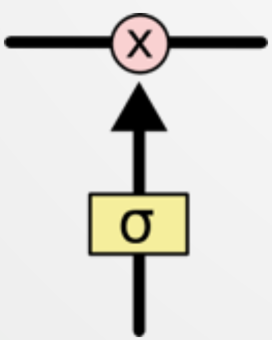
UNIVERSITY OF TORONTO

Long short-term memory (LSTM)

- In each **module**, in an LSTM, there are four interacting neural network layers.





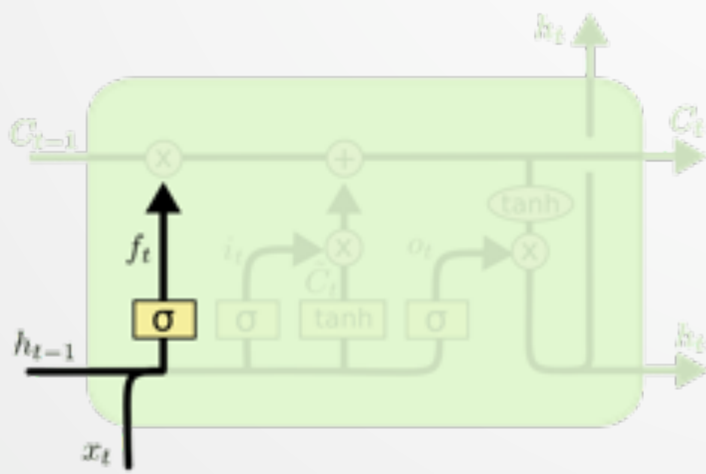


Gates decide what information should be withheld in the cell state.
They are a **sigmoid** followed by a pointwise ×.
Values near 0 block information; values near 1 pass information.

UNIVERSITY OF TORONTO

LSTM step 1: decide what to forget

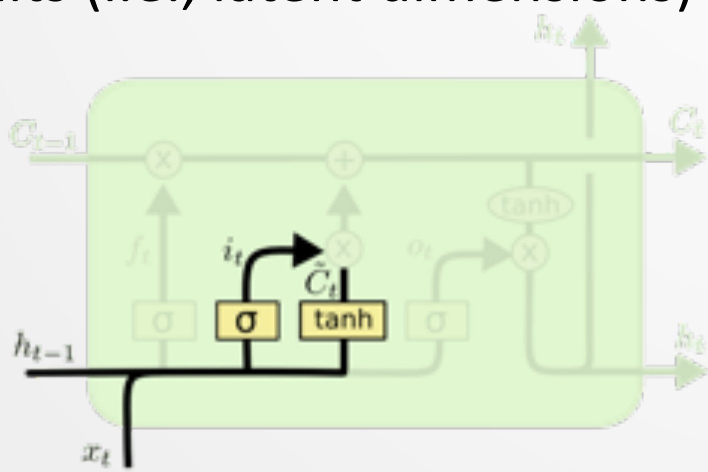
- The **forget gate layer** compares h_{t-1} and the current input x_t to decide which elements in cell state C_{t-1} to keep and which to turn off.
  - E.g., the cell state might 'remember' the number (sing./plural) of the current subject, in order to predict appropriately conjugated verbs, but decide to forget it when a new subject is mentioned at x_t.
    - (There's scanty evidence that such information is so explicit.)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

LSTM step 2: decide what to store

- The **input gate layer** has two steps.
  - First, a sigmoid layer σ decides which cell units to update.
  - Next, a tanh layer creates new candidate values $\widetilde{C}_t$.
  - E.g., the σ can turn on the 'number' units, and the tanh can push information on the current subject.
  - The σ layer is important – we don't want to push information on units (i.e., latent dimensions) for which we have no information.

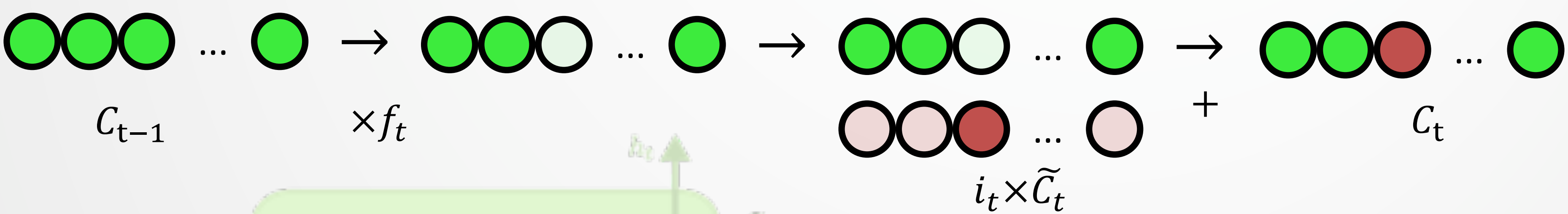


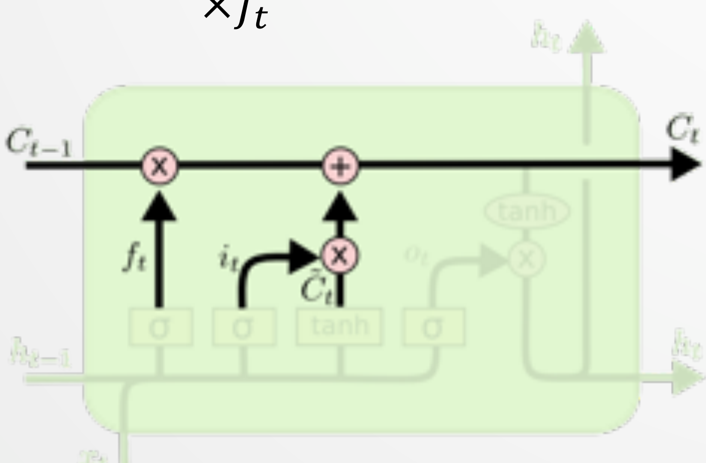
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

UNIVERSITY OF TORONTO

LSTM step 3: update the cell state

- Update C_{t-1} to C_t.
  - First, forget what we want to forget: multiply C_{t-1} by f_t.
  - Then, create a ‘mask vector’ of information we want to store, $i_t \times \widetilde{C}_t$.
  - Finally, write this information to the new cell state C_t.





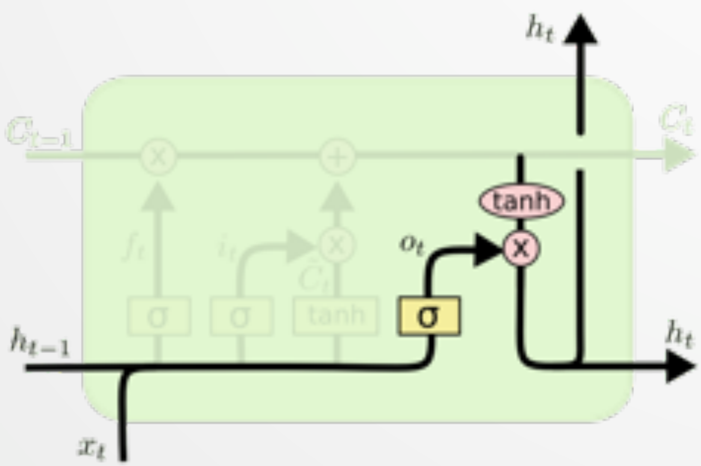
$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t$$

UNIVERSITY OF TORONTO

LSTM step 4: output and feedback

- Output something, o_t, based on the current x_t and h_{t-1}.
- Combine the output with the cell to give your h_t.
  - Normalize cell C_t on [-1,1] using tanh and combine with o_t

- In some sense, C_t is long-term memory and h_t is the short-term memory (hence the name).

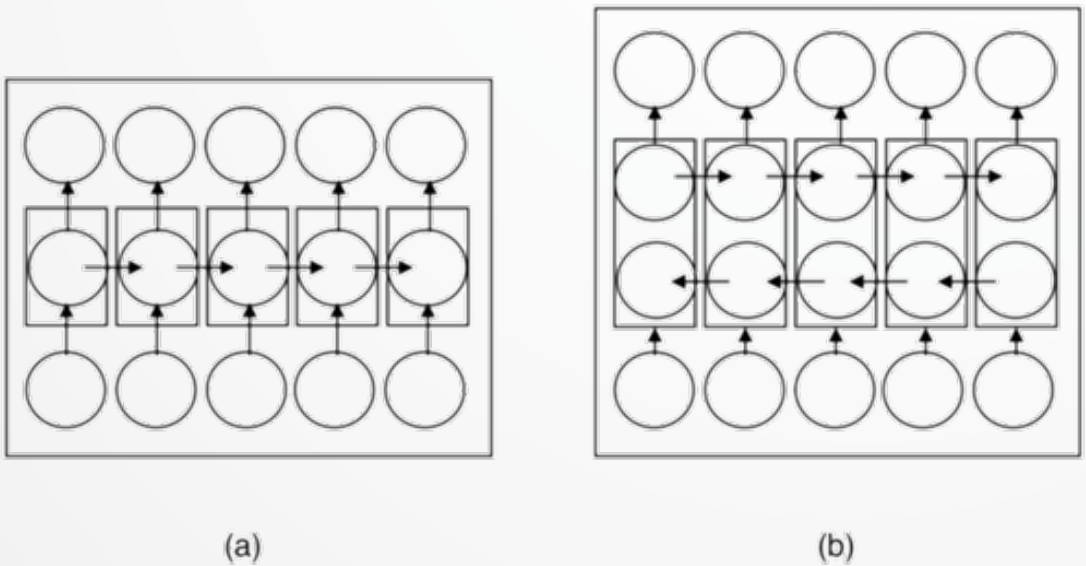


$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(C_t)$$

Variants of LSTMs

- There are various variations on LSTMs.
  - 'Bidirectional LSTMs' (and bidirectional RNNs generally), learn

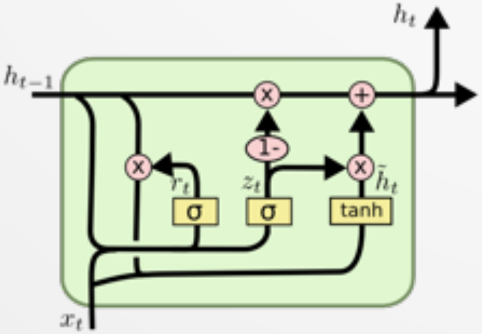


Structure overview
(a) unidirectional RNN
(b) bidirectional RNN

Schuster, Mike, and Kuldip K. Paliwal. (1997) Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on* **45(**11) (1997): 2673-2681.2.

Variants of LSTMs

- There are various variations on LSTMs.
  - Gers & Schmidhuber (2000) add ‘**peepholes**’ that allow all sigmoids to read the cell state.
  - We can **couple** the ‘forget’ and ‘input’ gates.
    - E.g., it’s a bit of a waste to decide to forget number, then decide to store a new number.
    - **Gated Recurrent units** (GRUs; [Cho *et al* (2014)]) go a step further and also merge the cell and hidden states.



$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$ **Update** gate

$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$ **Reset** gate (0: replace units in h_{t-1} with those in x_t)

$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$

$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

Are there examples where GRUs are used instead of LSTMs?

RECENT-ISH BREAKTHROUGHS

UNIVERSITY OF TORONTO

Deep contextualized representations

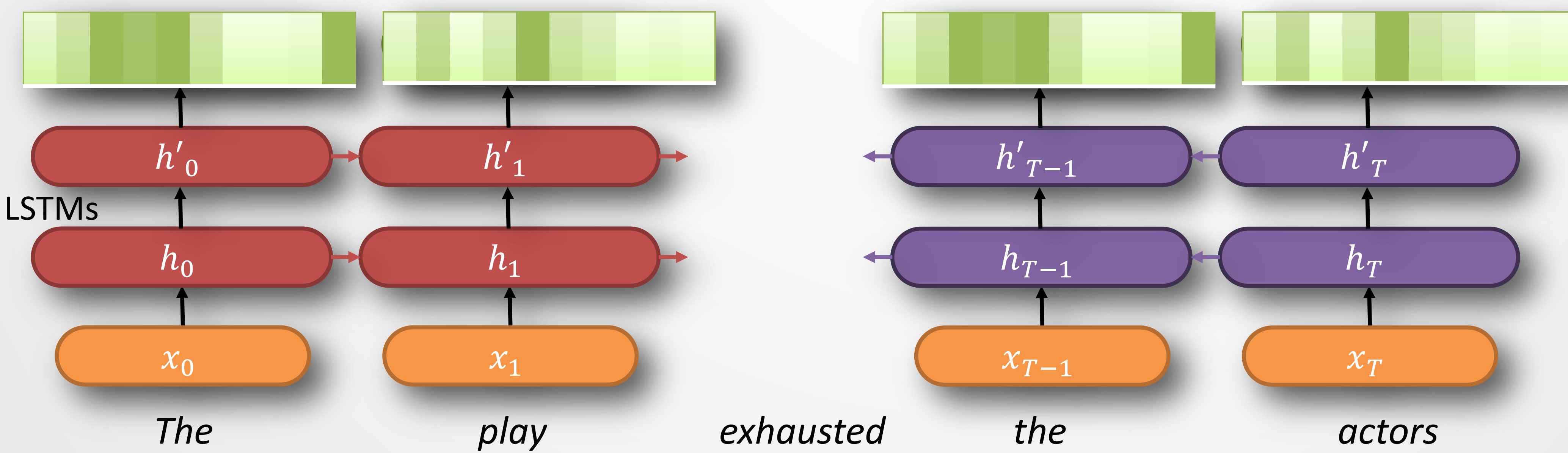
- What does the word *play* mean?

AllenNLP

Peters ME, Neumann M, Iyyer M, *et al.* (2018) Deep contextualized word representations. Published Online First: 2018. doi:10.18653/v1/N18-1202; http://arxiv.org/abs/1802.05365

ELMo: Embeddings from Language Models

- Instead of a fixed embedding for each word **type**, ELMo considers the entire sentence before embedding each **token**.
  - It uses a bi-directional LSTM trained on a specific task.
  - Outputs are softmax probabilities on words, as before.



UNIVERSITY OF TORONTO

ELMo: Embeddings from Language Models

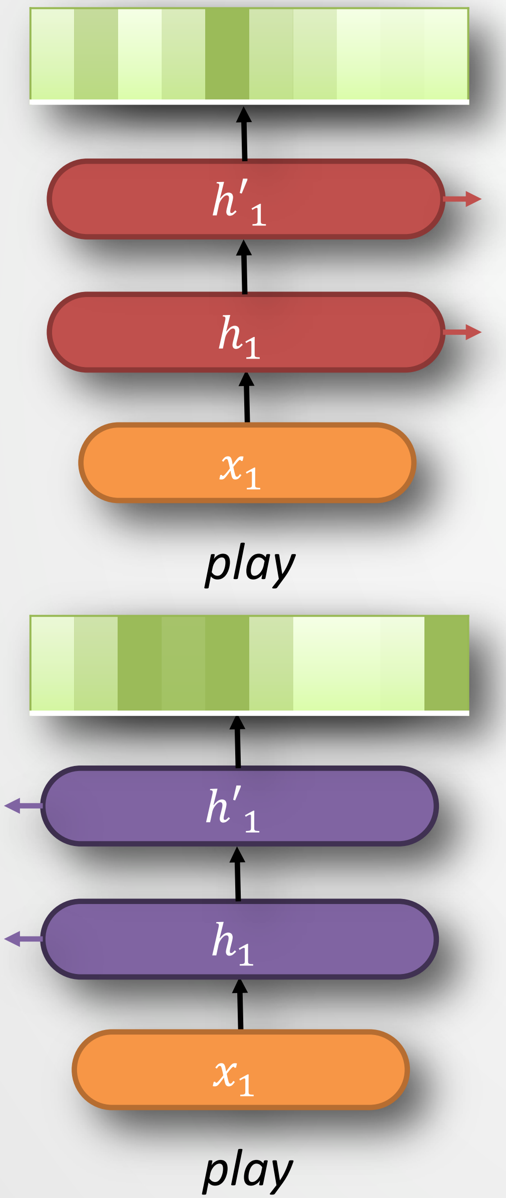
- Producing the final embedding for word token k.

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

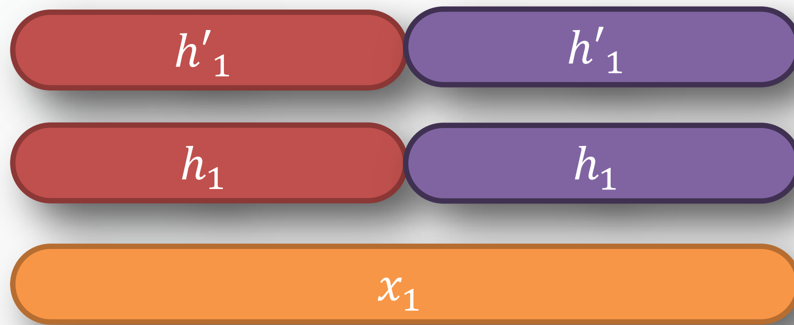
where R_K is the set of all L hidden layers, $\mathbf{h}_{k,j}$
s_j^{task} is the task's weight on the layer, and
γ^{task} is a weight on the entire task

UNIVERSITY OF TORONTO

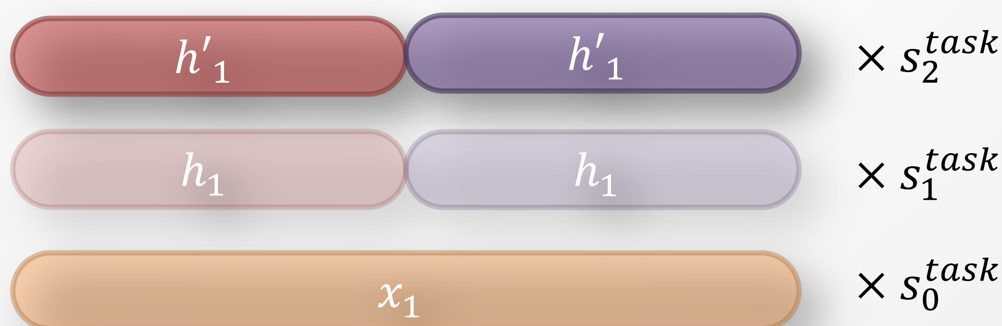
ELMo: Embeddings from Language Models



1. Concatenate



2. Multiply by weight vectors



3. Sum

UNIVERSITY OF TORONTO

ELMo: Embeddings from Language Models

- What does the word *play* mean?

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Peters ME, Neumann M, Iyyer M, *et al.* (2018) Deep contextualized word representations. Published Online First: 2018. doi:10.18653/v1/N18-1202; http://arxiv.org/abs/1802.05365

ELMo: Embeddings from Language Models

	Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
Q&A	SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
Textual entailment	SNLI	Chen et al. (2017)	88.6	88.0	88.7 $\pm$ 0.17	0.7 / 5.8%
Semantic role labelling	SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coreference resolution	Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
Name entity resolution	NER	Peters et al. (2017)	91.93 $\pm$ 0.19	90.15	92.22 $\pm$ 0.10	2.06 / 21%
Sentiment analysis	SST-5	McCann et al. (2017)	53.7	51.4	54.7 $\pm$ 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Peters ME, Neumann M, Iyyer M, *et al.* (2018) Deep contextualized word representations. Published Online First: 2018. doi:10.18653/v1/N18-1202; http://arxiv.org/abs/1802.05365

BERT: Bidirectional encoder representations from transformers

- Unlike ELMo, BERT is **deeply** bidirectional.
  - i.e., every embedding conditions every other in the next layer.

- This is difficult, because when predicting word x_t, you would already have 'seen' that word in modelling its own contexts.

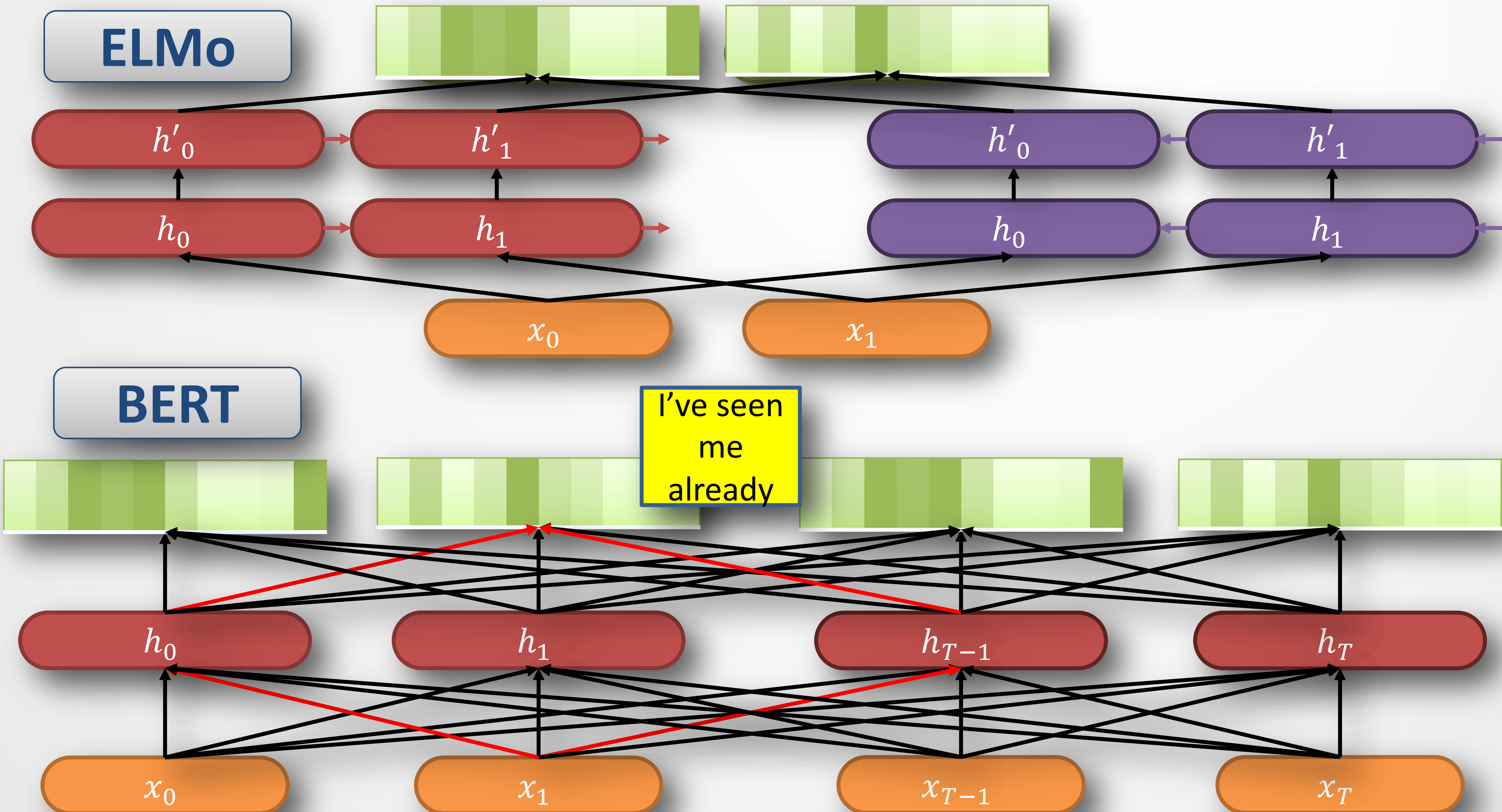
Code and models: https://github.com/google-research/bert

Google AI

Devlin J, Chang M-W, Lee K, *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805

UNIVERSITY OF TORONTO

BERT: Bidirectional encoder representations from transformers



BERT: Bidirectional encoder representations from transformers

- This can be solved by **masking** the word being predicted.

Input: The man went to the [MASK]$_1$. He bought a [MASK]$_2$ of milk .
Labels: [MASK]$_1$ = store; [MASK]$_2$ = gallon

  - (actually, 80% we use [MASK]. 10% we replace the target word with another actual word; 10% we keep the word as-is, to bias ‘towards the observation’.)

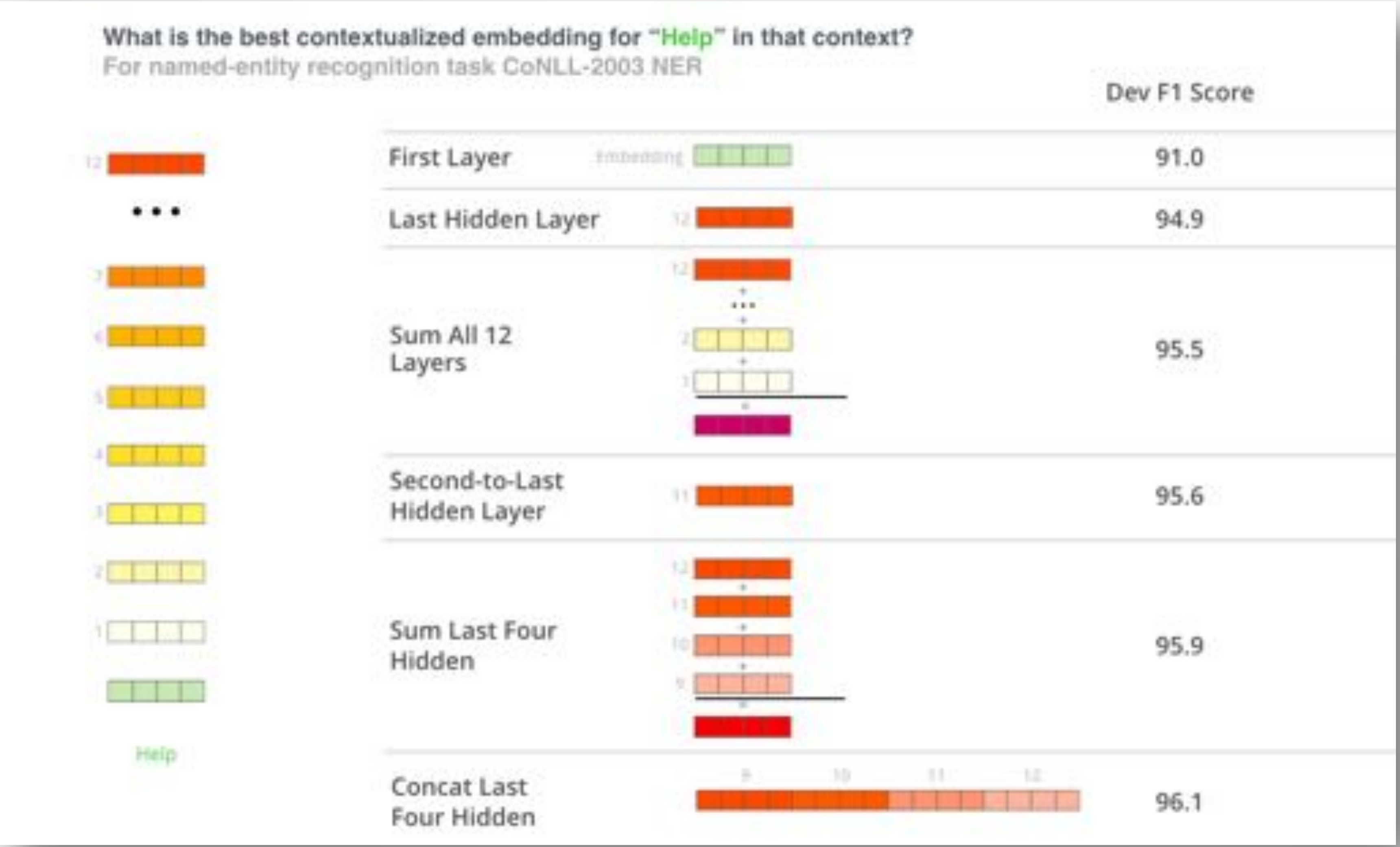
- We can also predict other relationships, like whether one sentence follows another.

Sentence A = The man went to the store. **Sentence B** = He bought a gallon of milk. **Label** = IsNextSentence	**Sentence A** = The man went to the store. **Sentence B** = Penguins are flightless. **Label** = NotNextSentence

  - (actually, you can fine-tune on *many* different tasks)

Aroca-Ouellette S, Rudzicz F (2020) On Losses for Modern Language Models, EMNLP.

BERT: Bidirectional encoder representations from transformers



(From http://jalammar.github.io/illustrated-bert/)

UNIVERSITY OF TORONTO

BERT: Bidirectional encoder representations from transformers

- The age of humans is over?

SQuAD1.1 Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

Humans

BERT

Aside – ClosedAI

- There are, of course, alternatives.

- **FastText**: Represent each word as a bag of character-grams
  Paper: https://arxiv.org/abs/1607.04606
  Code: https://fasttext.cc
- **ULMFit**: Model fine-tuning for classification tasks
  Paper: https://arxiv.org/abs/1801.06146
  Code: Here
- **GPT-2/3**: Spooky, closed uni-directional model
  Paper: Here
  Blog: Here

OTHER APPLICATIONS

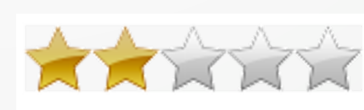
Sentiment analysis

- The traditional **bag-of-words** approach to sentiment analysis used dictionaries of *happy* and *sad* words, simple counts, and either *regression* or *binary* classification.

- But consider these:

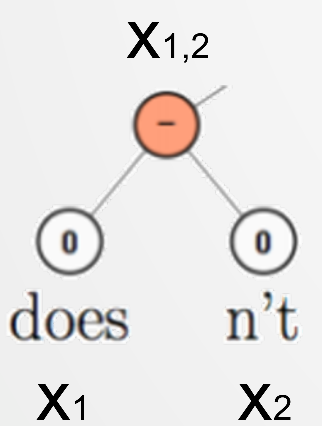
Best movie of the year 

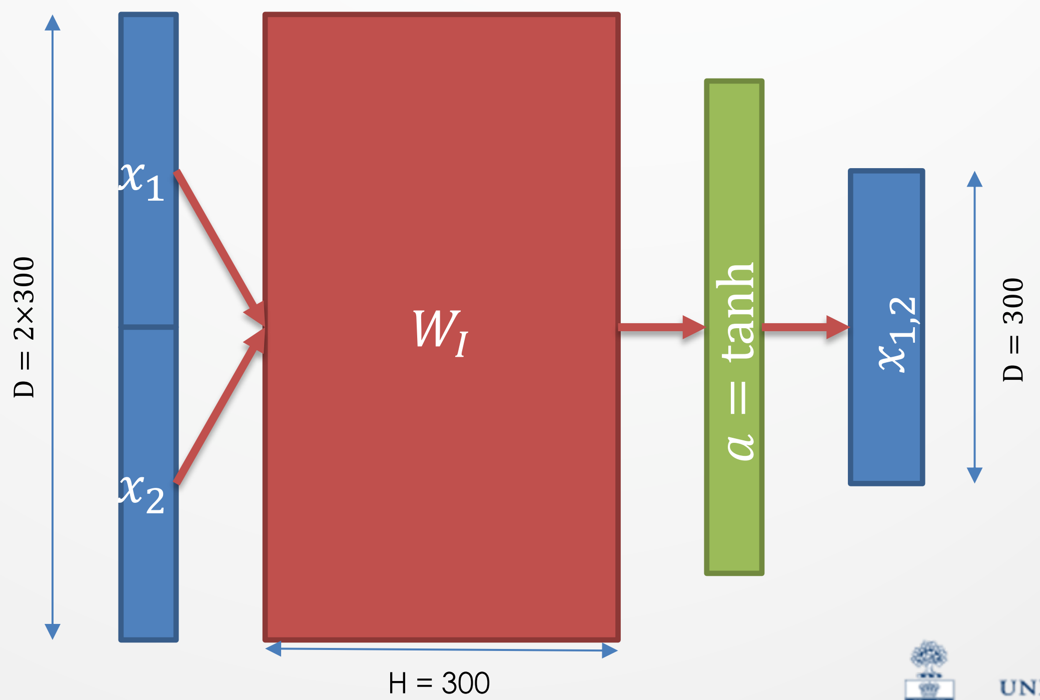
Slick and entertaining, despite a weak script 

Fun and sweet but ultimately unsatisfying 

Tree-based sentiment analysis

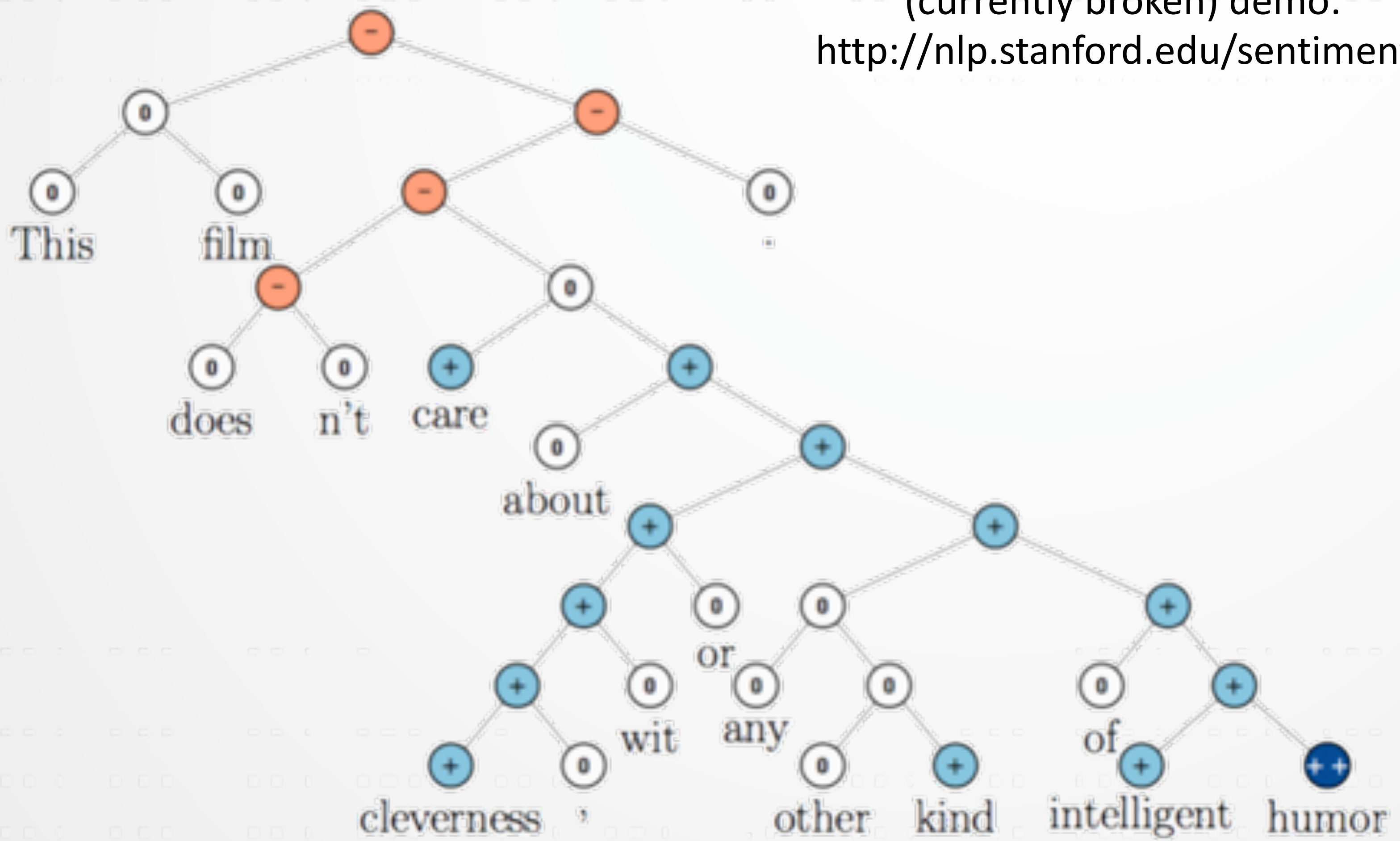
- We can **combine pairs** of words into phrase structures.
- Similarly, we can combine phrase and word structures hierarchically for classification.





Tree-based sentiment analysis

(currently broken) demo:
http://nlp.stanford.edu/sentiment/



UNIVERSITY OF TORONTO

Neural networks

- Research in neural networks is exciting, expansive, and explorative.
- We have many **hyperparameters** we can tweak
  (e.g., activation functions, number and size of layers).
- We have many **architectures** we can use
  (e.g., deep networks, LSTMs, attention mechanisms).

- Given the fevered hype, it’s important to retain our scientific skepticism.
  - What are our **biases** and expectations?
  - When are neural networks the **wrong choice**?
  - How are we actually **evaluating** these systems?