Sequence to Sequence Learning

Yonghui

Agenda

- Introduction
- Language modeling
- Machine translation (NMT)
- Speech recognition (ASR)
- Speech synthesis (TTS)
- Other sequence model applications
    - Image captioning
    - Syntactic parsing
    - Etc

Introduction

- Sequence to sequence learning:
    - Try to learn a mapping from one sequence to another sequence
- Examples include
    - Machine translation (MT)
    - Automatic speech recognition (ASR)
    - Speech synthesis (TTS)
    - Handwriting generation
- Seq2se learning using Encoder/decoder with attention model architecture has achieved state of the art results on many of the problems

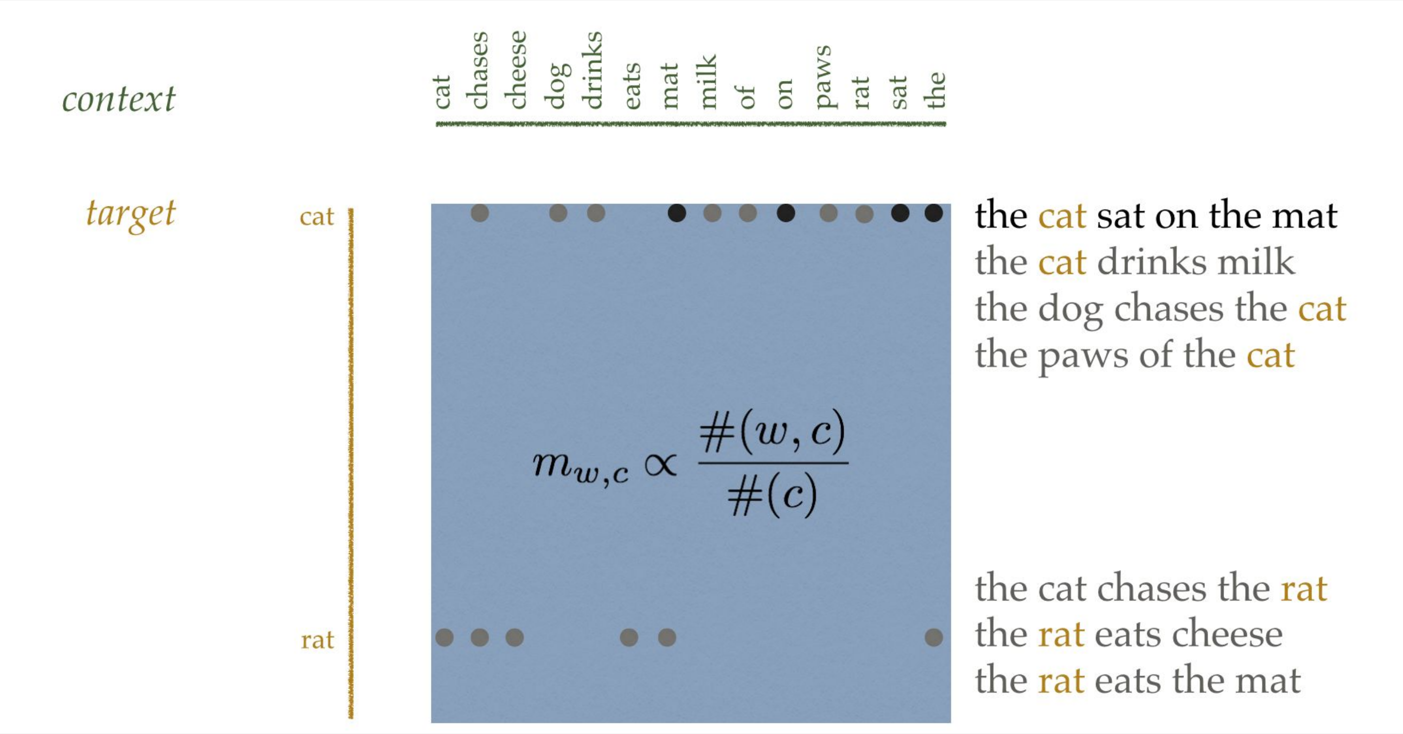
Language modeling

Language Modeling

context					*target*	$P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_{t-5})$
the	cat	sat	on	the	**mat**	0.15
$w_{t\text{-}5}$	$w_{t\text{-}4}$	$w_{t\text{-}3}$	$w_{t\text{-}2}$	$w_{t\text{-}1}$	w_t	
the	cat	sat	on	the	**rug**	0.12
the	cat	sat	on	the	**hat**	0.09
the	cat	sat	on	the	**dog**	0.01
the	cat	sat	on	the	**the**	0
the	cat	sat	on	the	**sat**	0
the	cat	sat	on	the	**robot**	?
the	cat	sat	on	the	**printer**	?

Slide Credit: Piotr Mirowski

n-grams



Slide Credit: Piotr Mirowski

The Chain Rule

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \| w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \| w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \| w_3, w_2, w_1)$
the	cat	sat	on	**the**	mat	$P(w_5 \| w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	**mat**	$P(w_6 \| w_5, w_4, w_3, w_2, w_1)$

Slide Credit: Piotr Mirowski

Markov assumption

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n+1})$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \| w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \| w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \| w_3, w_2)$
the	cat	sat	on	**the**	mat	$P(w_5 \| w_4, w_3)$
the	cat	sat	on	the	**mat**	$P(w_6 \| w_5, w_4)$

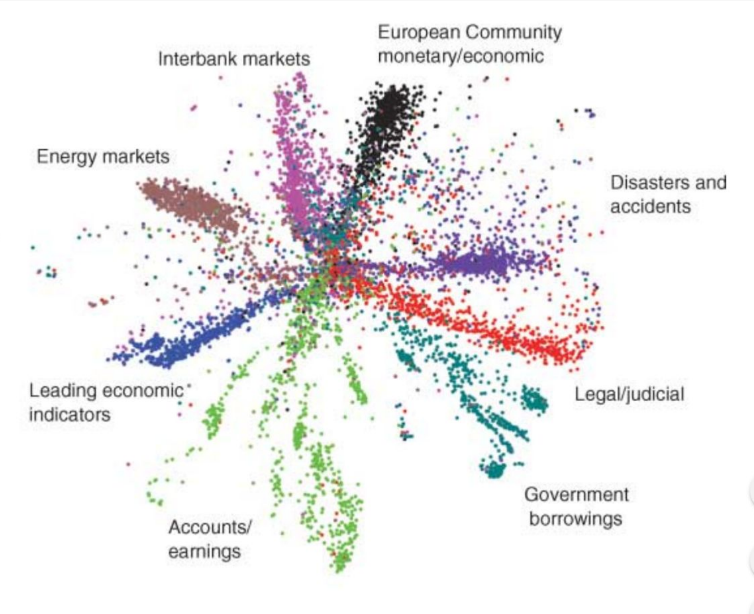
Slide Credit: Piotr Mirowski

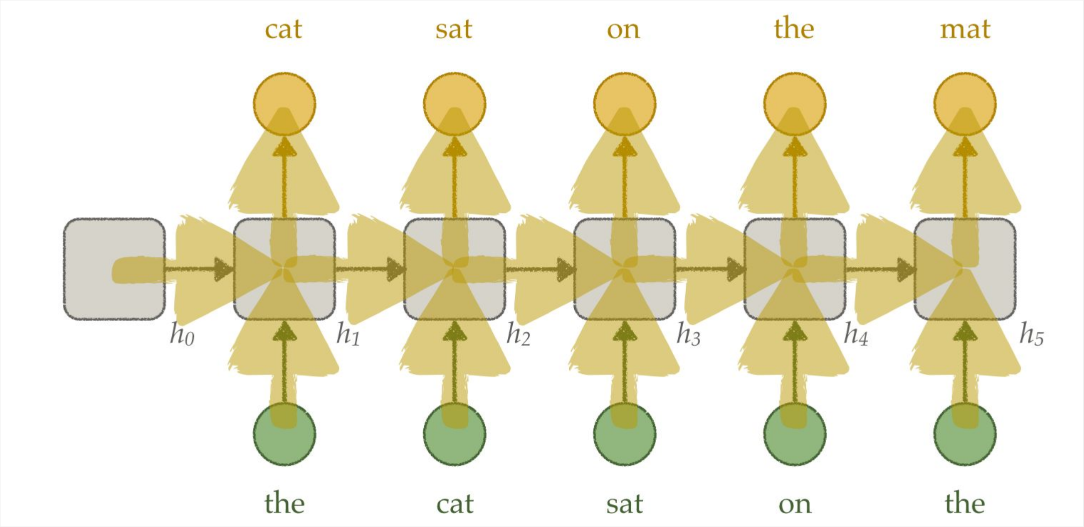
N-grams model improvements

- Very intuitive, easy to understand
- Sparsity
  - “w, c”, or “c” may not appear in a corpus at all, but the probability P(w|c) shouldn’t be 0
- Limited context
  - The length of the context is very limited
- Back-off models
  - Back off from trigram model to a bigram model, or from a bigram model to a unigram model.
  - Kneser–Ney smoothing
  - Katz's smoothing

Neural language model

- Two key ingredients: neural embeddings and recurrent neural networks

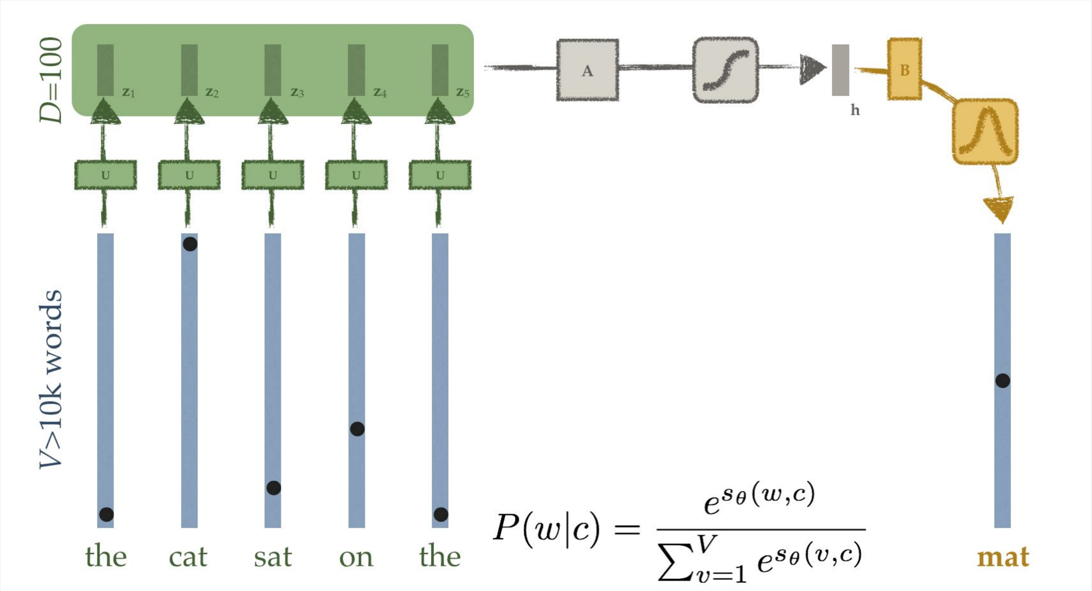




Slide Credit: Oriol Vinyals

Neural embedding

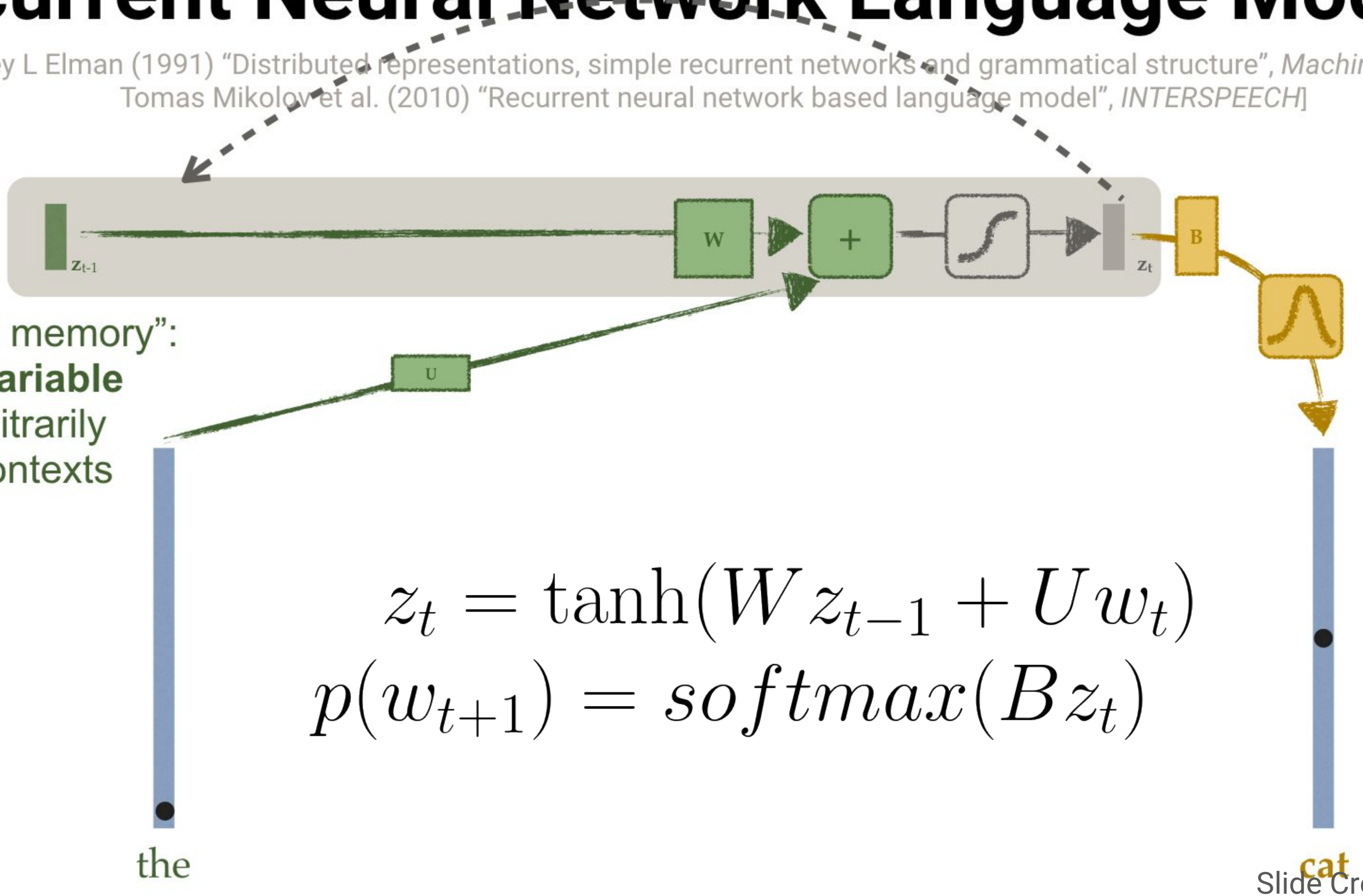
$$p(w_t | w_1, \ldots, w_{t-1}) = p_\theta(w_t | f_\theta(w_1, \ldots, w_{t-1}))$$



Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

[Jeffrey L Elman (1991) “Distributed representations, simple recurrent networks and grammatical structure”, *Machine Learning*; Tomas Mikolov et al. (2010) “Recurrent neural network based language model”, *INTERSPEECH*]



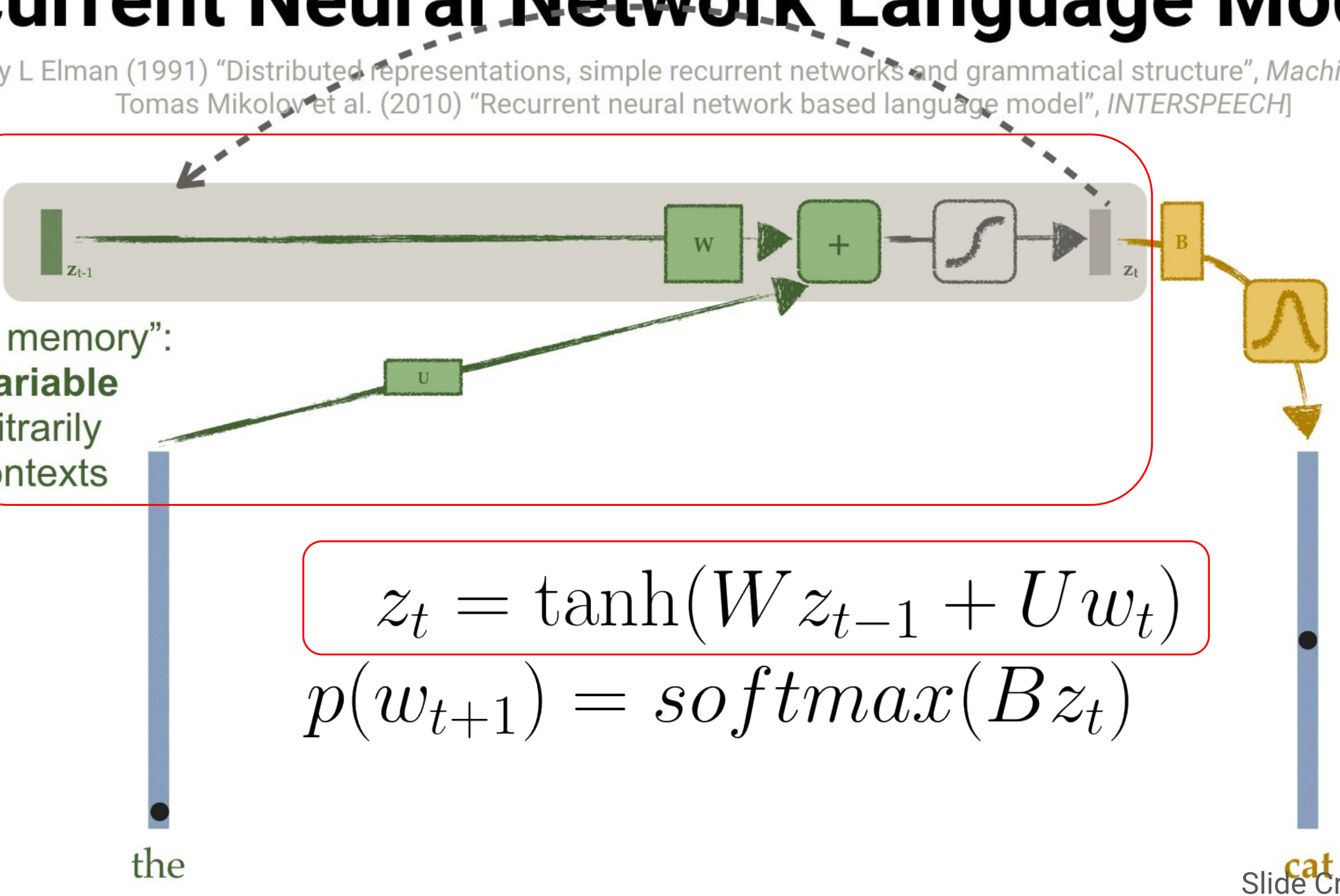
“persistent memory”:
state variable
for arbitrarily
long contexts

$$z_t = \tanh(W z_{t-1} + U w_t)$$
$$p(w_{t+1}) = softmax(B z_t)$$

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

[Jeffrey L Elman (1991) "Distributed representations, simple recurrent networks and grammatical structure", *Machine Learning*; Tomas Mikolov et al. (2010) "Recurrent neural network based language model", *INTERSPEECH*]



"persistent memory":
state variable
for arbitrarily
long contexts

$$\boxed{z_t = \tanh(W z_{t-1} + U w_t)}$$

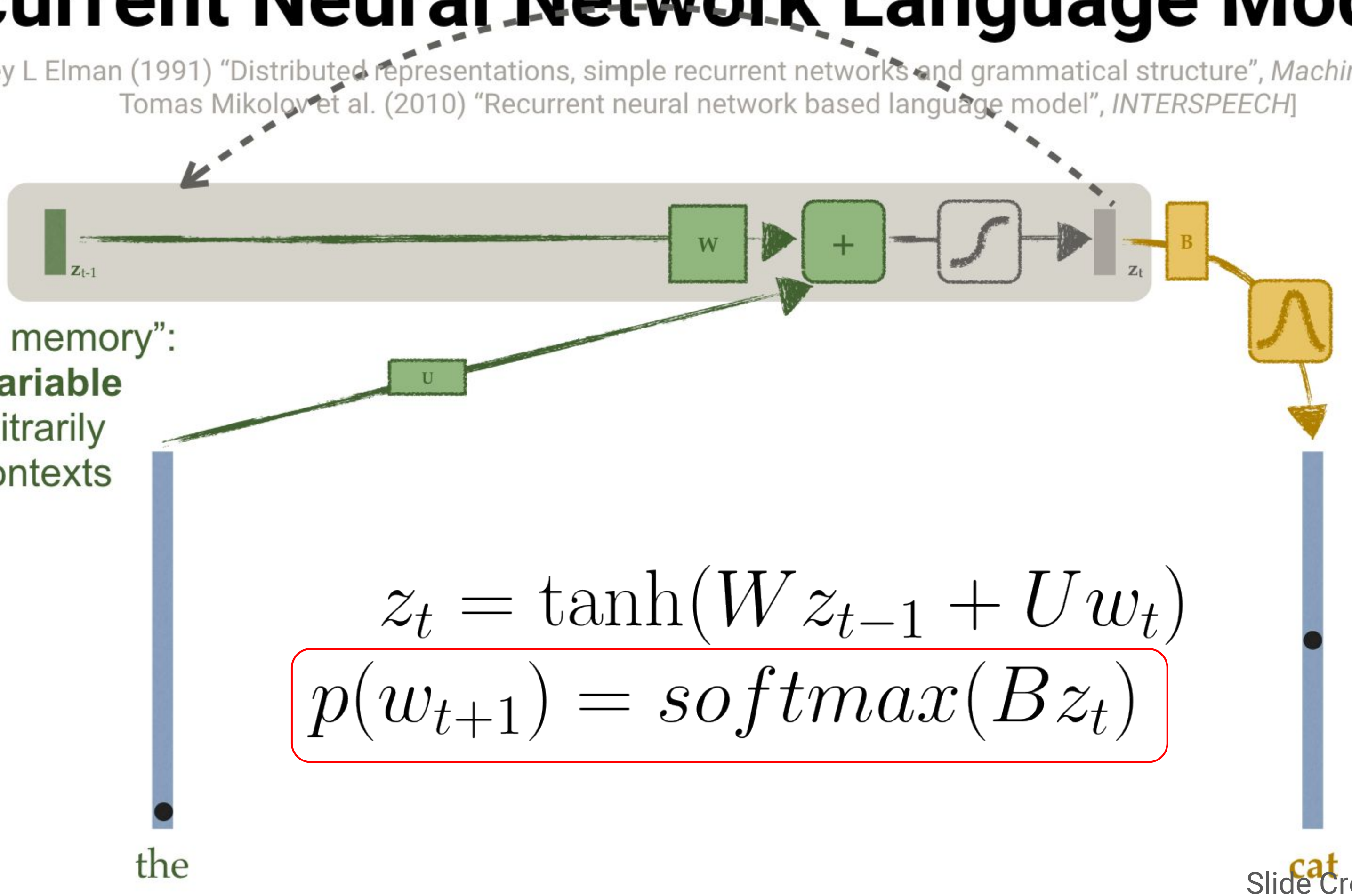
$$p(w_{t+1}) = softmax(B z_t)$$

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

[Jeffrey L Elman (1991) "Distributed representations, simple recurrent networks and grammatical structure", *Machine Learning*; Tomas Mikolov et al. (2010) "Recurrent neural network based language model", *INTERSPEECH*]

"persistent memory": **state variable** for arbitrarily long contexts

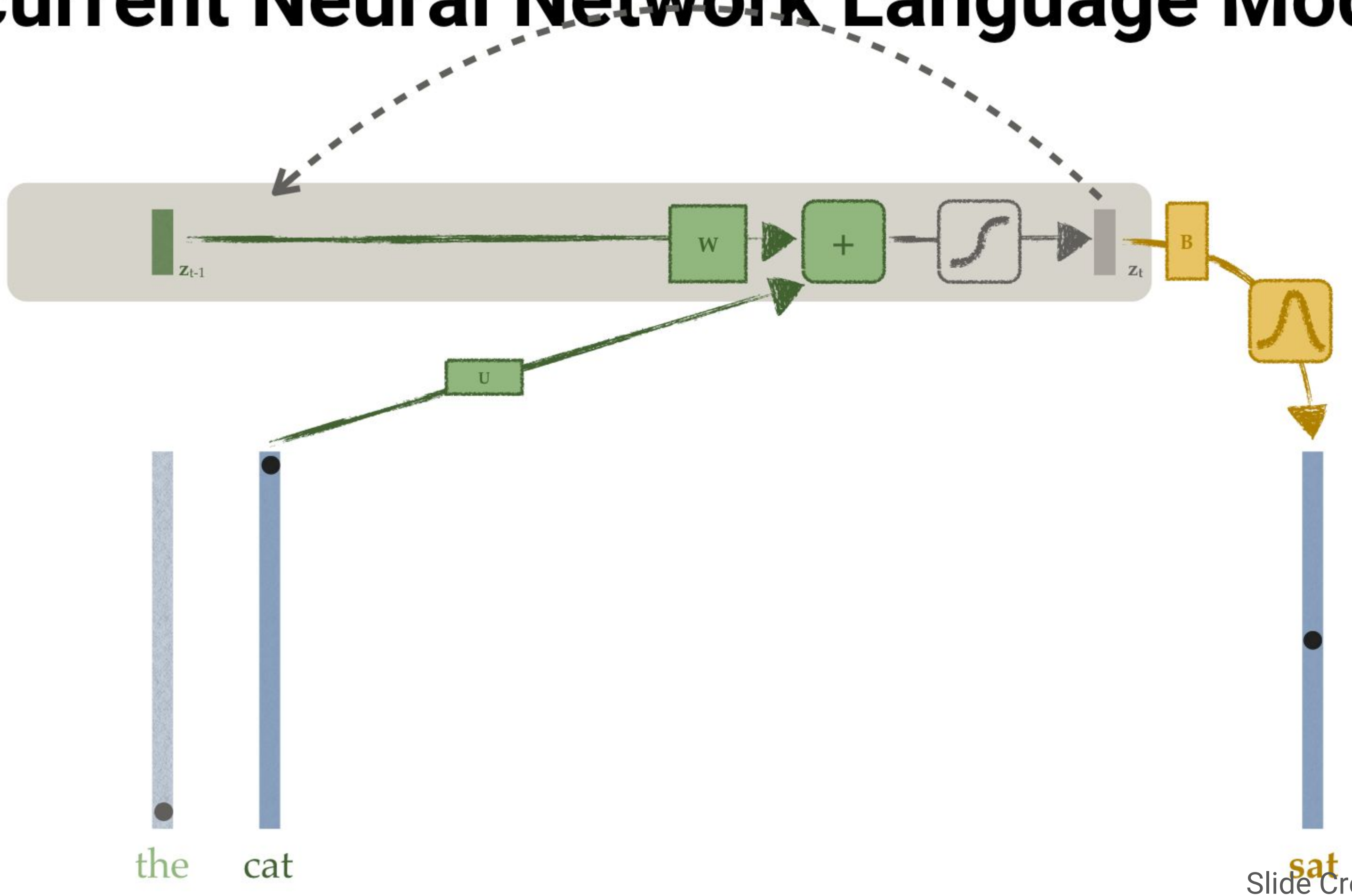


$$z_t = \tanh(W z_{t-1} + U w_t)$$

$$\boxed{p(w_{t+1}) = softmax(B z_t)}$$

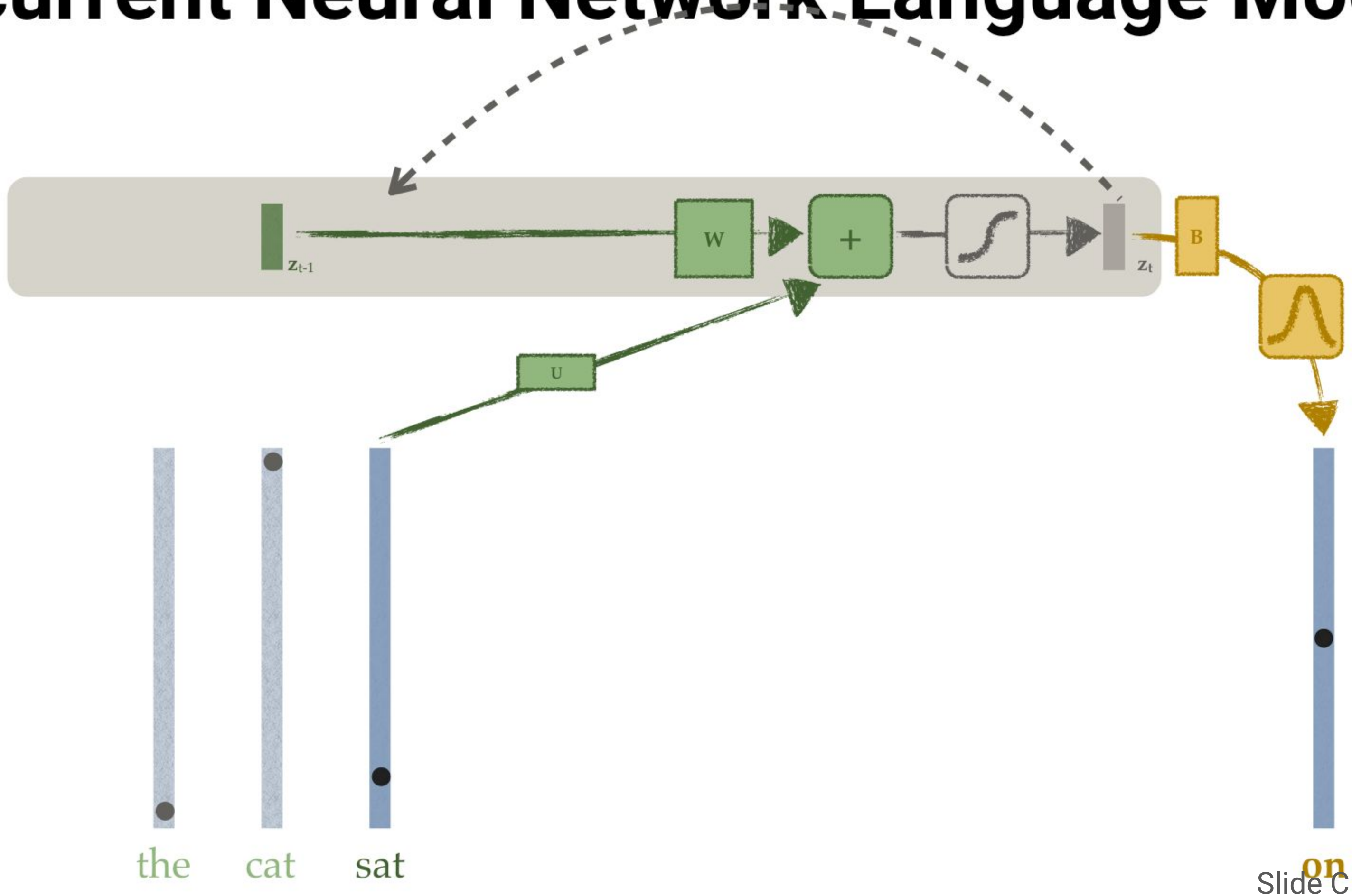
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



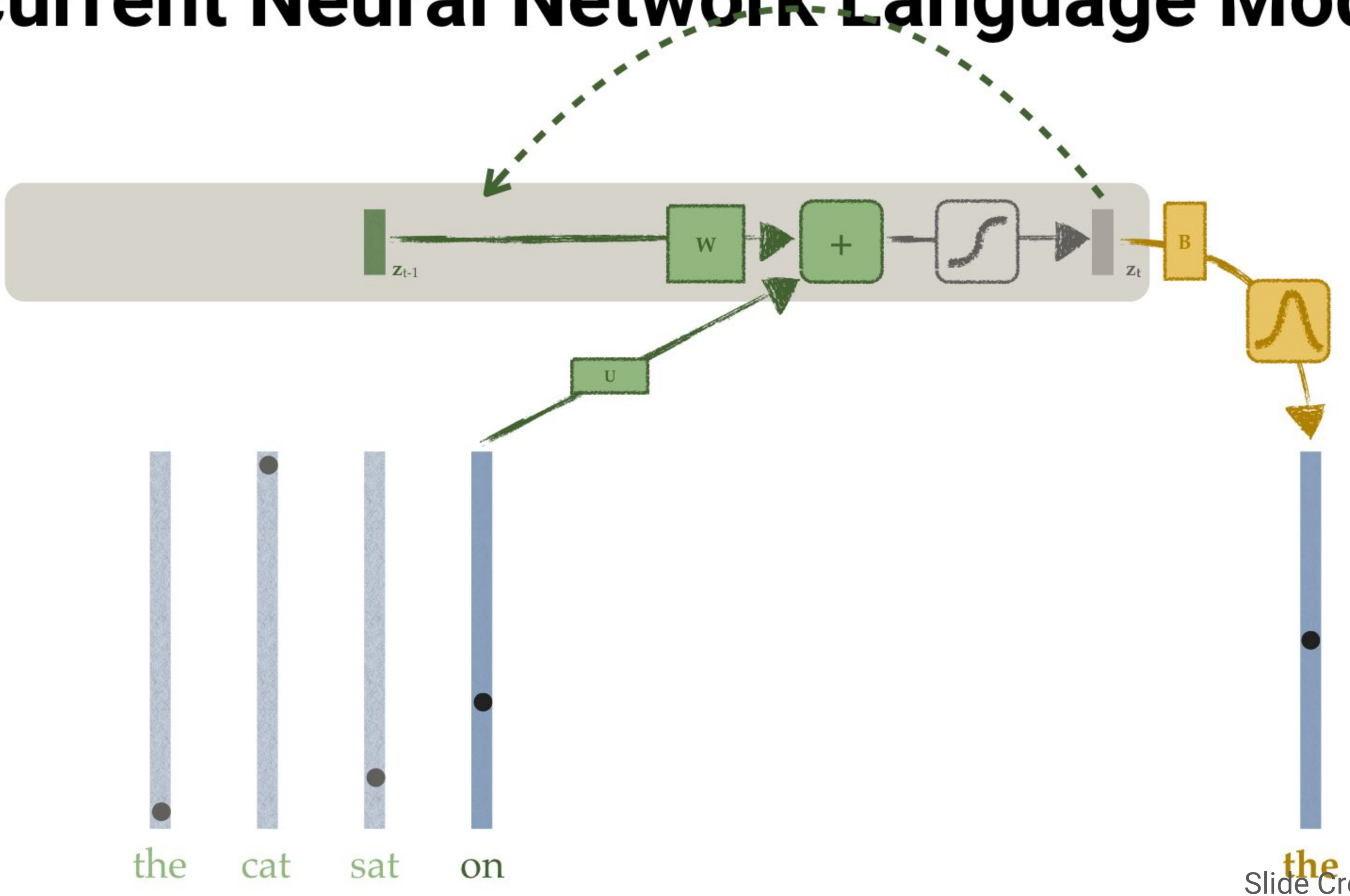
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



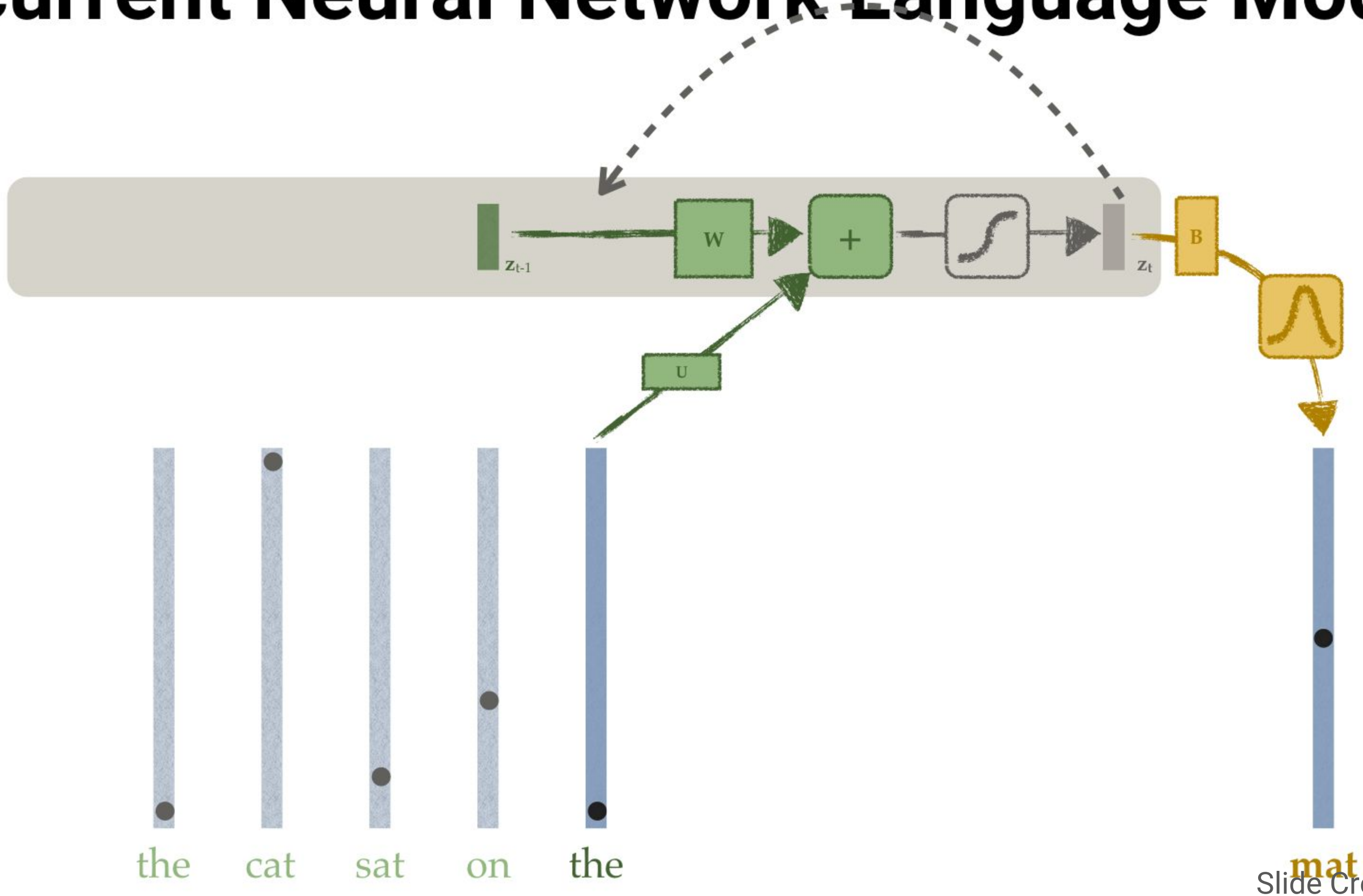
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



Slide Credit: Piotr Mirowski

What do we Optimize?

$$\theta^* = \arg\max_{\theta} E_{w \sim data} \log P_\theta(w_1, \ldots, w_T)$$

RNN language model

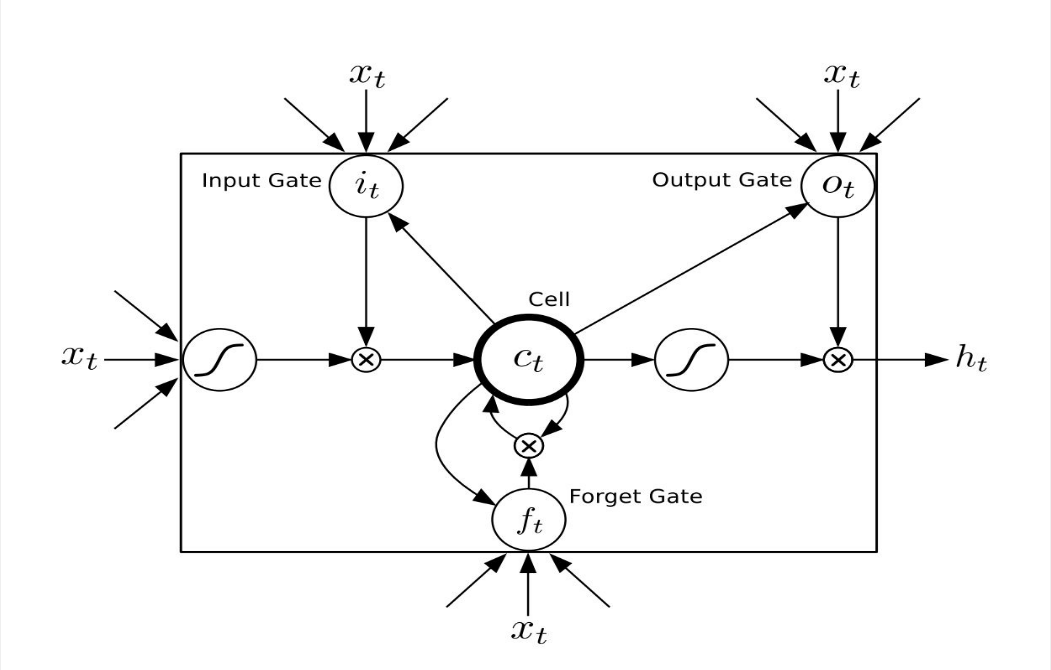
- Recurrent neural network based language model by Tomas Mikolov et al at Interspeech 2010

Model	PPL
KN 5gram	93.7
feedforward NN	85.1
recurrent NN	80.0
4xRNN + KN5	73.5

Exact architecture, e.g. num layers, num nodes and etc, used in the experiments is not clear.

- Simple experiment: 4M words from Switchboard corpus
- Feedforward networks used here are slightly different than what Bengio & Schwenk use

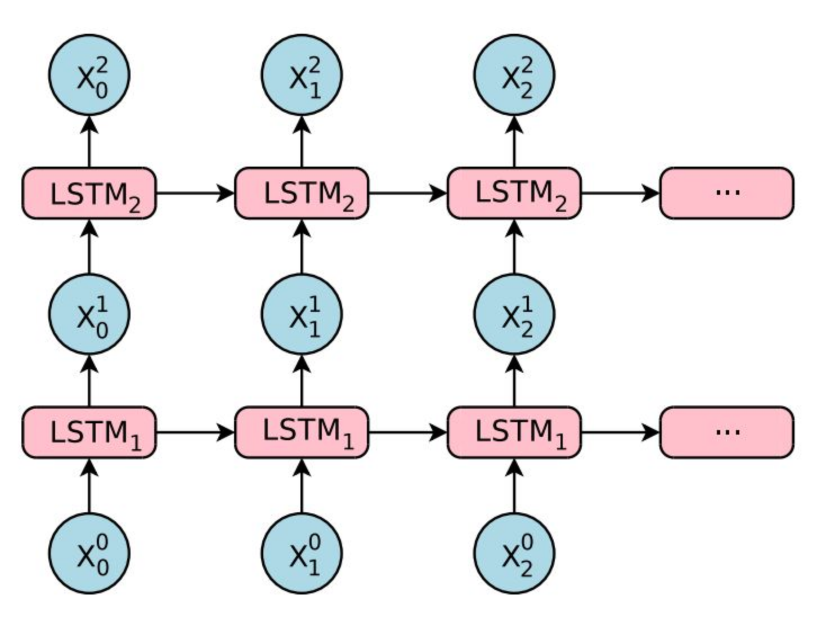
Long short-term memory network



$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)$$
$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$
$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)$$
$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$
$$h_t = o_t \tanh(c_t)$$

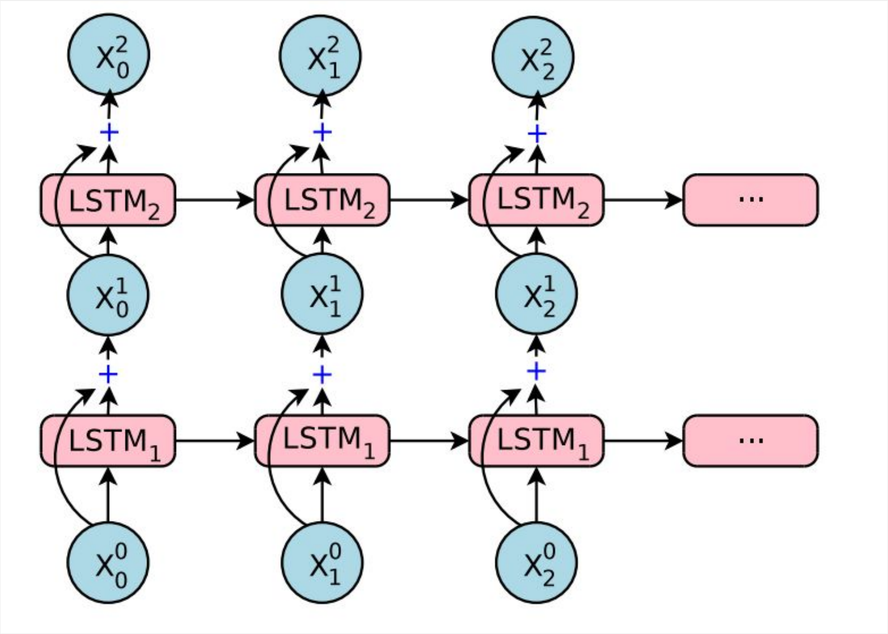
More powerful RNN

- To further improve the power of RNN networks, one can stack a few layers of RNN/LSTM.



Residual connection

- For deep RNN network, usually you need residual connections to stabilize/speed up training



A much better RNN language model

- Exploring the Limits of Language Modeling, by Rafal Jozefowicz

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8

The state of the art LM

- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, by Noam Shazeer and et al
- Stacked LSTM layers + Mixture of Expert layer in between

MOE

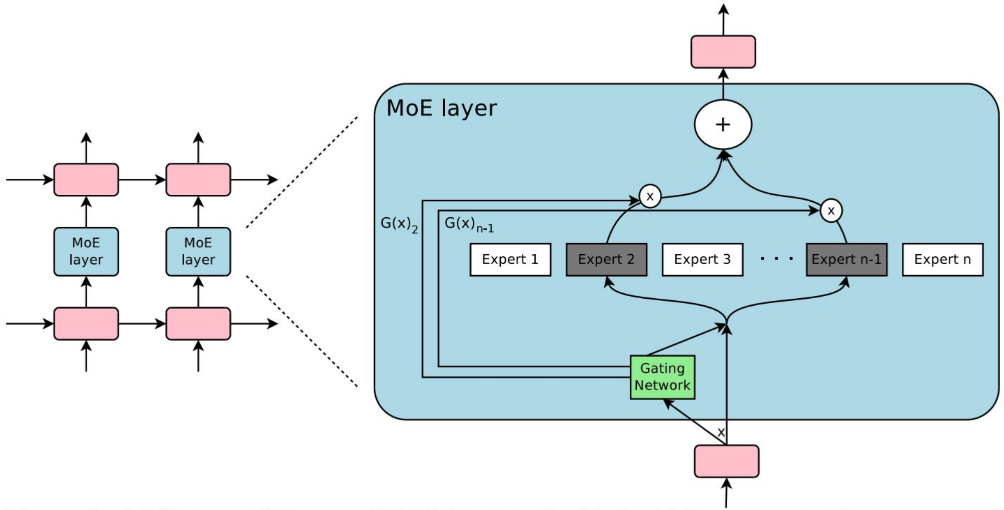


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$

MOE LM results

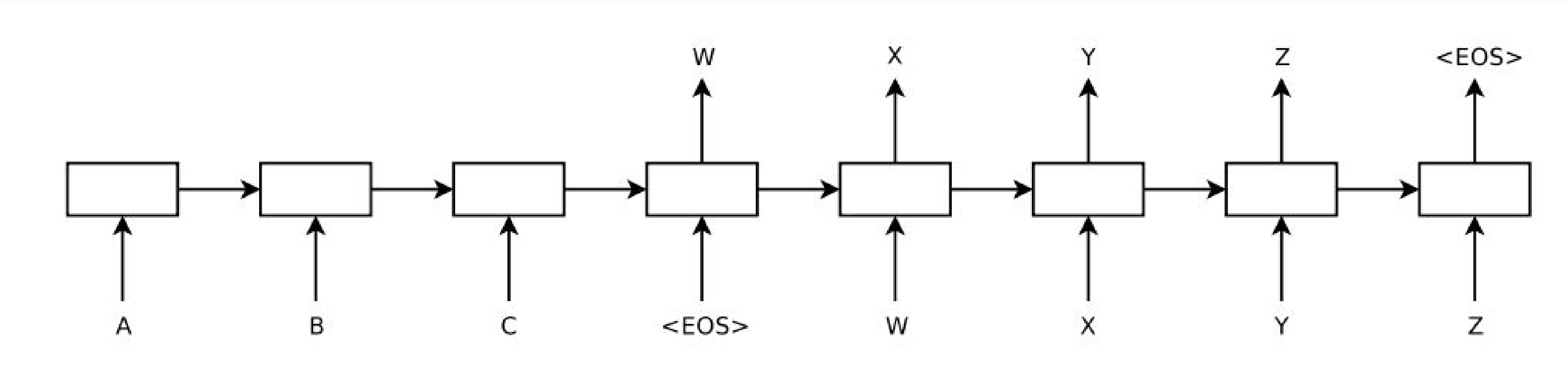
- Results on LM1B dataset

	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers	ops/timestep
Best Published Results	34.7	30.6	151 million	151 million
Low-Budget MoE Model	34.1		4303 million	8.9 million
Medium-Budget MoE Model	31.3		4313 million	33.8 million
High-Budget MoE Model	**28.0**		4371 million	142.7 million

Neural Machine Translation

seq2seq

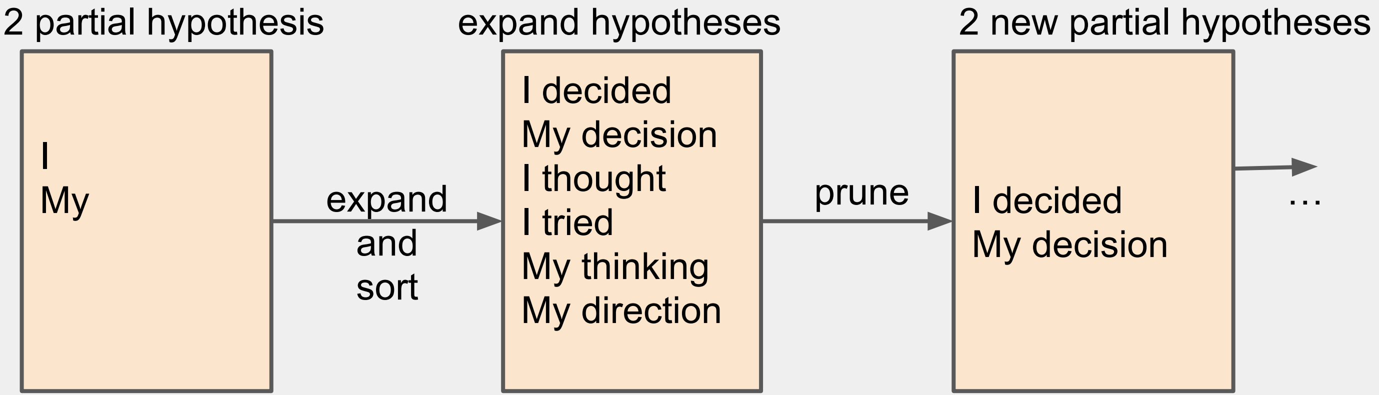
- Sequence to Sequence Learning with Neural Networks, by Ilya et al
  - Concatenate the source and target sequence
  - Only make prediction on the target sequence



$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Decoding in a Nutshell (Beam Size 2)

$$y^* = \arg\max_{y_1,\ldots,y_{T'}} P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$



Important Tricks

- When the model was first proposed, no one would have believed that it can solve the translation problem
- But it worked quite well
- Tricks that are important for the model quality
  - Reverse the source sequence
  - Deep lstms (4 layers lstm networks)

Seq2Seq experimental results

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	**34.81**

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.

Limitation of the model

- Information bottleneck
  - Regardless of the sequence length, source sequence is encoded using fixed size vectors
  - Solution: Attention
- Out of vocabulary (OOV) problems
  - Fixed size vocab, constrained by computational budget and GPU RAM.
  - All words not in vocab will be mapped to the same <UNK> token.
  - More of a problem for morphologically rich languages like Russian and Polish.
  - Solution: WordPiece or BytePair encoding

Attention

- *Generating Sequences With Recurrent Neural Networks*, by Alex Graves
  - Motonic, only moves left to right
  - Proposed to solve the handwriting synthesis issue
  - It is referred to Gaussian Mixture Model (GMM) Attention in the literature
- Generalized by Dzmitry Bahdanau in his paper “Neural Machine Translation by Jointly Learning to Align and Translate”
  - No more monotonicity constraints

Online handwriting synthesis

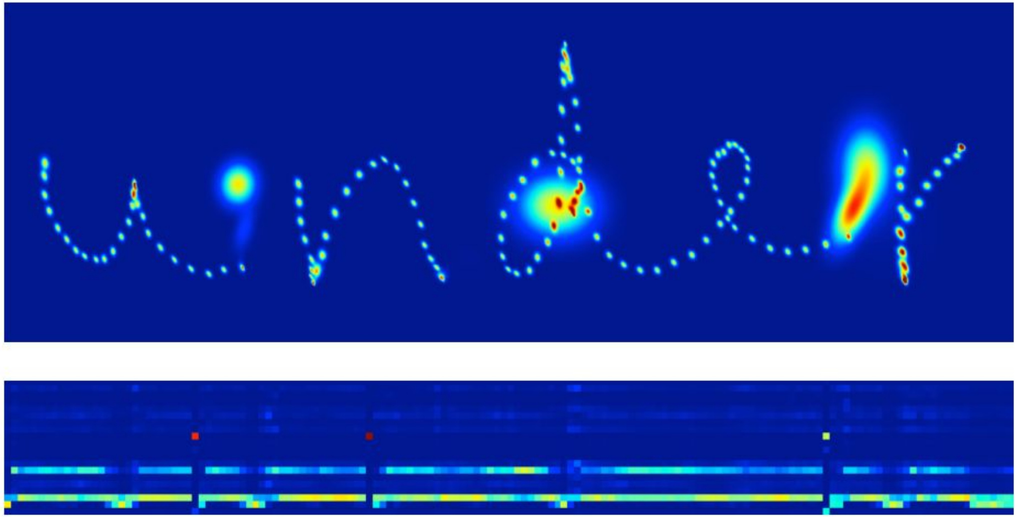


Figure 14: **Mixture density outputs for handwriting synthesis.** The top heatmap shows the predictive distributions for the pen locations, the bottom heatmap shows the mixture component weights. Comparison with Fig. 10 indicates that the synthesis network makes more precise predictions (with smaller density blobs) than the prediction-only network, especially at the ends of strokes, where the synthesis network has the advantage of knowing which letter comes next.

GMM Attention

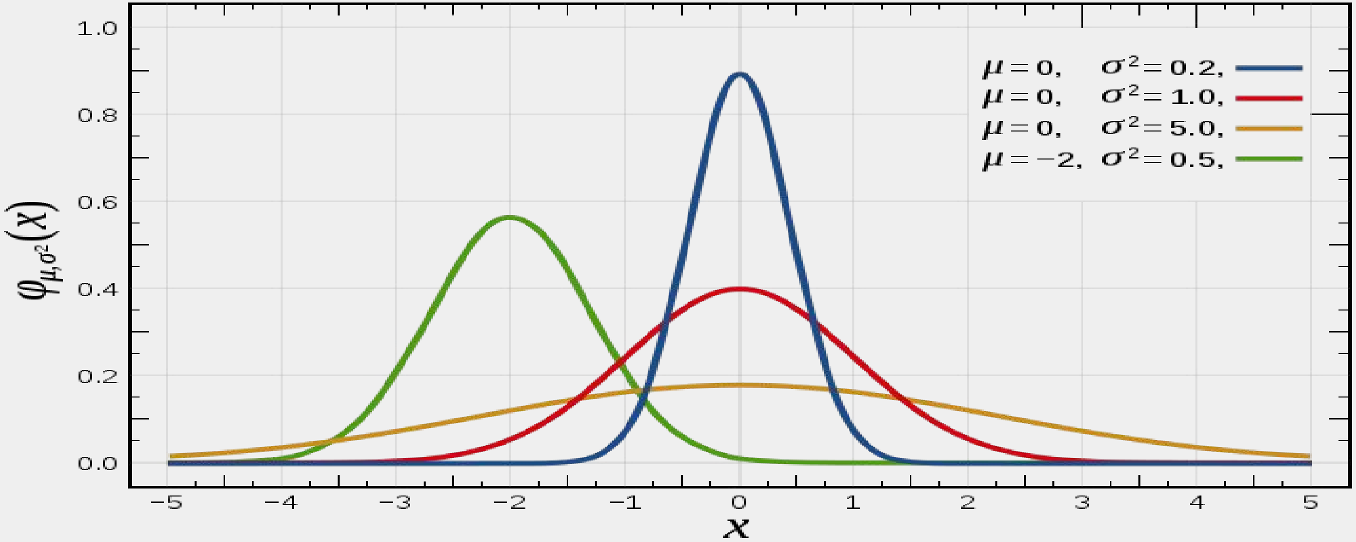
- K Gaussian components
- Each defines a density distribution over the character sequence

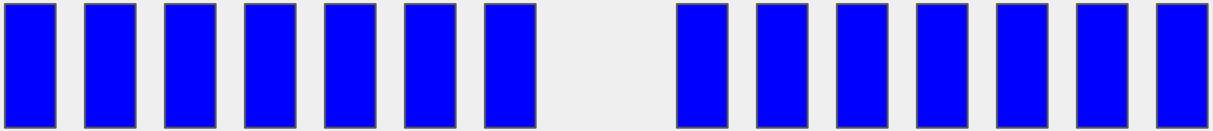
Given a length U character sequence $\mathbf{c}$ and a length T data sequence $\mathbf{x}$, the soft window w_t into $\mathbf{c}$ at timestep t ($1 \leq t \leq T$) is defined by the following discrete convolution with a mixture of K Gaussian functions

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right) \tag{46}$$

$$w_t = \sum_{u=1}^{U} \phi(t,u) c_u \tag{47}$$

Gaussian Mixture Model





t h e c a t

GMM attention

- GMM params are updated at every timestep.
- Attention params are estimated using decoder rnn hidden states
- Modeled as shift from the previous center. Hence monotonicity guarantee

$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1p} h_t^1 + b_p$$

$$\alpha_t = \exp\left(\hat{\alpha}_t\right)$$

$$\beta_t = \exp\left(\hat{\beta}_t\right)$$

$$\boxed{\kappa_t = \kappa_{t-1} + \exp\left(\hat{\kappa}_t\right)}$$

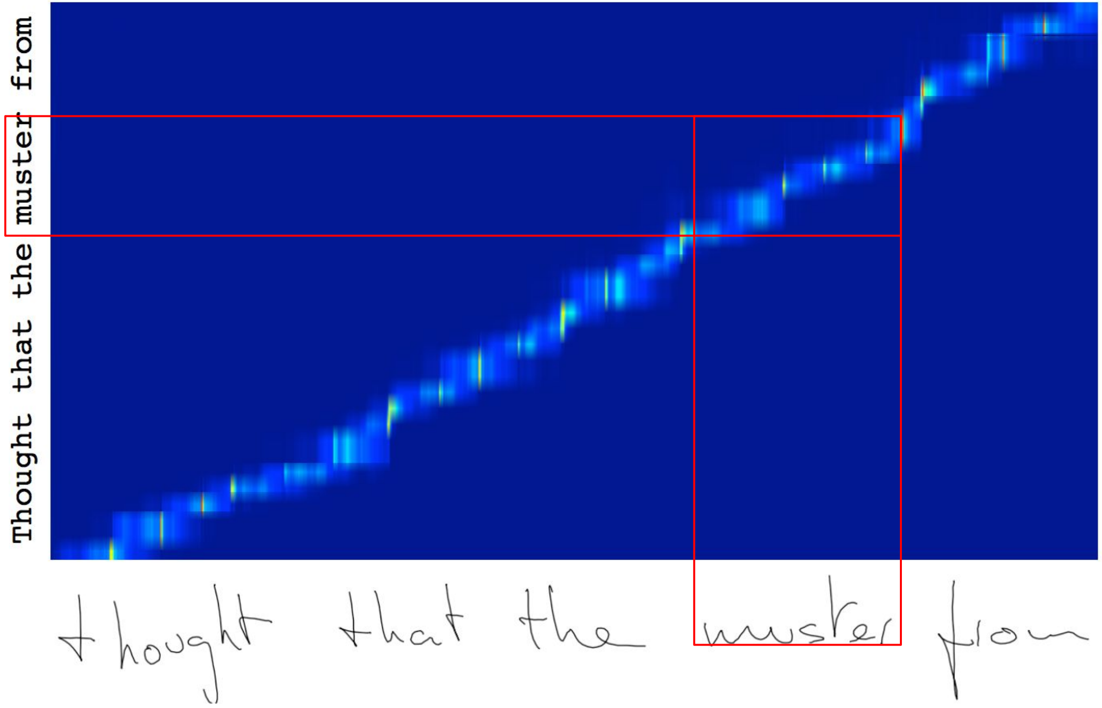
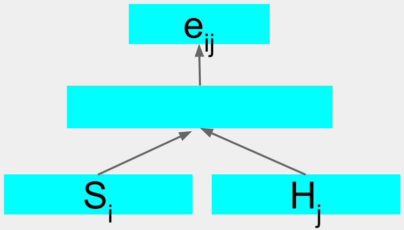


Figure 13: **Window weights during a handwriting synthesis sequence** Each point on the map shows the value of $\phi(t, u)$, where t indexes the pen trace along the horizontal axis and u indexes the text character along the vertical axis. The bright line is the alignment chosen by the network between the characters and the writing. Notice that the line spreads out at the boundaries between characters; this means the network receives information about next and previous letters as it makes transitions, which helps guide its predictions.

A generalization of GMM attention

- *Neural Machine Translation by Jointly Learning to Align and Translate*, by Dzmitry Bahdanau, et al

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$



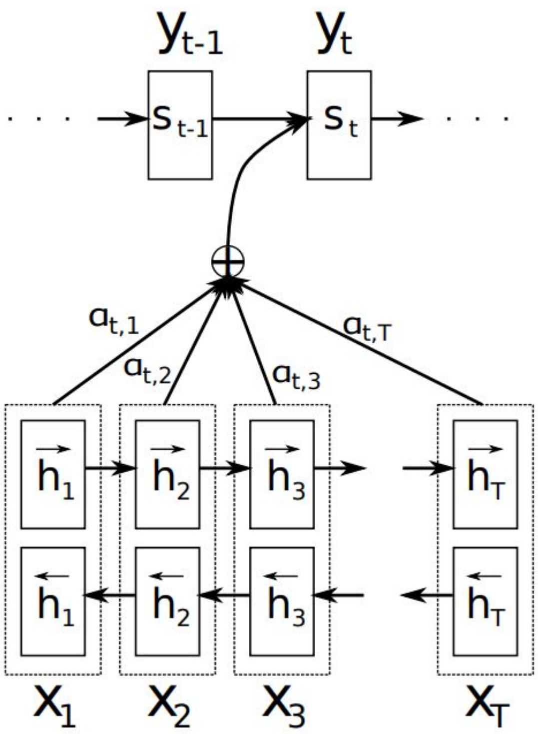


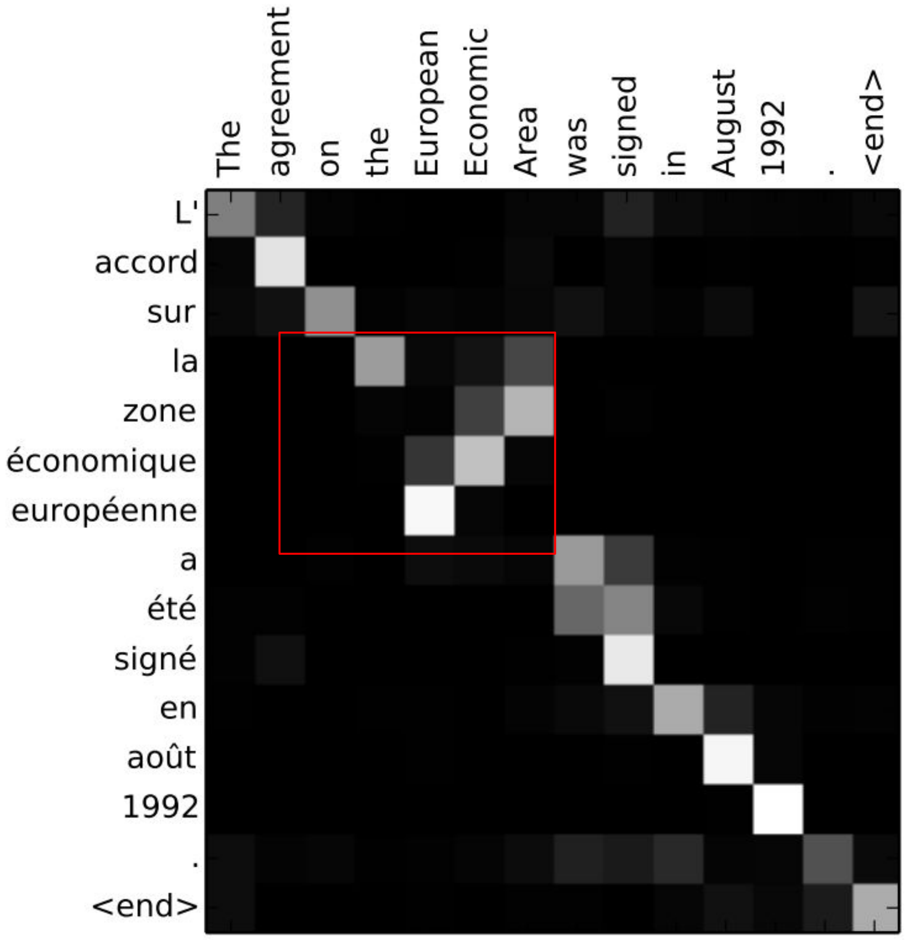
Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention benefits

- At any point of decoding, the model only focuses on the most relevant part of the source, much like how human do
- Encoding of the source sequence is now distributed over all the source words.
  - Longer sequence is encoded using more bits

Attention visulization

Attention probability matrix can be nicely visualized



(a)

Experimental result

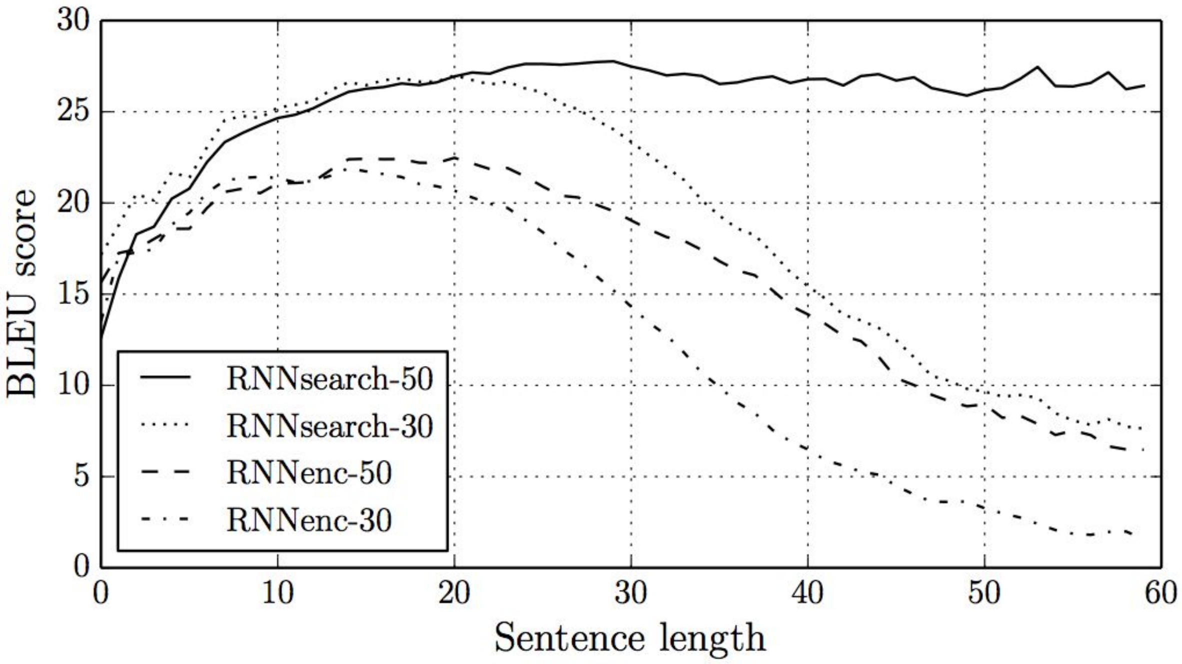


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Convolution Attention

- *Attention-Based Models for Speech Recognition*, by Jan Chorowski et al
- Key idea: explicitly incorporate a location information
- Encourages attention to gradually move forward

We extend this content-based attention mechanism of the original model to be location-aware by making it take into account the alignment produced at the previous step. First, we extract k vectors $f_{i,j} \in \mathbb{R}^k$ for every position j of the previous alignment α_{i-1} by convolving it with a matrix $F \in \mathbb{R}^{k \times r}$:

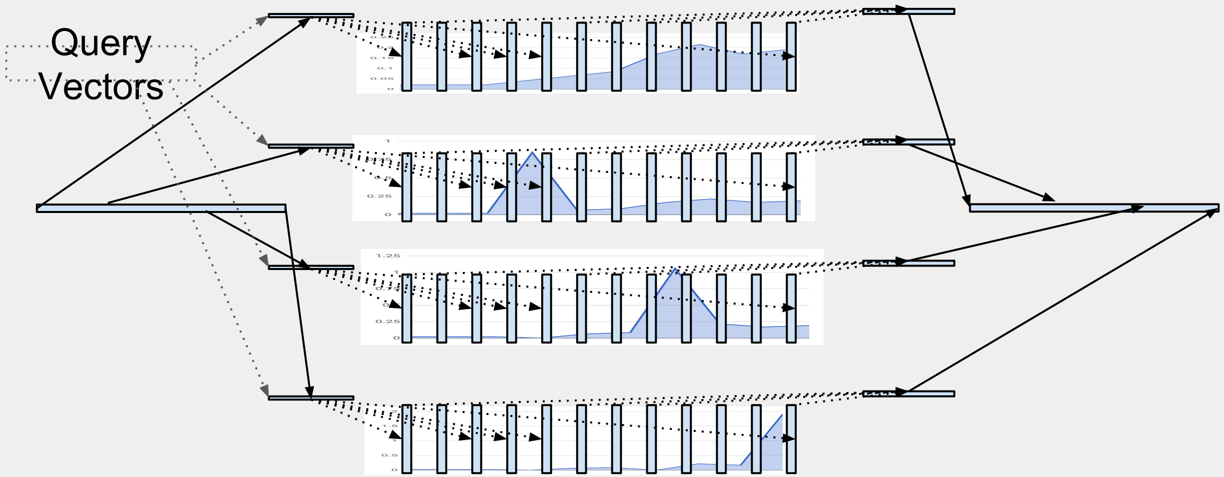
$$\boxed{f_i = F * \alpha_{i-1}.} \tag{8}$$

These additional vectors $f_{i,j}$ are then used by the scoring mechanism $e_{i,j}$:

$$e_{i,j} = w^\top \tanh(W s_{i-1} + V h_j + \boxed{U f_{i,j}} + b) \tag{9}$$

Multi-headed attention

- Simple extension to single headed attention
- Multiple attention runs in parallel. Each individual attention may focus on a different region in the input



Motonic Attention

- *Online and Linear-Time Attention by Enforcing Monotonic Alignments*, by Colin
- Linear time complexity
- Guarantees monotonicity

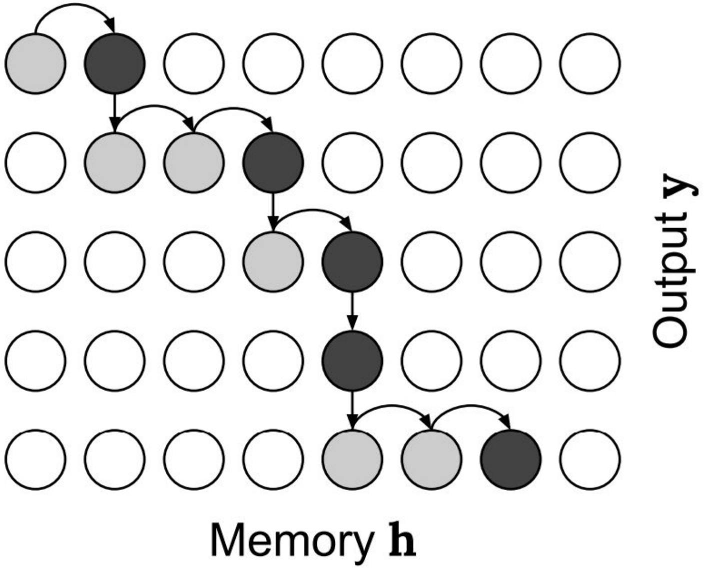


Figure 2. Schematic of our novel monotonic stochastic decoding process. At each output timestep, the decoder inspects memory entries (indicated in gray) from left-to-right starting from where it left off at the previous output timestep and chooses a single one (indicated in black). A black node indicates that memory element h_j is aligned to output y_i. White nodes indicate that a particular input-output alignment was not considered because it violates monotonicity. Arrows indicate the order of processing and dependence between memory entries and output timesteps.

Solve the OOV problem

- Move to sub-word unit
- Character model
  - Sequence length too long
- Wordpiece model strikes a good balance between character model and word model
  - Common words are still just just one wordpiece
  - Rare words are often decomposed into a morphologically meaningful way, word-root and suffix

WordPiece model / Bytepair Encoding

- *Japanese and Korean voice search*, by Mike Schuster
- Same as bytepair encoding, which Rico Sennrich first adopted for NMT
    - *Neural Machine Translation of Rare Words with Subword Units*
-
- Minimal description length
    - Find an encoding of words subject to vocab size limit such that a corpus can be encoded using minimal number of tokens

Simple greedy algorithm

- Very simple algorithm
  - Start from all characters
  - Iteratively merge most frequent bi-grams
  - Stop when vocab size is reached
- Most common word is a single wordpiece

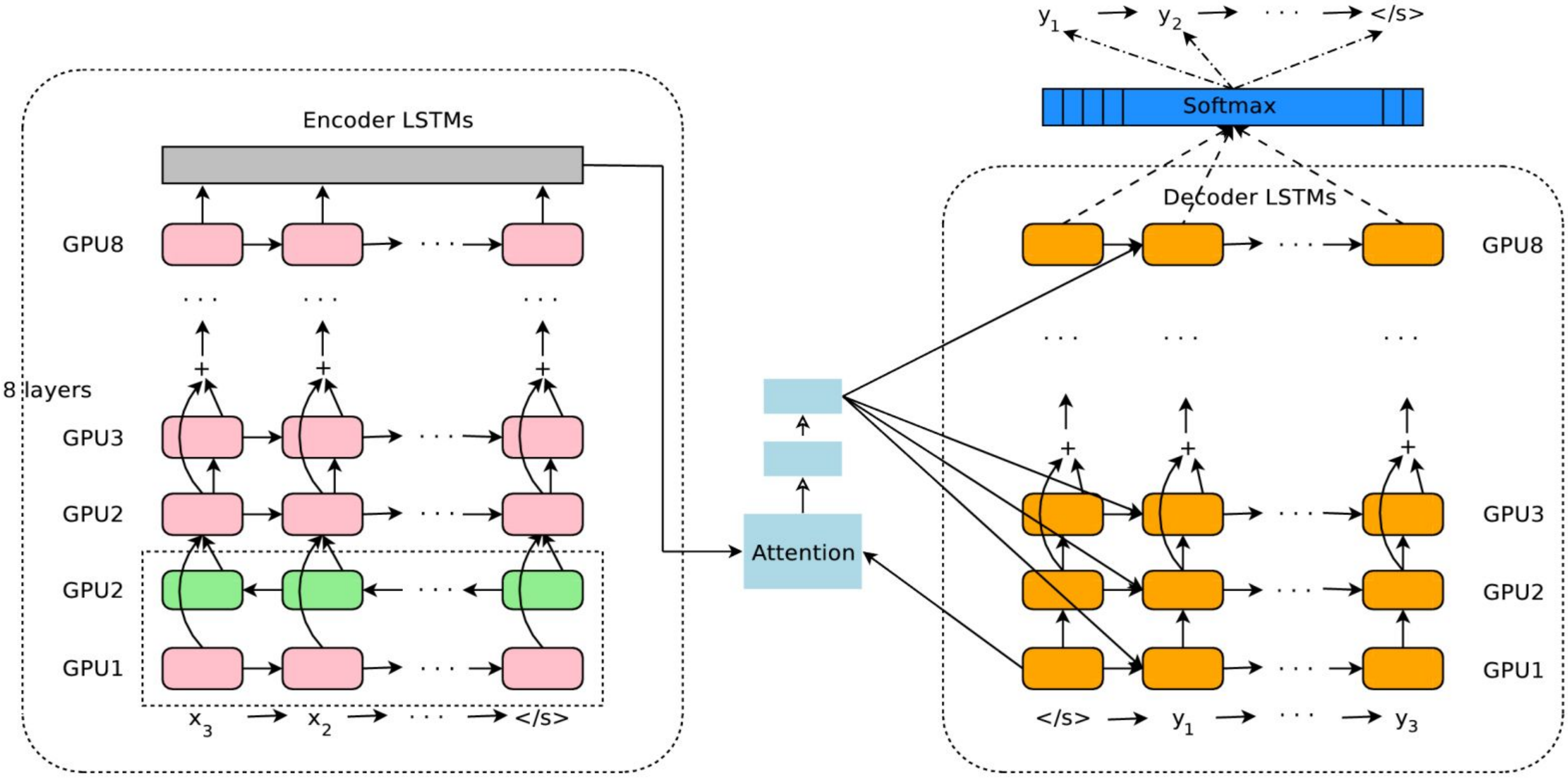
- **Word:** Jet makers feud over seat width with big orders at stake
- **wordpieces:** _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

One more important trick

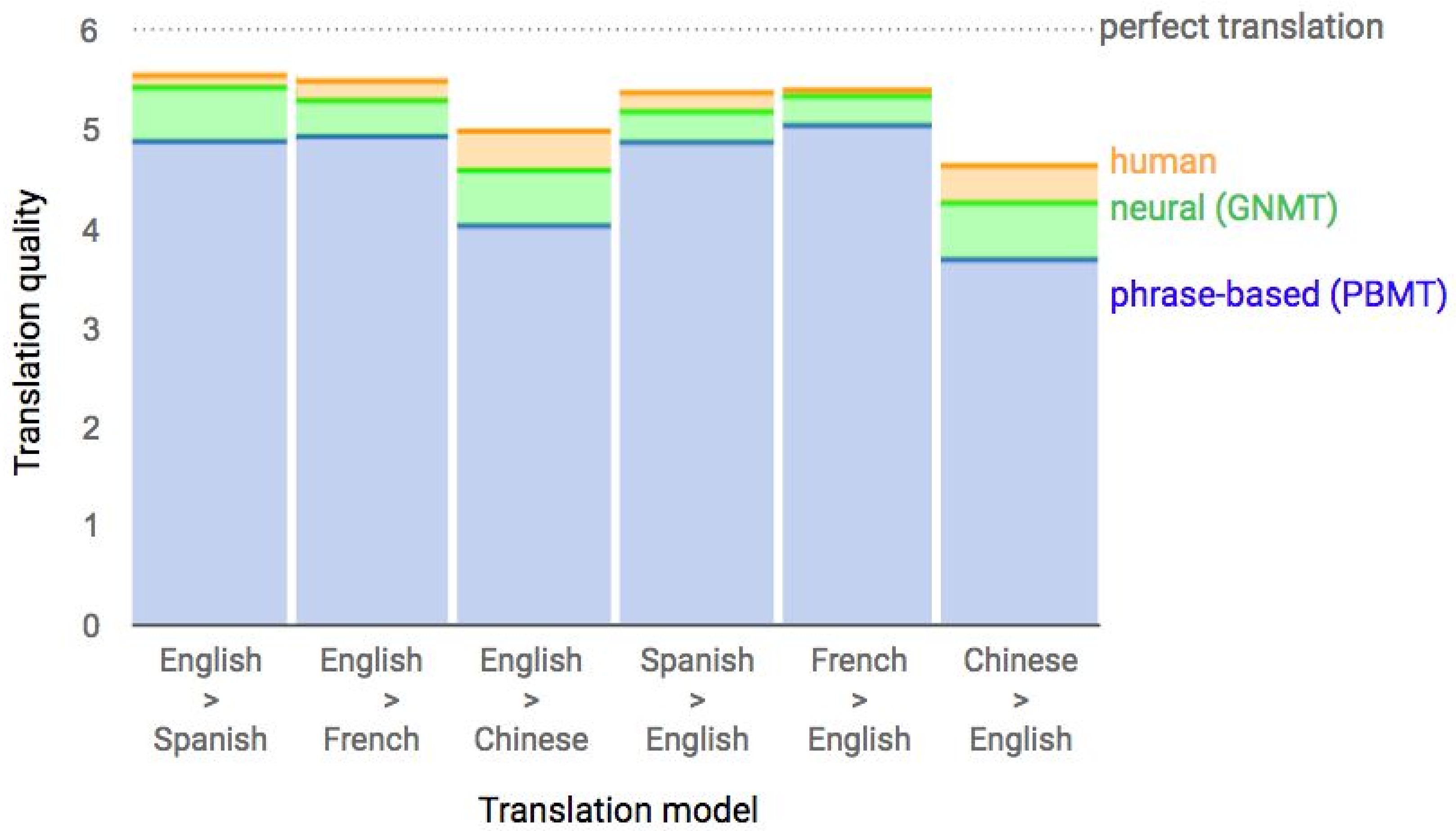
- Rare words are often copied from source to target in verbatim
- To facilitate such copying, source and target should share the same WordPiece model
    - Rare words will be segmented exactly the same way

Google Neural Machine Translation

- All the techniques combined
  - Very deep lstm stack
  - Attention
  - Wordpiece model
- Large scale NMT
  - Carefully engineered
  - Pieces are carefully tuned and assembled together
- Some novel inventions
  - Better decoding algorithm
  - Quantized training and inference
  - ...



Encoder LSTMs
GPU8
GPU3
GPU2
GPU2
GPU1
8 layers
x3 → x2 → · · · → </s>
Attention
y1 → y2 → · · · → </s>
Softmax
Decoder LSTMs
GPU8
GPU3
GPU2
GPU1
</s> → y1 → · · · → y3

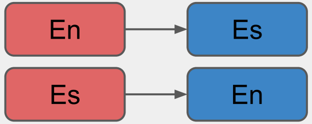


6
5
4
3
2
1
0
Translation quality
perfect translation
human
neural (GNMT)
phrase-based (PBMT)
English > Spanish
English > French
English > Chinese
Spanish > English
French > English
Chinese > English
Translation model

Multilingual Model

- Model several language pairs in single model
  - We ran first experiments in 2/2016, surprisingly this worked!
- Prepend source with additional token to indicate target language
  - Translate to Spanish:
    - <2es> How are you </s> -> Cómo estás </s>
  - Translate to English:
    - <2en> Cómo estás </s> -> How are you </s>
- No other changes to model architecture!
  - Extremely simple and effective
  - Usually with shared WPM for source/target
- Benefits
  - simplifies model deployment
  - improves translation quality

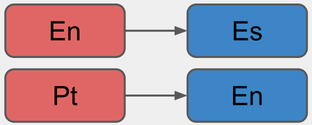
Multilingual Model



Single	Multi
34.5	**35.1**
38.0	37.3

Translation:
<2es> How are you </s> Cómo estás </s>
<2en> Cómo estás </s> How are you </s>

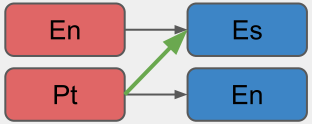
Zero-Shot Translation



single	multi
34.5	**35.0**
44.5	43.7

Zero-shot (pt->es):

<2es> Como você está </s> Cómo estás </s>



23.0 BLEU

Some more recent NMT models

- MOE model, from Google Brain
- Conv seq2seq model, from Facebook
- Transformer Model, from Google Brain

Mixture of experts NMT model

- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, by Noam Shazeer et al
- Based on the GNMT model architecture
  - Add a Mixture of Exports layer between the first and second LSTM layers
  - Adds a lot of params, but actually reduces number of compute

Table 2: Results on WMT'14 En→ Fr newstest2014 (bold values represent best results).

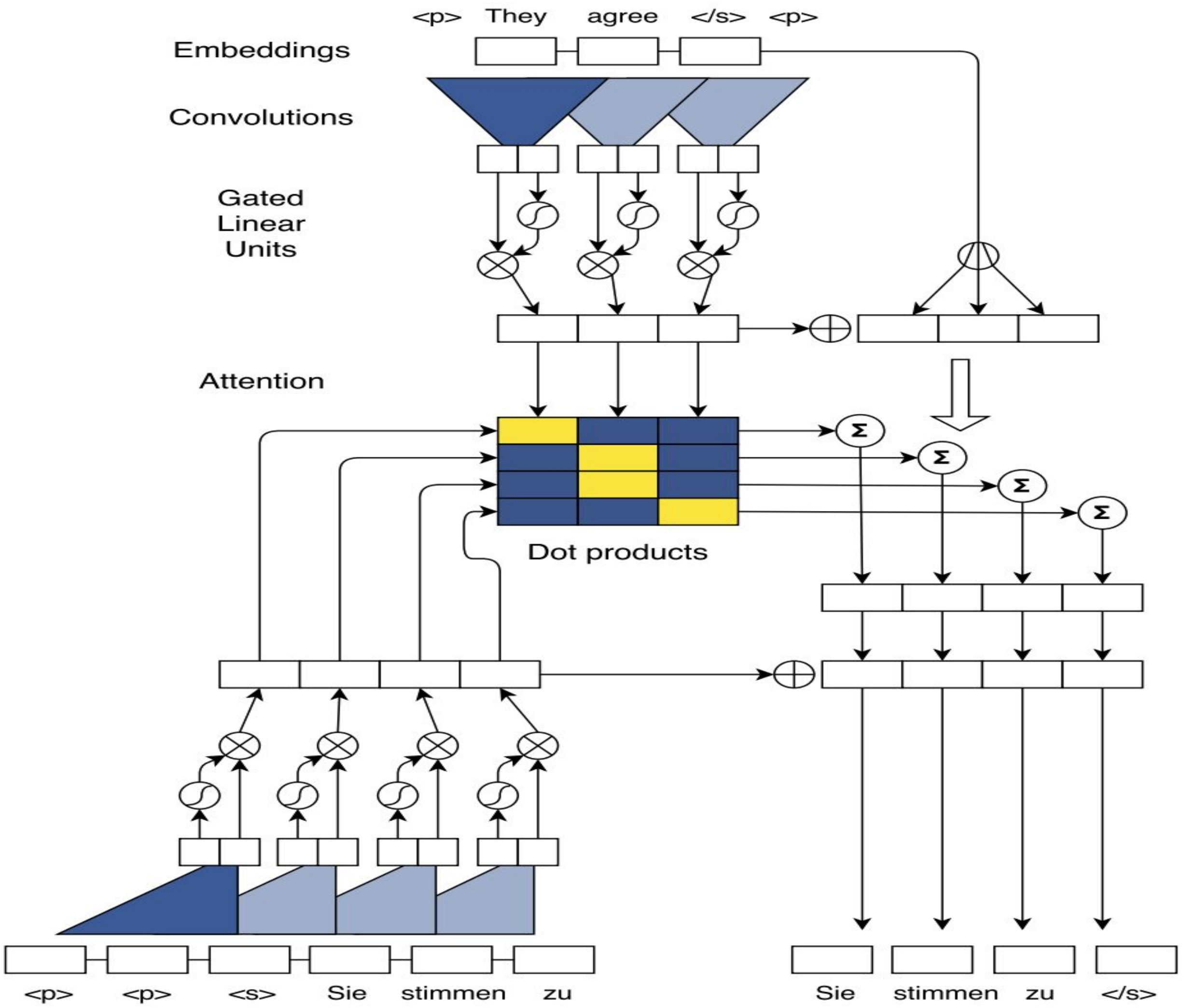
Model	Test Perplexity	Test BLEU	ops/timenstep	Total #Parameters	Training Time
MoE with 2048 Experts	2.69	40.35	85M	8.7B	3 days/64 k40s
MoE with 2048 Experts (longer training)	**2.63**	**40.56**	85M	8.7B	6 days/64 k40s
GNMT (Wu et al., 2016)	2.79	39.22	214M	278M	6 days/96 k80s
GNMT+RL (Wu et al., 2016)	2.96	39.92	214M	278M	6 days/96 k80s
PBMT (Durrani et al., 2014)		37.0			
LSTM (6-layer) (Luong et al., 2015b)		31.5			
LSTM (6-layer+PosUnk) (Luong et al., 2015b)		33.1			
DeepAtt (Zhou et al., 2016)		37.7			
DeepAtt+PosUnk (Zhou et al., 2016)		39.2			

Conv sequence to sequence model

- Convolutional Sequence to Sequence Learning, by Jonas Gehring et al
- Main difference from GNMT, lstm is replaced by gated linear units

$$v([A\ B]) = A \otimes \sigma(B)$$

- Other difference from GNMT
    - Attention at every decoding layer
    - Dot product based attention
    - Position embedding


<p> They agree </s> <p>
Embeddings
Convolutions
Gated Linear Units
Attention
Dot products
<p> <p> <s> Sie stimmen zu
Sie stimmen zu </s>

Conv sequence to sequence learning

- Very nice results
- Appears to be very sensitive to params initialization, hard to reproduce

WMT'14 English-French	BLEU
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51

Transformer Model

- *Attention is all you need,* by Ashish and Noam from the Google Brain team.
- Building blocks
  - Multi-headed Self-attention
  - Multi-layer Multi-headed Attention
  - Hand engineered position embedding
  - Residual connections and layer normalization to stabalize model training
- It is the current state of the art

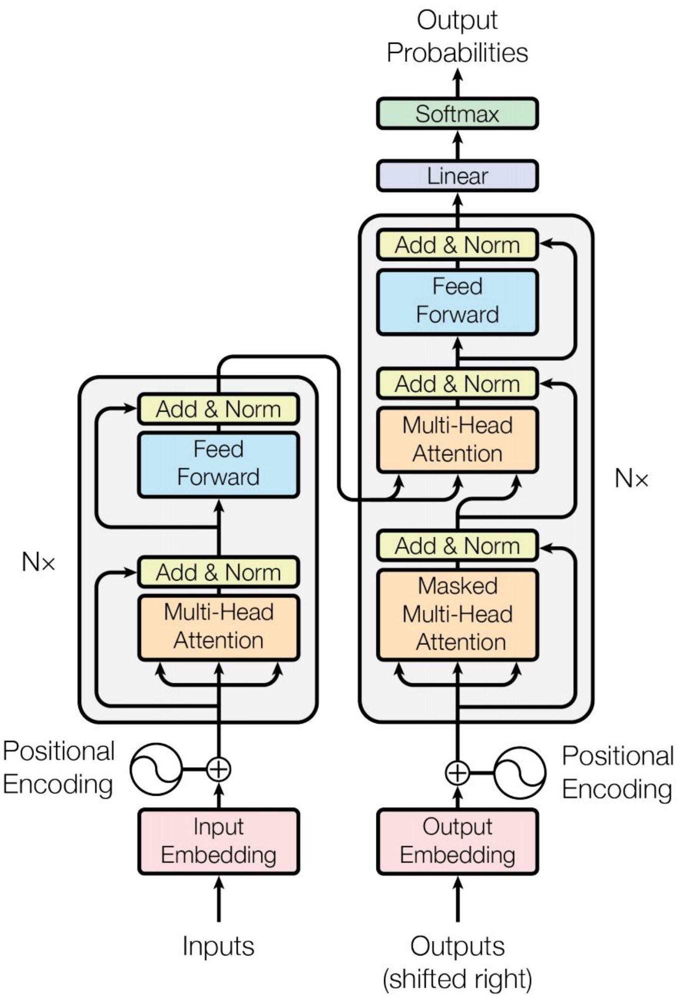
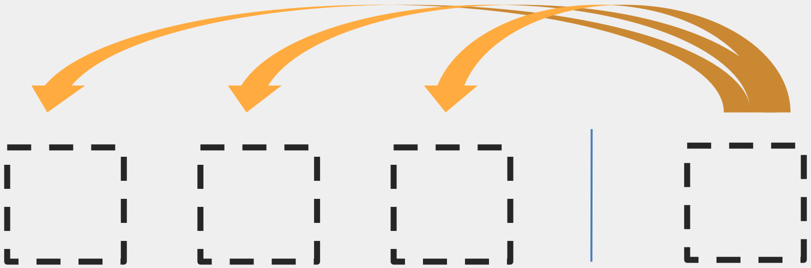
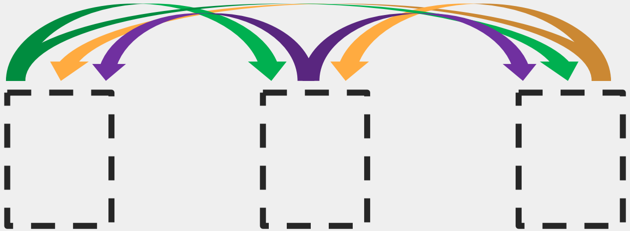


Figure 1: The Transformer - model architecture.

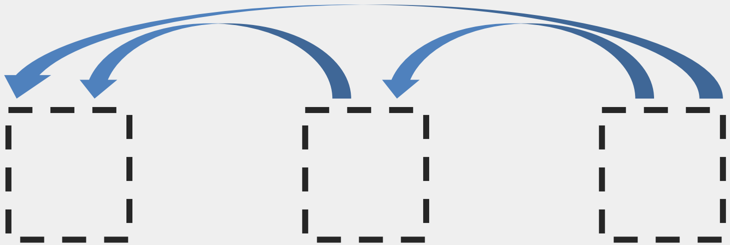
Three ways of attention



Encoder-Decoder Attention

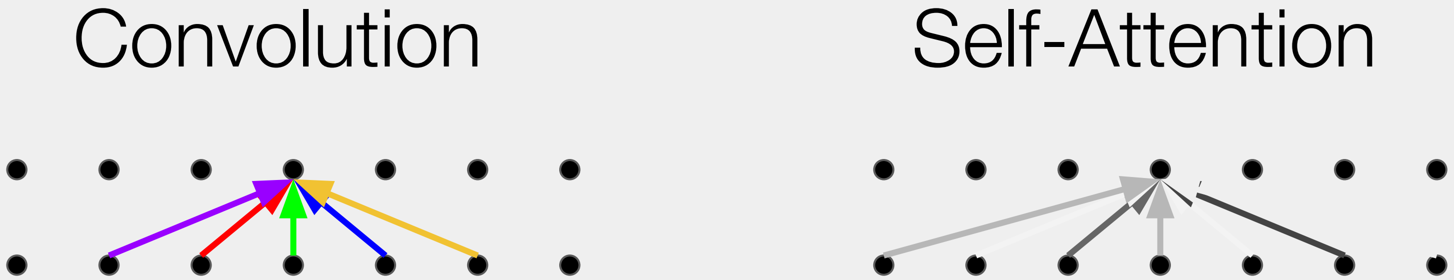


Encoder Self-Attention



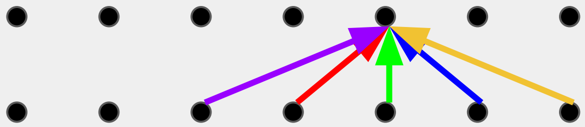
MaskedDecoder Self-Attention

Self-Attention



Self-Attention

Convolution



Self-Attention

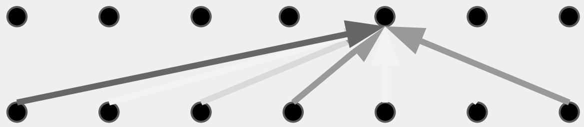


Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Further reading

- Neural Machine Translation, book chapter

Speech Recognition

Sequence models in ASR

- Covers a few popular end-to-end ASR models: CTC, RNN-T, LAS
- CTC is by Alex Graves
  - *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*, by Alex Graves
- RNN-T is by Alex Graves
  - *Sequence Transduction with Recurrent Neural Networks*, by Alex Graves
- LAS is by Google Brain and Yoshua's group
  - *Listen, Attend and Spell*, by William Chan et al
  - *Attention-Based Models for Speech Recognition,* by Jan Chorowski et al

Speech frontend

- To transform the raw speech signal into a format that is more suitable for ASR
  - Keeps only the essential information in speech
- Trend is speech frontend is getting simpler and simpler
  - Modern ASR system is capable of extracting useful information from noisy signal directly.
  - Complex frontend runs the risk of throwing away useful information
- Research suggest that you can even get rid of the ASR directly.
  - *Learning the Speech Front-end With Raw Waveform CLDNNs,* by Tara and et al.

Log mel filter bank energies

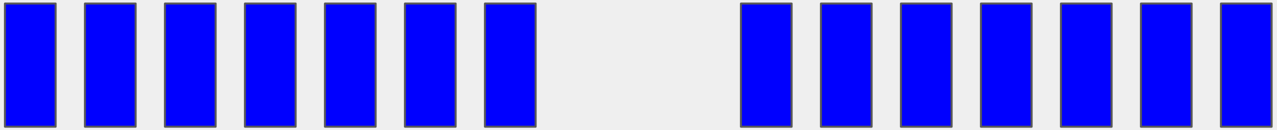
- Take a short time fourier transform of the speech signal with an appropriate window size and shift. Typical window size and shift are 25ms and 10ms respectively.
- Convert frequency energies from linear scale to a mel scale, using triangular overlapping windows.
- Take the log of the energy

CTC model

- *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*, by Alex Graves
- It is a quite old model, proposed in 2005
- Used in Baidu's DeepSpeech and DeepSpeech2 systems

Alignment

- Model output is frame synchronous
    - The model output one label symbol at every frame
- To match the output sequence to the input sequence in length
    - blank symbol, "_"
    - repeated symbols

The cat sat on the mat	Label sequence
t h h _ _ e e … _ _ m _ a t t	Alignment #1
t t _ h e e _ … m m a _ t _ _	Alignment #2
	Spectrogram

CTC loss

- CTC loss assumes "temporal independence" in the output layer
- Given an alignment of label sequence to input sequence, P(alignment | input) is trivial to compute
  - It is simply the product of P(label symbol) at each frame
- P(label | input) = sum_{all valid alignments}(P(alignment | input))
  - Exponentially many valid alignments
  - Dynamic programming algorithm available to compute this sum exactly
- CTC loss = $E_{\{label, input\}}$log(P(label | input))

CTC loss

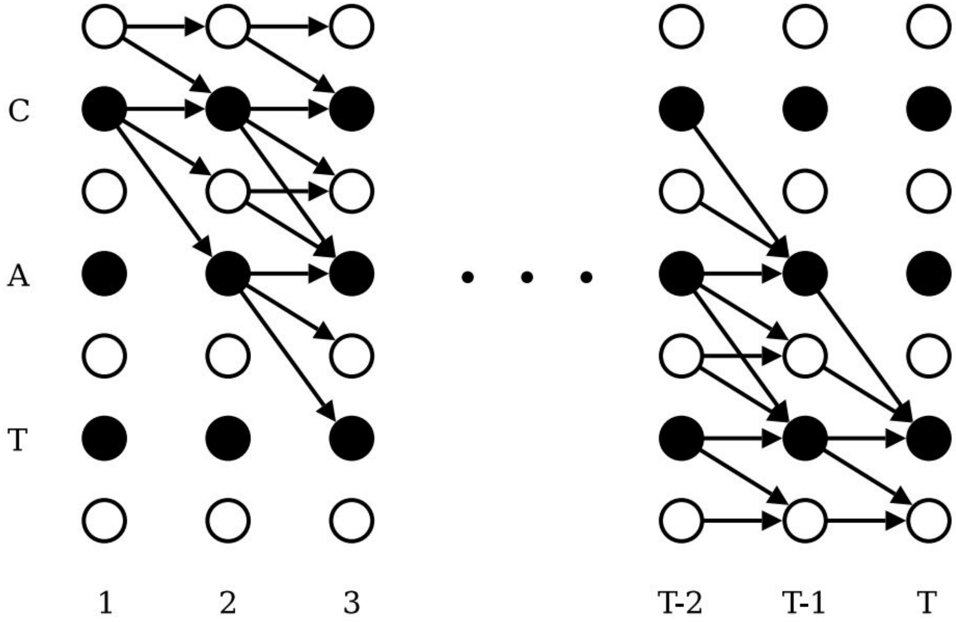


Figure 3. **illustration of the forward backward algorithm applied to the labelling 'CAT'.** Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

CTC decoding

- Greedy decoding
  - Find an alignment such that P(alignment|input) is maximized
  - Take the most probable symbol at each time step
  - Remove repeated symbols and blank symbols
- Dynamic programming based decoding
  - Find the label sequence such that P(label|input) is maximized
  - Can use an algorithm similar to the one used in training

Nice extension of the CTC model

- *Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling*, by Hairong Liu et al
- Segmentation is fixed in standard CTC, e.g. “CAT” to “C”, “A” and “T”.
- Key idea is to allow the network to learn a valid segmentation at the same time
    - CAT -> “C”, “A”, “T”
    - CAT -> “C”, “AT”
    - CAT -> “CA”, “T”
    - CAT -> “CAT”

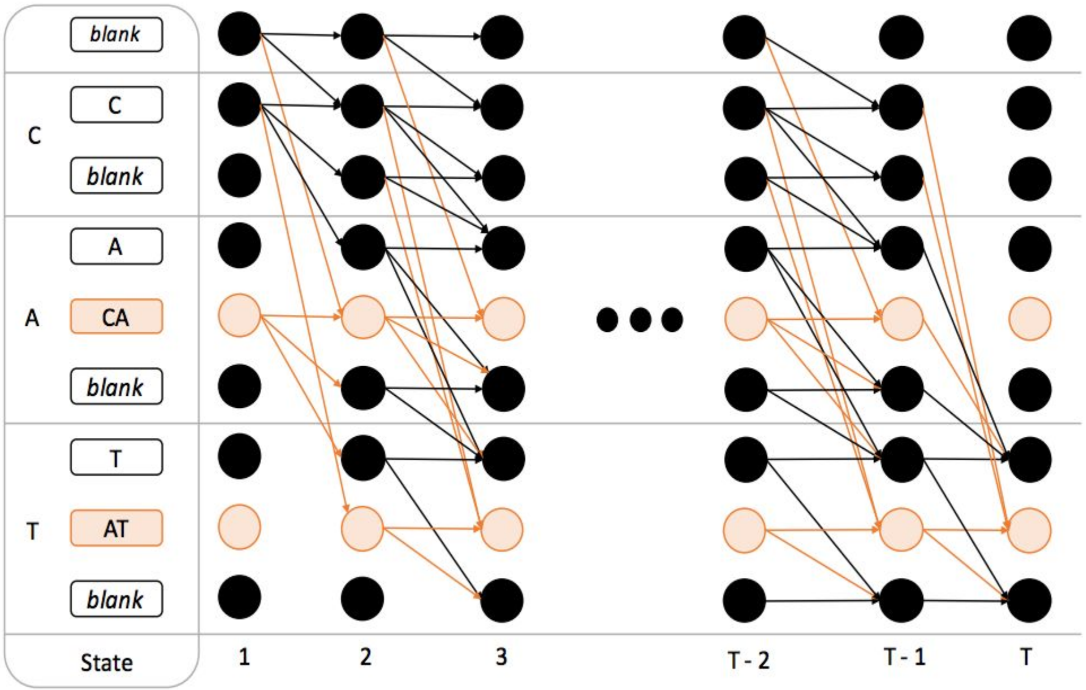


Figure 1. Illustration of the states and the forward-backward transitions for the label 'CAT'. Here we let G be the set of all uni-grams and bi-grams of the English alphabet. The set of all valid states S for the label l = 'CAT' are listed to the left. The set of states and transitions that are common to both vanilla and Gram-CTC are in black, and those that are unique to Gram-CTC are in orange. In general, any extension that collapses back to l is a valid transition - For example, we can transition into ('CAT', 1) from ('CAT', 1), ('CA', 2), ('CA', 1) and ('CA', 0) but not from ('CAT', 0) or ('CAT', 2)

CTC

- It is fairly stable to train
  - Guaranteed monotonicity in alignment
- Natualy produces an alignment as a by-product
  - Can be useful for other applications
- Very weak language model
  - Usually need to combine with a strong external language model for it to perform well

RNN-T

- Proposed to address the limitations in the CTC model, namely the temporal independence assumption
- Two recurrent neural networks
  - One on the source sequence
  - The other on the label sequence
- Target side RNN allows the model to learn a much better build-in language model
- A separate joint network on top to combine the source sequence RNN and the target sequence RNN

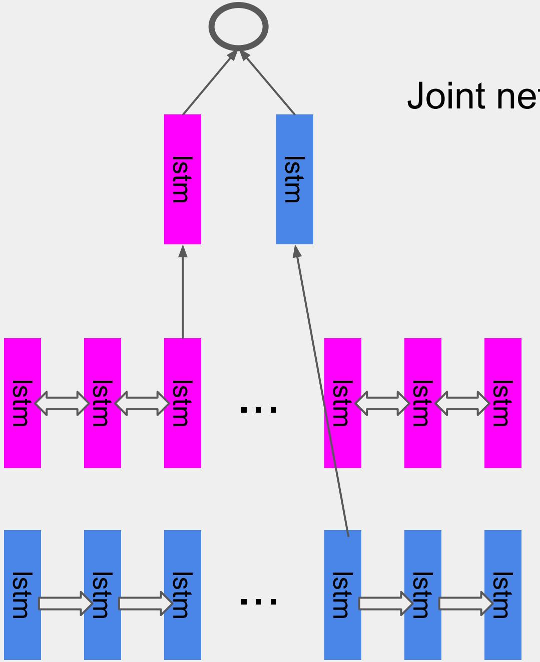
RNN-T

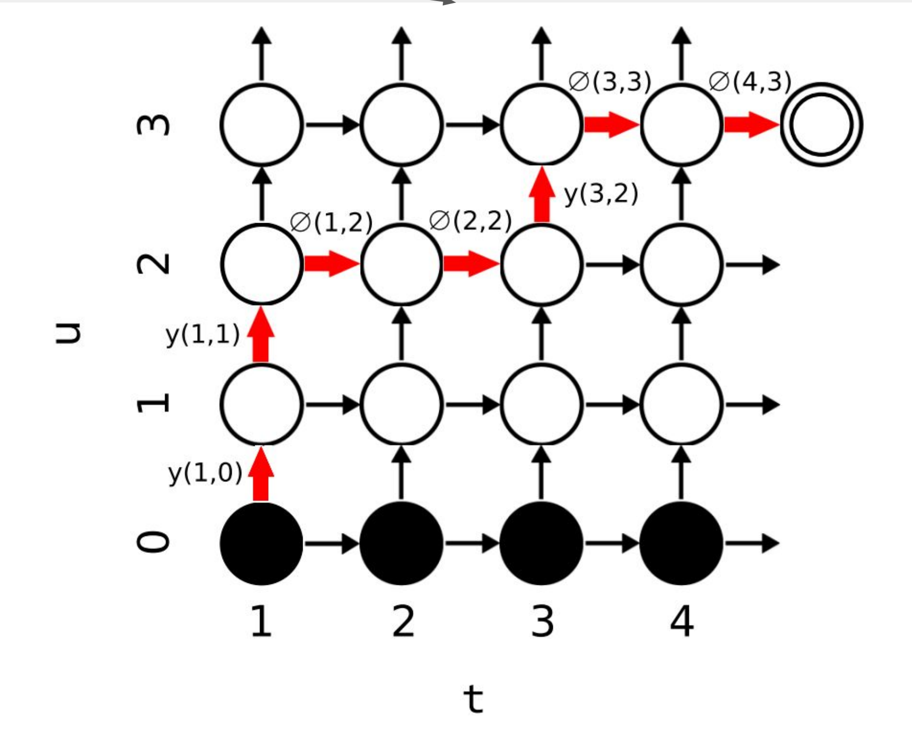
u * t joint networks
organized into a lattice

Joint network

Source RNN

Target RNN





RNN-T

- A special symbol “null” is introduced
- When “null” is generated, the model moves one step forward in the source sequence
- At training time, dynamic programming is used to sum up probabilities in all the path
- Each prediction path is (u + t) steps long

RNN-T

- Definitely stronger built-in language model
- It is harder to train, but one can pre-train the source RNN with a CTC loss, and target RNN using a LM loss on some large text corpus
- Decoder RNN is completely independent of the source

Attention based models

- Very similar to the translation model
  - speech as input instead of text
- Gaining popularity
  - Attention-Based Models for Speech Recognition, by Jan Chorowski et al
  - Listen, Attend and Spell, by William Chan et al
  - Very Deep Convolutional Networks for End-to-End Speech Recognition, by Yu Zhang et al
  - Latent Sequence Decompositions, by William Chan et al
- Research from my group suggest attention based models are very promising for ASR

Attention based models

- Offers the best modeling flexibility
  - Decoder has strictly more information as it conditions on the encoding of the source
  - No artificial statistical assumptions, like temporal independence assumption in CTC, or the independence assumption among the joint networks in RNN-T
- In theory should be the best one to use
  - In practice depends very much on model tuning as well

LAS

- Listen: Acoustic model
- Attend: Dynamic time warping
- Spell: language model

https://arxiv.org/pdf/1508.01211.pdf

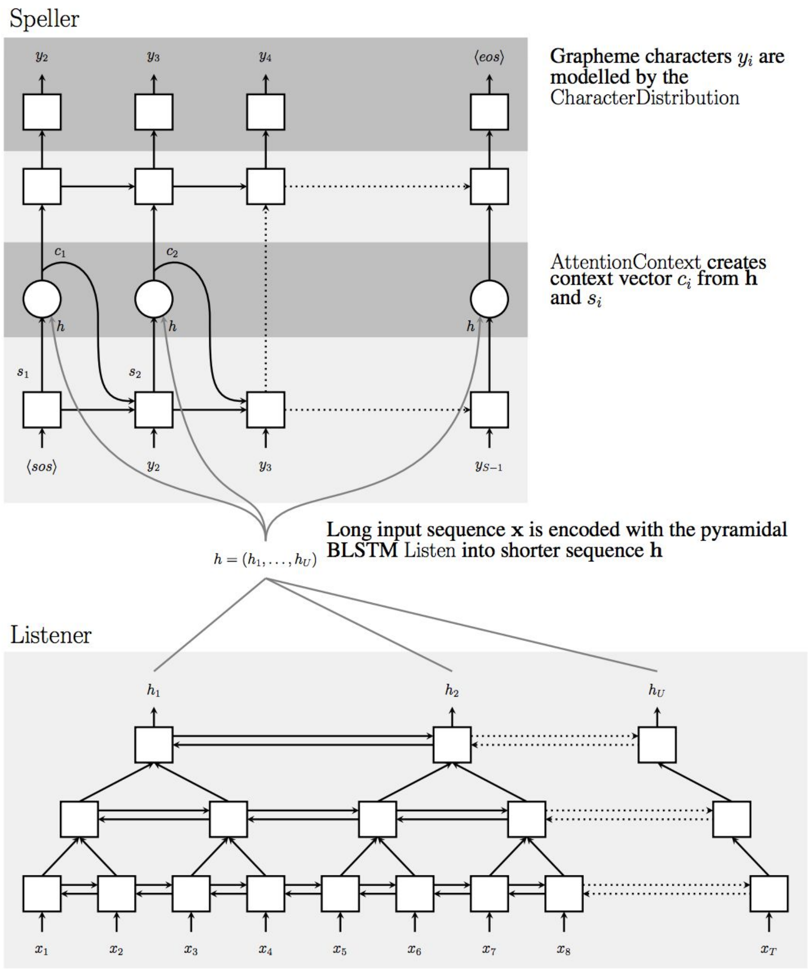
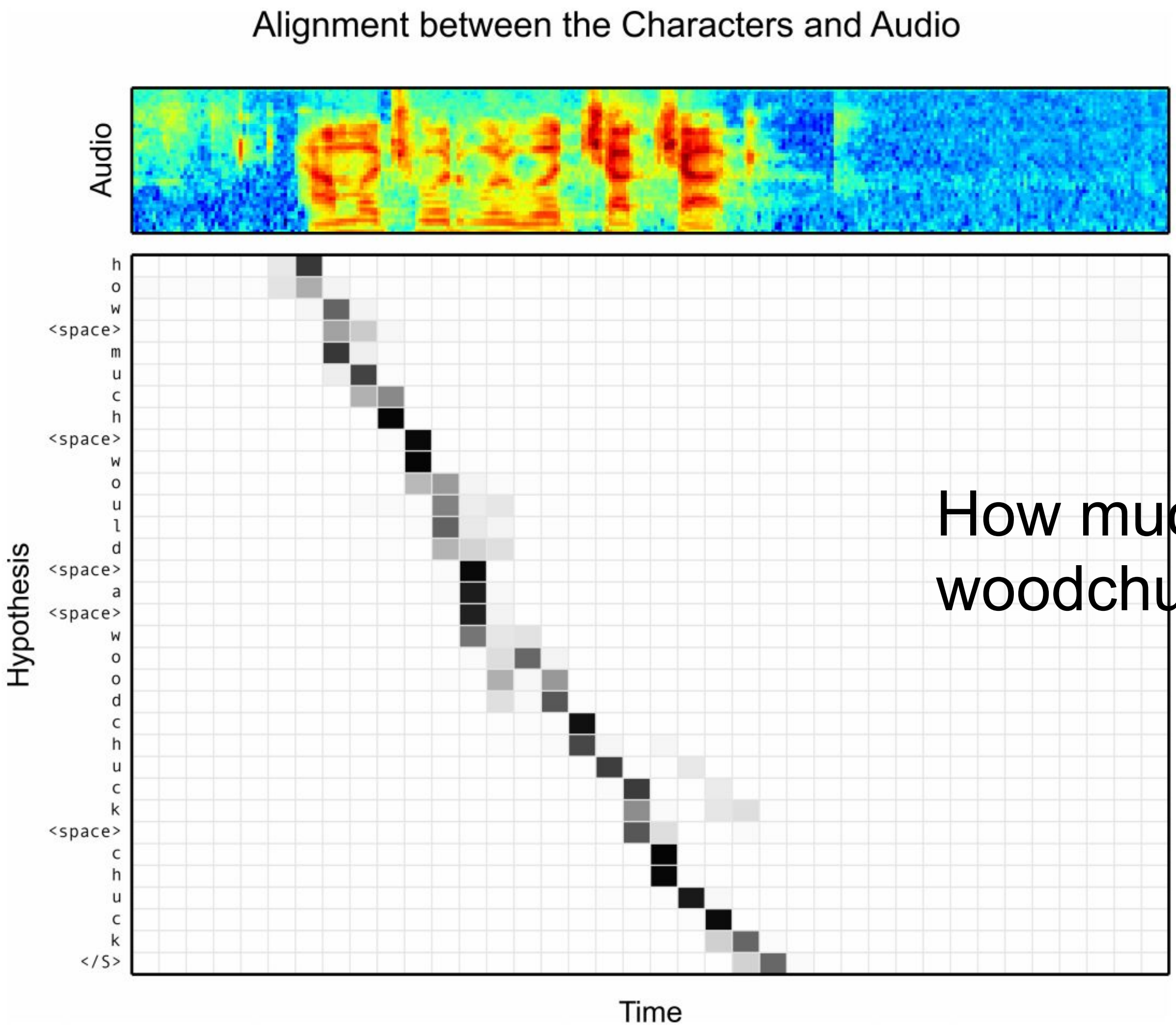
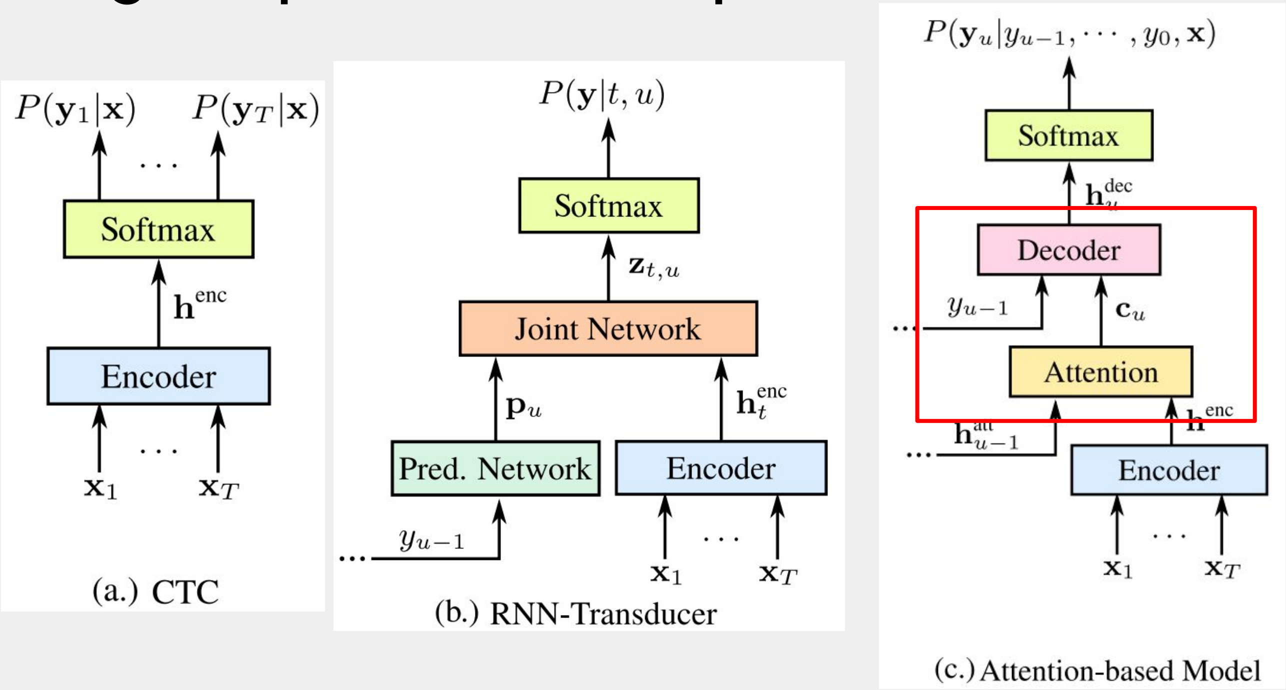


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence $\mathbf{x}$ into high level features $\mathbf{h}$, the speller is an attention-based decoder generating the $\mathbf{y}$ characters from $\mathbf{h}$.



How much would a woodchuck chuck

Comparing Sequence-to-Sequence Models



Attention-based models have the encoder feeding the decoder with acoustic information.

A Comparison of Sequence-to-Sequence Models for Speech Recognition,

Table 1: *WERs (%) on various test sets for the models compared in this work. The attention-based model with two decoder layers is the single best sequence-to-sequence model.*

Model	Clean		Noisy		numeric
	dict	vs	dict	vs	
Baseline Uni. CDP	6.4	9.9	8.7	14.6	11.4
Baseline BiDi. CDP	5.4	8.6	6.9	-	11.4
End-to-end systems					
CTC-grapheme[3]	39.4	53.4	-	-	-
RNN Transducer	6.6	12.8	8.5	22.0	9.9
RNN Trans. with att.	6.5	12.5	8.4	21.5	9.7
Att. 1-layer dec.	6.6	11.7	8.7	20.6	9.0
Att. 2-layer dec.	**6.3**	**11.2**	**8.1**	**19.7**	**8.7**

CTC

RNN-T

LAS

https://arxiv.org/pdf/1703.08581.pdf

	Architecture	SWBD WER	CH WER
Published	Iterated-CTC [29]	11.3	18.7
	BLSTM + LF MMI [21]	8.5	15.3
	LACE + LF MMI [4] [28]	8.3	14.8
	Dilated convolutions [25]	7.7	14.5
	CTC + Gram-CTC [17]	7.3	14.7
	BLSTM + Feature fusion[23]	**7.2**	**12.7**
Ours	CTC [17]	9.0	17.7
	RNN-Transducer		
	Beam Search **NO LM**	8.5	16.4
	Beam Search + LM	8.1	17.5
	Attention		
	Beam Search **NO LM**	8.6	17.8
	Beam Search + LM	8.6	17.8

Table 1. WER comparison against previous published results on Fisher-Switchboard Hub5'00 benchmark using *in-domain* data. We only list results using single models here. All the previous works reported WER using language models. We don't leverage any speaker information in our models, though it has been shown to reduce WER in previous works [28, 25].

https://arxiv.org/pdf/1707.07413.pdf

Speech Synthesis

Speech synthesis

- It is the reverse problem of speech recognition
- Existed for a long time
- Straightforward solution is Concatenative TTS
  - Stitch together pieces of recorded speech, with smoothing at the boundaries
  - Quite intelligible
  - Prosody is not natural, sounds very robotic
  - Widely used

Neural based TTS

- Fully neural network based TTS system is getting popular
  - Wavenet from Google DeepMind
  - TacoTron from Google
  - DeepVoice from Baidu
  - VoiceLoop from Facebook
- Wavenet can been seen as a high quality vocoder
- TacoTron, DeepVoice and VoiceLoop are all end to end TTS systems
- Will focus on DeepVoice and TacoTron in this talk

DeepVoice

- Follows more traditional approach
- It is a pipelined system
  - text -> phoneme
  - phoneme -> duration
  - phoneme + duration -> Pictch
  - Wavenet like vocoder

DeepVoice

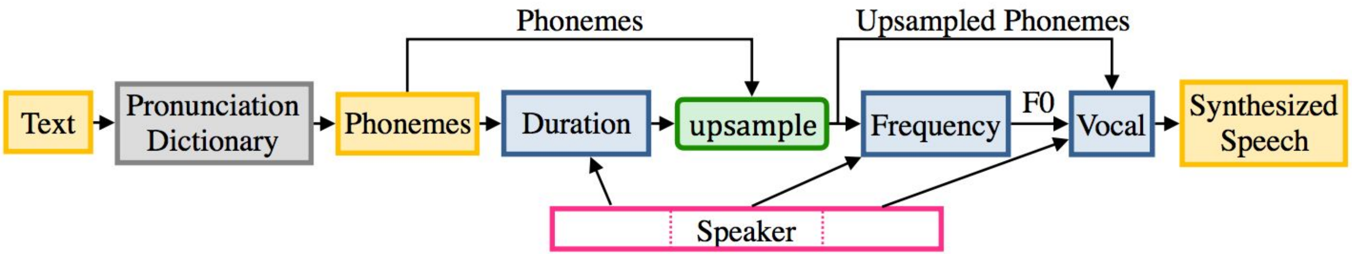


Figure 1: Inference system diagram: first text-phonemes dictionary conversion, second predict phoneme durations, third upsample and generate F_0, finally feed F_0 and phonemes to vocal model.

TacoTron

- It is just one end to end model
- The model predicts linear spectrograms from text directly
  - So no explicit duration model, pitch model and etc
- Used fixed Griffin-Lim algorithm to convert from spectrograms to speech
  - Griffin-Lim as inverse FFT
- Could benefit from a better vocoder
  - e.g. Wavenet
- Some examples can be found:
  - https://google.github.io/tacotron/

Model Architecture

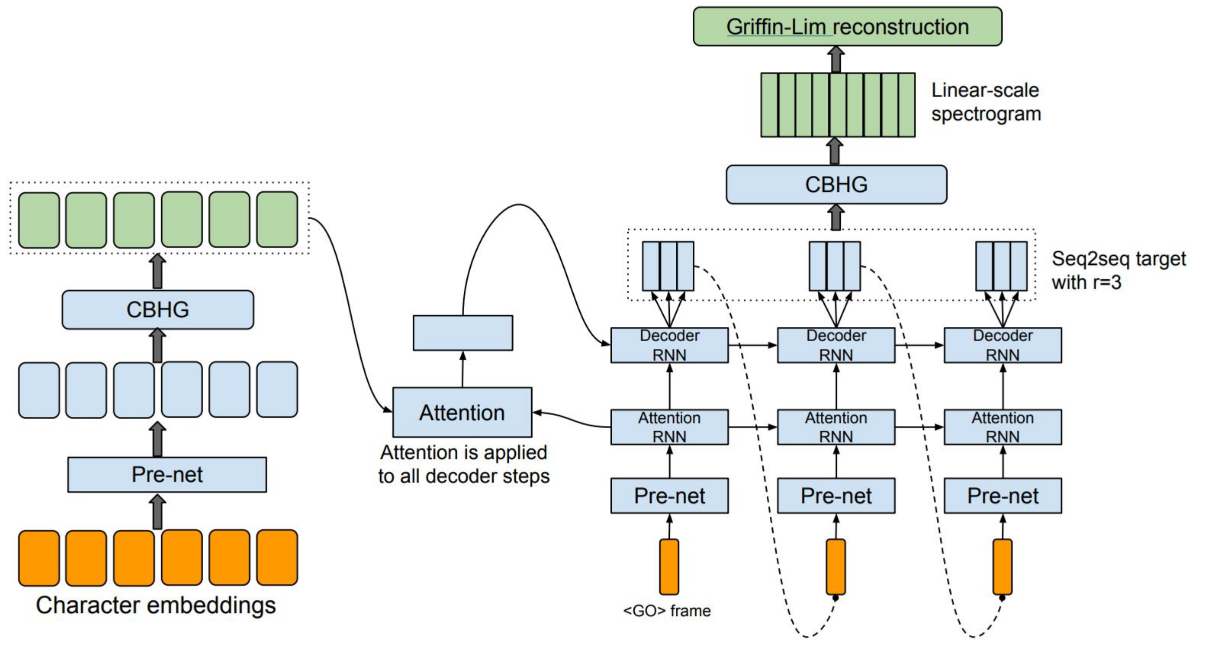


Figure 1: *Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

Other applications of sequence models

What else we can do with sequence models?

- Image captioning
  - *Show and Tell: A Neural Image Caption Generator*, by Oriol Vinyals
- Syntactic constituency parsing
  - *Grammar as a Foreign Language,* by Oriol Vinyals
- Handwriting synthesis
  - *Generating Sequences With Recurrent Neural Networks*, by Alex Graves
- Many more ...

Image Captioning

p(English | French)

$\downarrow$

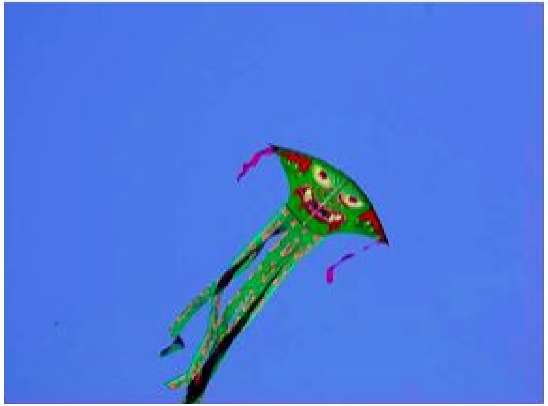
p(English | Image)

1. Vinyals, O., et al. "Show and Tell: A Neural Image Caption Generator." *CVPR* (2015).
2. Mao, J., et al. "Deep captioning with multimodal recurrent neural networks (m-rnn)." *ICLR (2015).*
3. Karpathy, A., Li, F., "Deep visual-semantic alignments for generating image descriptions." *CVPR (2015).*

A man holding a tennis racquet on a tennis court.

Two pizzas sitting on top of a stove top oven

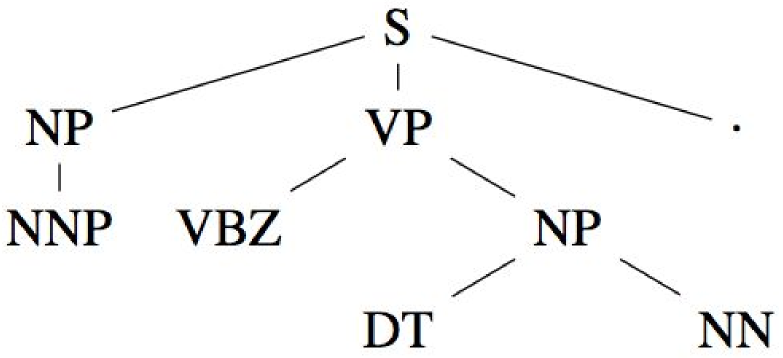
A group of young people playing a game of Frisbee



A man flying through the air while riding a snowboard

Syntactic parsing

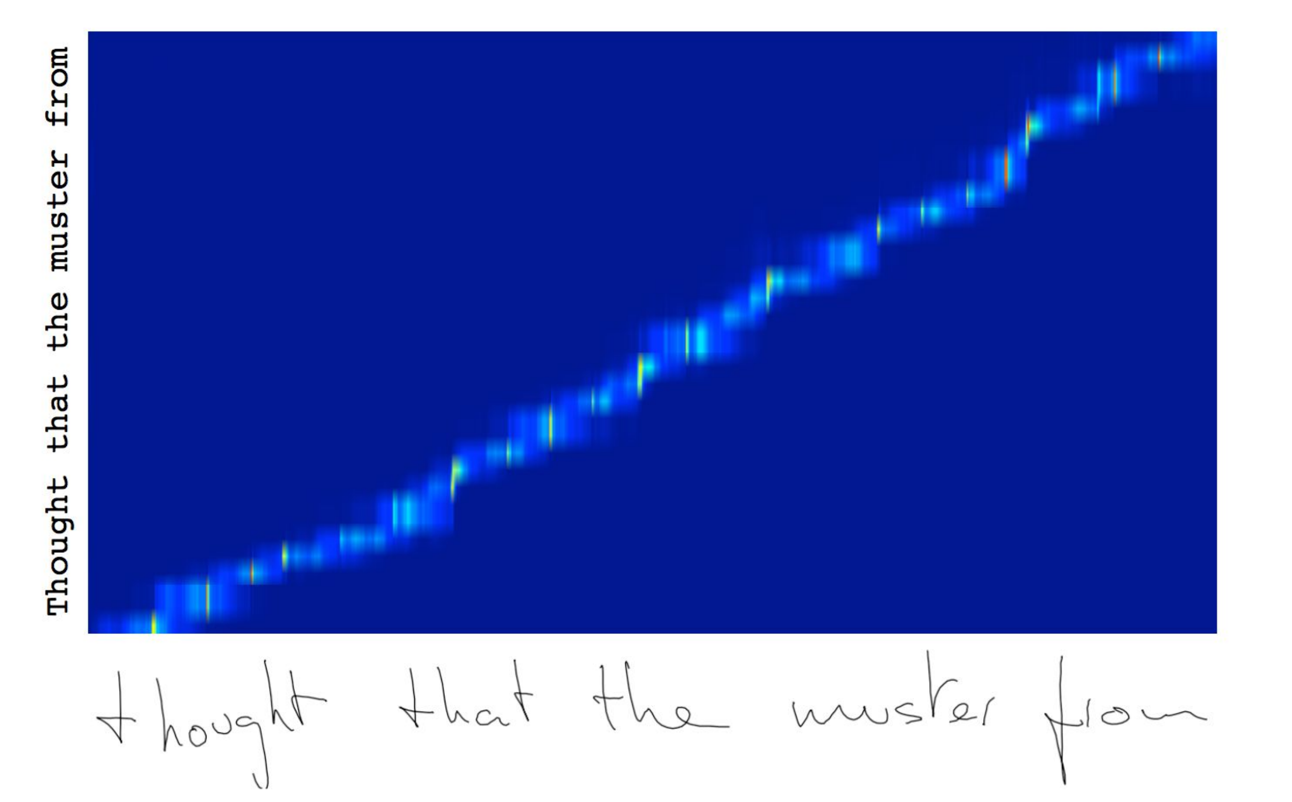
John has a dog . →



John has a dog . → (S (NP NNP $)_{NP}$ (VP VBZ (NP DT NN $)_{NP}$ $)_{VP}$. $)_{S}$

Figure 2: Example parsing task and its linearization.

Handwriting synthesis



Conclusion

- Introduced many popular sequence models for language modeling, for machine translation, automatic speech recognition, speech synthesis, and others
- Advanced topics that I didn’t cover
    - Scheduled sampling
    - Sequence training to directly optimize for the final metric
    - Online sequence models
    - Language model integration into NMT, ASR and etc

Q & A