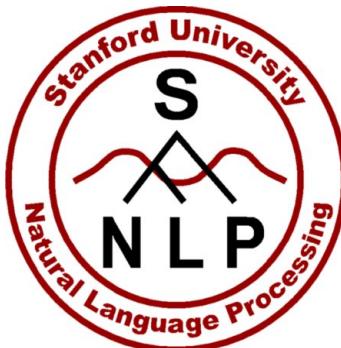


# Natural Language Processing with Deep Learning

CS224N/Ling284



Lecture 9:  
Recap and  
Fancy Recurrent Neural Networks  
for Machine Translation

Christopher Manning and **Richard Socher**

# Overview

- Recap of most important concepts & equations
- Machine translation
- Fancy RNN Models tackling MT:
  - Gated Recurrent Units by Cho et al. (2014)
  - Long-Short-Term-Memories  
by Hochreiter and Schmidhuber (1997)

*Advanced, cutting edge, blast from the past*

# Recap of most important concepts

Word2Vec 
$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

Glove 
$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

Nnet & Max-margin 
$$s = U^T f(Wx + b)$$
  
$$J = \max(0, 1 - s + s_c)$$

# Recap of most important concepts

Multilayer Nnet

$$\begin{aligned} x &= z^{(1)} = a^{(1)} \\ z^{(2)} &= W^{(1)}x + b^{(1)} \end{aligned}$$

&

$$a^{(2)} = f(z^{(2)})$$

Backprop

$$\begin{aligned} z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f(z^{(3)}) \\ s &= U^T a^{(3)} \end{aligned}$$

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) \circ f'(z^{(l)}),$$

$$\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}$$

# Recap of most important concepts

## Recurrent Neural Networks

$$\begin{aligned} h_t &= \sigma \left( W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]} \right) \\ \hat{y}_t &= \text{softmax} \left( W^{(S)} h_t \right) \end{aligned}$$

Cross Entropy Error

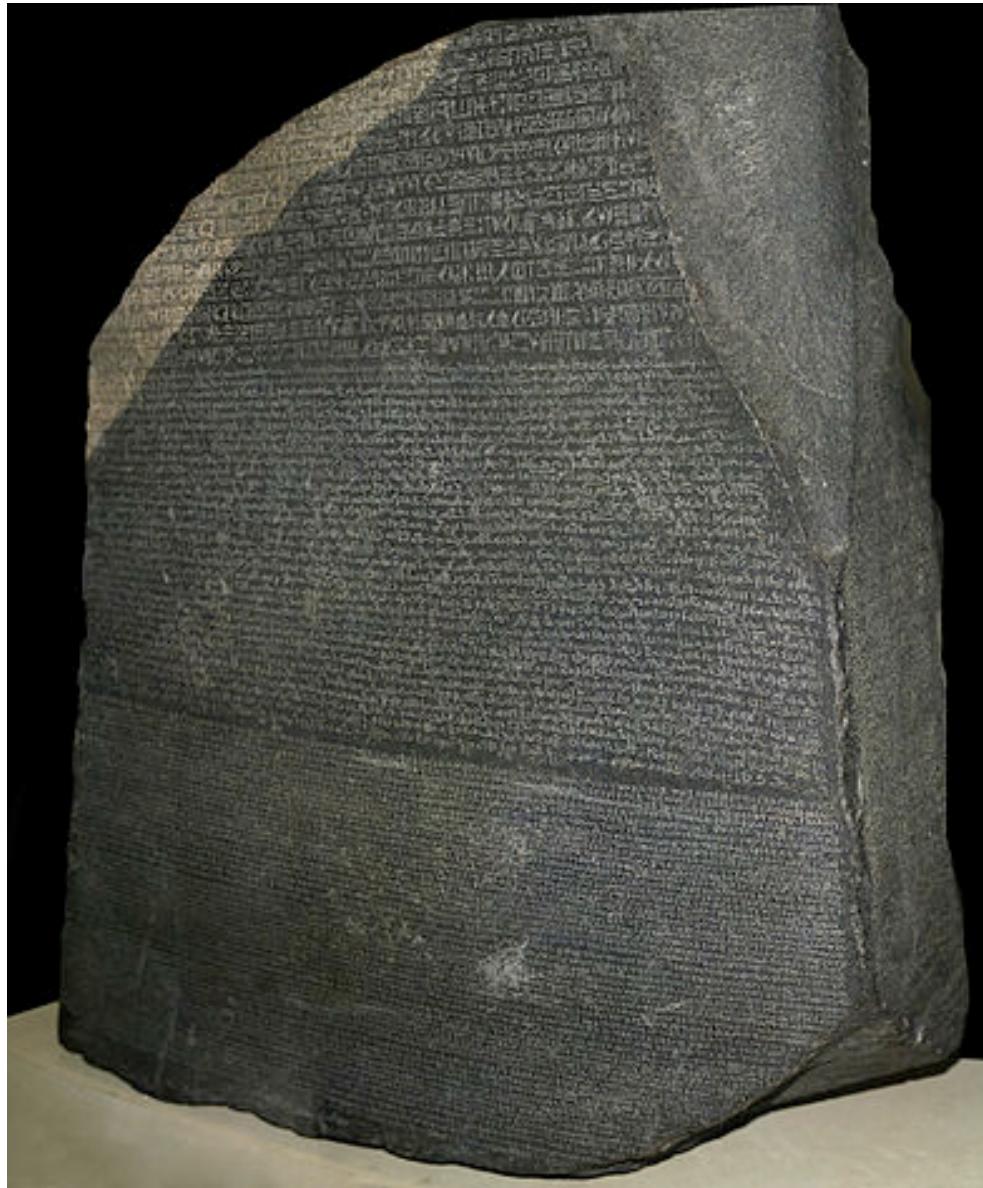
$$J^{(t)}(\theta) = - \sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Mini-batched SGD

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J_{t:t+B}(\theta)$$

# Machine Translation

- Methods are statistical
- Use parallel corpora
  - European Parliament
- First parallel corpus:
  - Rosetta Stone →
- Traditional systems are very complex



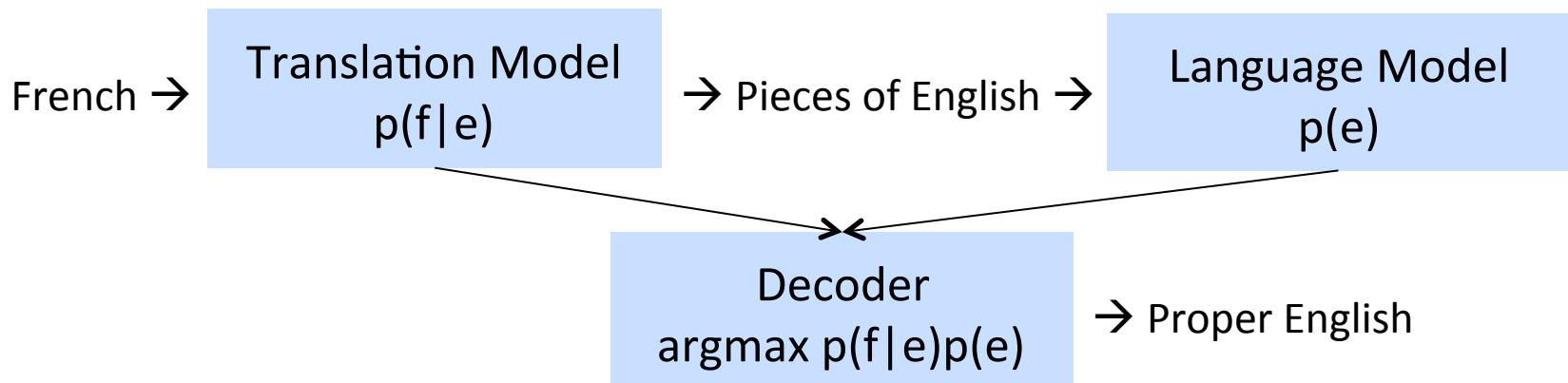
Picture from Wikipedia

# Current statistical machine translation systems

- Source language  $f$ , e.g. French
- Target language  $e$ , e.g. English
- Probabilistic formulation (using Bayes rule)

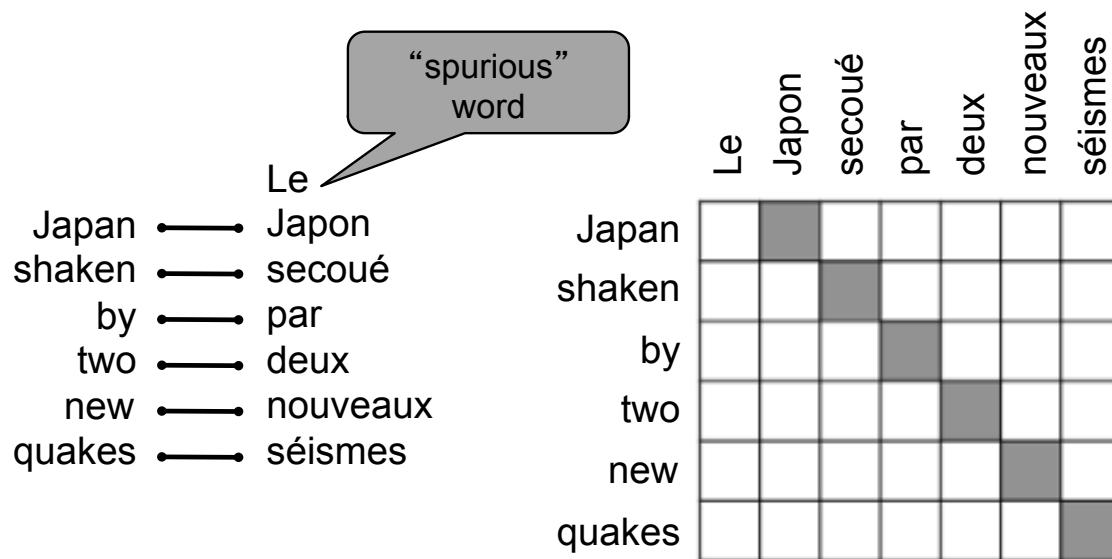
$$\hat{e} = \operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e p(f|e)p(e)$$

- Translation model  $p(f|e)$  trained on parallel corpus
- Language model  $p(e)$  trained on English only corpus (lots, free!)



## Step 1: Alignment

Goal: know which word or phrases in source language would translate to what words or phrases in target language? → Hard already!



Alignment examples from Chris Manning/CS224n

# Step 1: Alignment

“zero fertility” word  
not translated

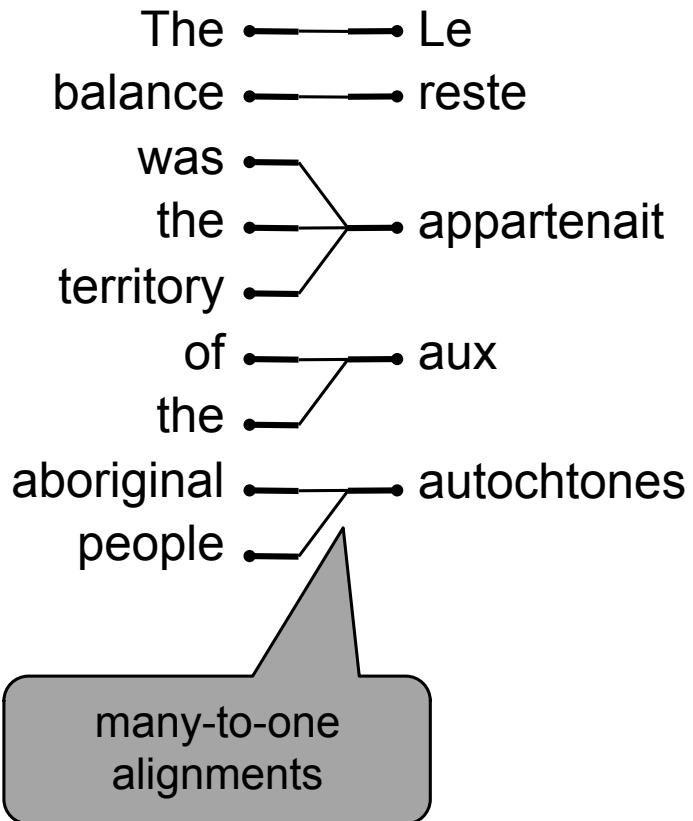
And                    Le  
the                  programme  
program            a  
has                été  
been              mis  
implemented      en  
                     application

one-to-many  
alignment

Le	programme	a	été	mis	en	application
And						
the						
program						
has						
been						
implemented						

# Step 1: Alignment

Really hard :/



	Le	reste	appartenait	aux	autochtones
The	■				
balance		■			
was			■		
the				■	
territory					■
of				■	
the					■
aboriginal					■
people					■

# Step 1: Alignment

The                    Les  
poor                 pauvres  
don't                sont  
have                 démunis  
any  
money

many-to-many  
alignment

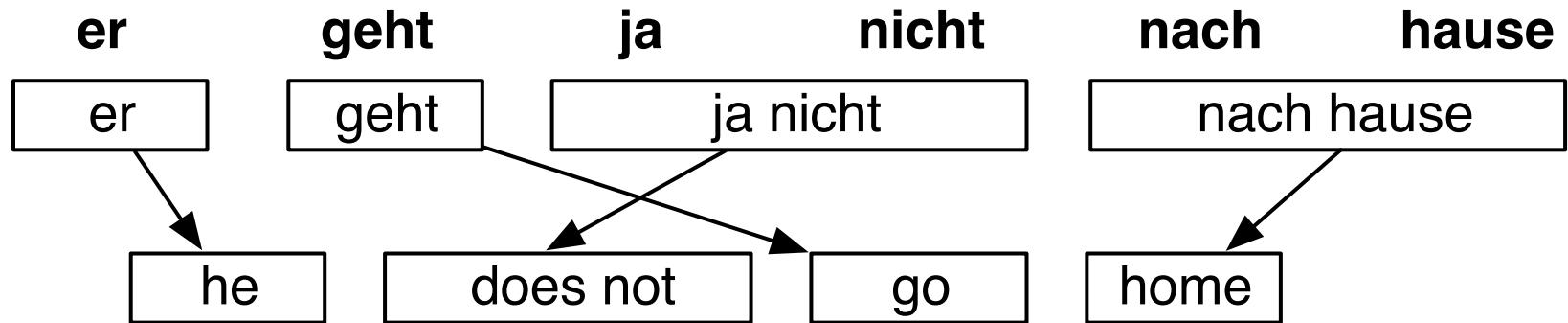
Les                 pauvres  
sont                démunis

The			
poor			
don't			
have			
any			
money			

phrase  
alignment

# Step 1: Alignment

- We could spend an entire lecture on alignment models
- Not only single words but could use phrases, syntax
- Then consider reordering of translated phrases

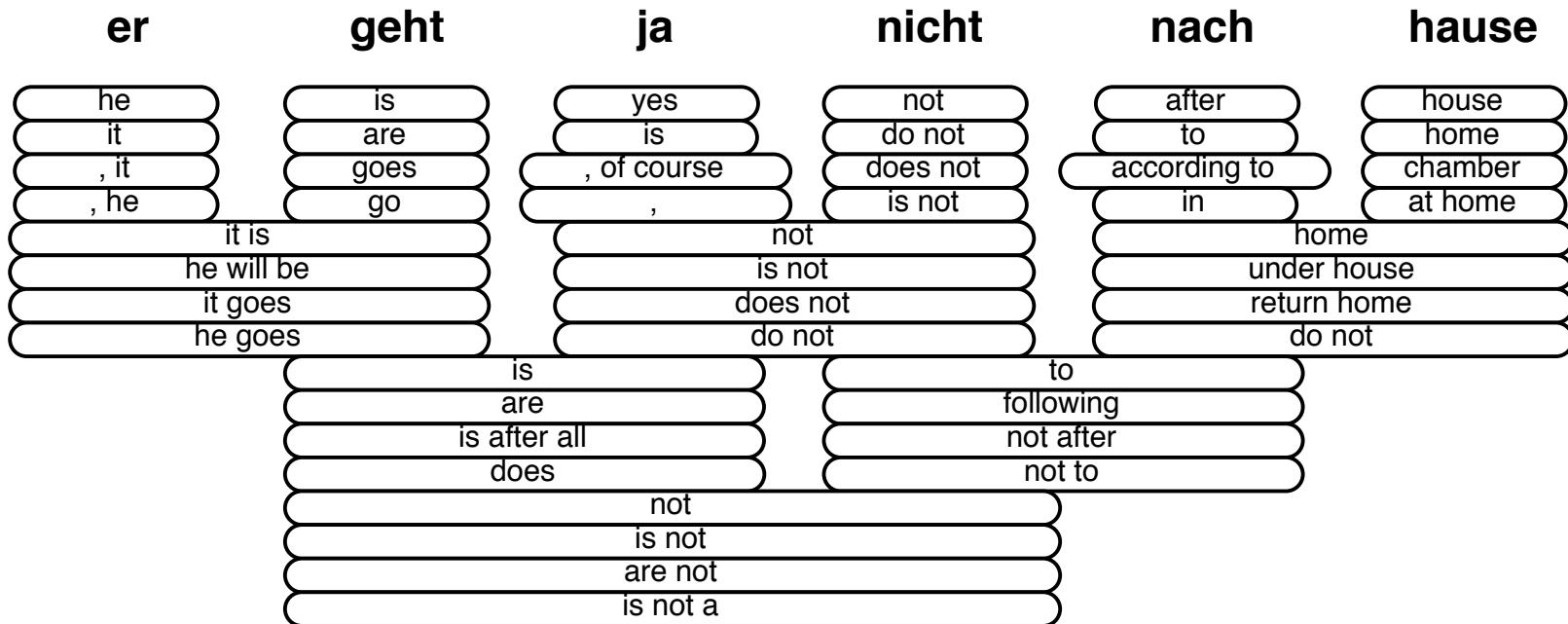


Example from Philipp Koehn

# After many steps

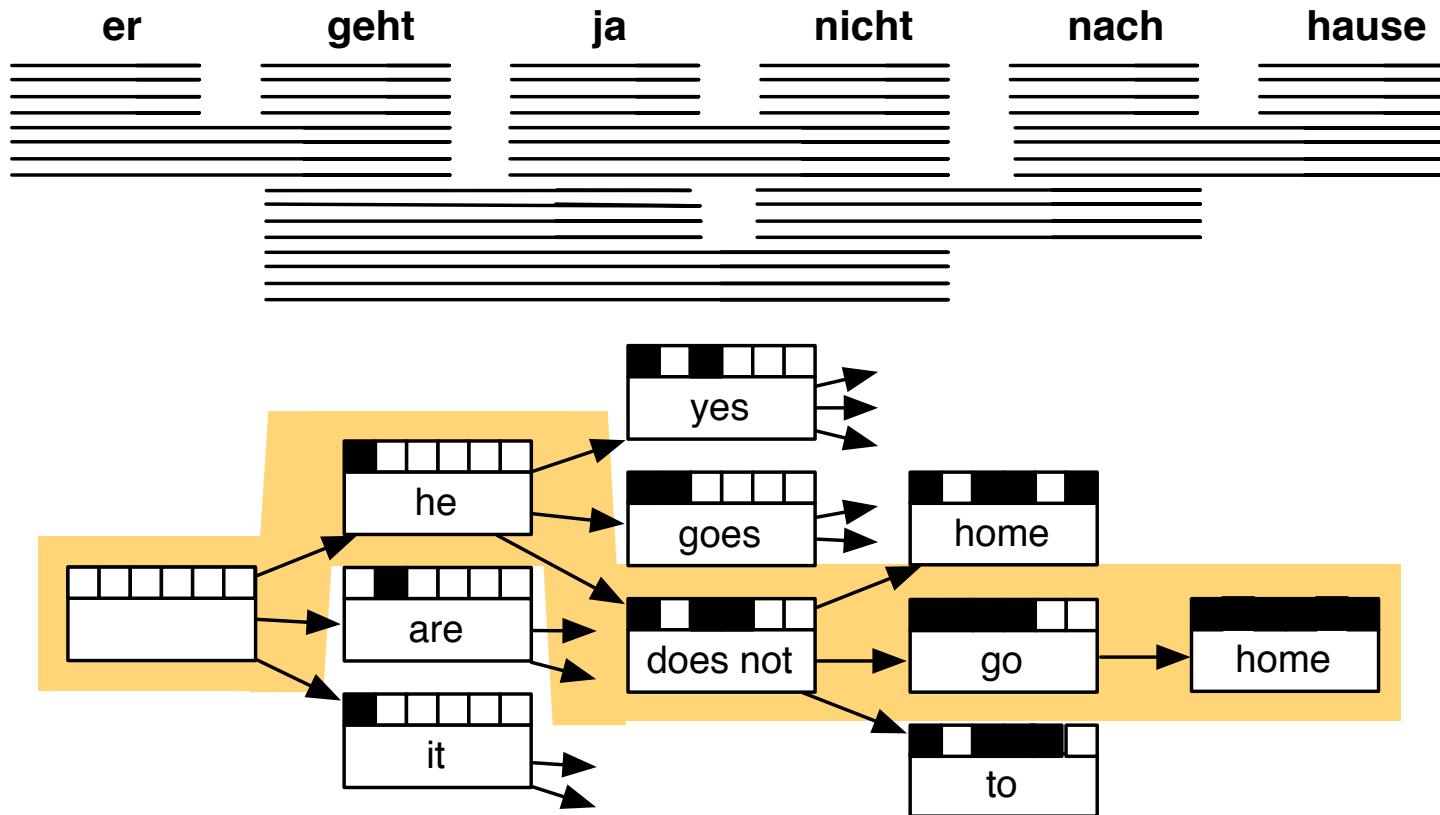
Each phrase in source language has many possible translations resulting in large search space:

## Translation Options



# Decode: Search for best of many hypotheses

Hard search problem that also includes language model

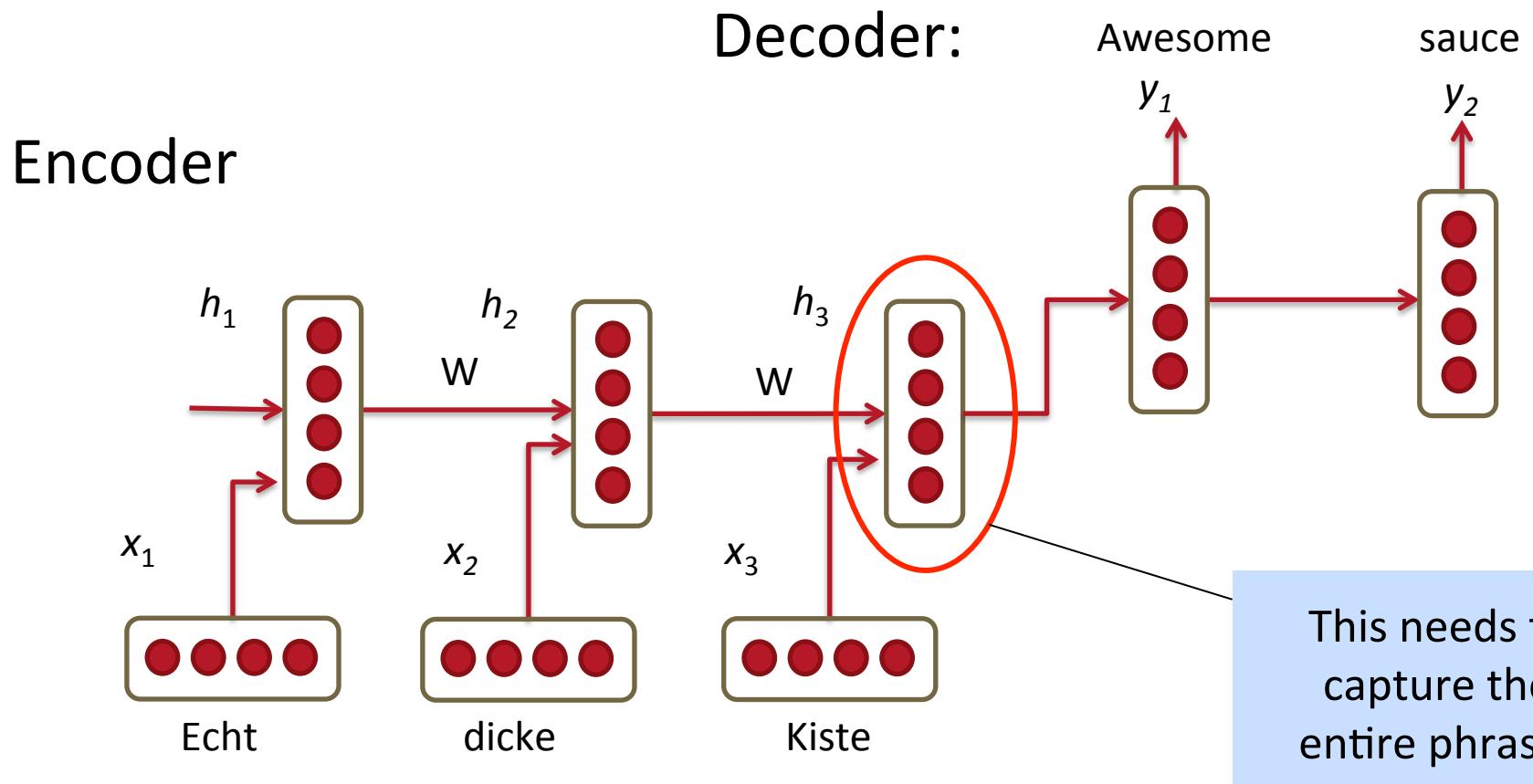


## Traditional MT

- Skipped hundreds of important details
- A lot of human feature engineering
- Very complex systems
- Many different, independent machine learning problems

# Deep learning to the rescue! ... ?

Maybe, we could translate directly with an RNN?



# MT with RNNs – Simplest Model

Encoder:  $h_t = \phi(h_{t-1}, x_t) = f\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$

Decoder:  $h_t = \phi(h_{t-1}) = f\left(W^{(hh)}h_{t-1}\right)$

$$y_t = \text{softmax}\left(W^{(S)}h_t\right)$$

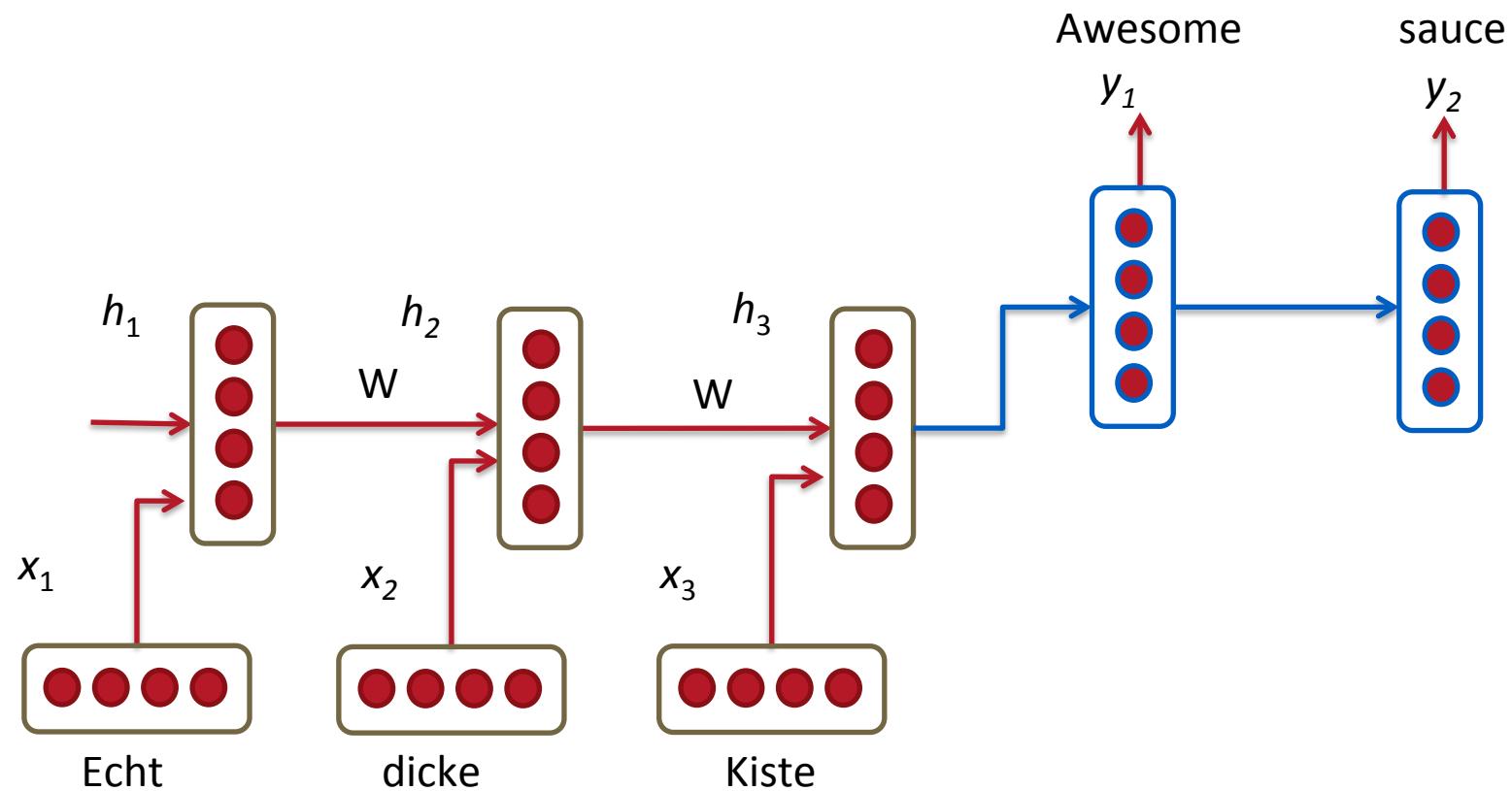
Minimize cross entropy error for all target words  
conditioned on source words

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log p_{\theta}(y^{(n)} | x^{(n)})$$

It's not quite that simple ;)

# RNN Translation Model Extensions

1. Train different RNN weights for encoding and decoding



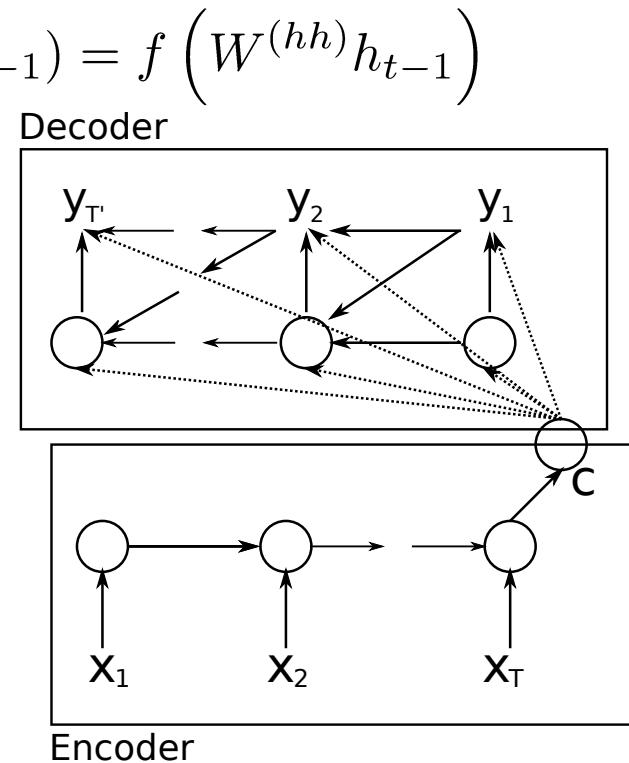
# RNN Translation Model Extensions

Notation: Each input of  $\phi$  has its own linear transformation matrix. Simple:  $h_t = \phi(h_{t-1}) = f\left(W^{(hh)}h_{t-1}\right)$

## 2. Compute every hidden state in decoder from

- Previous hidden state (standard)
- Last hidden vector of encoder  $c=h_T$
- Previous predicted output word  $y_{t-1}$

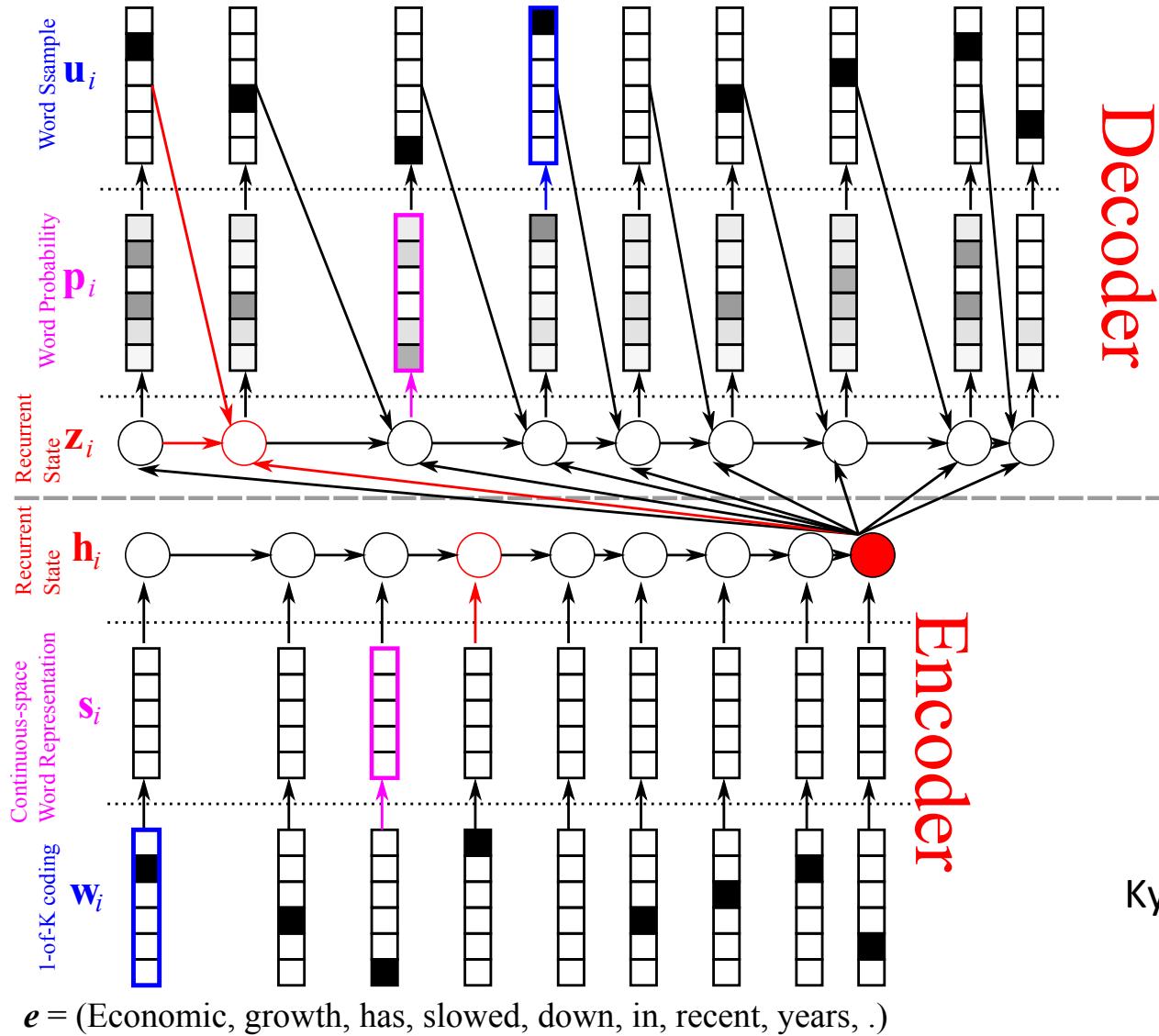
$$h_{D,t} = \phi_D(h_{t-1}, c, y_{t-1})$$



Cho et al. 2014

# Different picture, same idea

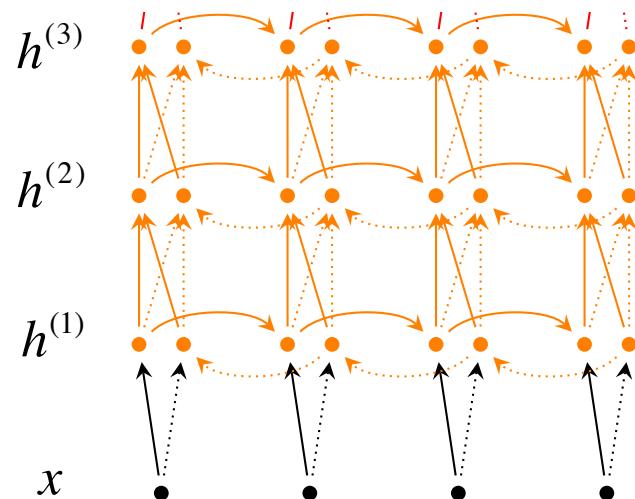
$f = (\text{La}, \text{croissance}, \text{économique}, \text{s'est}, \text{ralentie}, \text{ces}, \text{dernières}, \text{années}, .)$



Kyunghyun Cho et al. 2014

# RNN Translation Model Extensions

3. Train stacked/deep RNNs with multiple layers
4. Potentially train bidirectional encoder
5. Train input sequence in reverse order for simple optimization problem: Instead of  $A\ B\ C \rightarrow X\ Y$ , train with  $C\ B\ A \rightarrow X\ Y$



## 6. Main Improvement: Better Units

- More complex hidden unit computation in recurrence!
- Gated Recurrent Units (GRU)  
introduced by Cho et al. 2014 (see reading list)
- Main ideas:
  - keep around memories to capture long distance dependencies
  - allow error messages to flow at different strengths depending on the inputs

# GRUs

- Standard RNN computes hidden layer at next time step directly:
$$h_t = f \left( W^{(hh)} h_{t-1} + W^{(hx)} x_t \right)$$
- GRU first computes an update **gate** (another layer) based on current input word vector and hidden state

$$z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$

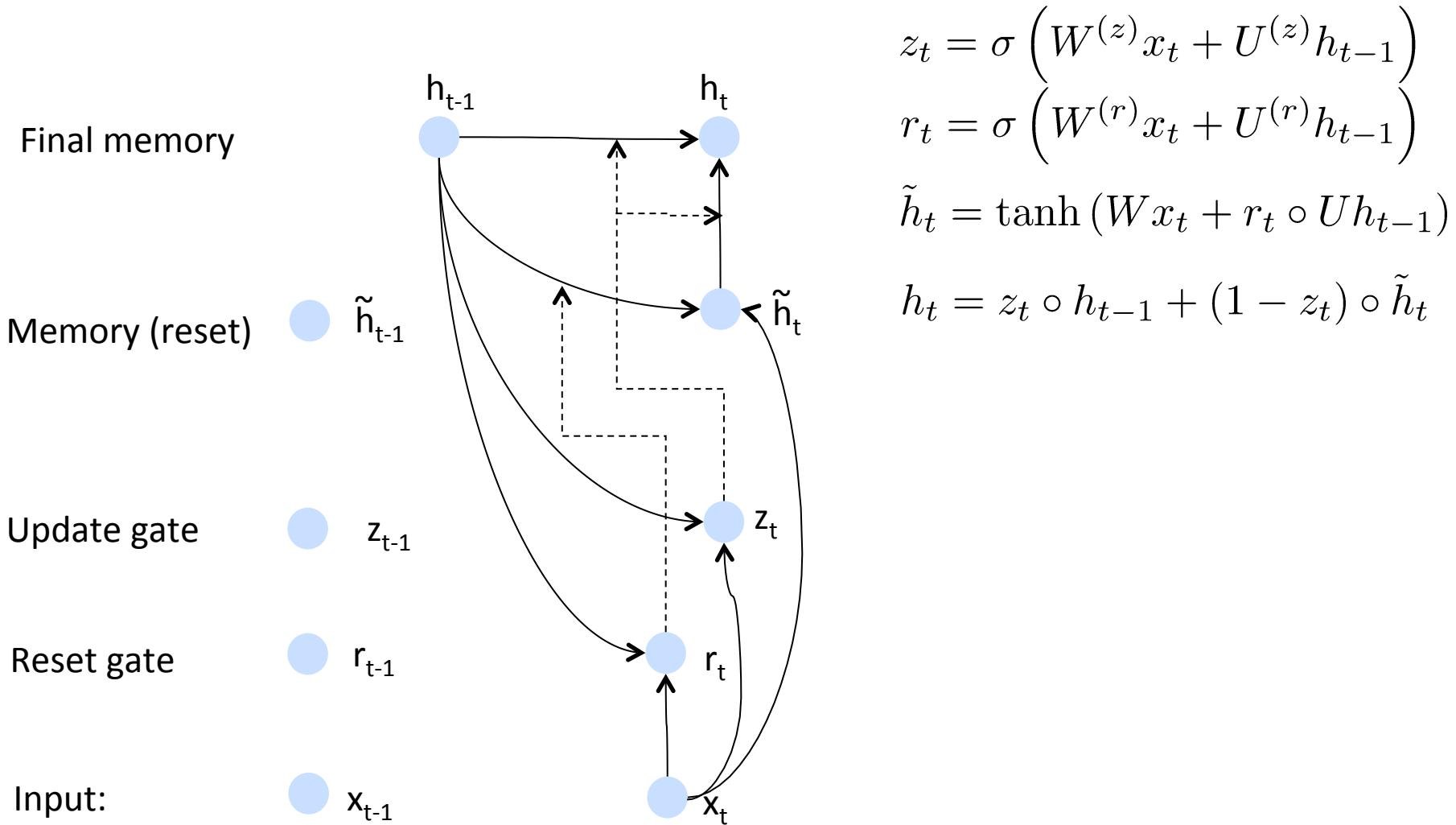
- Compute reset gate similarly but with different weights

$$r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$

# GRUs

- Update gate 
$$z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$
- Reset gate 
$$r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$
- New memory content:  $\tilde{h}_t = \tanh (W x_t + r_t \circ U h_{t-1})$   
If reset gate unit is  $\sim 0$ , then this ignores previous memory and only stores the new word information
- Final memory at time step combines current and previous time steps: 
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

# Attempt at a clean illustration



# GRU intuition

- If reset is close to 0, ignore previous hidden state  
→ Allows model to drop information that is irrelevant in the future
- Update gate  $z$  controls how much of past state should matter now.
  - If  $z$  close to 1, then we can copy information in that unit through many time steps! **Less vanishing gradient!**
- Units with short-term dependencies often have reset gates very active

$$z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$

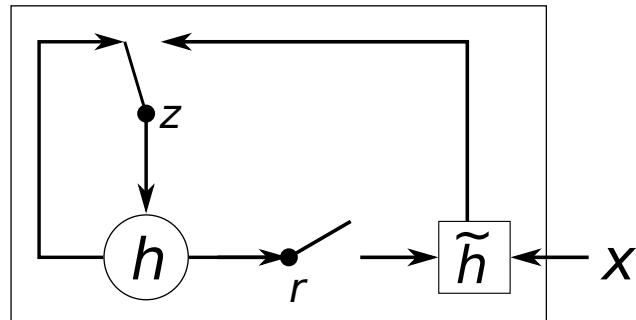
$$r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$

$$\tilde{h}_t = \tanh (W x_t + r_t \circ U h_{t-1})$$

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

# GRU intuition

- Units with long term dependencies have active update gates  $z$
- Illustration:



- Derivative of  $\frac{\partial}{\partial x_1} x_1 x_2$  ? → rest is same chain rule, but implement with **modularization** or automatic differentiation

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

$$\tilde{h}_t = \tanh(Wx_t + r_t \circ Uh_{t-1})$$

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

# Long-short-term-memories (LSTMs)

- We can make the units even more complex
- Allow each time step to modify
  - Input gate (current cell matters)
  - Forget (gate 0, forget past)
  - Output (how much cell is exposed)
  - New memory cell
- Final memory cell:
- Final hidden state:

$$i_t = \sigma \left( W^{(i)}x_t + U^{(i)}h_{t-1} \right)$$

$$f_t = \sigma \left( W^{(f)}x_t + U^{(f)}h_{t-1} \right)$$

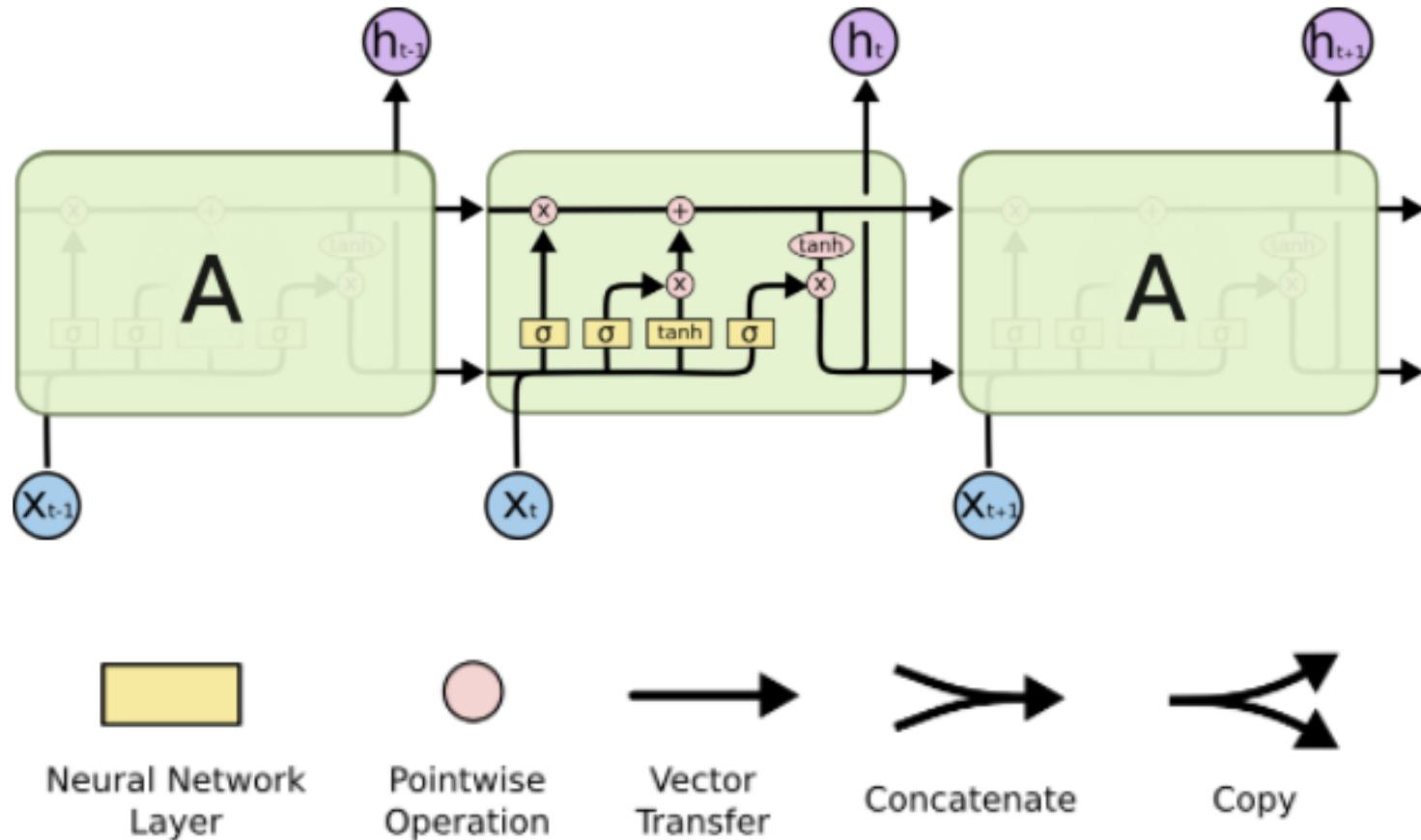
$$o_t = \sigma \left( W^{(o)}x_t + U^{(o)}h_{t-1} \right)$$

$$\tilde{c}_t = \tanh \left( W^{(c)}x_t + U^{(c)}h_{t-1} \right)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

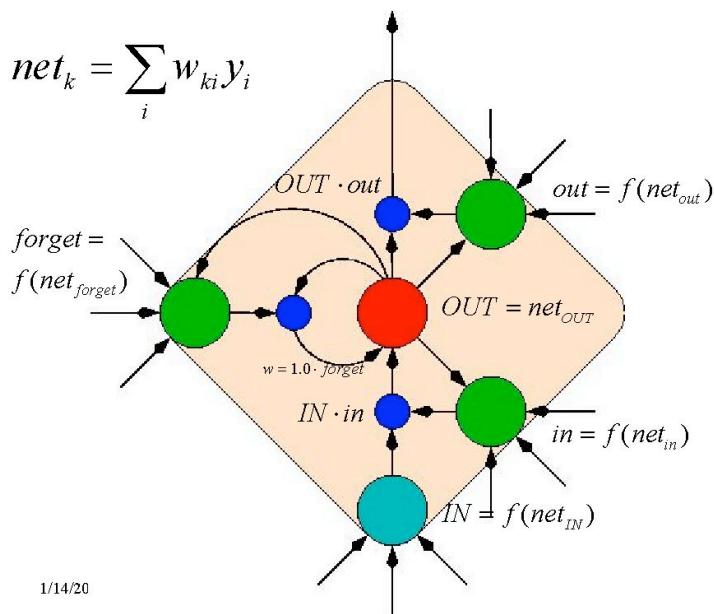
$$h_t = o_t \circ \tanh(c_t)$$

# Some visualizations

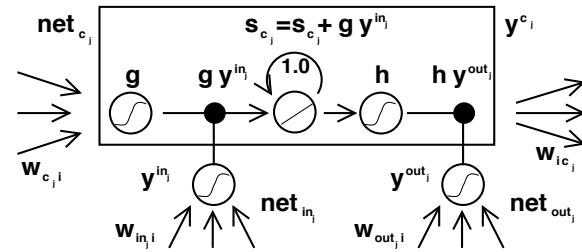


By Chris Ola: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

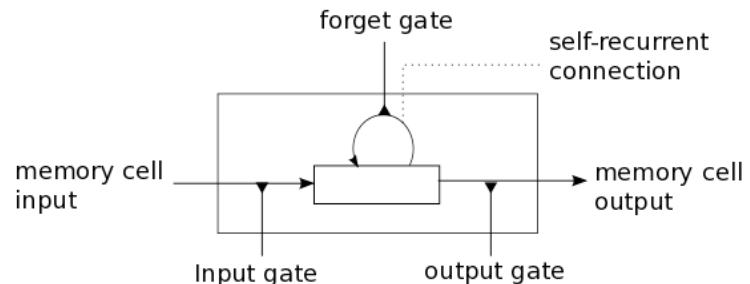
# Most illustrations a bit overwhelming ;)



<http://people.idsia.ch/~juergen/lstm/sld017.htm>



Long Short-Term Memory by Hochreiter and Schmidhuber (1997)



<http://deeplearning.net/tutorial/lstm.html>

Intuition: memory cells can keep information intact, unless inputs makes them forget it or overwrite it with new input.

Cell can decide to output this information or just store it

# LSTMs are currently very hip!

- En vogue default model for most sequence labeling tasks
- Very powerful, especially when stacked and made even deeper (each hidden layer is already computed by a deep internal network)
- Most useful if you have lots and lots of data

# Deep LSTMs compared to traditional systems 2015

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	<b>34.81</b>

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	<b>37.0</b>
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	<b>36.5</b>
Oracle Rescoring of the Baseline 1000-best lists	~45

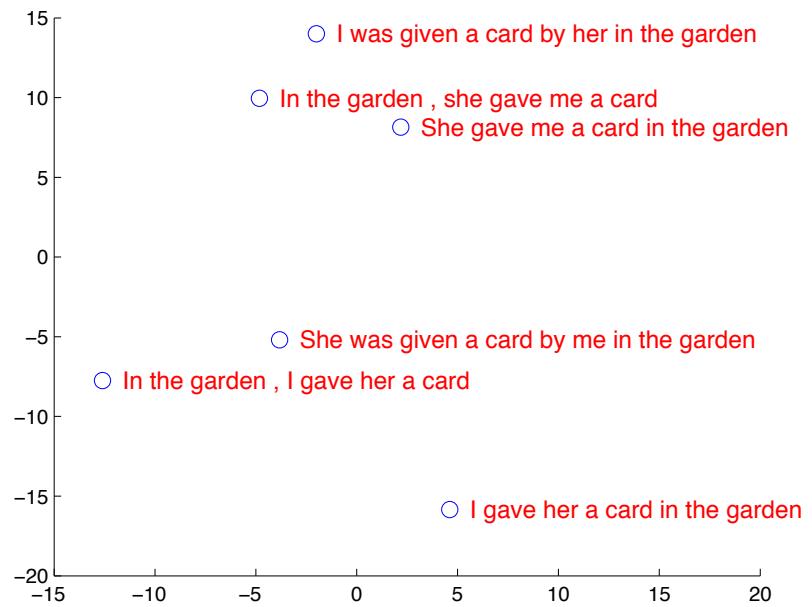
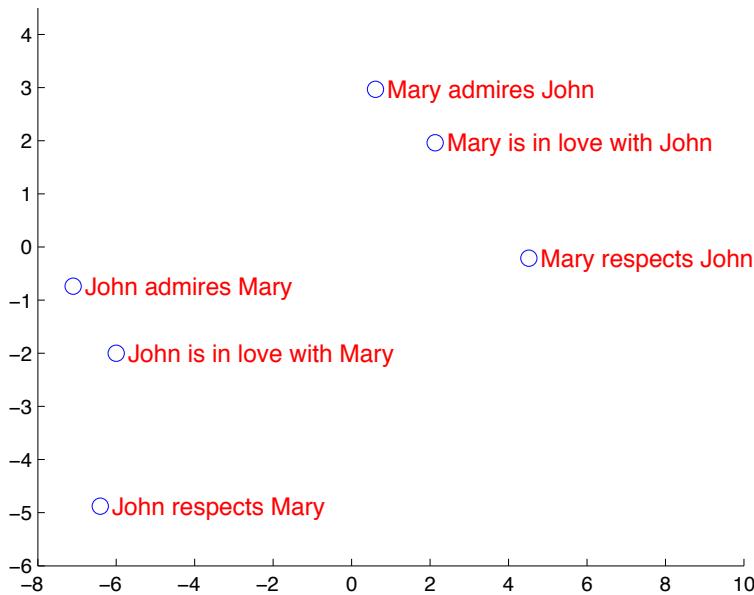
# Deep LSTMs (with a lot more tweaks)

WMT 2016 competition results from last year

Scored Systems					
System	Submitter	System Notes	Constraint	Run Notes	BLEU
<a href="#">uedin-nmt-ensemble</a> <small>(Details)</small>	rsennrich <i>University of Edinburgh</i>	BPE neural MT system with monolingual training data (back-translated), ensemble of 4, reranked with right-to-left model.	yes		34.8
<a href="#">metamind-ensemble</a> <small>(Details)</small>	jekbradbury <i>Salesforce MetaMind</i>	Neural MT system based on Luong 2015 and Sennrich 2015, using Morfessor for subword splitting, with back-translated monolingual augmentation. Ensemble of 3 checkpoints from one run plus 1 Y-LSTM (see entry).	yes		32.8
<a href="#">uedin-nmt-single</a> <small>(Details)</small>	rsennrich <i>University of Edinburgh</i>	BPE neural MT system with monolingual training data (back-translated). single model. (contrastive)	yes		32.2
<a href="#">KIT</a> <small>(Details)</small>	niehues <i>KIT</i>	Phrase-based MT with NMT in rescoring	yes		29.7
<a href="#">uedin-pbt-wmt16-en-de</a> <small>(Details)</small>	Matthias Huck <i>University of Edinburgh</i>	Phrase-based Moses	yes		29.1
<a href="#">Moses Phrase-Based</a> <small>(Details)</small>	jhu-smt <i>Johns Hopkins University</i>	Phrase-based model, word clusters for all model components (LM, OSM, LR, sparse features), neural network joint model, large cc LM	yes	[26-7]	29.0
<a href="#">uedin-pbt-wmt16-en-de-contrastive</a> <small>(Details)</small>	Matthias Huck <i>University of Edinburgh</i>	Phrase-based Moses (contrastive, 2015 system)	yes		29.0

# Deep LSTM for Machine Translation

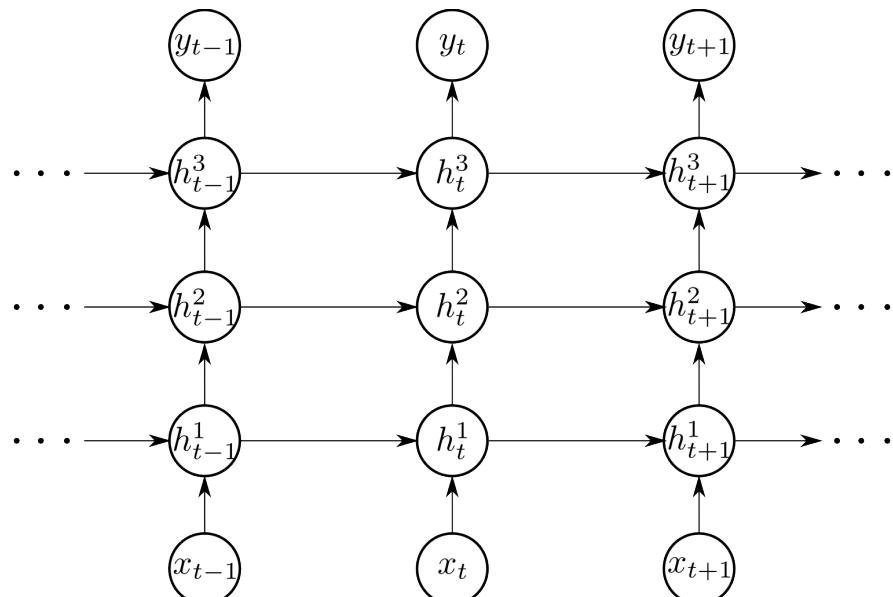
PCA of vectors from last time step hidden layer



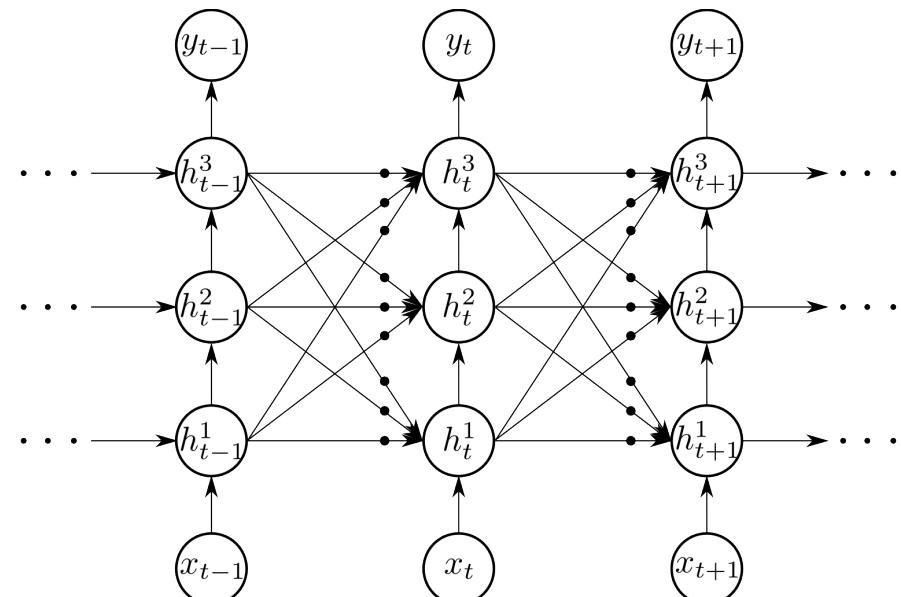
Sequence to Sequence Learning by Sutskever et al. 2014

# Further Improvements: More Gates!

Gated Feedback Recurrent Neural Networks, Chung et al. 2015



(a) Conventional stacked RNN



(b) Gated Feedback RNN

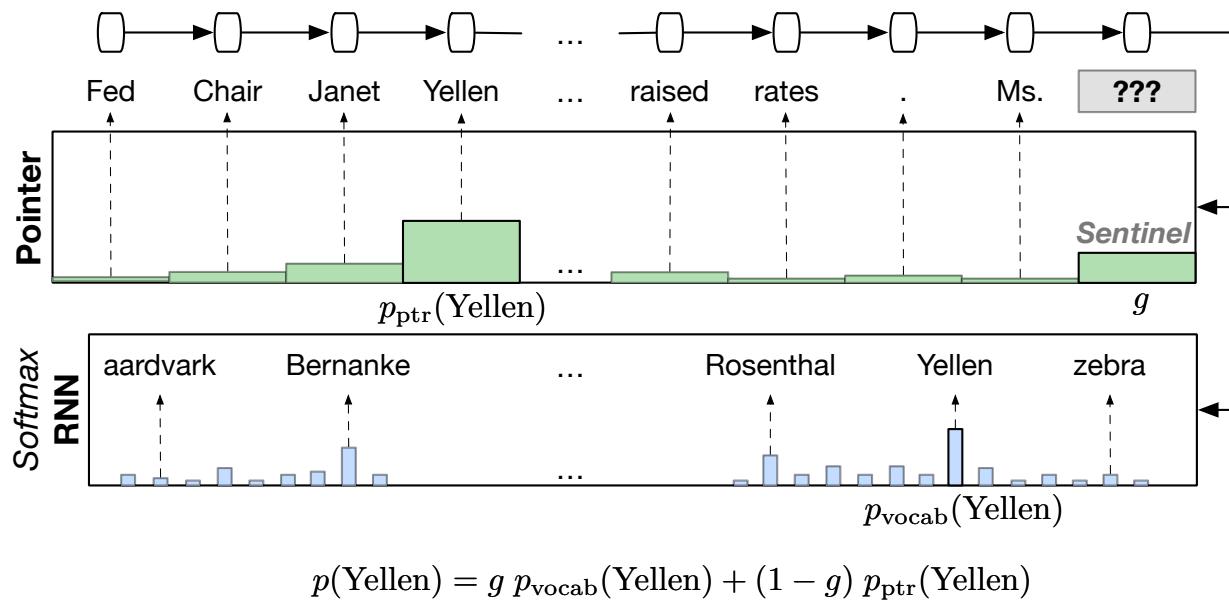
A recent  
improvement to  
RNNs

# **Problem with Softmax: No Zero Shot Word Predictions**

- Answers can only be predicted if they were seen during training and part of the softmax
- But it's natural to learn new words in an active conversation and systems should be able to pick them up

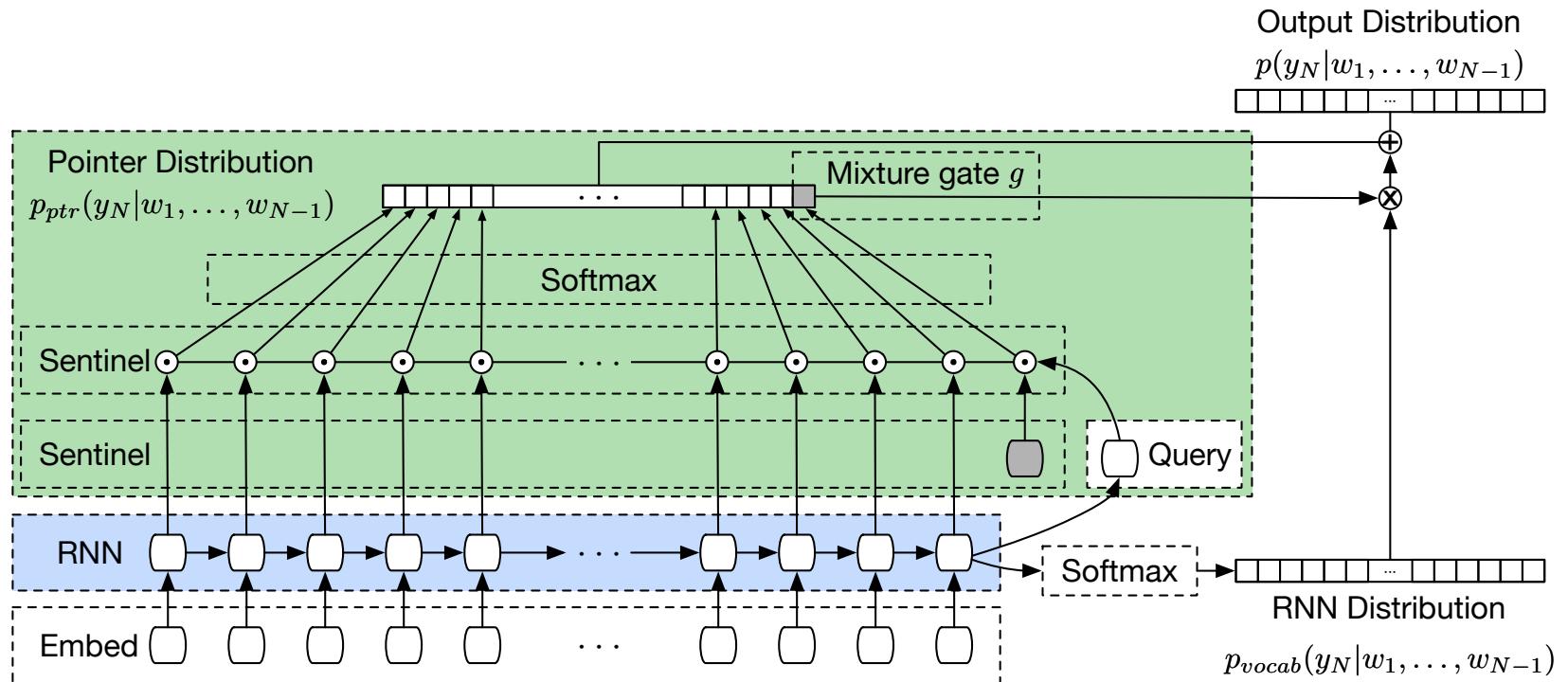
# Tackling Obstacle by Predicting Unseen Words

- Idea: Mixture Model of softmax and pointers:



- Pointer Sentinel Mixture Models by Stephen Merity, Caiming Xiong, James Bradbury, Richard Socher

# Pointer-Sentinel Model - Details



$$p(y_i|x_i) = g p_{vocab}(y_i|x_i) + (1 - g) p_{ptr}(y_i|x_i)$$

$$z_i = q^T h_i, \quad p_{ptr}(w) = \sum_{i \in I(w,x)} a_i,$$

$$a = \text{softmax}(z),$$

# Pointer Sentinel for Language Modeling

Model	Parameters	Validation	Test
Mikolov & Zweig (2012) - KN-5	2M <sup>‡</sup>	—	141.2
Mikolov & Zweig (2012) - KN5 + cache	2M <sup>‡</sup>	—	125.7
Mikolov & Zweig (2012) - RNN	6M <sup>‡</sup>	—	124.7
Mikolov & Zweig (2012) - RNN-LDA	7M <sup>‡</sup>	—	113.7
Mikolov & Zweig (2012) - RNN-LDA + KN-5 + cache	9M <sup>‡</sup>	—	92.0
Pascanu et al. (2013a) - Deep RNN	6M	—	107.5
Cheng et al. (2014) - Sum-Prod Net	5M <sup>‡</sup>	—	100.0
Zaremba et al. (2014) - LSTM (medium)	20M	86.2	82.7
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal (2015) - Variational LSTM (medium, untied)	20M	$81.9 \pm 0.2$	$79.7 \pm 0.1$
Gal (2015) - Variational LSTM (medium, untied, MC)	20M	—	$78.6 \pm 0.1$
Gal (2015) - Variational LSTM (large, untied)	66M	$77.9 \pm 0.3$	$75.2 \pm 0.2$
Gal (2015) - Variational LSTM (large, untied, MC)	66M	—	$73.4 \pm 0.0$
Kim et al. (2016) - CharCNN	19M	—	78.9
Zilly et al. (2016) - Variational RHN	32M	72.8	71.3
Zoneout + Variational LSTM (medium)	20M	84.4	80.6
Pointer Sentinel-LSTM (medium)	21M	72.4	<b>70.9</b>

# Summary

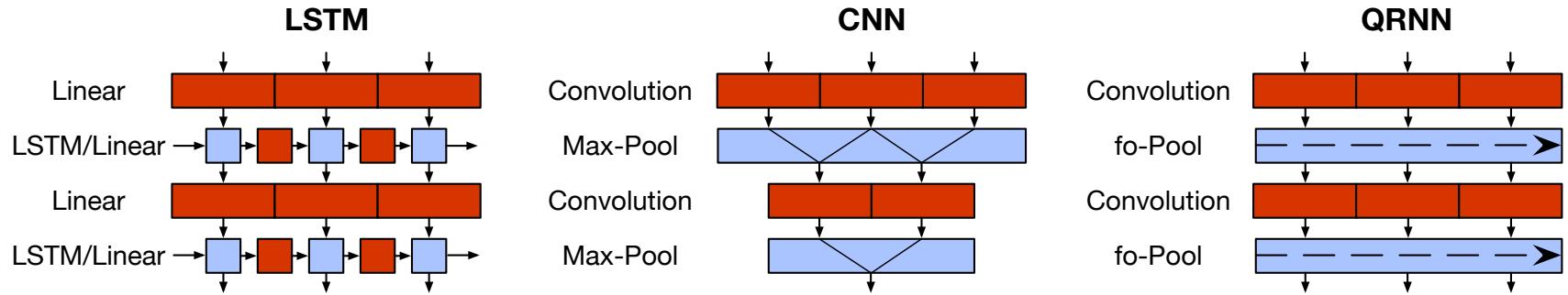
- Recurrent Neural Networks are powerful
- A lot of ongoing work right now
- Gated Recurrent Units even better
- LSTMs maybe even better (jury still out)
- This was an advanced lecture → gain intuition, encourage exploration
- Next up: Midterm review

Another recent  
improvement to  
“RNNs”

# RNNs are Slow

- RNNs are the basic building block for deepNLP
- Idea: Take the best and parallelizable parts of RNNs and CNNs
- Quasi-Recurrent Neural Networks by  
James Bradbury, Stephen Merity, Caiming Xiong & Richard Socher

# Quasi-Recurrent Neural Network



- Parallelism computation across time:

$$\mathbf{z}_t = \tanh(\mathbf{W}_z^1 \mathbf{x}_{t-1} + \mathbf{W}_z^2 \mathbf{x}_t)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f^1 \mathbf{x}_{t-1} + \mathbf{W}_f^2 \mathbf{x}_t)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o^1 \mathbf{x}_{t-1} + \mathbf{W}_o^2 \mathbf{x}_t).$$

$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X})$$

$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X})$$

$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X}),$$

- Element-wise gated recurrence for parallelism across channels:

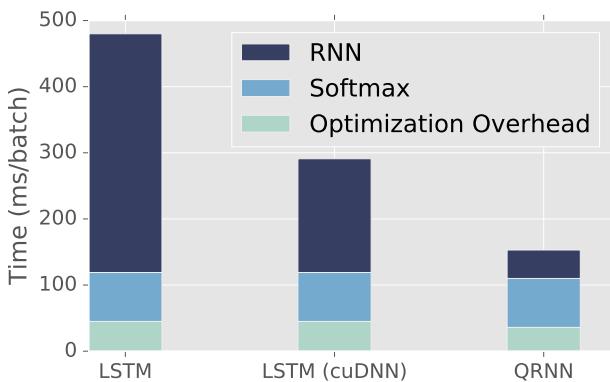
$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t,$$

# Q-RNNs for Language Modeling

- Better

Model	Parameters	Validation	Test
LSTM (medium) (Zaremba et al., 2014)	20M	86.2	82.7
Variational LSTM (medium) (Gal & Ghahramani, 2016)	20M	81.9	79.7
LSTM with CharCNN embeddings (Kim et al., 2016)	19M	—	78.9
Zoneout + Variational LSTM (medium) (Merity et al., 2016)	20M	84.4	80.6
<i>Our models</i>			
LSTM (medium)	20M	85.7	82.0
QRNN (medium)	18M	82.9	79.9
QRNN + zoneout ( $p = 0.1$ ) (medium)	18M	82.1	78.3

- Faster



Batch size	Sequence length				
	32	64	128	256	512
8	<b>5.5x</b>	<b>8.8x</b>	<b>11.0x</b>	<b>12.4x</b>	<b>16.9x</b>
16	<b>5.5x</b>	<b>6.7x</b>	<b>7.8x</b>	<b>8.3x</b>	<b>10.8x</b>
32	<b>4.2x</b>	<b>4.5x</b>	<b>4.9x</b>	<b>4.9x</b>	<b>6.4x</b>
64	<b>3.0x</b>	<b>3.0x</b>	<b>3.0x</b>	<b>3.0x</b>	<b>3.7x</b>
128	<b>2.1x</b>	<b>1.9x</b>	<b>2.0x</b>	<b>2.0x</b>	<b>2.4x</b>
256	<b>1.4x</b>	<b>1.4x</b>	<b>1.3x</b>	<b>1.3x</b>	<b>1.3x</b>

# Q-RNNs for Sentiment Analysis

- Often better and faster than LSTMs
- More interpretable
- Example:
- Initial positive review
- *Review starts out positive*  
At 117: “*not exactly a bad story*”  
At 158: “*I recommend this movie to everyone, even if you’ve never played the game*”

Model	Time / Epoch (s)	Test Acc (%)
BSVM-bi (Wang & Manning, 2012)	—	91.2
2 layer sequential BoW CNN (Johnson & Zhang, 2014)	—	92.3
Ensemble of RNNs and NB-SVM (Mesnil et al., 2014)	—	92.6
2-layer LSTM (Longpre et al., 2016)	—	87.6
Residual 2-layer bi-LSTM (Longpre et al., 2016)	—	90.1
<i>Our models</i>		
Deeply connected 4-layer LSTM (cuDNN optimized)	480	90.9
Deeply connected 4-layer QRNN	150	91.4
D.C. 4-layer QRNN with $k = 4$	160	91.1

