

# Social Media & Text Analysis

## lecture 9 - Deep Learning for NLP

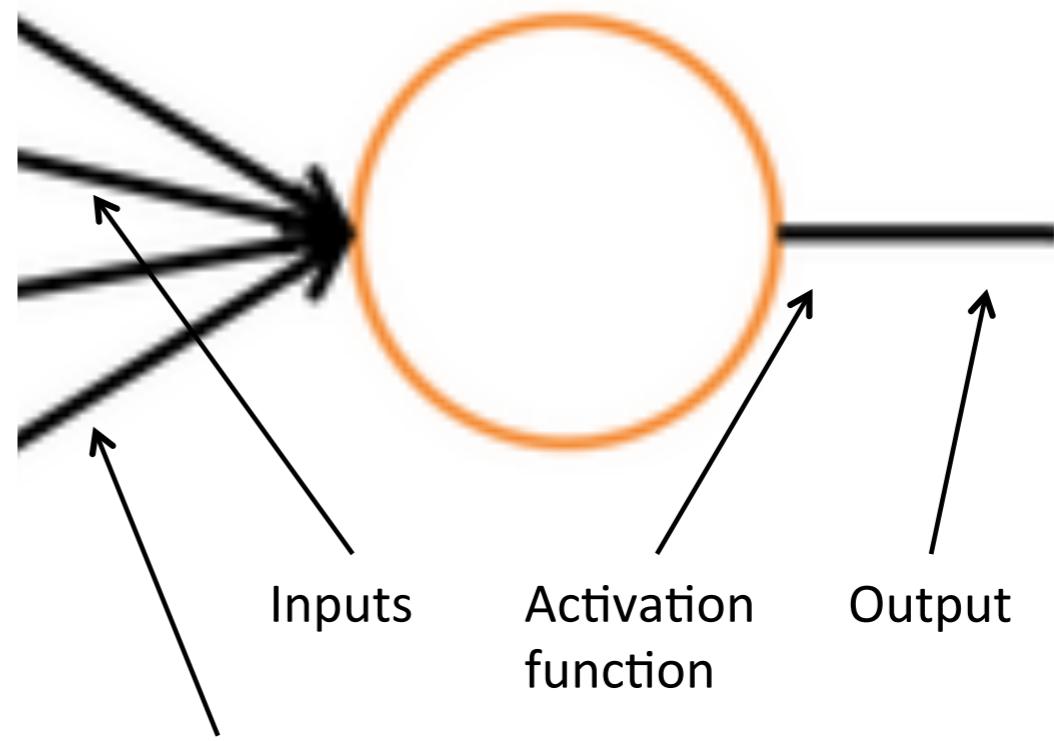
**CSE 5539-0010 Ohio State University**  
**Instructor: Alan Ritter**  
**Website: [socialmedia-class.org](http://socialmedia-class.org)**

Many slides are adapted from Richard Socher, Greg Durret, Chris Dyer, Dan Jurafsky, Chris Manning

# A Neuron

- If you know Logistic Regression, then you already understand a basic neural network neuron!

**A single neuron**  
A computational unit with  $n$  (3) inputs  
and 1 output  
and parameters  $W, b$



Bias unit corresponds to intercept term

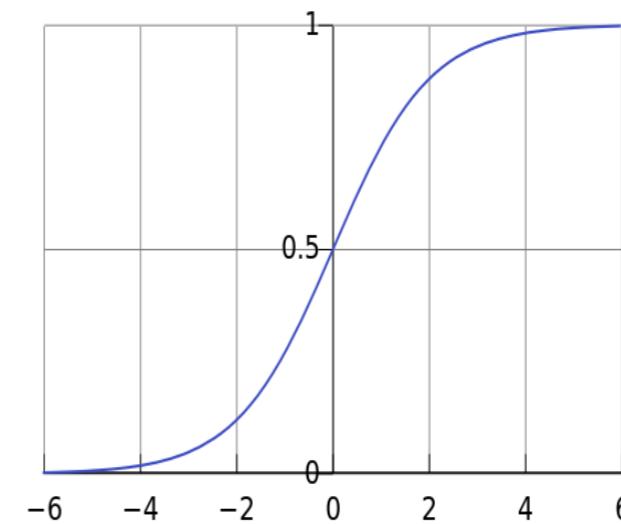
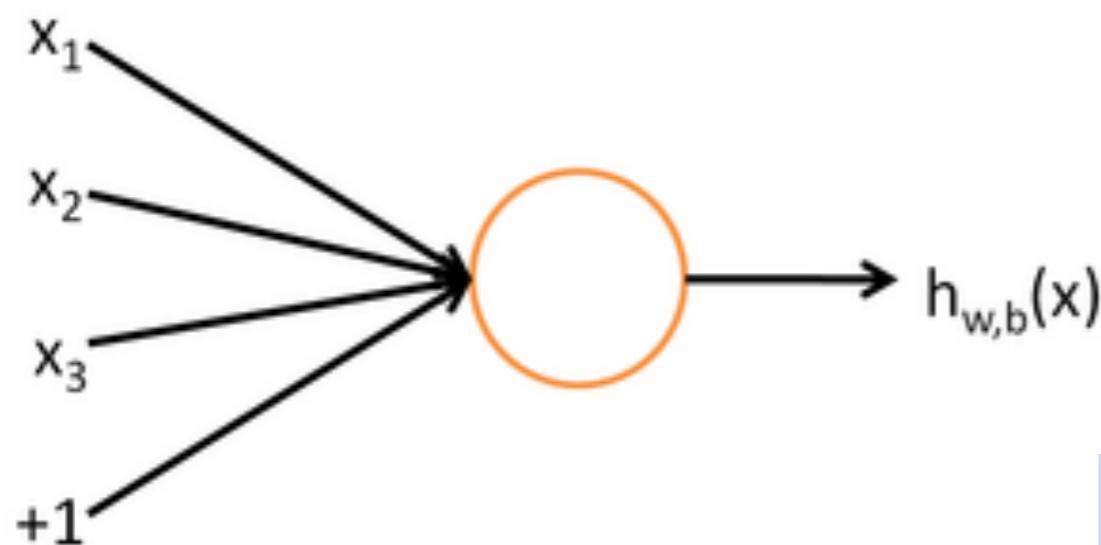
# A Neuron

is essentially a binary logistic regression unit

$$h_{w,b}(x) = f(w^\top x + b)$$

*b:* We can have an “always on” feature, which gives a class prior, or separate it out, as a bias term

$$f(z) = \frac{1}{1 + e^{-z}}$$

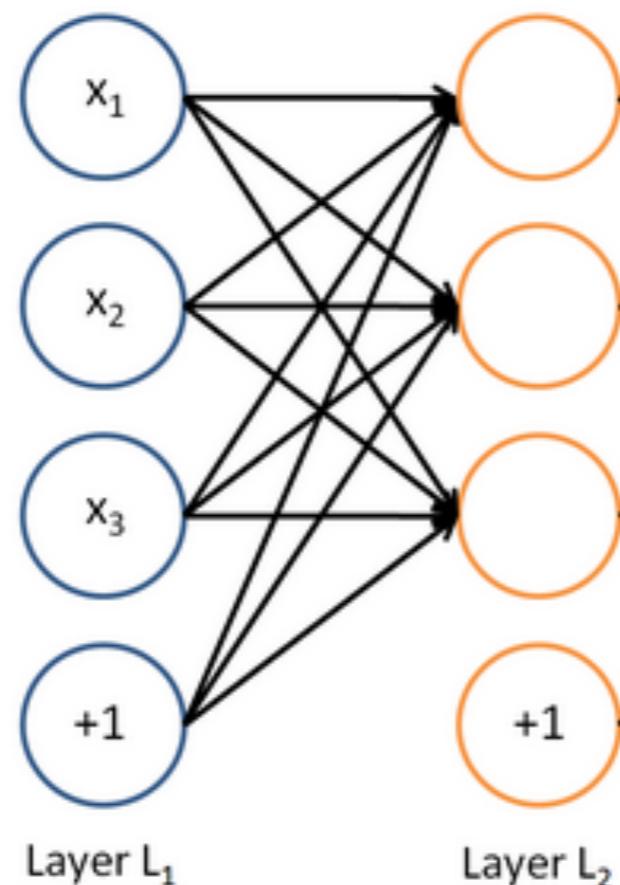


*w, b* are the parameters of this neuron  
i.e., this logistic regression model

# A Neural Network

= running several logistic regressions at the same time

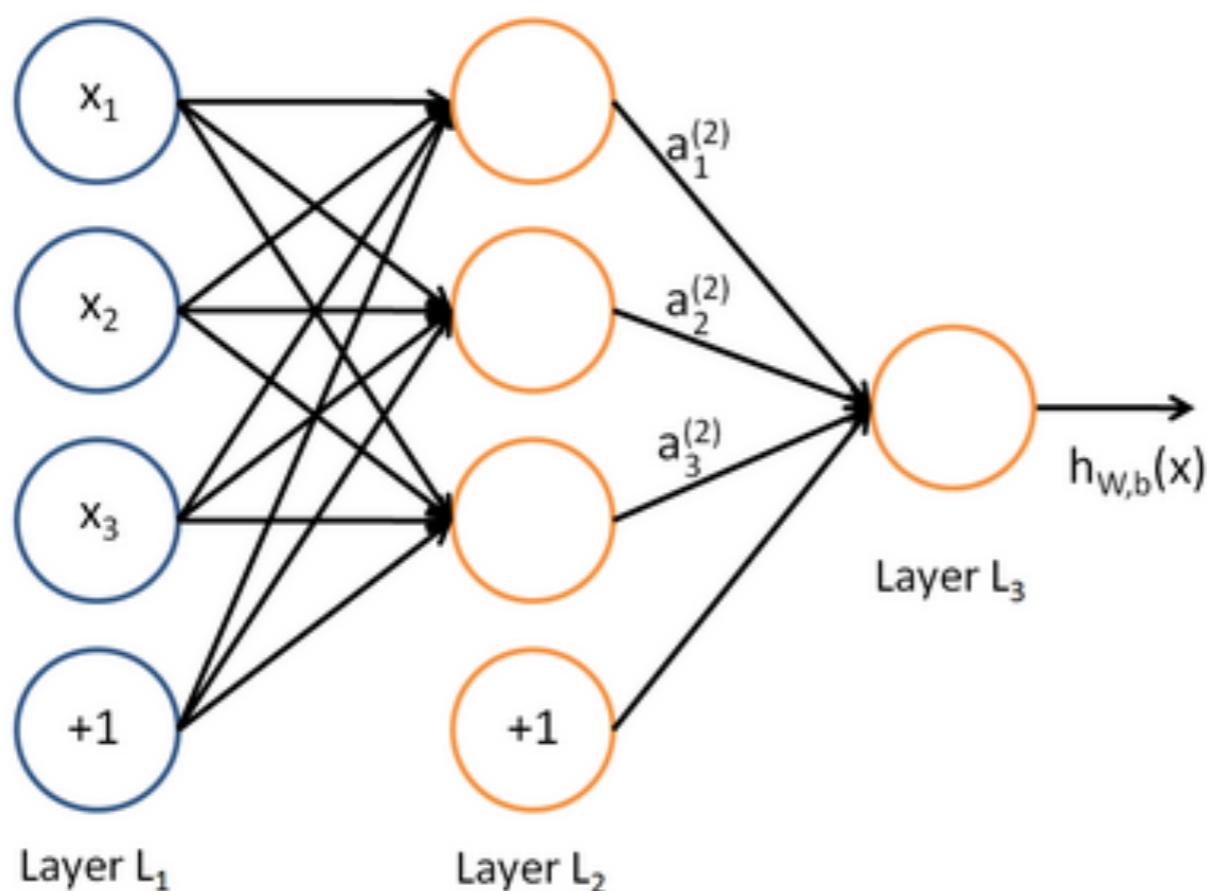
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs ...



# A Neural Network

= running several logistic regressions at the same time

... which we can feed into another logistic regression function

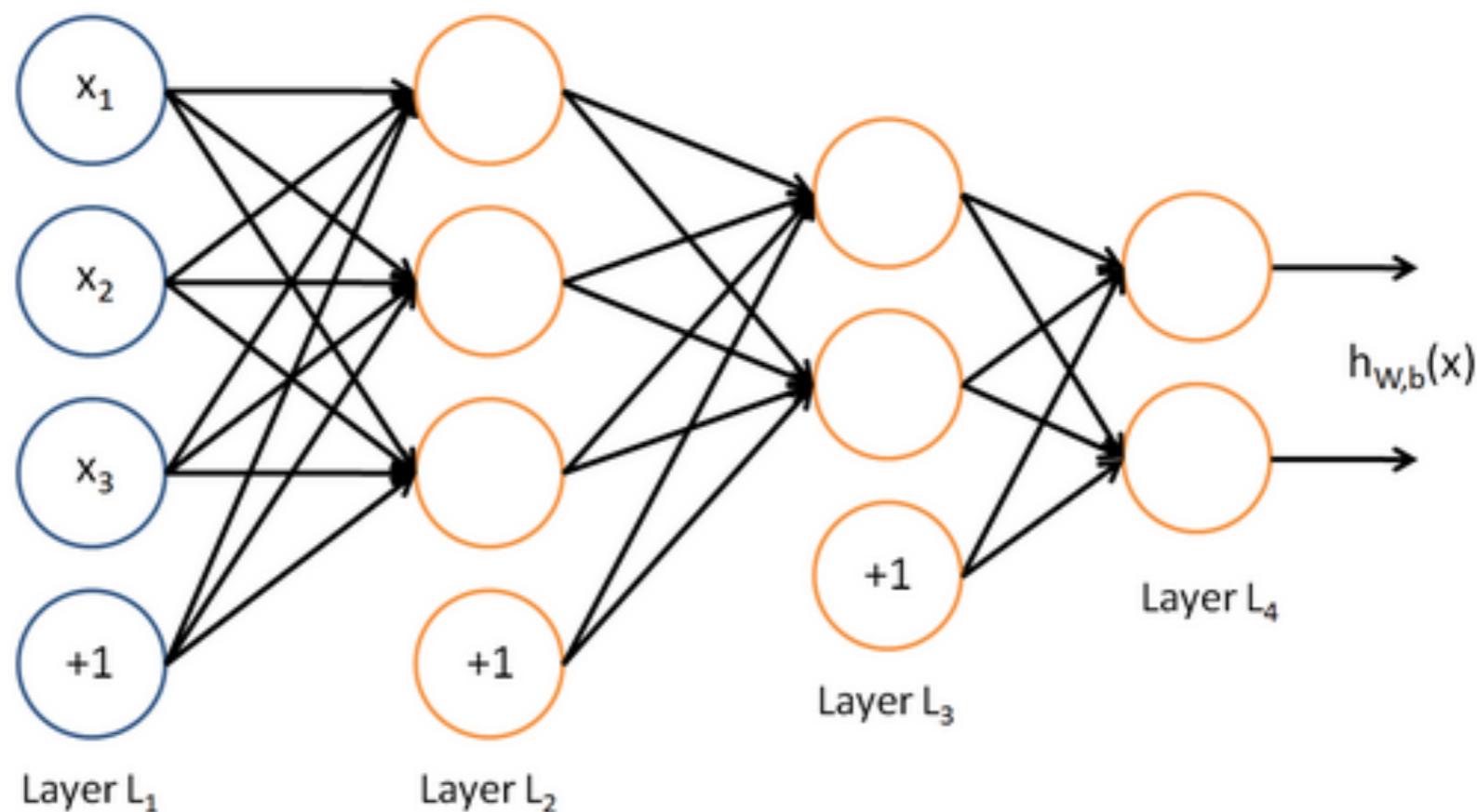


*It is the loss function that will direct what the intermediate hidden variables should be, so as to do a good job at predicting the targets for the next layer, etc.*

# A Neural Network

= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network....



# $f$ : Activation Function

We have

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$

$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

etc.

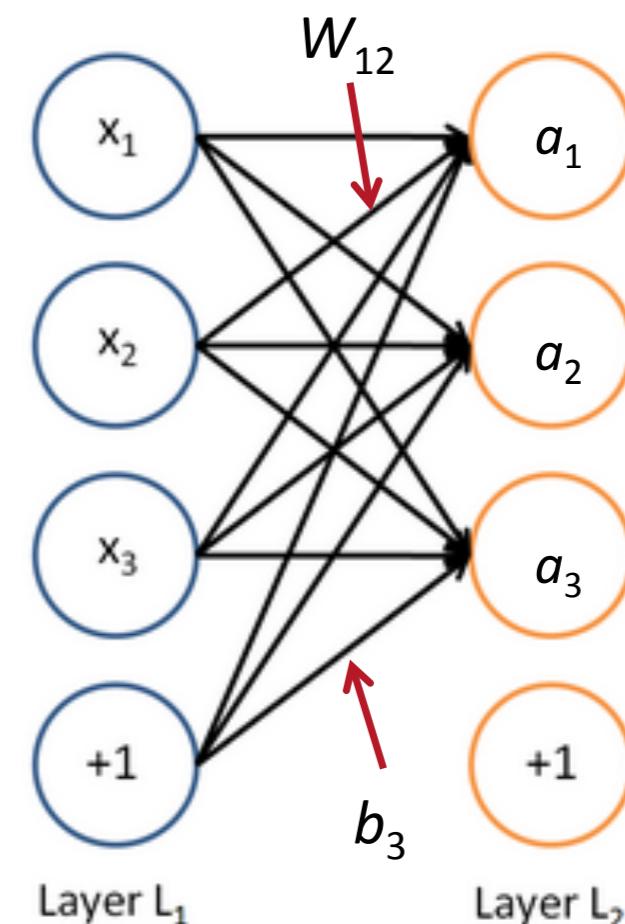
In matrix notation

$$z = Wx + b$$

$$a = f(z)$$

where  $f$  is applied element-wise:

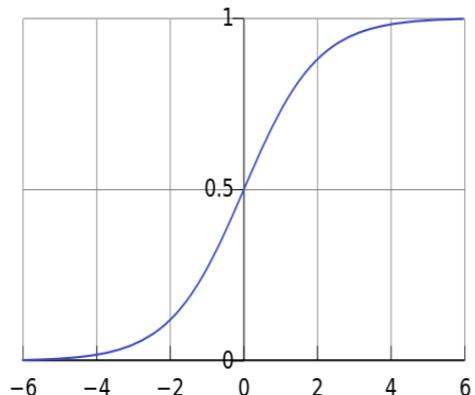
$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$



# Activation Function

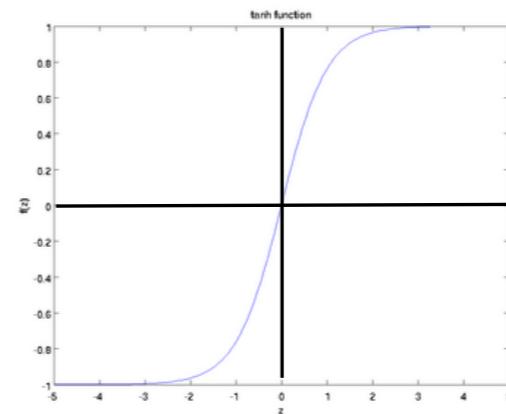
logistic (“sigmoid”)

$$f(z) = \frac{1}{1 + \exp(-z)}.$$



tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$



$$f'(z) = f(z)(1 - f(z))$$

$$f'(z) = 1 - f(z)^2$$

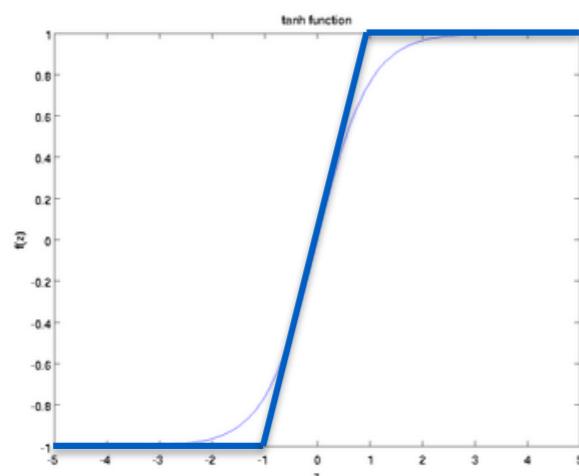
tanh is just a rescaled and shifted sigmoid

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

# Activation Function

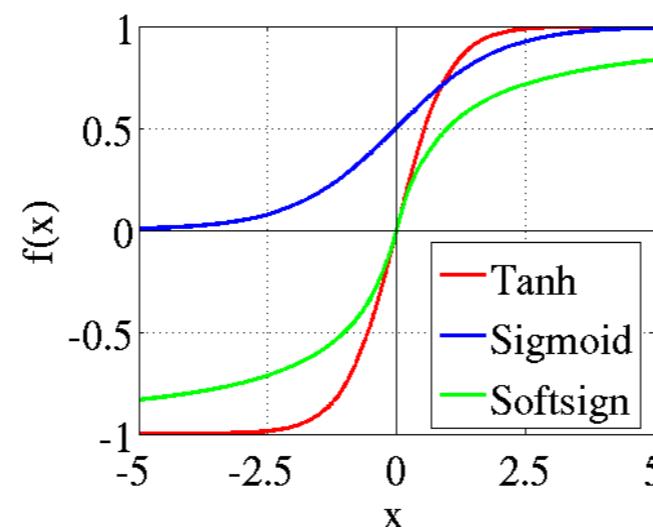
hard tanh

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$



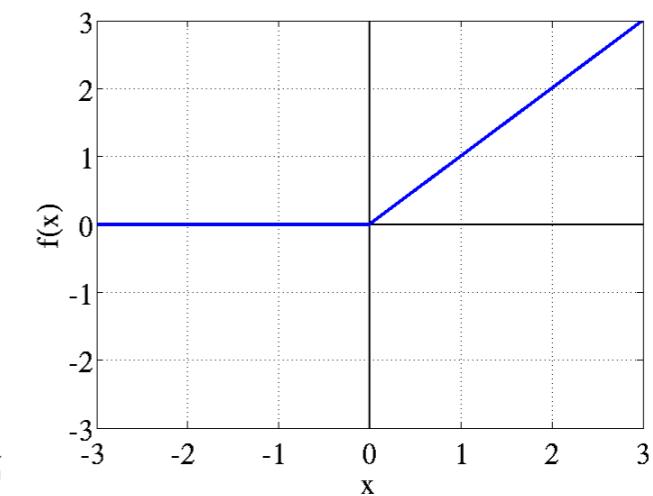
soft sign

$$\text{softsign}(z) = \frac{a}{1+|a|}$$



rectified linear (ReLU)

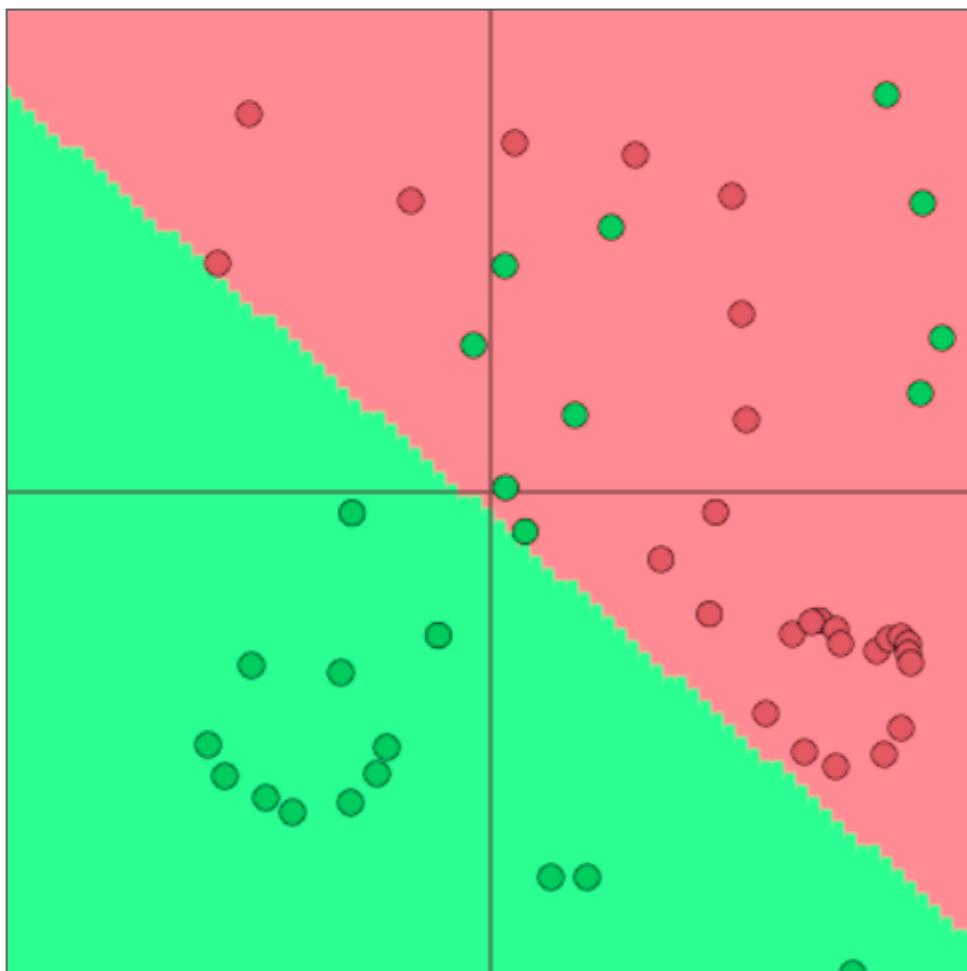
$$\text{rect}(z) = \max(z, 0)$$



- hard tanh similar but computationally cheaper than tanh and saturates hard.
- Glorot and Bengio, *AISTATS 2011* discuss softsign and rectifier

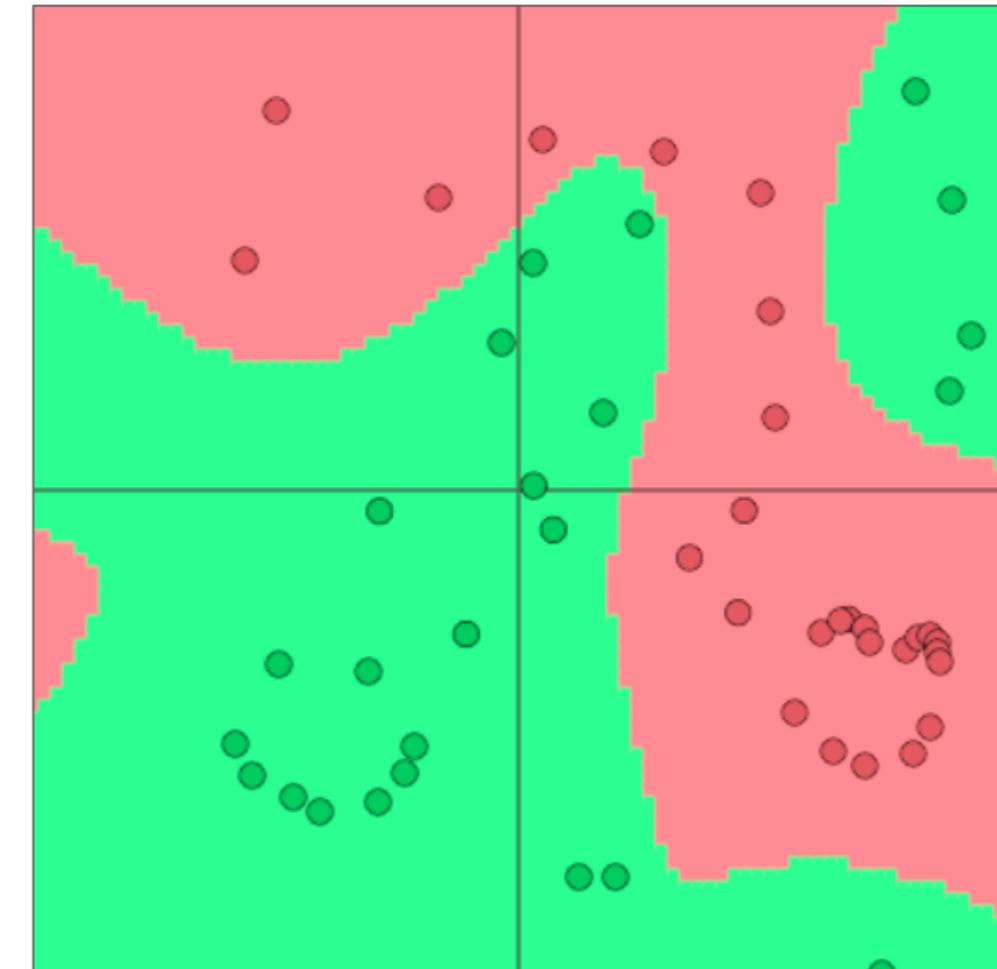
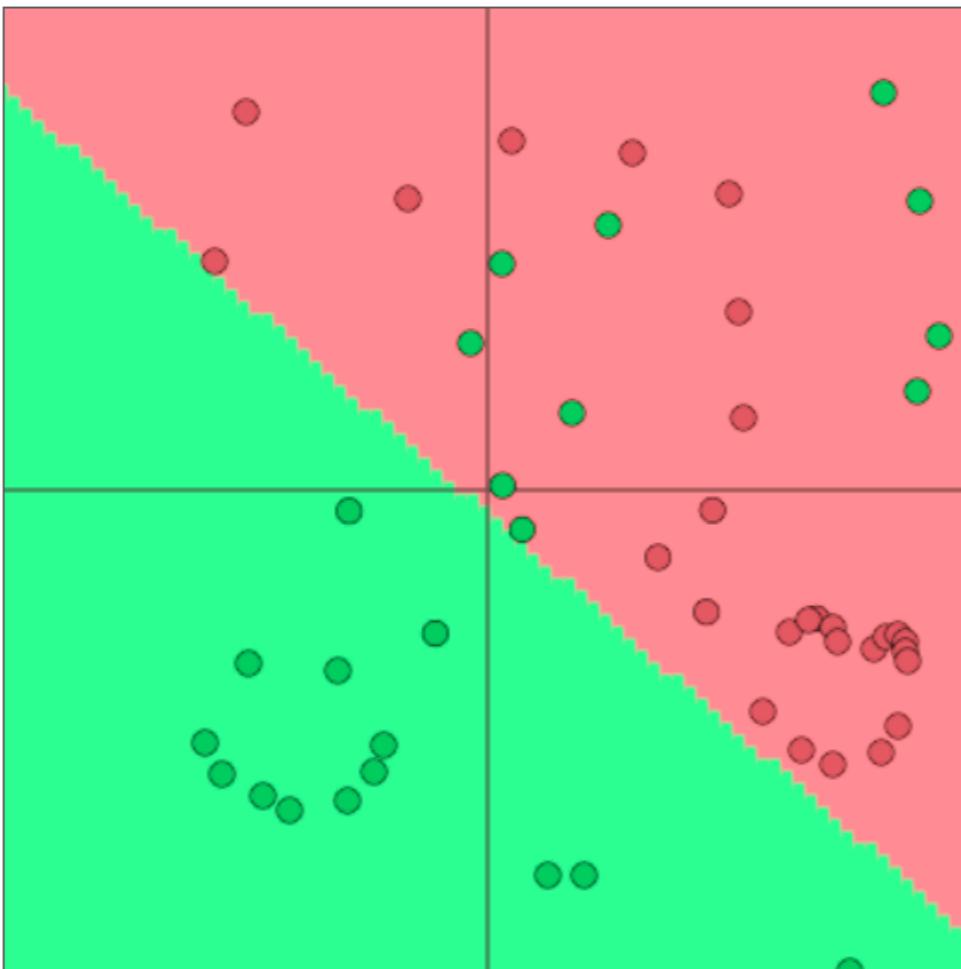
# Non-linearity

- Logistic (Softmax) Regression only gives linear decision boundaries

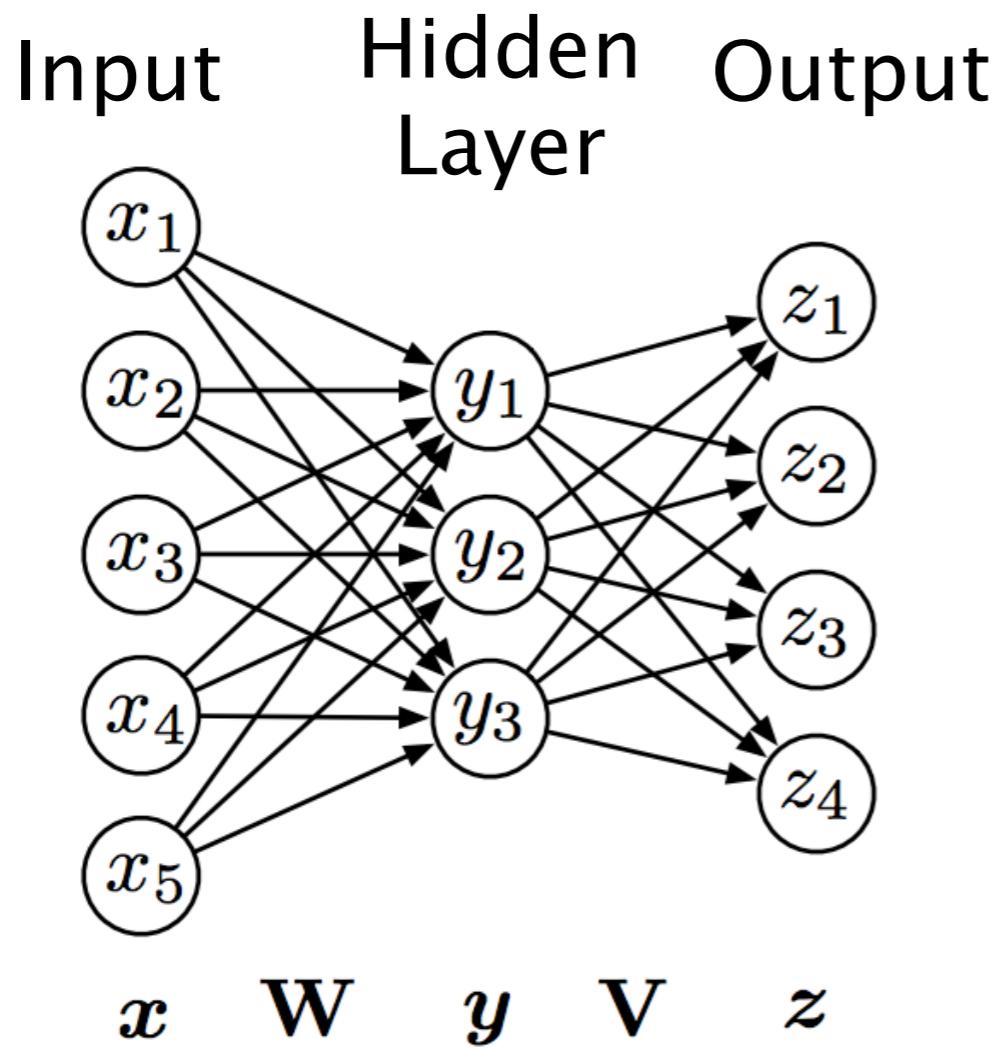


# Non-linearity

- Neural networks can learn much more complex functions and nonlinear decision boundaries!



# Non-linearity



$$\mathbf{y} = g(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = g(\mathbf{V}g(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{c})$$

output of first layer

With no nonlinearity:

$$\mathbf{z} = \mathbf{V}\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{b} + \mathbf{c}$$

Equivalent to  $\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{d}$

What about Word2vec  
(Skip-gram and CBOW)?

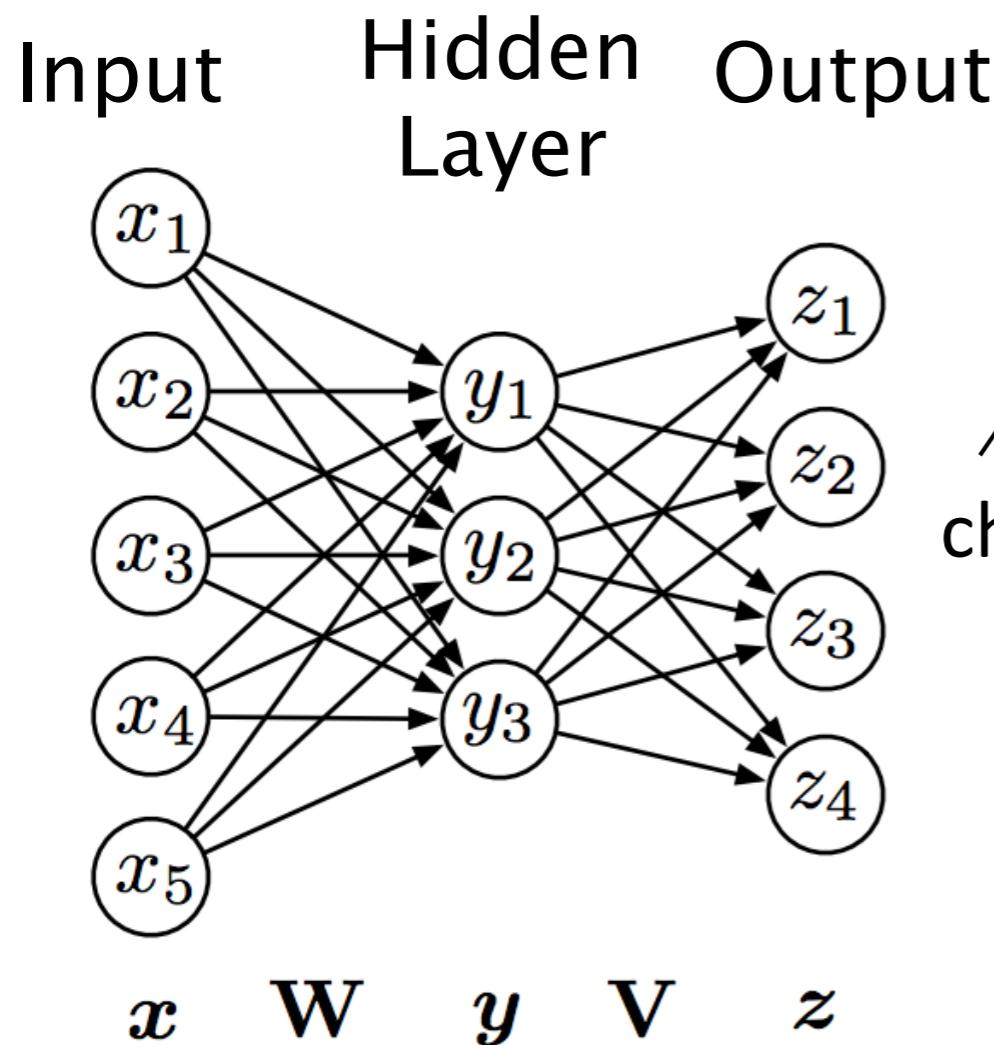
# So, what about Word2vec (Skip-gram and CBOW)?

It is not deep learning — but “shallow” neural networks.

It is — in fact — a log-linear model (softmax regression).

So, it is faster over larger dataset yielding better embeddings.

# Learning Neural Networks



$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

change in output w.r.t. hidden  
change in hidden w.r.t. input  
change in output w.r.t. input

Computing these looks like running this network in reverse (backpropagation)

# Strategy for Successful NNs

- Select network structure appropriate for problem
  - Structure: Single words, fixed windows, sentence based, document level; bag of words, recursive vs. recurrent, CNN, ...
  - Nonlinearity
- Check for implementation bugs with gradient checks
- Parameter initialization
- Optimization tricks
- Should get close to 100% accuracy/precision/recall/etc... on training data
  - Tune number of iterations on dev data

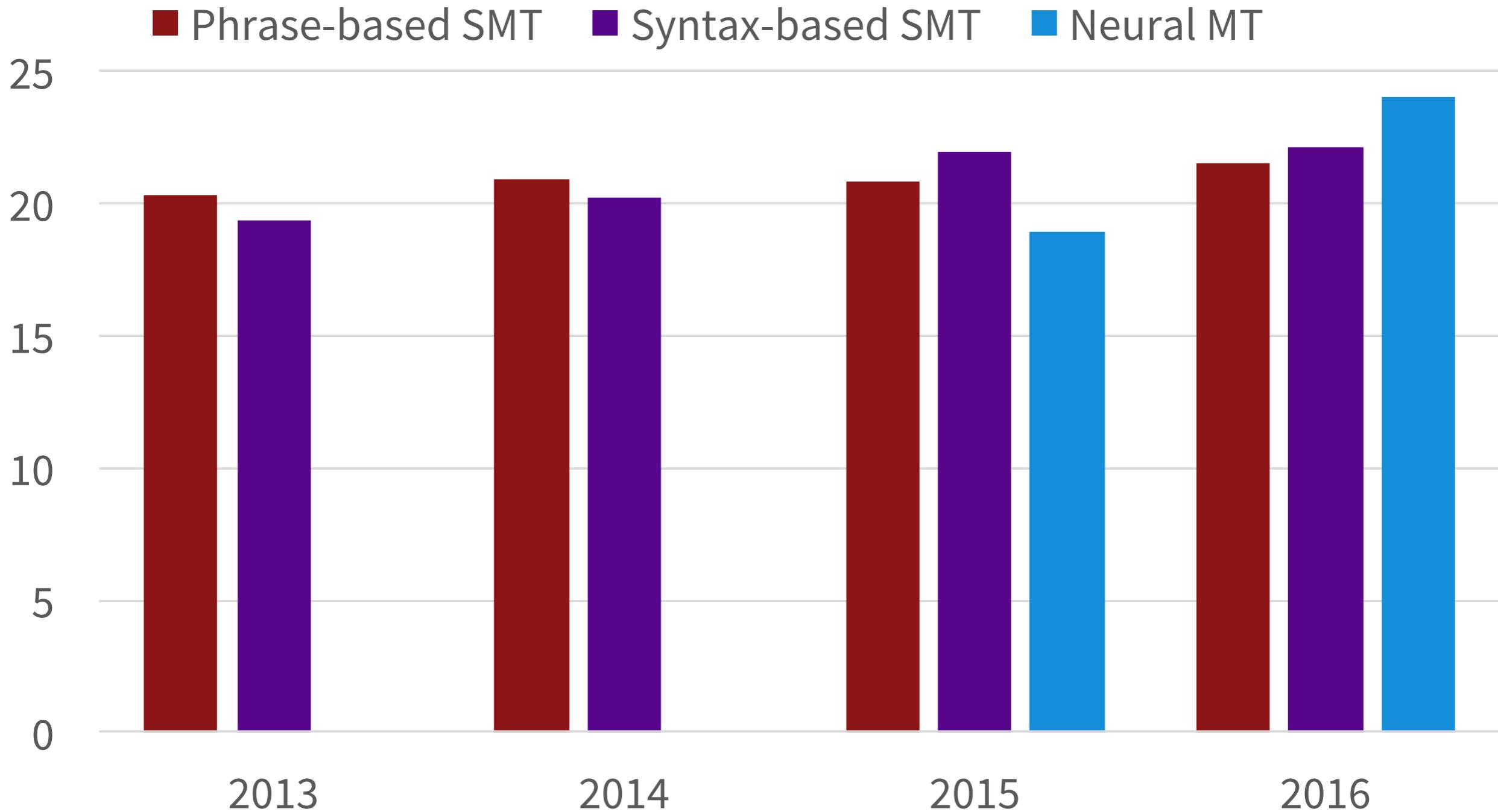
# Neural Machine Translation

Neural MT went from a fringe research activity in 2014 to the widely-adopted leading way to do MT in 2016.

Amazing!

# Progress in Machine Translation

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



From [Sennrich 2016, [http://www.meta-net.eu/events/meta-forum-2016/slides/09\\_sennrich.pdf](http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf)]

# What is Neural MT (NMT)?

Neural Machine Translation is the approach of modeling the entire MT process via one big artificial neural network\*

\*But sometimes we compromise this goal a little

# The three big wins of Neural MT

## 1. End-to-end training

All parameters are simultaneously optimized to minimize a loss function on the network's output

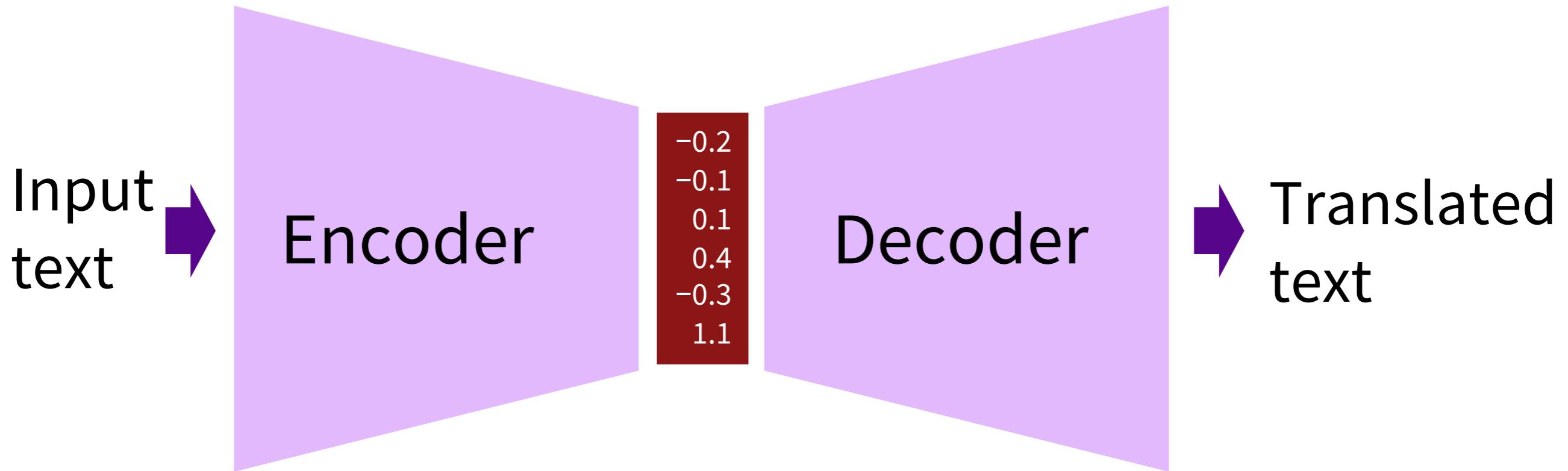
## 2. Distributed representations share strength

Better exploitation of word and phrase similarities

## 3. Better exploitation of context

NMT can use a much bigger context – both source and partial target text – to translate more accurately

# Neural encoder-decoder architectures



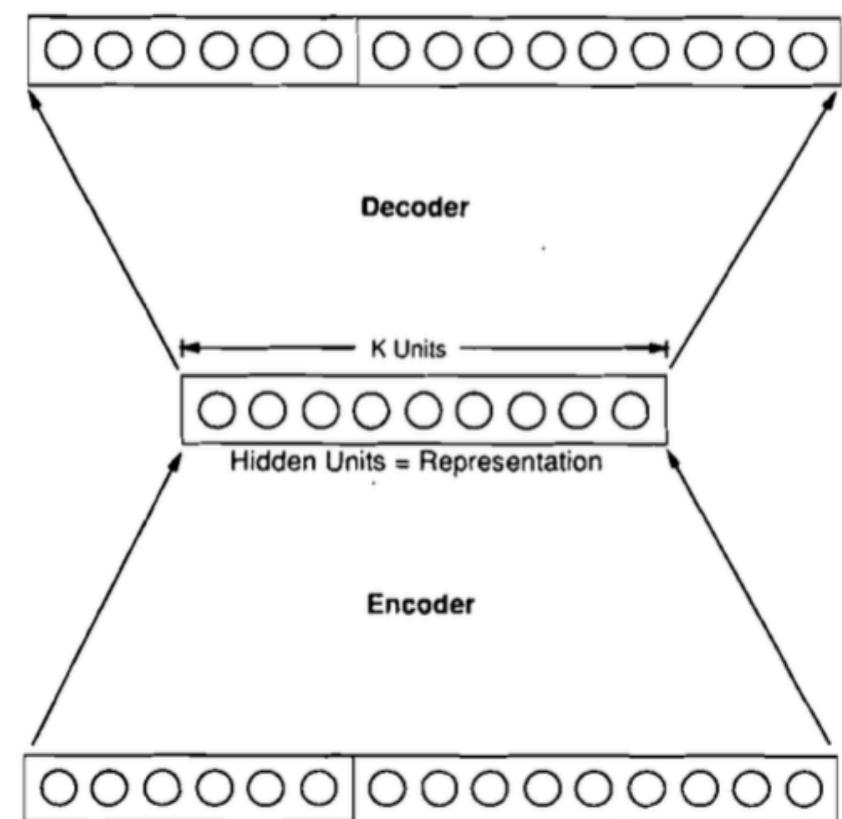
# Neural MT: The Bronze Age

[Allen 1987 IEEE 1<sup>st</sup> ICNN]

3310 En-Es pairs constructed on 31  
En, 40 Es words, max 10/11 word  
sentence; 33 used as test set

The grandfather offered the little girl a book →  
El abuelo le ofrecio un libro a la nina pequena

Binary encoding of words – 50  
inputs, 66 outputs; 1 or 3 hidden  
150-unit layers. Ave WER: 1.3 words



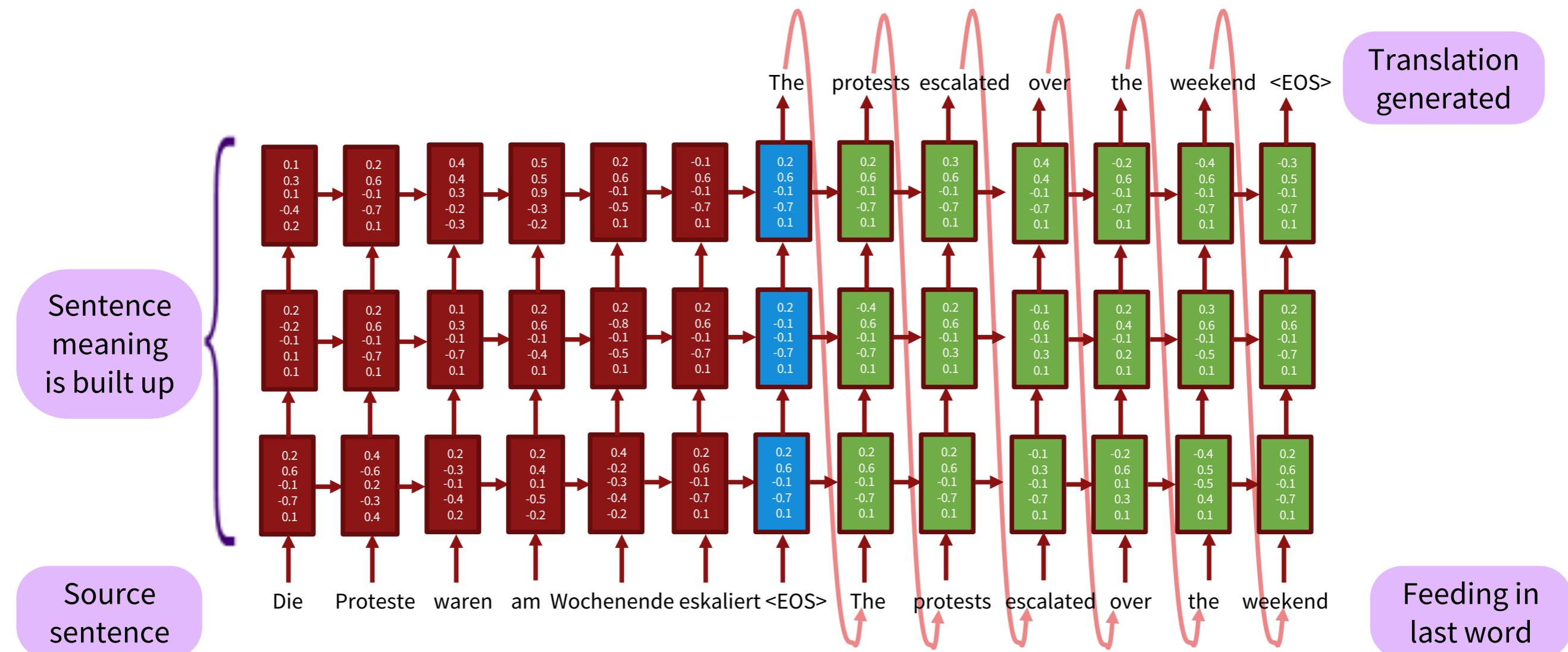
# **Several Studies on Natural Language and Back-Propagation**

**Robert B. Allen**  
**Bell Communications Research**  
**Morristown, NJ 07960 U.S.A.**  
**rba@bellcore.com**

Recent developments in neural algorithms provide a new approach to natural language processing. Two sets of brief studies show how networks may be developed for processing simple demonstratives and analogies. Two longer studies consider pronoun reference and natural language translation. Taken together, the studies provide additional support for the applicability of these algorithms to natural language processing.

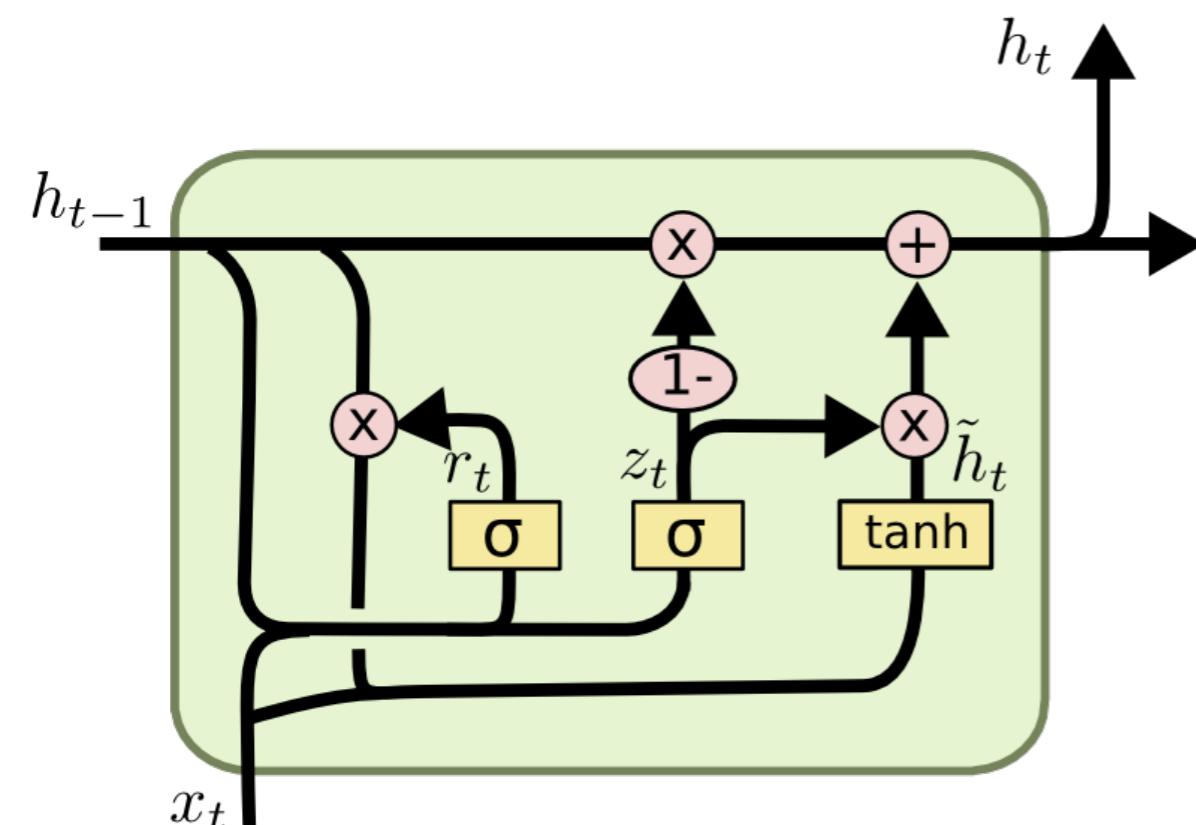
# Modern Sequence Models for NMT

[Sutskever et al. 2014, Bahdanau et al. 2014, et seq.]  
following [Jordan 1986] and more closely [Elman 1990]



A deep recurrent neural network

# Long Short-Term Memory Networks (LSTM)

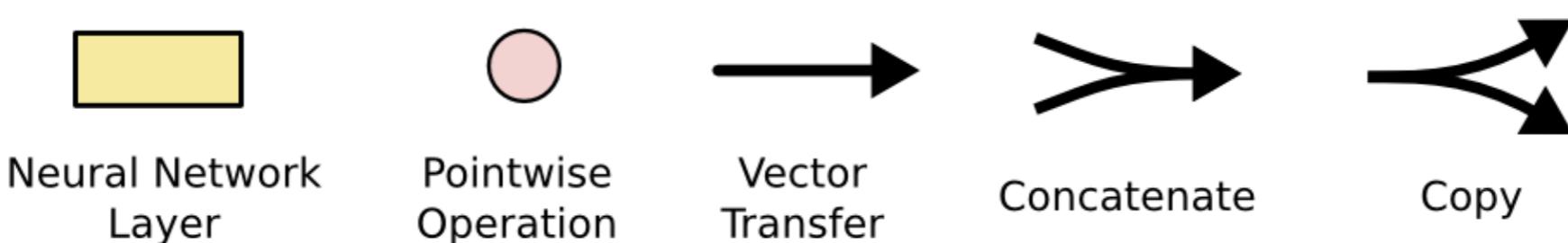


$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

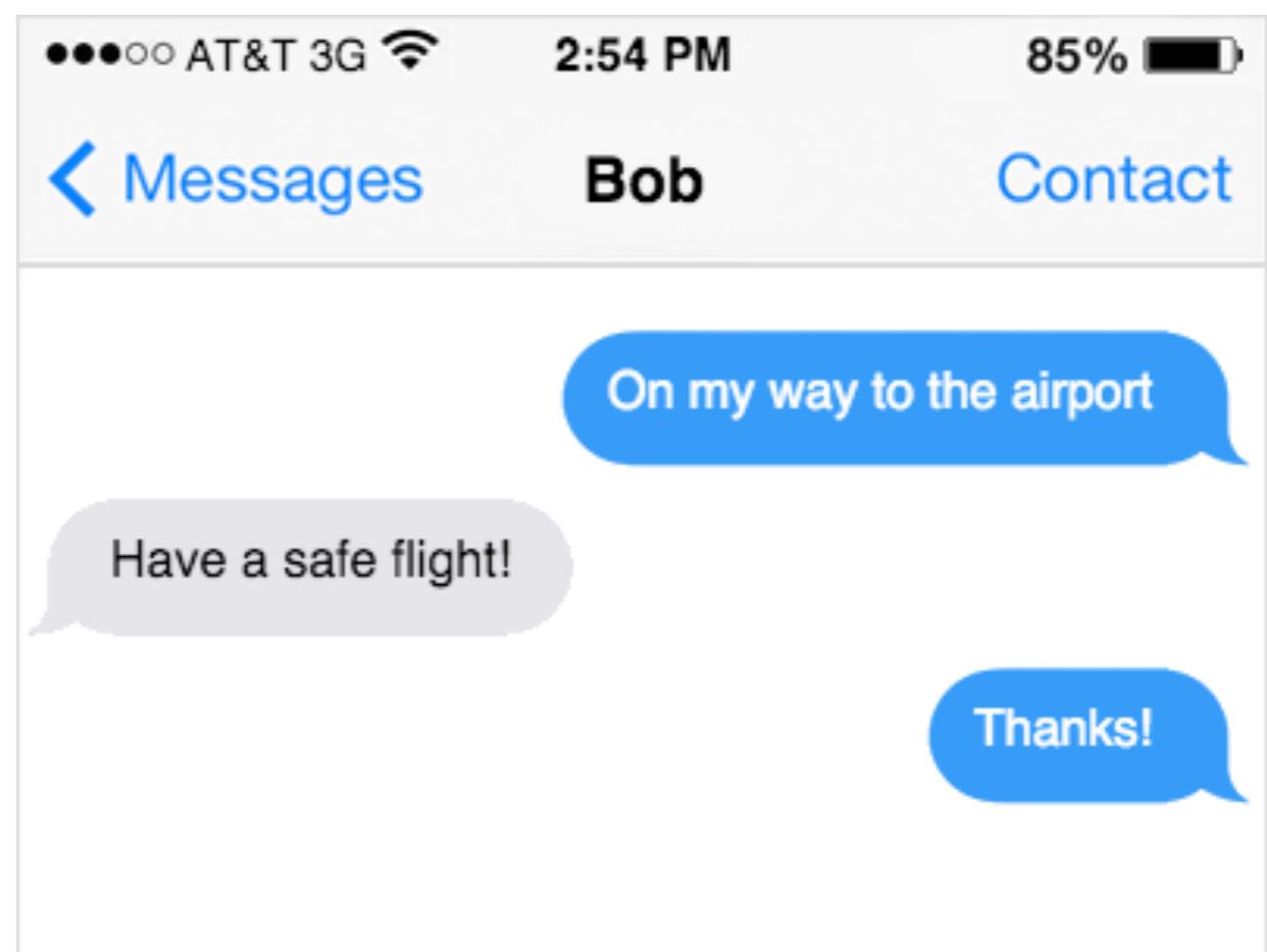
$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

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



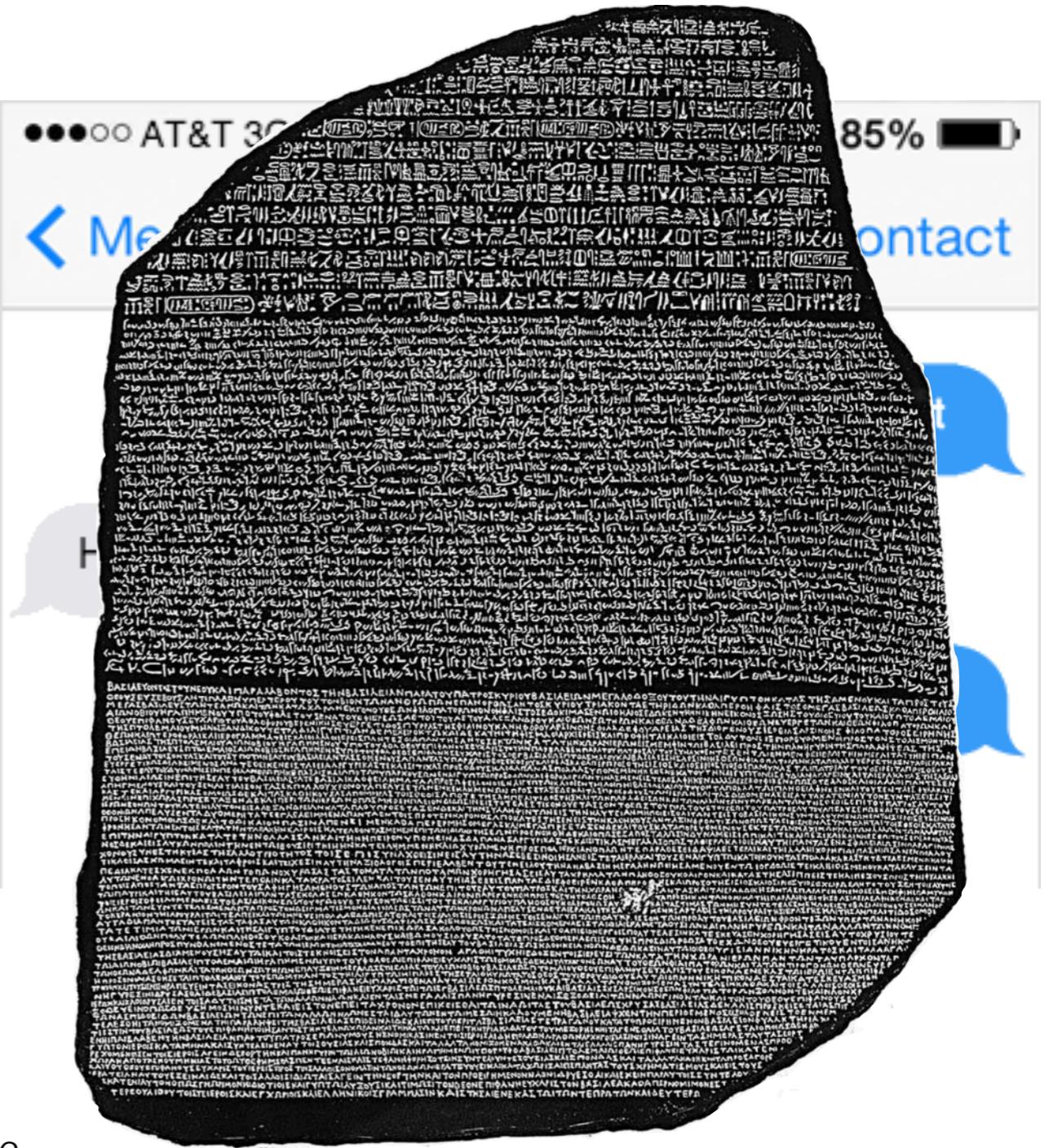
# Data-Driven Conversation

- **Twitter:** ~ 500 Million Public SMS-Style Conversations ***per Month***
- **Goal:** Learn conversational agents directly from massive volumes of data.



# Data-Driven Conversation

- **Twitter:** ~ 500 Million Public SMS-Style Conversations *per Month*
- **Goal:** Learn conversational agents directly from massive volumes of data.



# Noisy Channel Model

Input:

**Who wants to come over for dinner tomorrow?**

# Noisy Channel Model

Input:

**Who wants to come over for dinner tomorrow?**



Output:

Yum ! I



# Noisy Channel Model

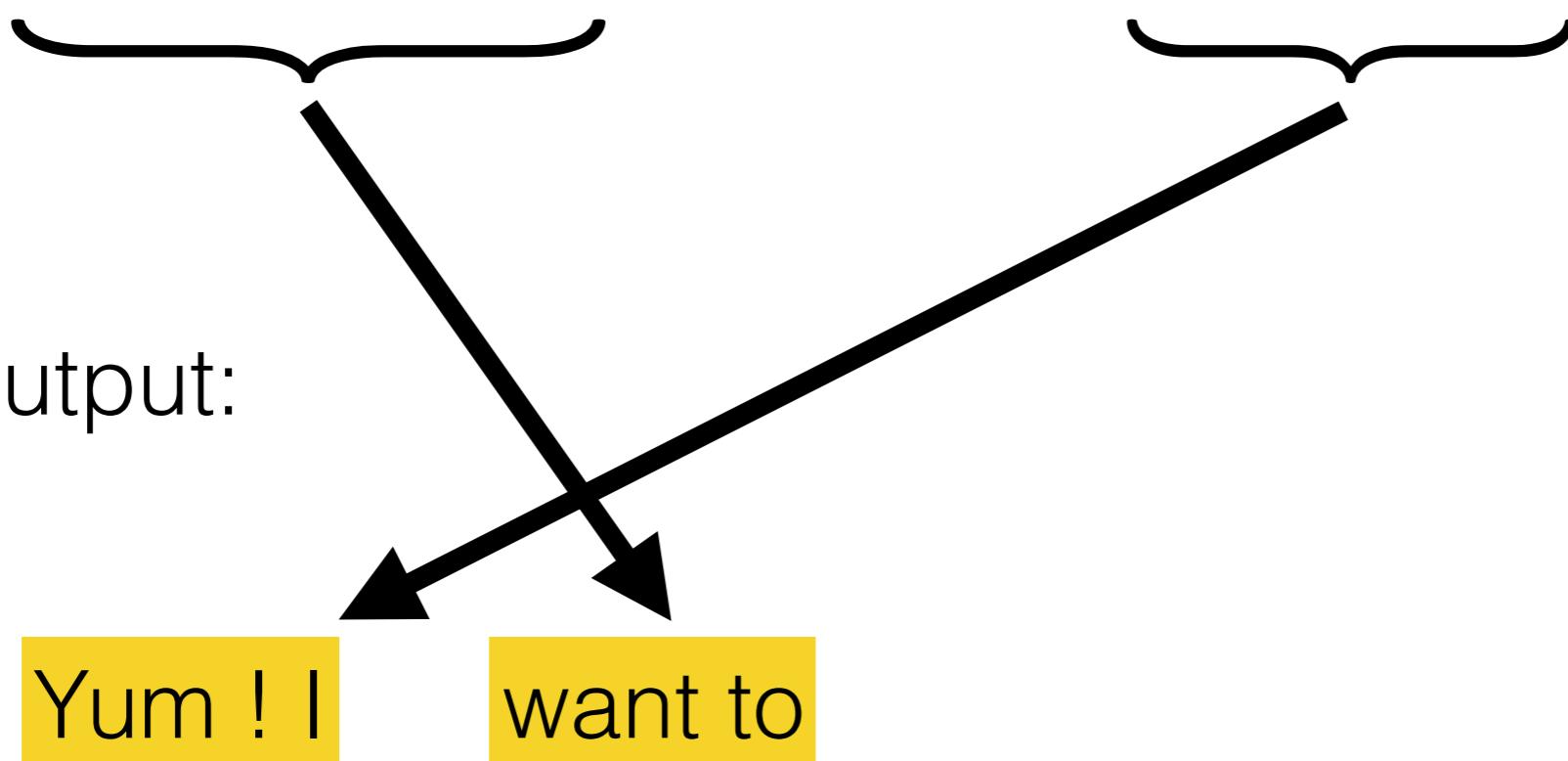
Input:

**Who wants to come over for dinner tomorrow?**

Output:

Yum ! I

want to



# Noisy Channel Model

Input:

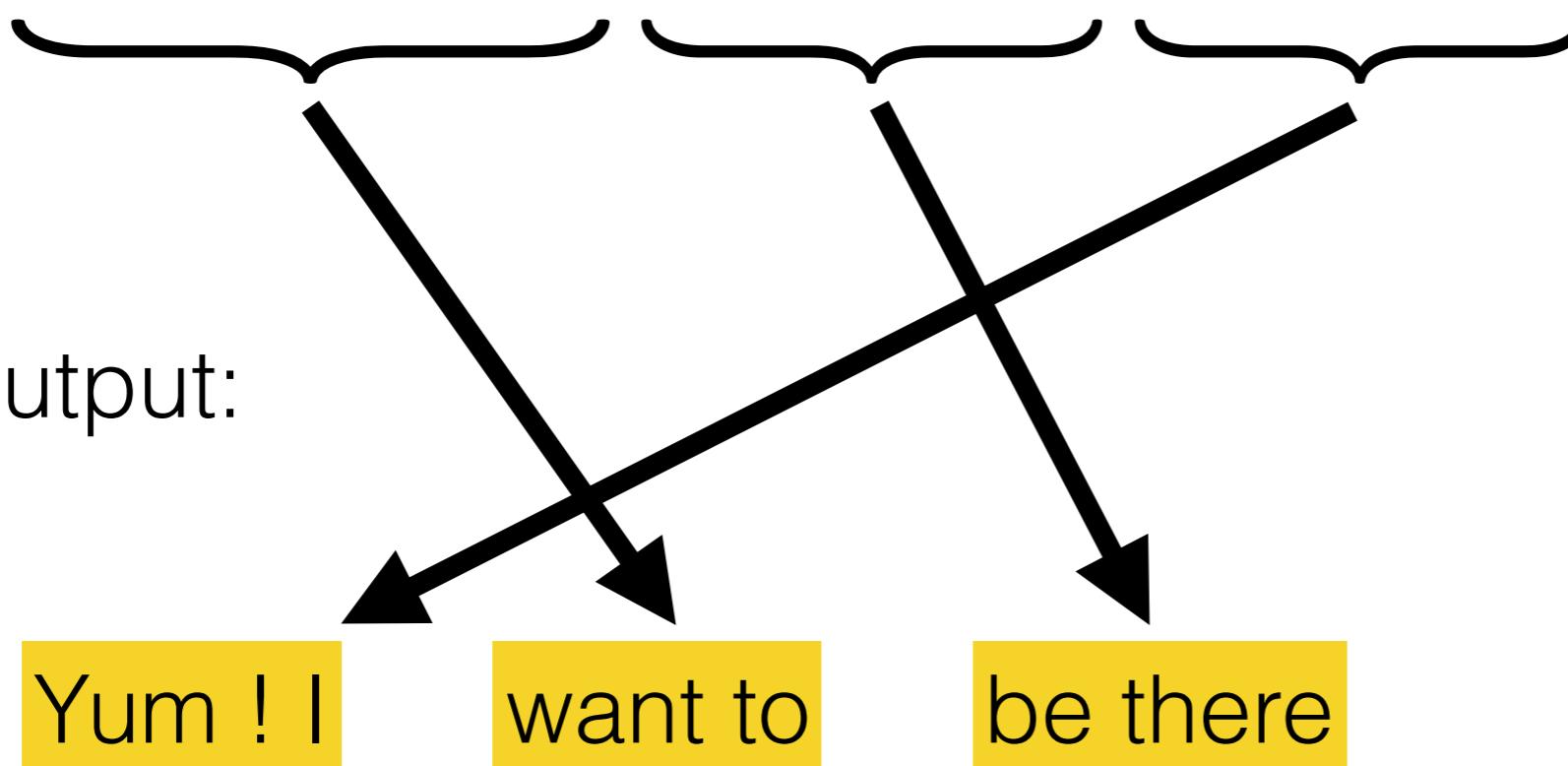
**Who wants to come over for dinner tomorrow?**

Output:

Yum ! I

want to

be there

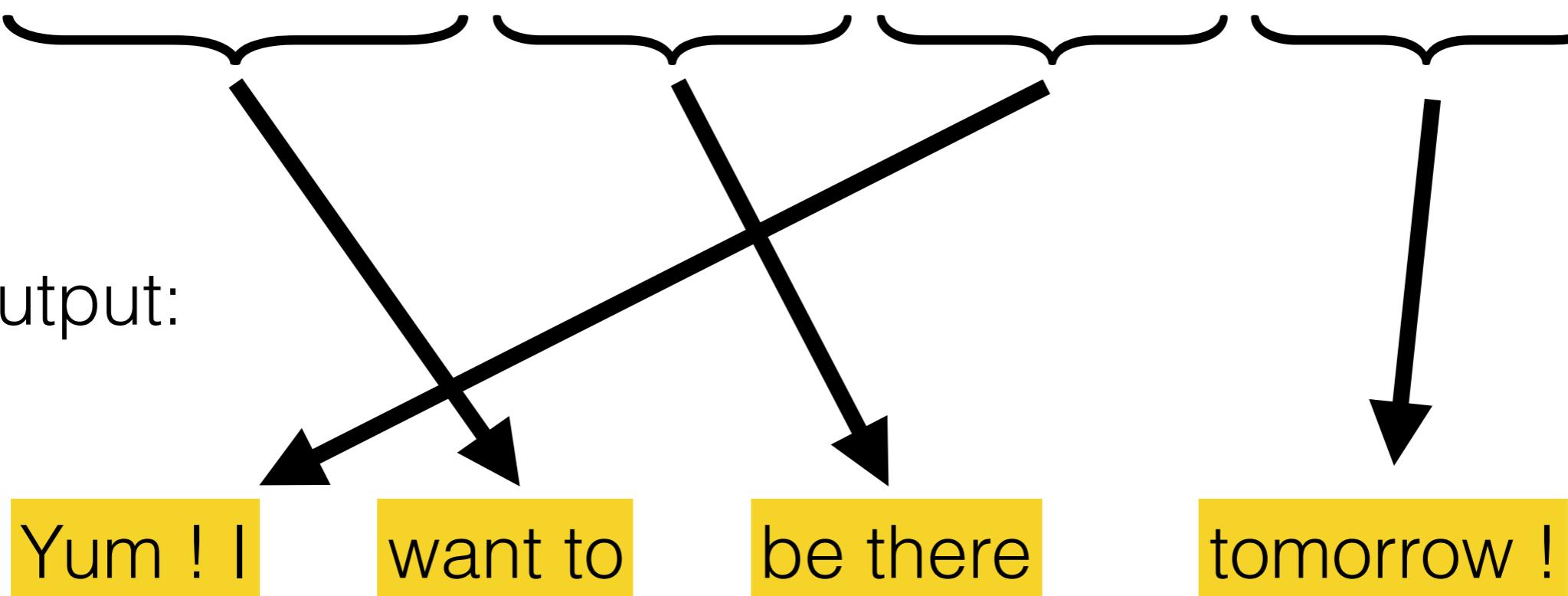


# Noisy Channel Model

Input:

**Who wants to come over for dinner tomorrow?**

Output:



# Neural Conversation

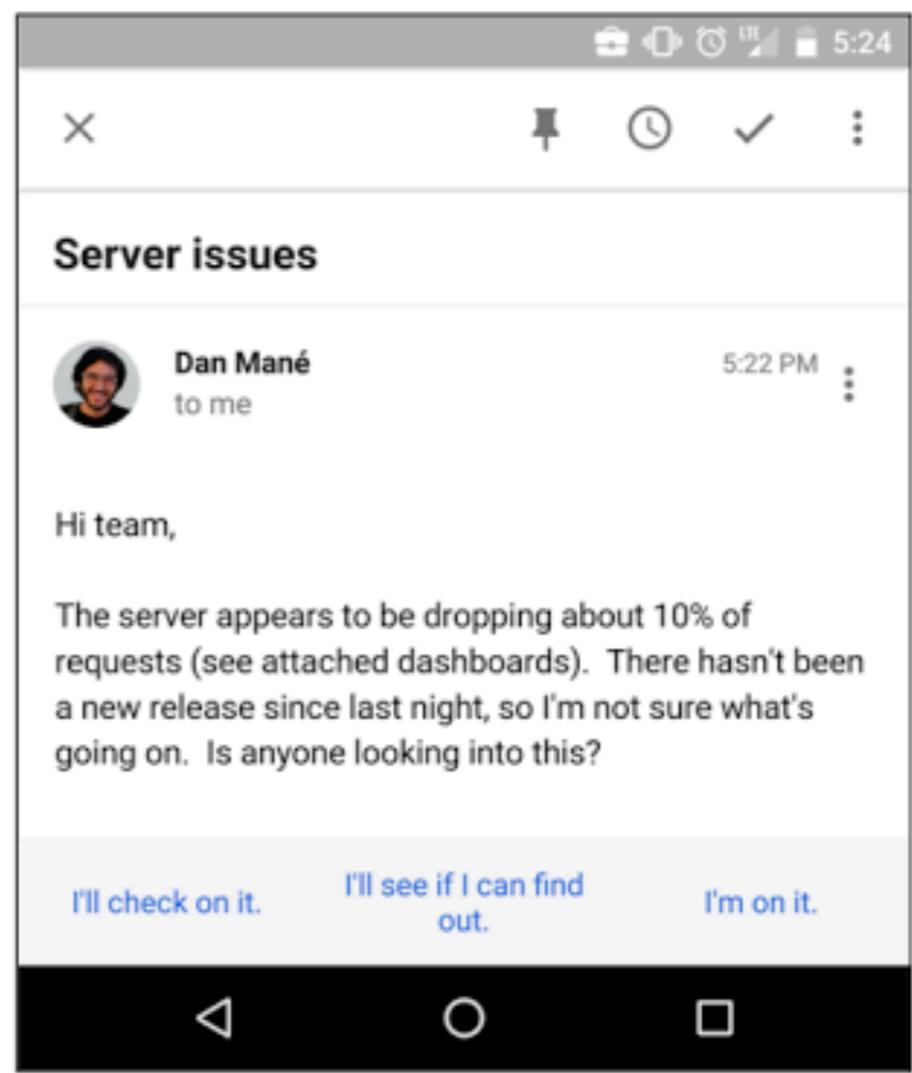


## Google Research Blog

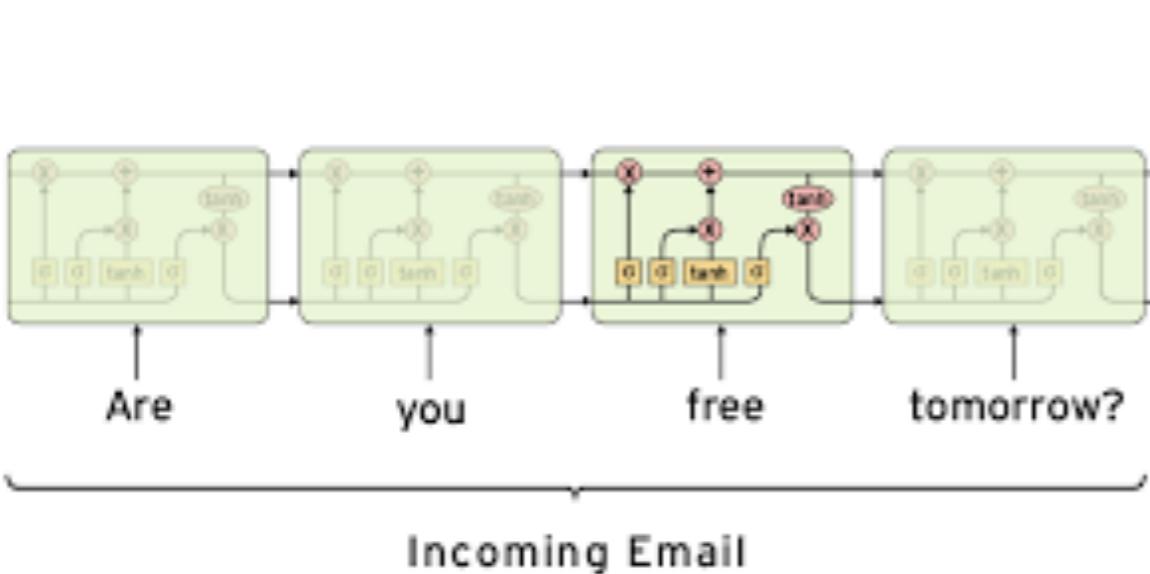
Computer, respond to this email.

Tuesday, November 03, 2015

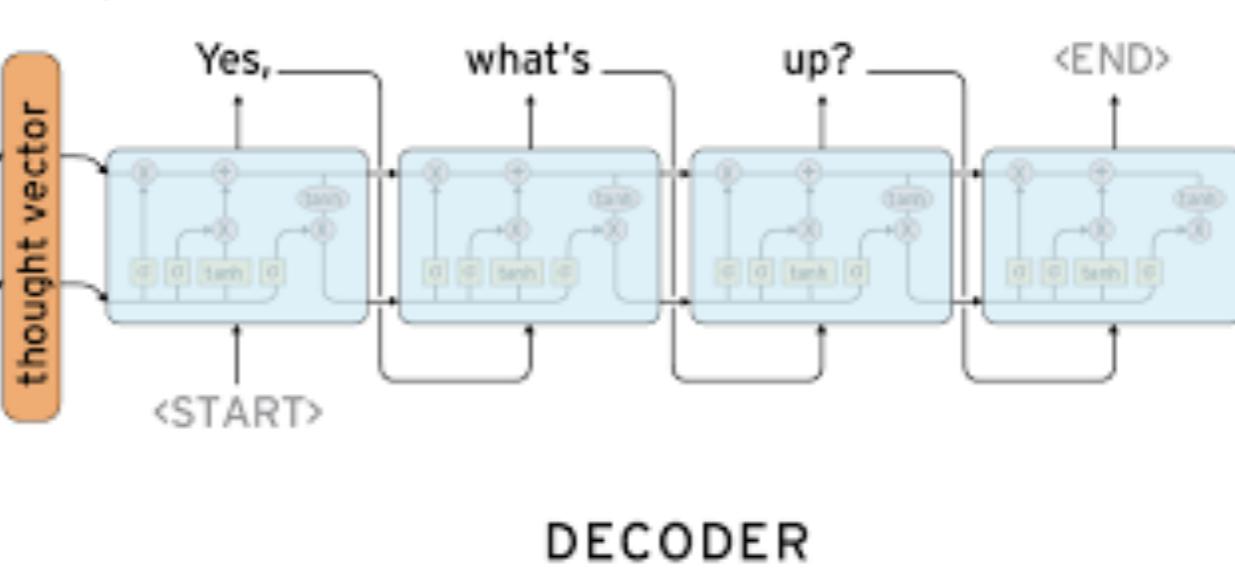
Posted by Greg Corrado\*, Senior Research Scientist



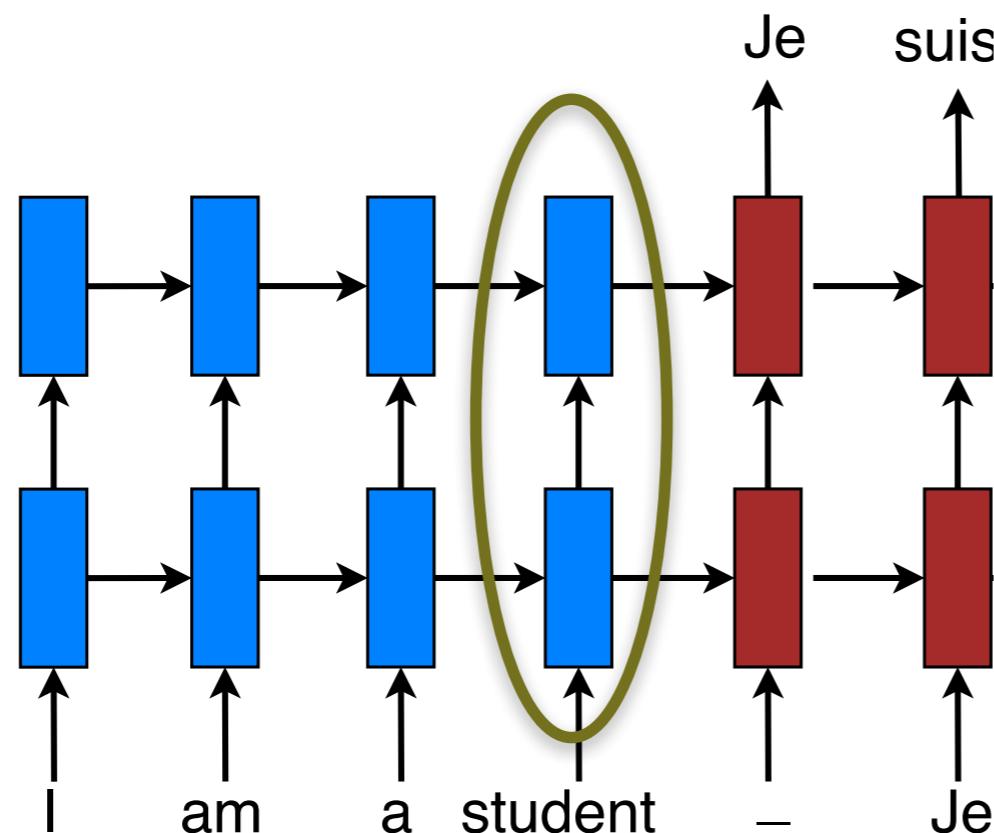
ENCODER



Reply



# Vanilla seq2seq & long sentences

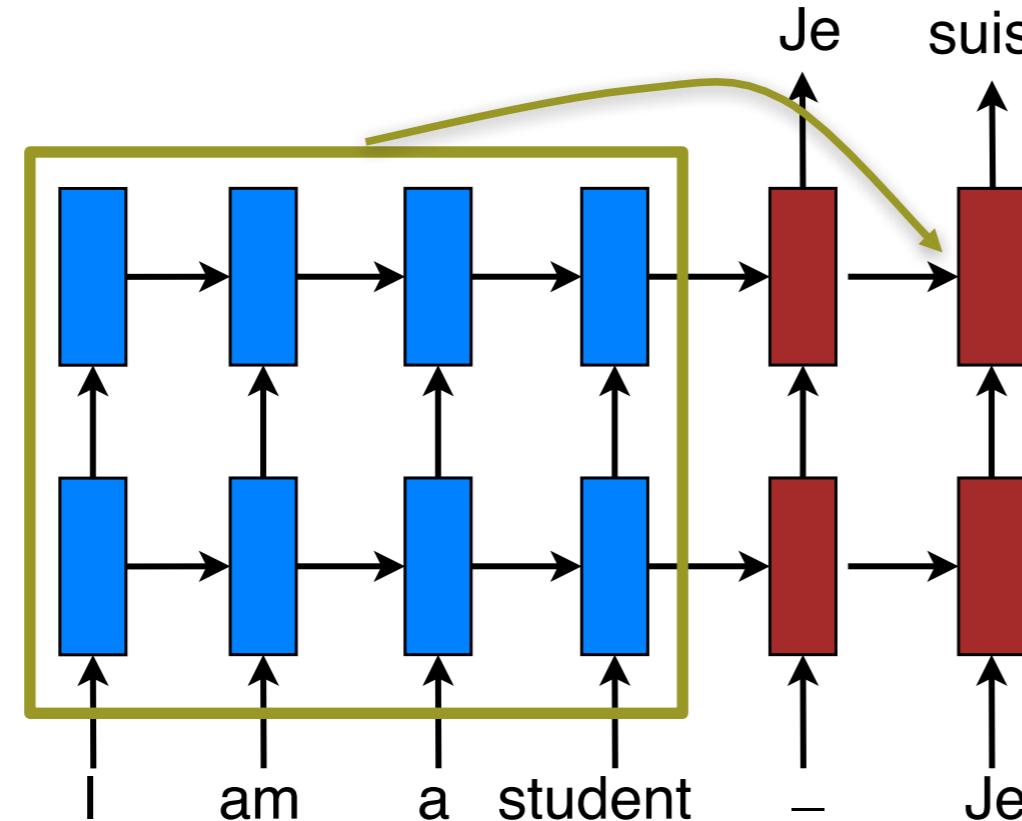


Problem: fixed-dimensional representations

# Attention Mechanism

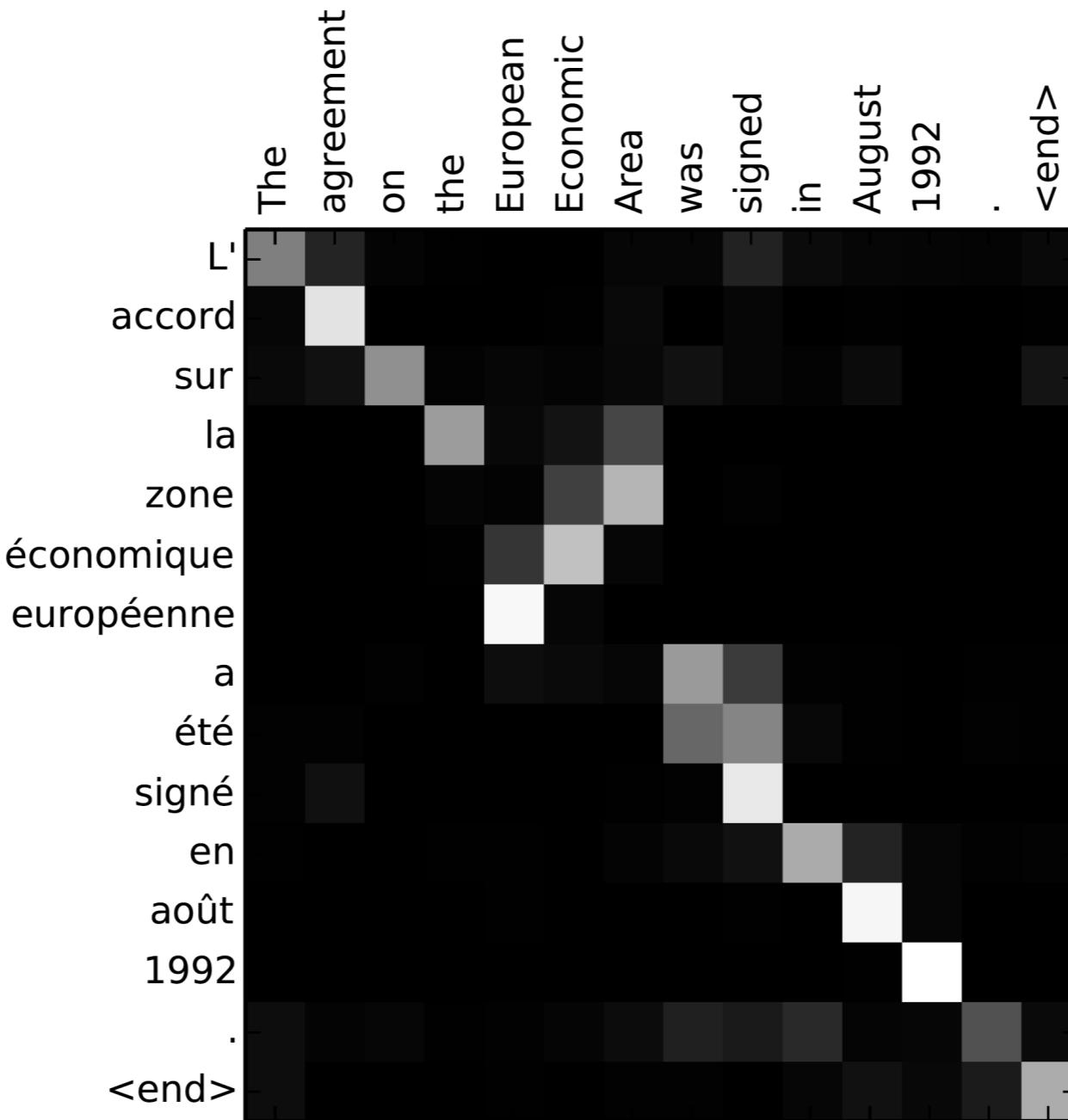
Started in computer vision!  
[Larochelle & Hinton, 2010],  
[Denil, Bazzani, Larochelle,  
Freitas, 2012]

Pool of  
source  
states



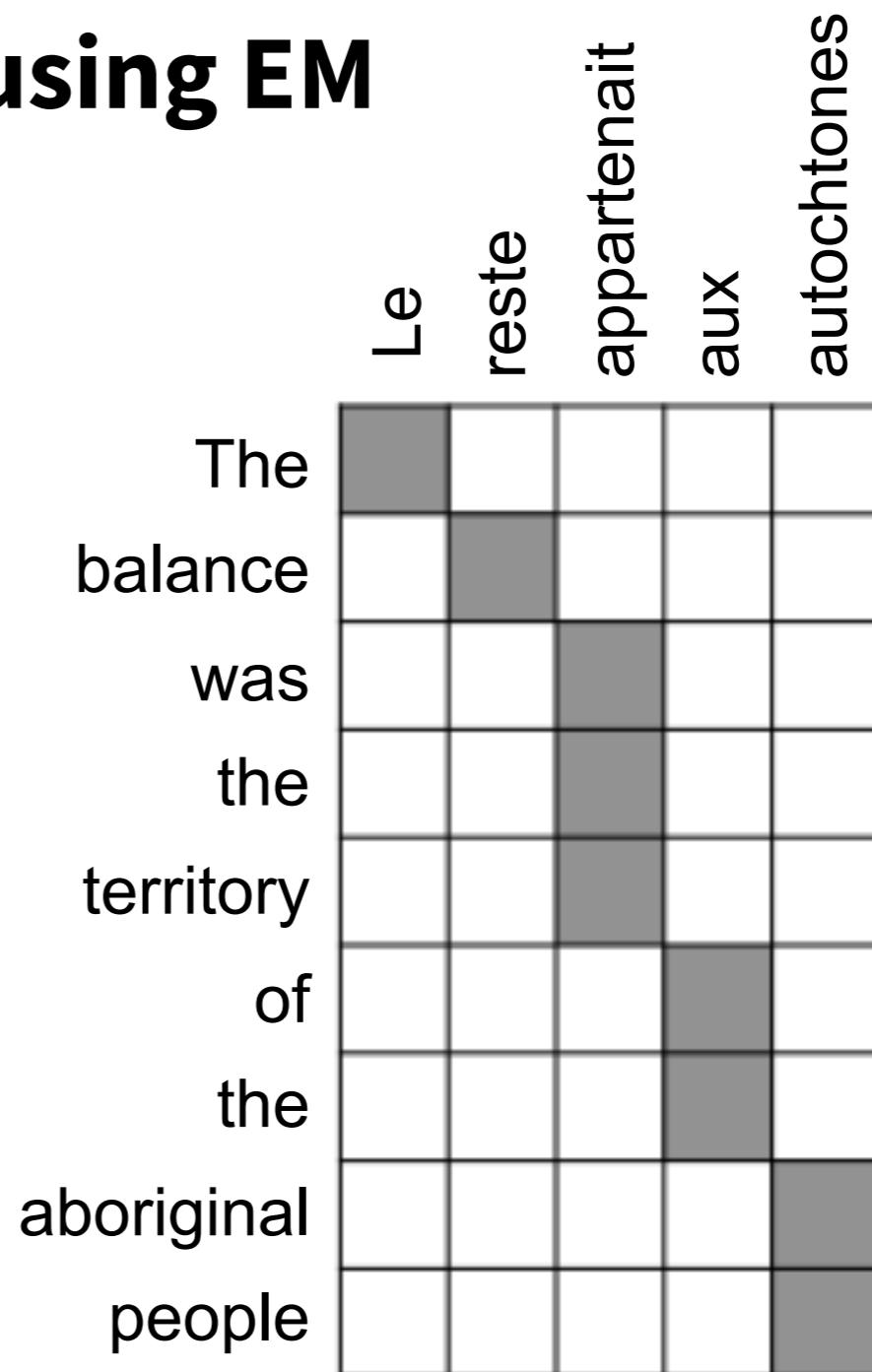
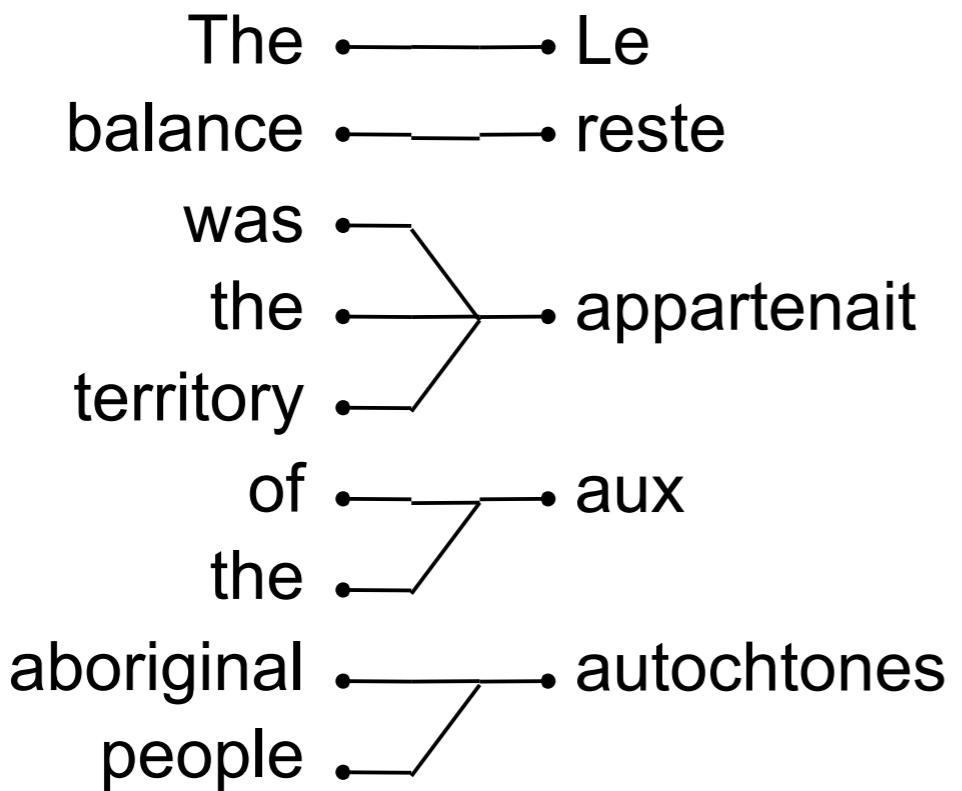
- **Solution:** random access memory
  - Retrieve as needed.

# Learning both translation & alignment



# Word alignments

Phrase-based SMT aligned words in a preprocessing-step, usually using EM

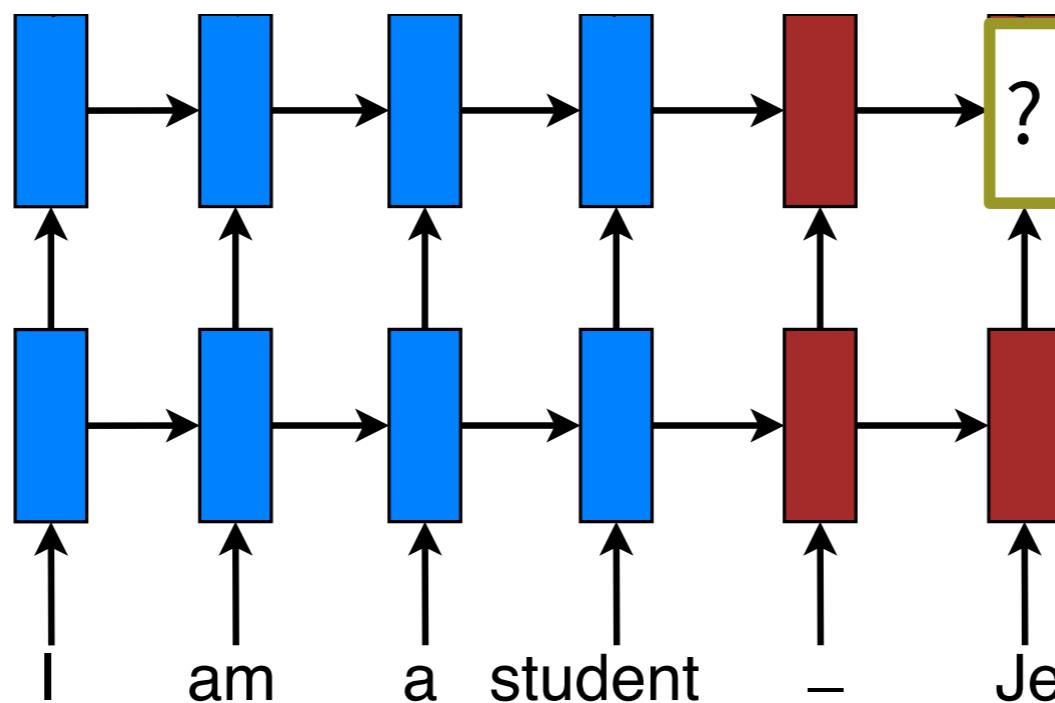


→ Models of attention

[Bahdanau et al. 2014; ICLR 2015]

Part 3b later

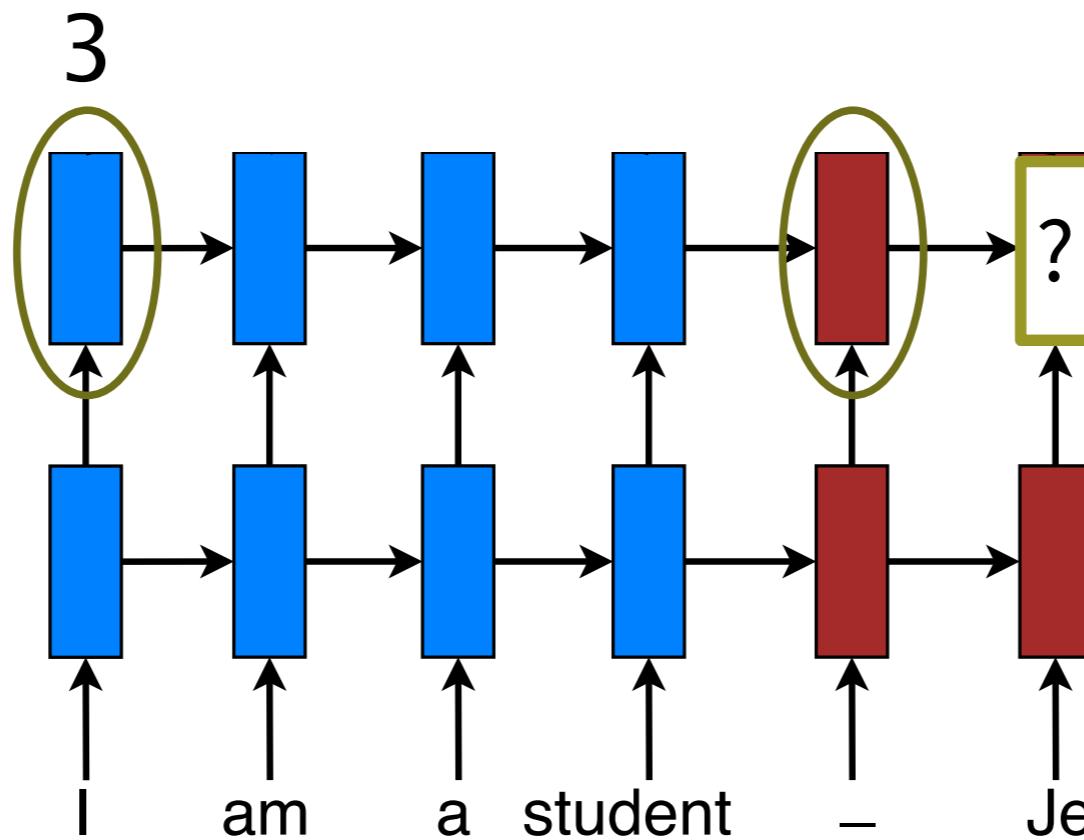
# Attention Mechanism



Simplified version of (Bahdanau et al., 2015)

# Attention Mechanism - Scoring

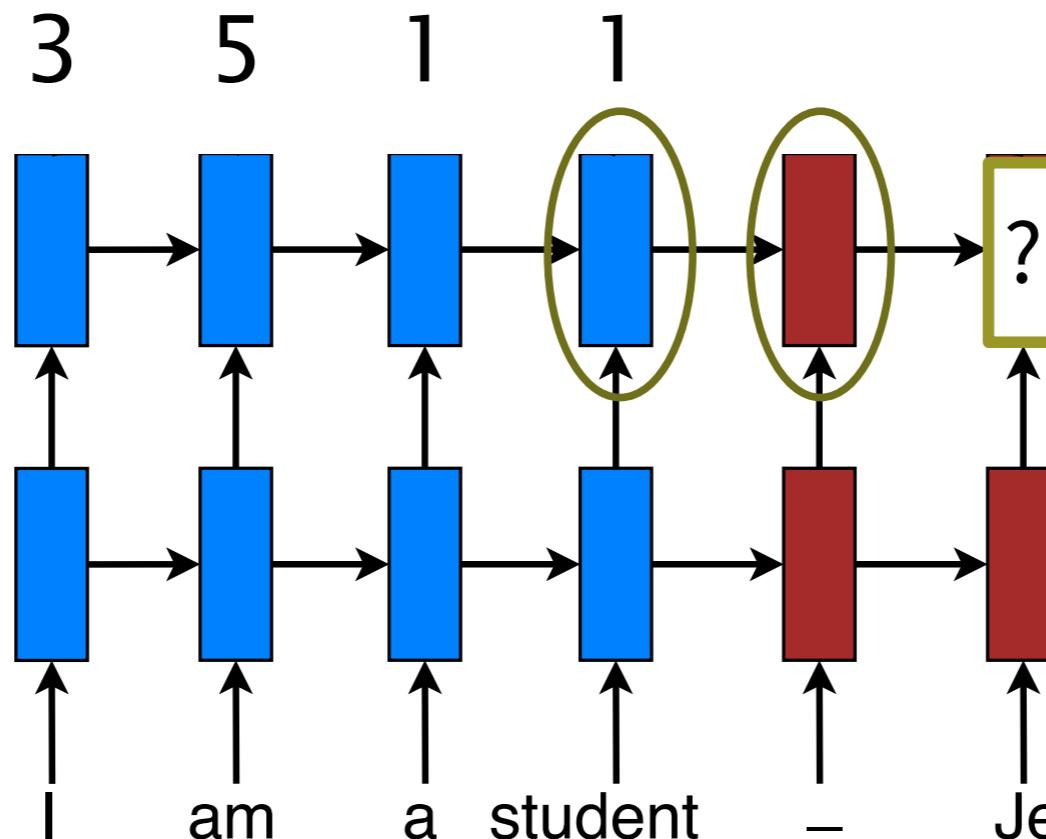
$$\text{score}(\mathbf{h}_{t-1}, \bar{\mathbf{h}}_s)$$



- Compare target and source hidden states.

# Attention Mechanism - Scoring

$$\text{score}(h_{t-1}, \bar{h}_s)$$

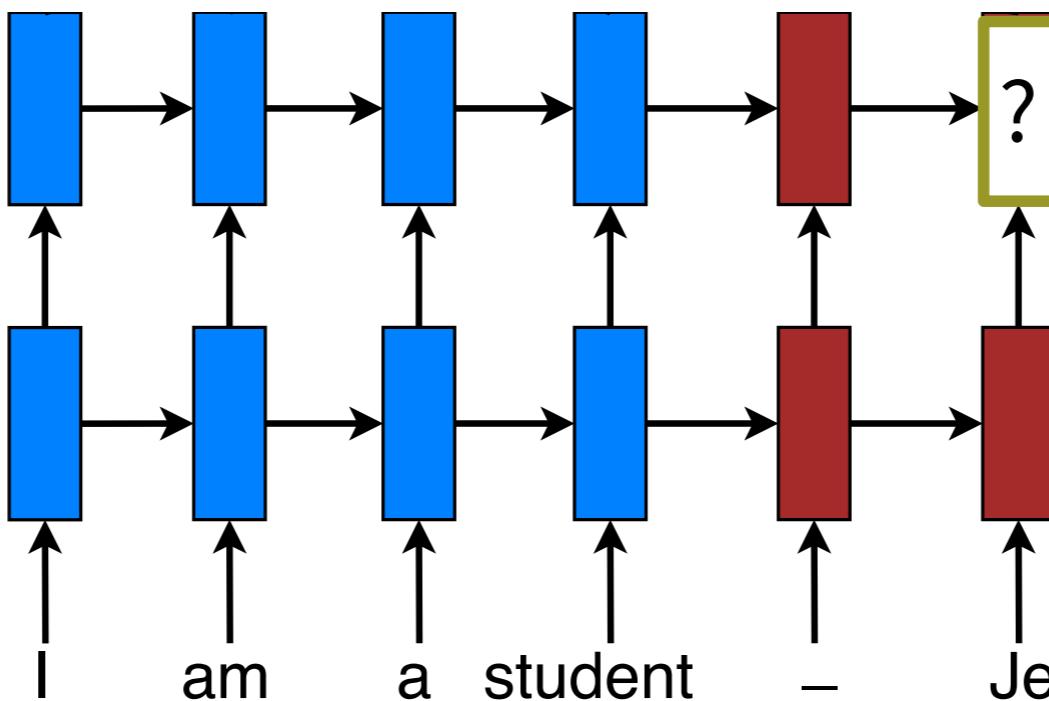


- Compare target and source hidden states.

# Attention Mechanism – Normalization

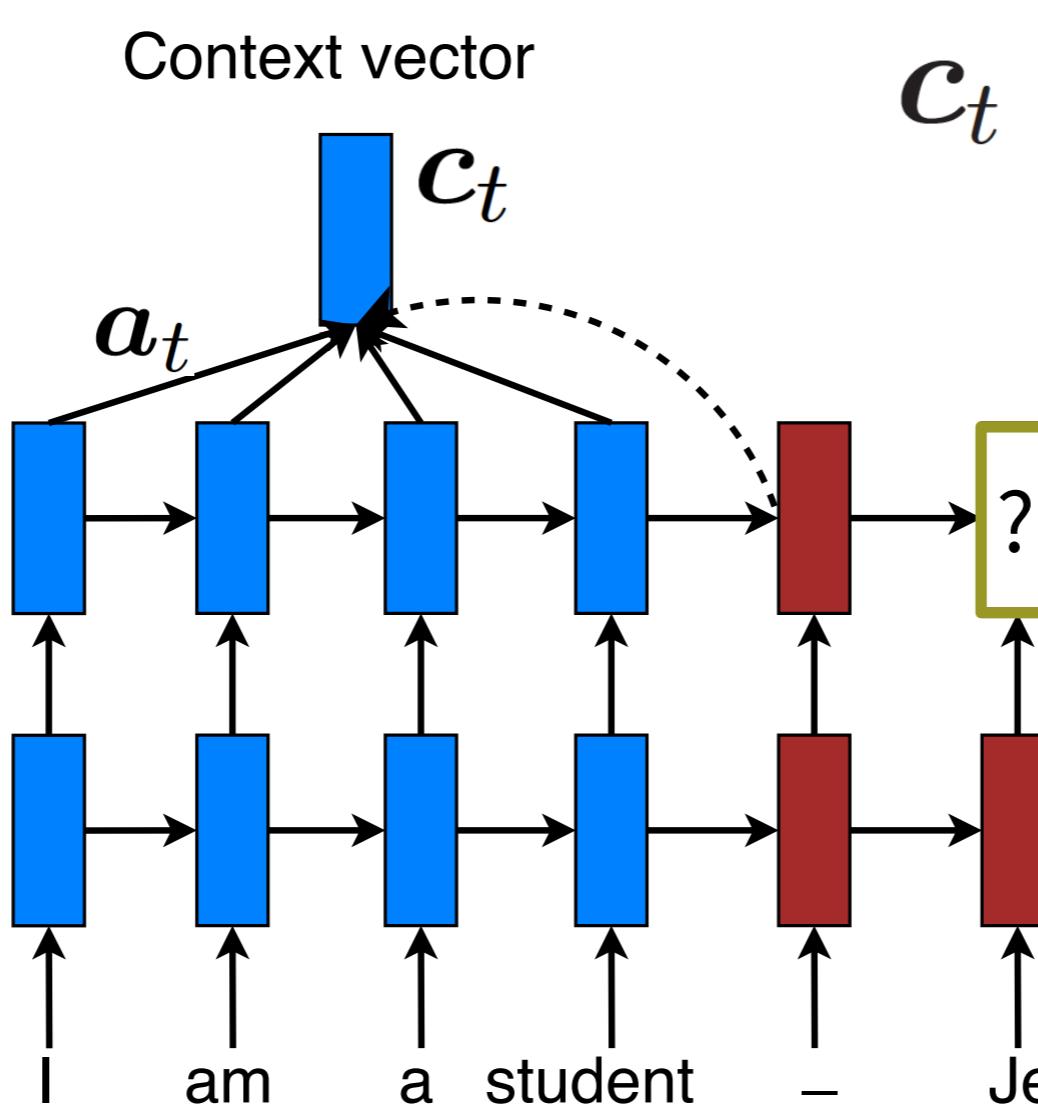
$$a_t(s) = \frac{e^{\text{score}(s)}}{\sum_{s'} e^{\text{score}(s')}}$$

$a_t$  0.3 0.5 0.1 0.1



- Convert into alignment weights.

# Attention Mechanism - Context

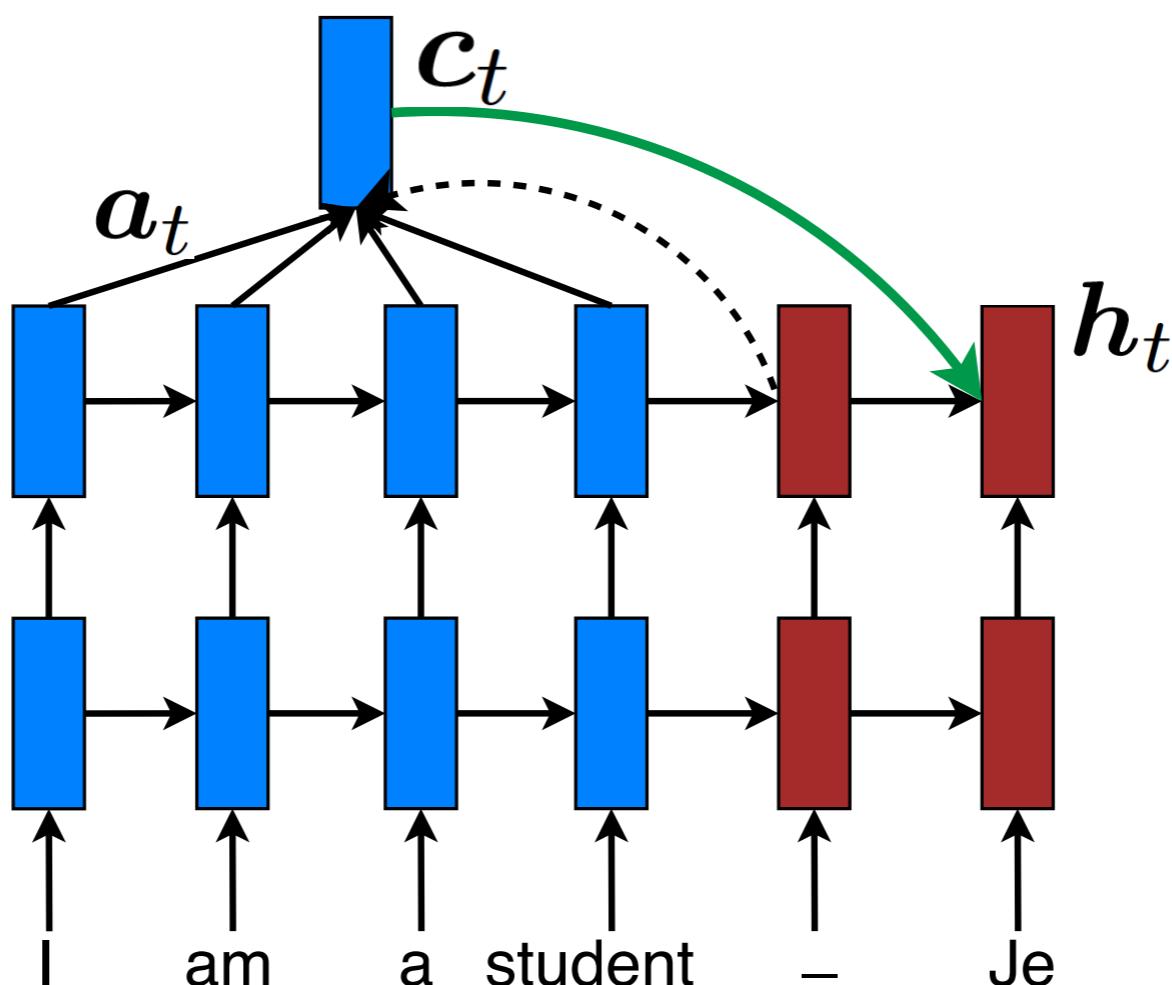


$$c_t = \sum_s a_t(s) \bar{h}_s$$

- Build context vector: weighted average.

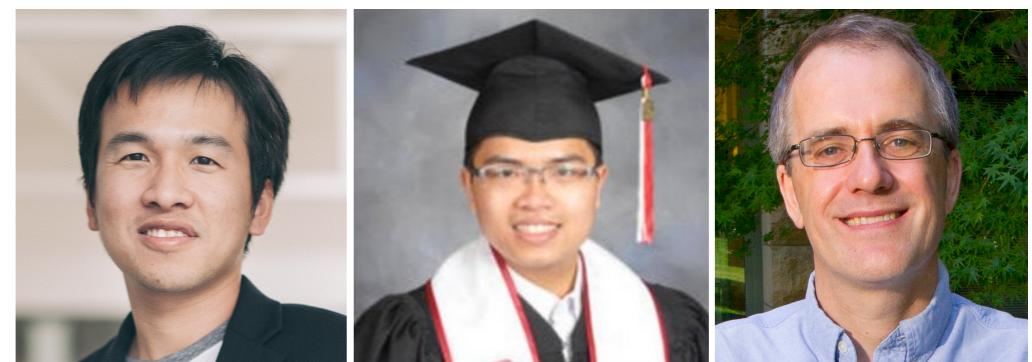
# Attention Mechanism - *Hidden State*

Context vector

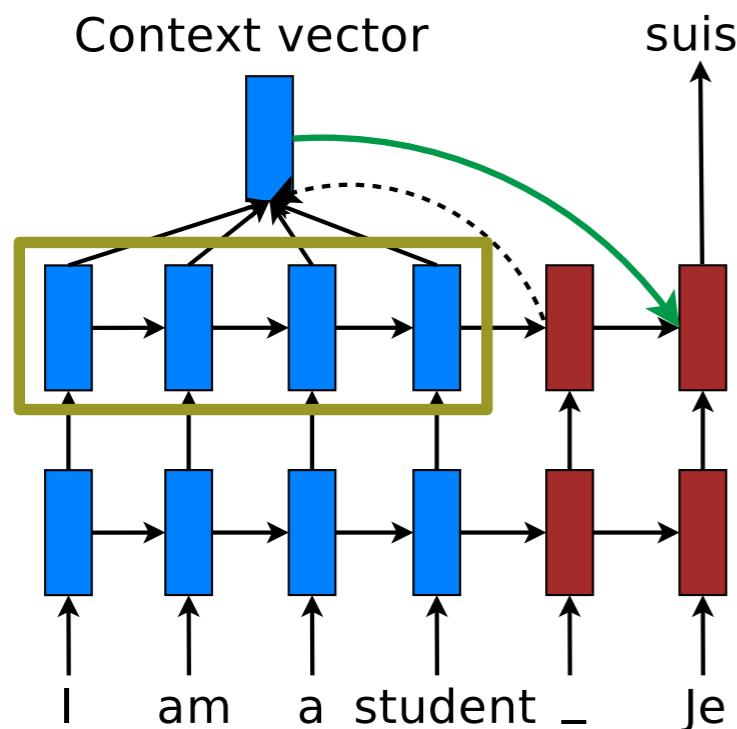


- Compute the next hidden state.

# Attention Mechanisms+



- Simplified mechanism & more functions:



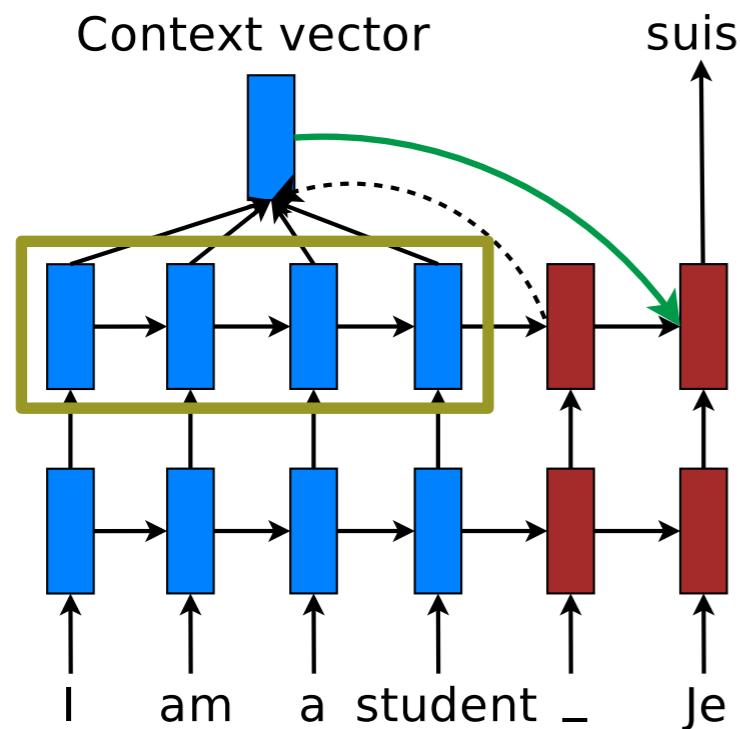
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \bar{\mathbf{h}}_s \\ \mathbf{h}_t^\top \mathbf{W}_a \bar{\mathbf{h}}_s \\ \mathbf{v}_a^\top \tanh (\mathbf{W}_a [\mathbf{h}_t; \bar{\mathbf{h}}_s]) \end{cases}$$

*Thang Luong, Hieu Pham, and Chris Manning. Effective Approaches to Attention-based Neural Machine Translation. EMNLP'15.*

# Attention Mechanisms+



- Simplified mechanism & more functions:



Bilinear form:  
well-adopted.

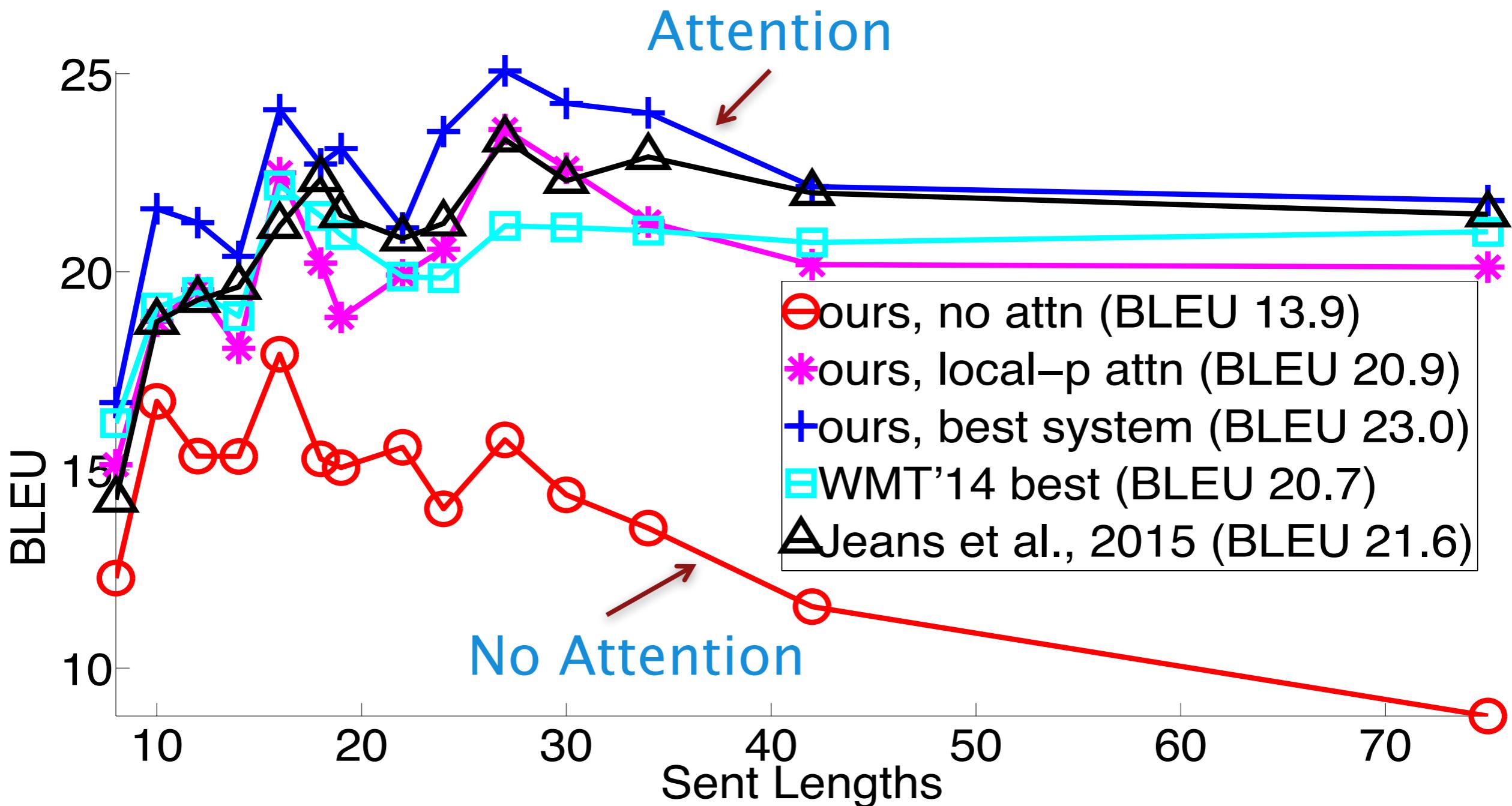
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \bar{\mathbf{h}}_s \\ \mathbf{h}_t^\top \mathbf{W}_a \bar{\mathbf{h}}_s \\ \mathbf{v}_a^\top \tanh (\mathbf{W}_a [\mathbf{h}_t; \bar{\mathbf{h}}_s]) \end{cases}$$

GitHub, Inc. [US] <https://github.com/harvardnlp/seq2seq-attn>

## Sequence-to-Sequence Learning with Attentional Neural Networks

The attention model is from [Effective Approaches to Attention-based Neural Machine Translation](#), Luong et al. EMNLP 2015. We use the *global-general-attention* model with the *input-feeding* approach from the paper. Input-feeding is optional and can be turned off.

# Better Translation of Long Sentences



# Neural Network Toolkits

## ★ **PyTorch**: <http://pytorch.org/>

- Facebook AI Research and many others

## • **Tensorflow**: <https://www.tensorflow.org/>

- By Google, actively maintained, bindings for many languages

## • **DyNet**: <https://github.com/clab/dynet>

- CMU and other individual researchers, dynamic structures that change for every training instance

## • **Caffe**: <http://caffe.berkeleyvision.org/>

- UC Berkeley, for vision

## • **Theano**: <http://deeplearning.net/software/theano>

- University of Montreal, less and less maintained

# Next Time:

- \* **Last Class!**
- \* **Convolutional Neural Networks, Sentiment Analysis**
- \* **Reading Assignment:**
  - \* **Overcoming Language Variation in Sentment Analysis with Social Attention**