

10417/10617

**Intermediate Deep Learning:
Fall2023**

Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

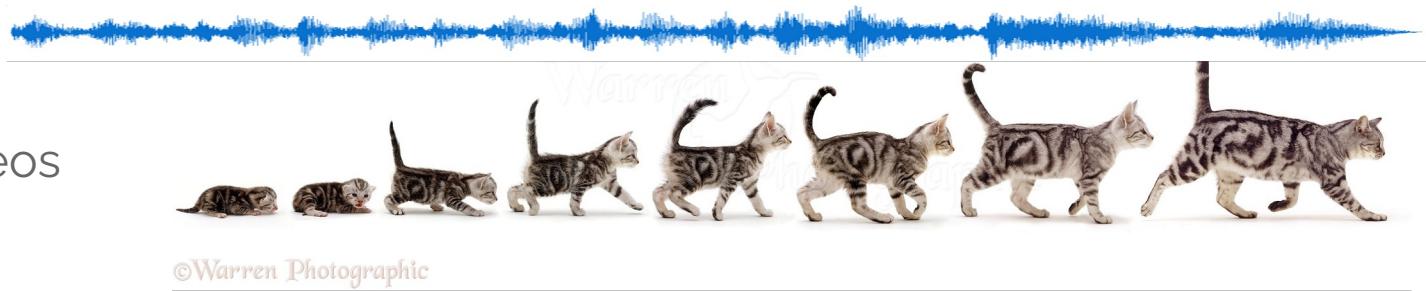
Sequence to Sequence

Sequences

- Words, Letters

50 years ago, the fathers of artificial intelligence convinced everybody that logic was the key to intelligence. Somehow we had to get computers to do logical reasoning. The alternative approach, which they thought was crazy, was to forget logic and try and understand how networks of brain cells learn things. Curiously, two people who rejected the logic based approach to AI were Turing and Von Neumann. If either of them had lived I think things would have turned out differently... now neural networks are everywhere and the crazy approach is winning.

- Speech

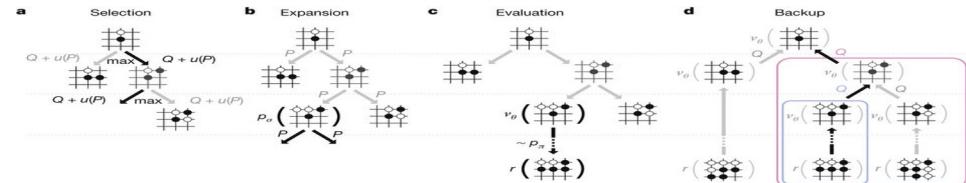


- Images, Videos

- Programs

```
while (*d++ = *s++);
```

- Sequential Decision Making (RL)



Classical Models for Sequence Prediction

- Sequence prediction was classically handled as a structured prediction task
 - Most were built on conditional independence assumptions
 - Others such as DAGGER were based on supervisory signals and auxiliary information

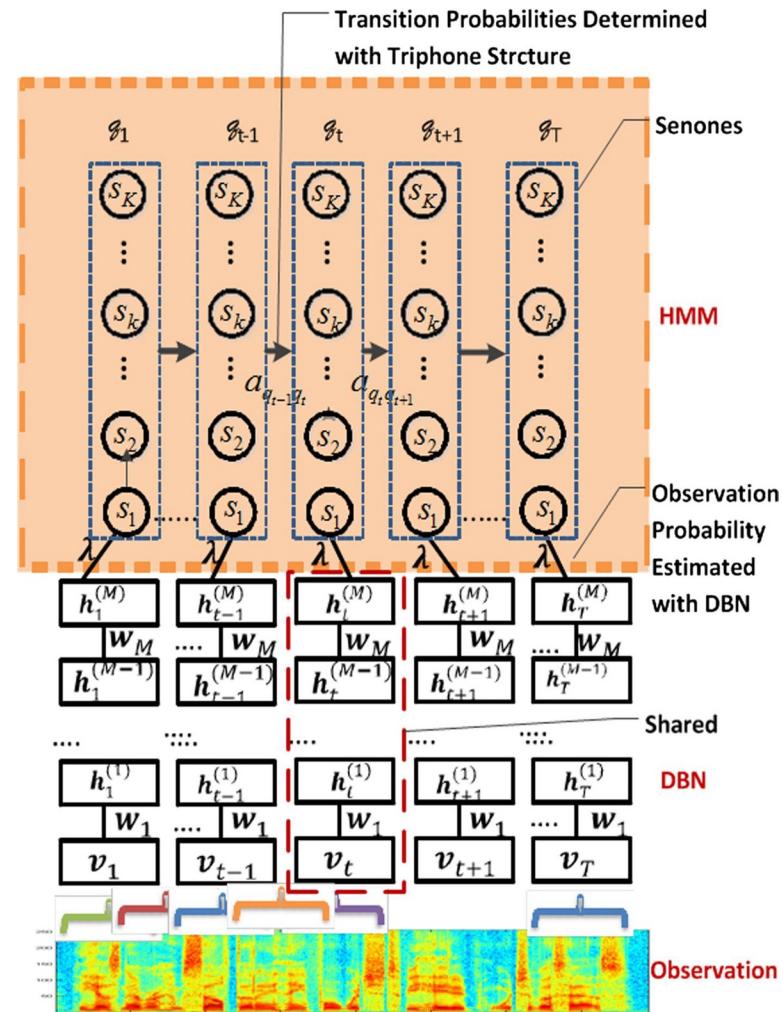
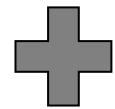


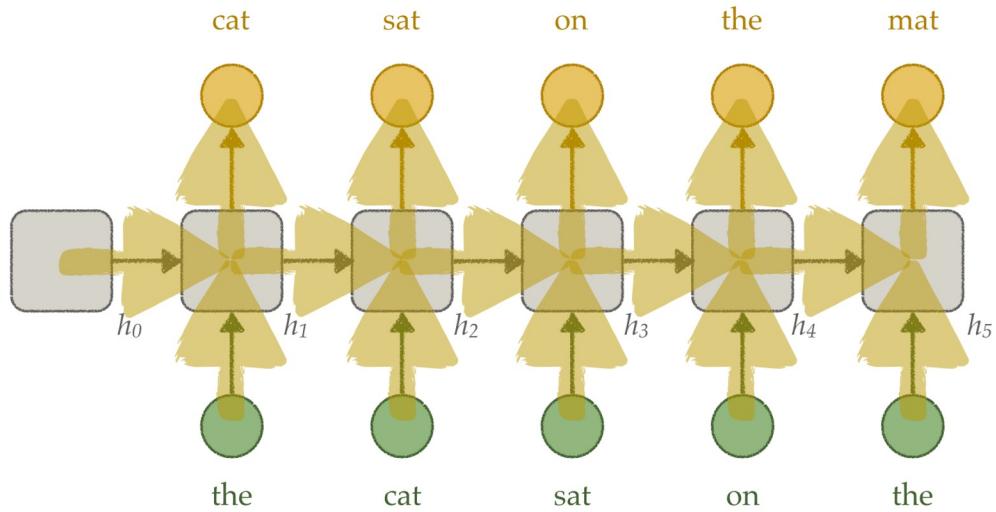
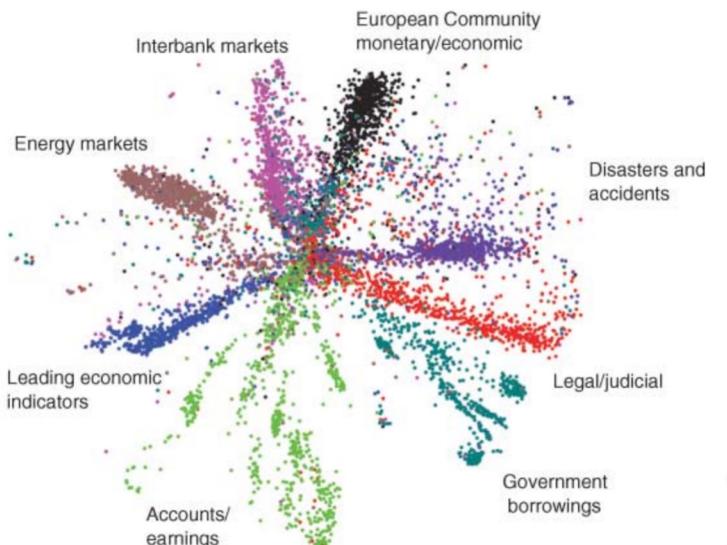
Figure credit: Li Deng

Two Key Ingredients

Neural Embeddings



Recurrent Language Models



Hinton, G., Salakhutdinov, R. "Reducing the Dimensionality of Data with Neural Networks." *Science* (2006)

Mikolov, T., et al. "Recurrent neural network based language model." *Interspeech* (2010)

Language Models

<i>context</i>					<i>target</i>	$P(w_t w_{t-1}, w_{t-2}, \dots w_{t-5})$
the	cat	sat	on	the	mat	0.15
w_{t-5}	w_{t-4}	w_{t-3}	w_{t-2}	w_{t-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	?

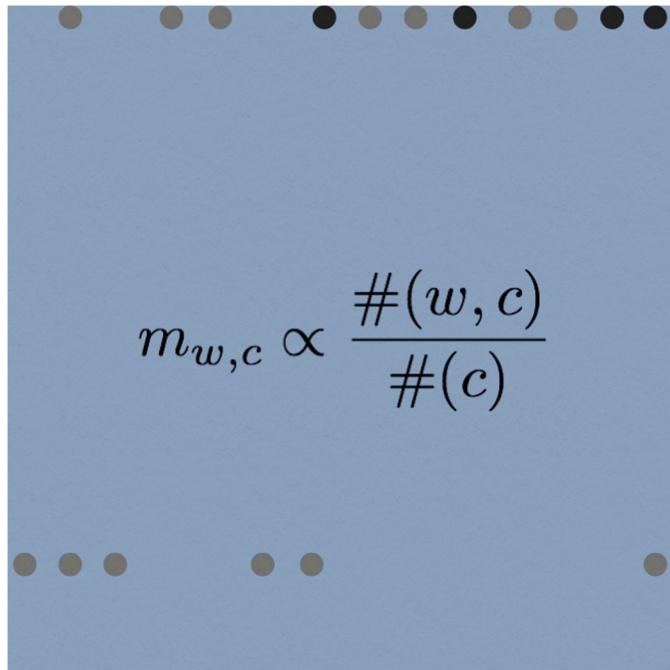
N-grams

context

cat chases cheese dog drinks eats mat milk of on paws rat sat the

target

cat
rat



the cat sat on the mat
the cat drinks milk
the dog chases the cat
the paws of the cat

the cat chases the rat
the rat eats cheese
the rat eats the mat

N-grams

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

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	$P(w_6 w_5, w_4)$

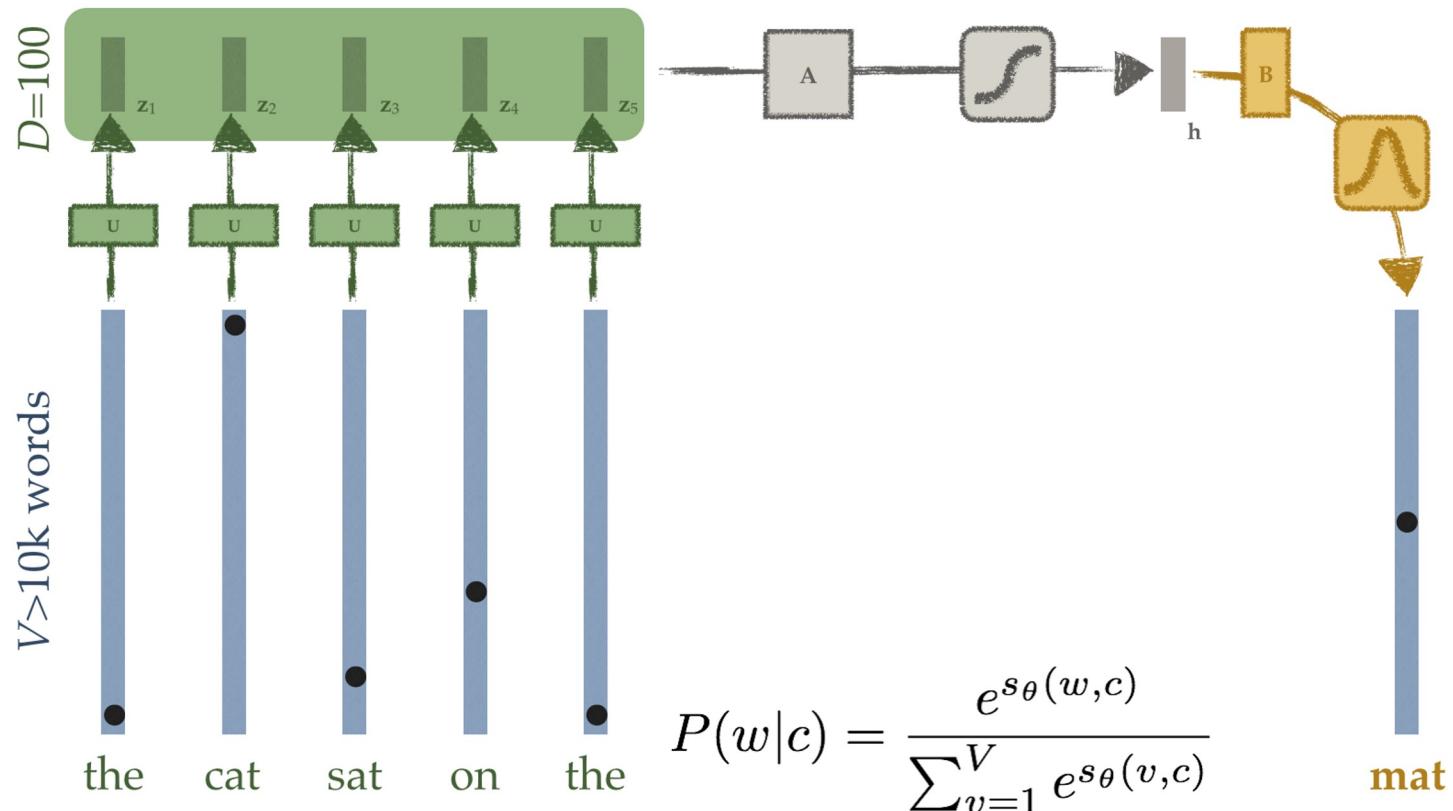
Chain Rule

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

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	$P(w_6 w_5, w_4, w_3, w_2, w_1)$

Key Insight: Vectorizing Context

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



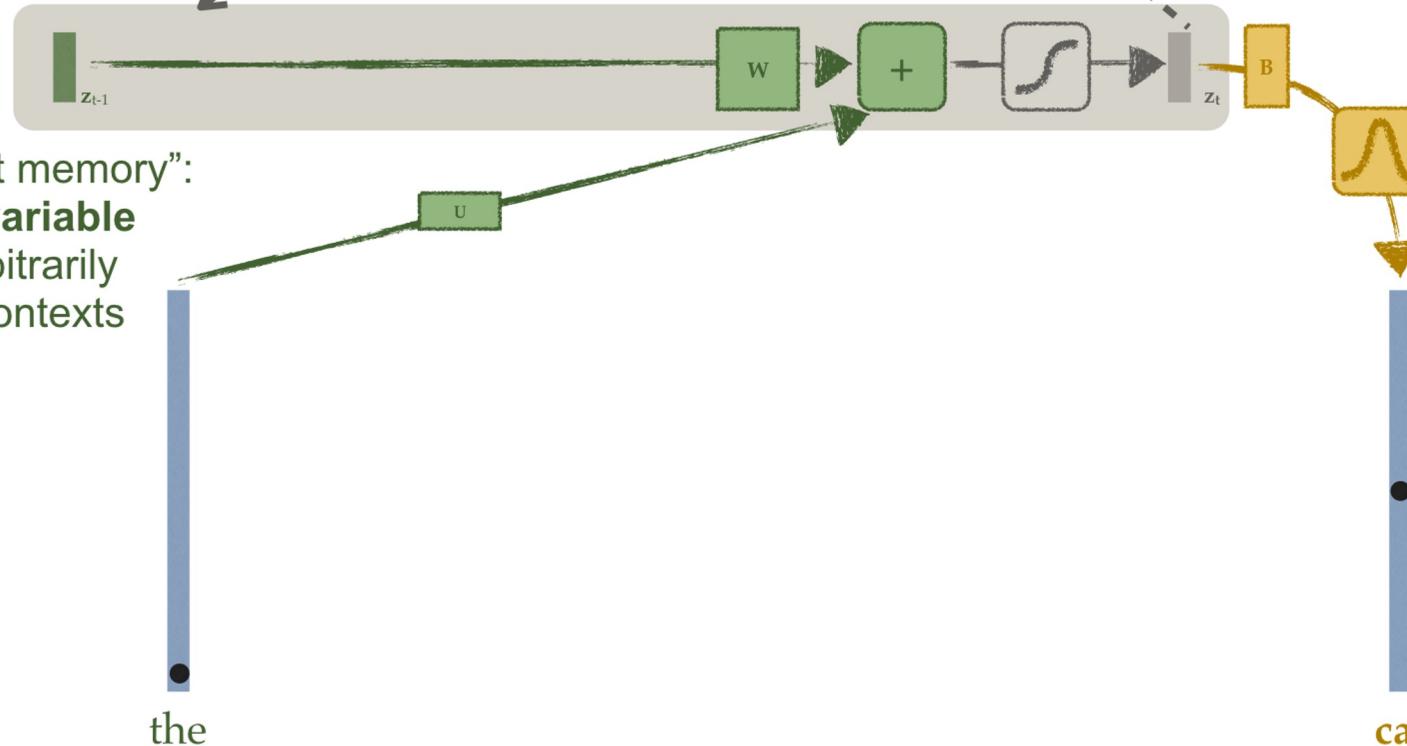
Recurrent Neural Network Language Models

[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*]

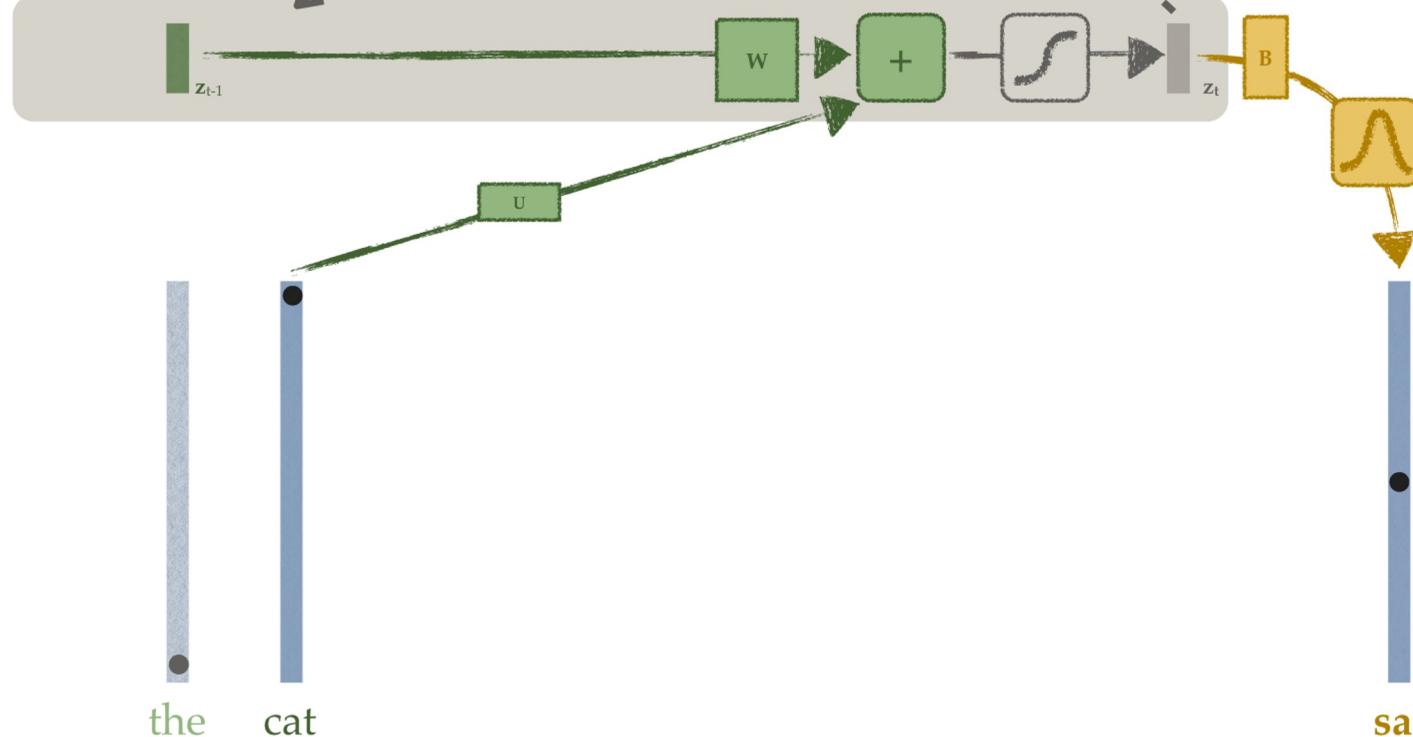
"persistent memory":

state variable

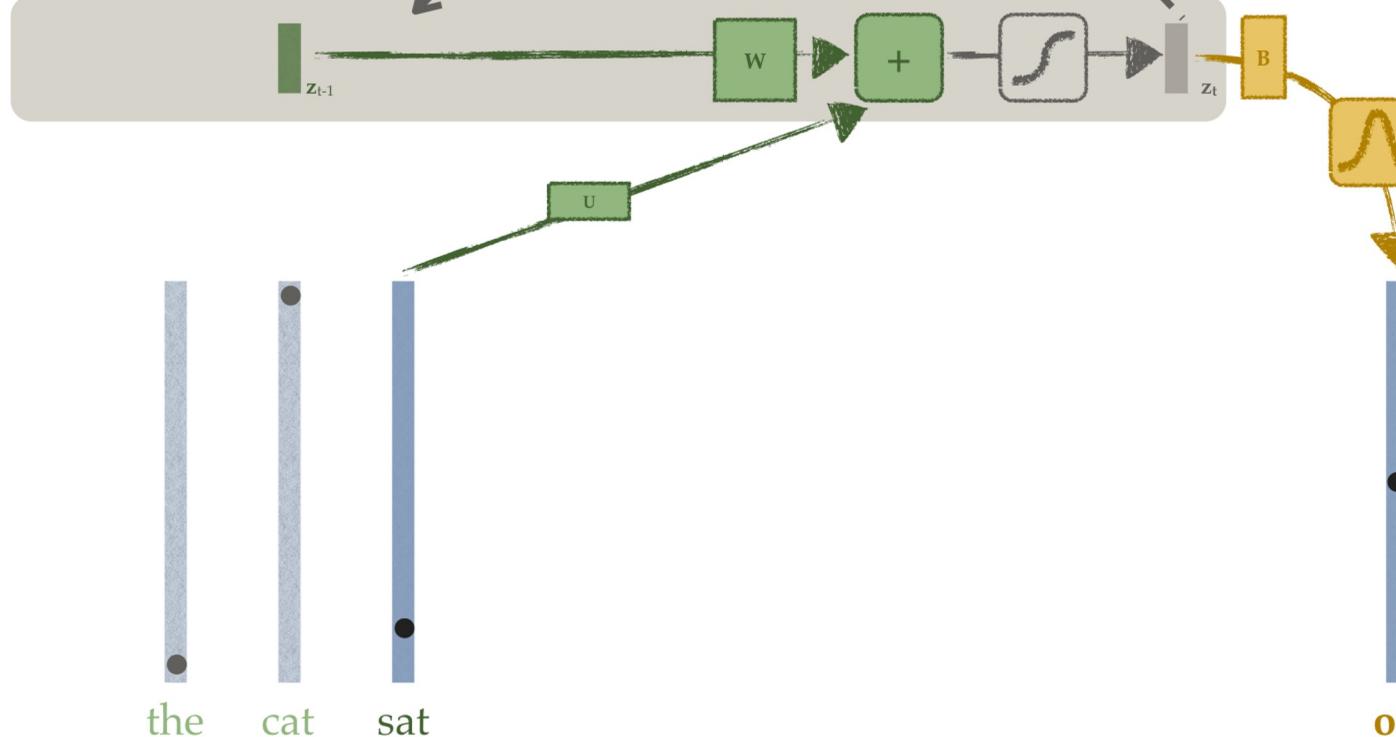
for arbitrarily
long contexts



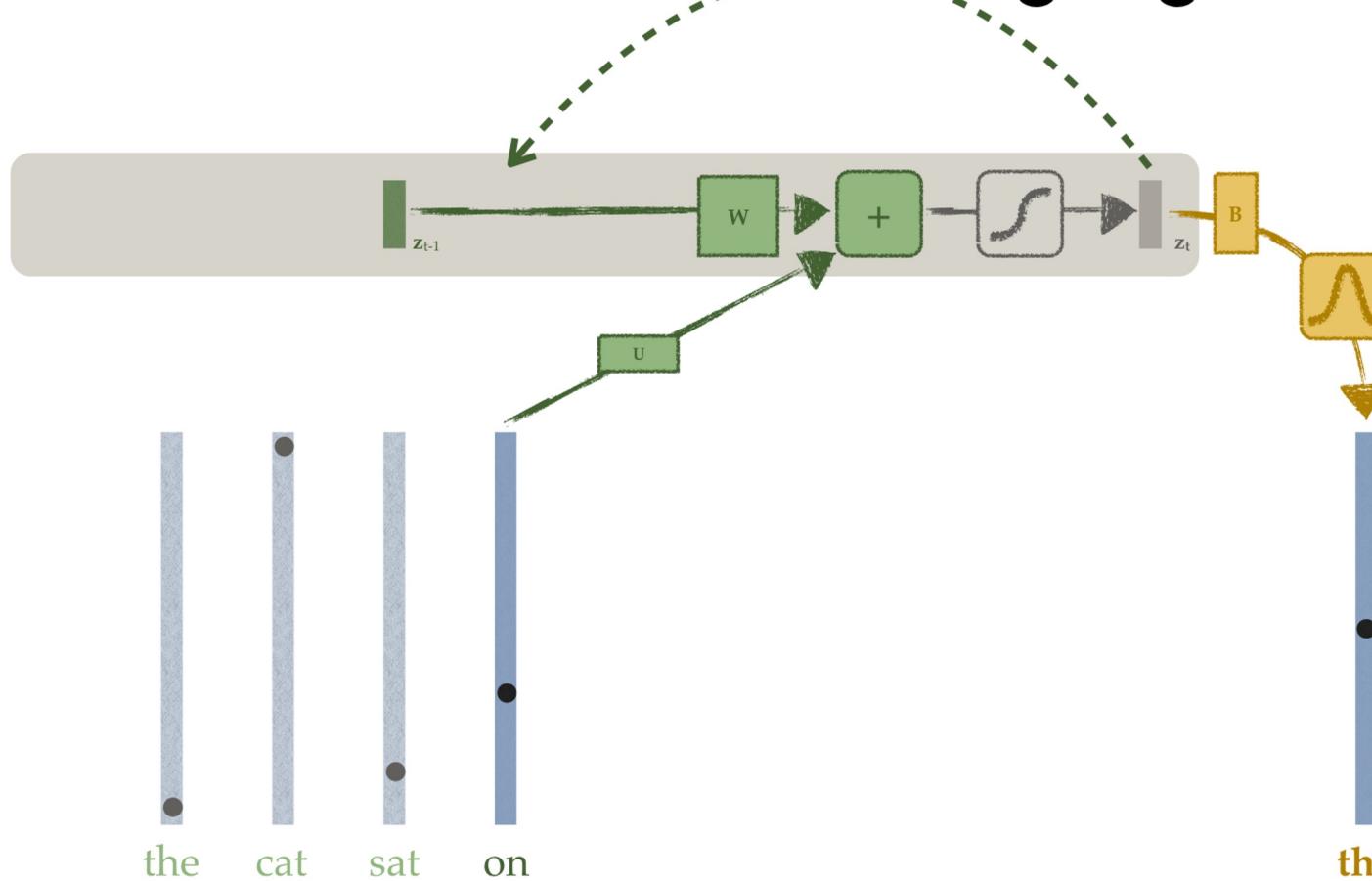
Recurrent Neural Network Language Models



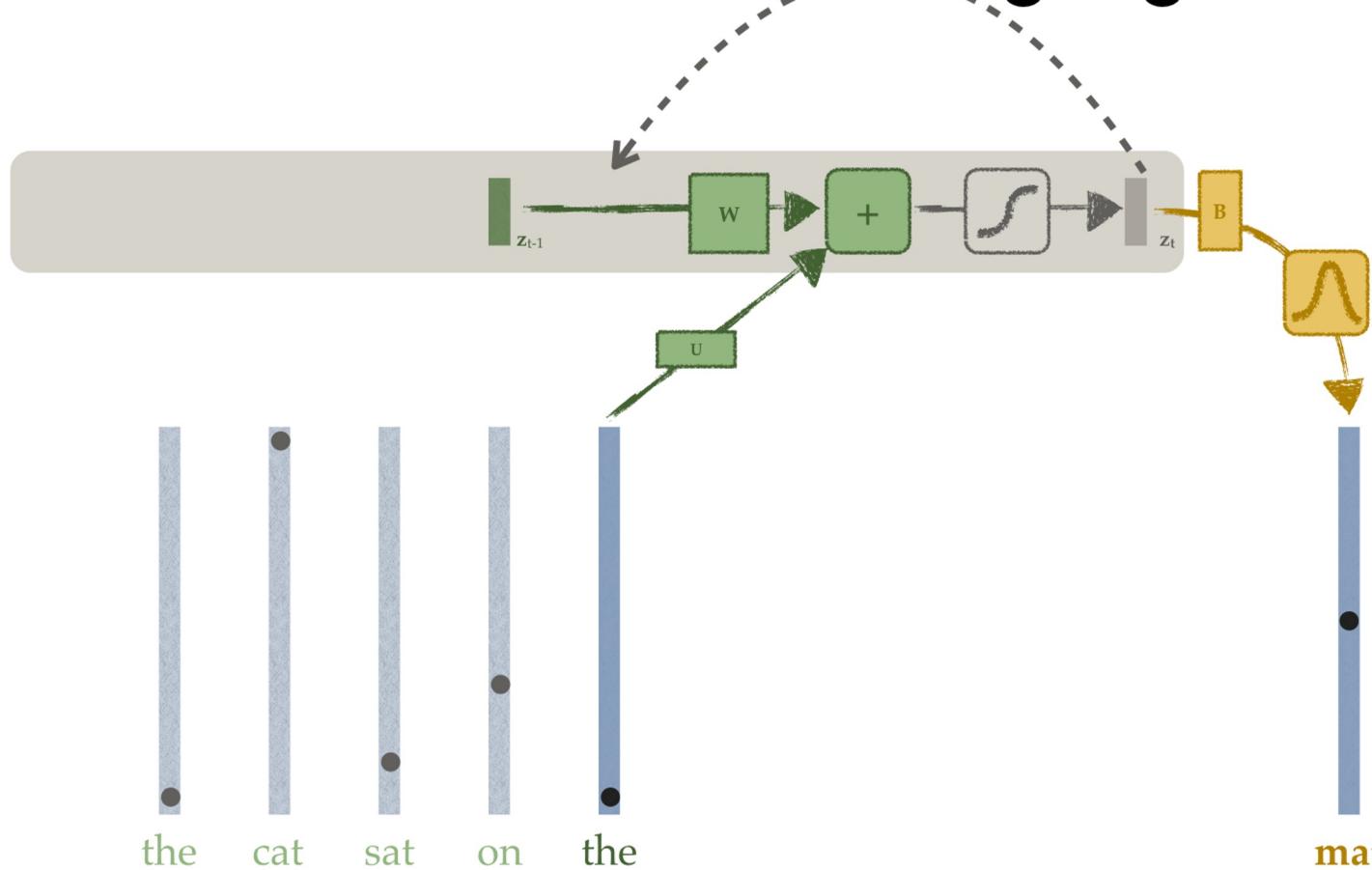
Recurrent Neural Network Language Models



Recurrent Neural Network Language Models



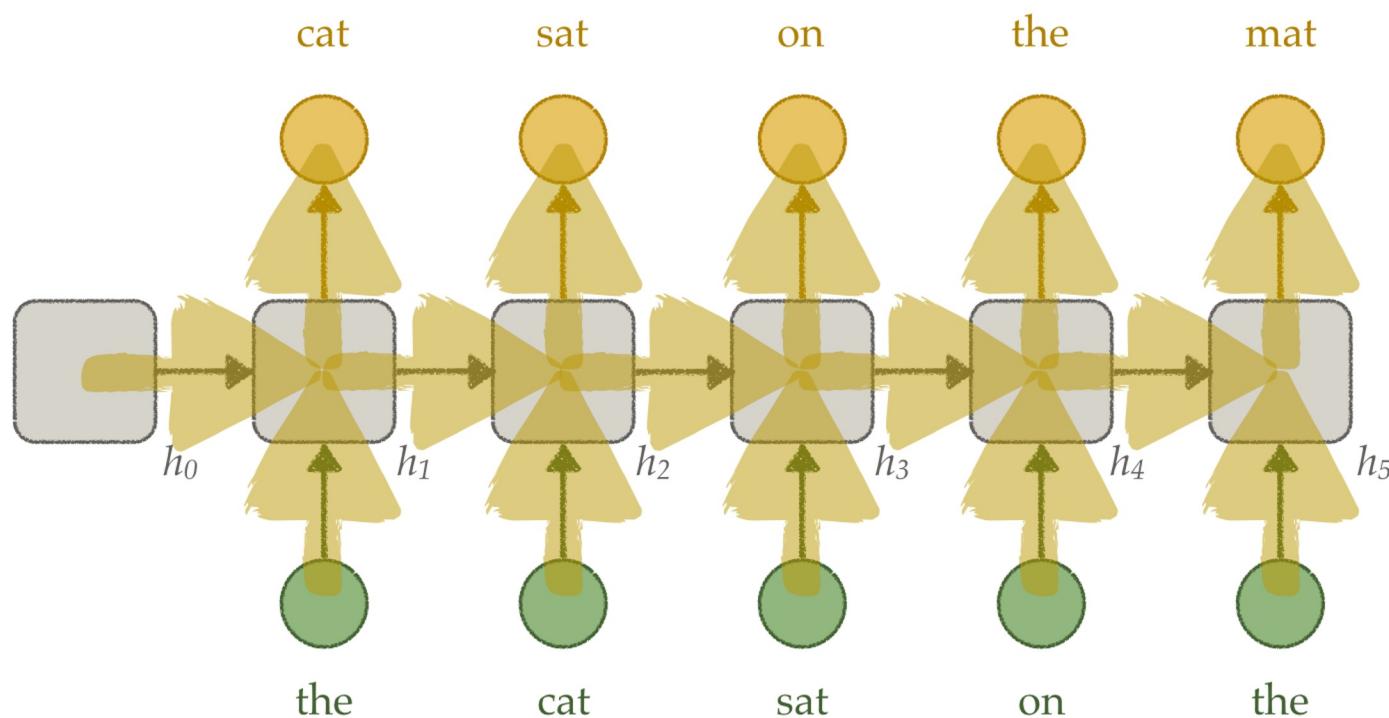
Recurrent Neural Network Language Models



What do we Optimize?

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

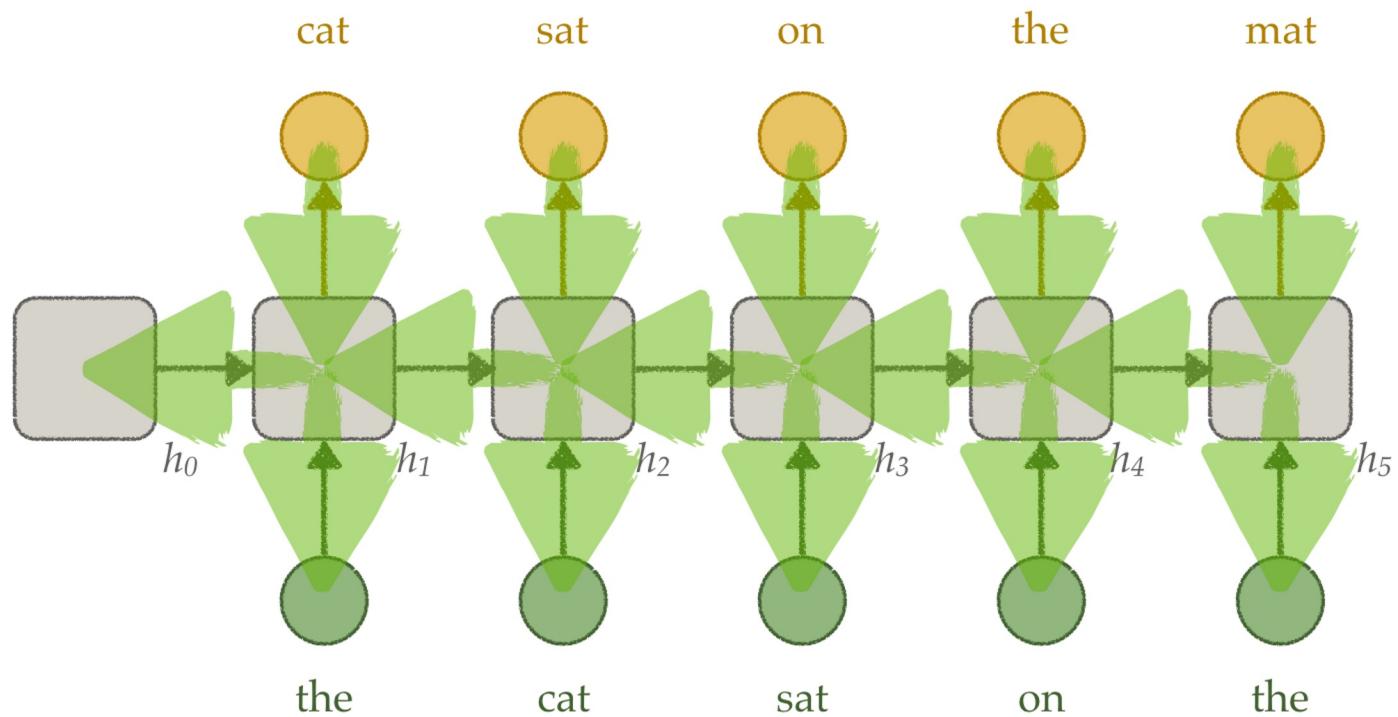
Recurrent Neural Network Language Models



Learning Sequences – Piotr Mirowski

- Forward Pass

Recurrent Neural Network Language Models



Learning Sequences — Piotr Mirowski

- Backward Pass

Seq2Seq

Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig
Microsoft Research
Redmond, WA, USA
michael.auli,mqgalley,chrisq,gzweig@microsoft.com

Abstract

We present a joint language and translation model based on a recurrent neural network which predicts target words based on an unbounded history of both source and target words. The writer independence assumption of this model result in a vastly improved performance over previous work based on forward/backward language or translation models. We tackle this issue with a new lattice reordering algorithm and demonstrate its effectiveness empirically. Our joint model builds on a well known recurrent neural network language model (Mikolov, 2012) augmented by a layer of additional inputs from the source language. These inputs are the previously computed n-gram features of the source language. Our best results improve the output of a system trained on WMT 2012 French-English data by up to 1.5 BLEU, and by 1.1 BLEU on average across several test sets.

1 Introduction

Recently, several feed-forward neural network based language and translation models have achieved impressive accuracy improvements on statistical machine translation tasks (Allauzen et al., 2011; Le et al., 2012b; Schwenk et al., 2012). In this paper we focus on recurrent neural network architectures, which have recently advanced the state-of-the-art in language modeling (Mikolov et al., 2010; Mikolov et al., 2011e; Mikolov, 2012), outperforming multi-layer feed-forward based networks in both perplexity and word error rate in speech recognition (Ariyoshi et al., 2012; Sundermeyer et al., 2013). The major attraction of recurrent architectures is their potential to capture long-span dependencies since

1046

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1044–1054, Seattle, Washington, USA, 18–21 October 2013. ©2013 Association for Computational Linguistics.

Recurrent Continuous Translation Models

Nal Kalchbrenner **Phil Blunsom**
Department of Computer Science
University of Oxford
`{nal.kalchbrenner,phil.blunsom}@cs.ox.ac.uk`

Abstract

We introduce a class of models, called Continuous Translation Models, which are based purely on continuous representations for words, phrases and sentences and do not rely on alignments or phrase trees.

The model is based on a word and a conditioning aspect. The generation of the translation is modelled with a target Recurrent Neural Network, while the generation of the source sentence is modelled with a Convolutional Language Model. Through various experiments we show that our models obtain better performance with respect to translations that are > 43% lower than that of state-of-the-art alignment-based translation models. We also show that our model is less sensitive to the word order, syntax, and meaning of the source sentence, despite lacking alignment. Finally we show that they match a state-of-the-art when retraining on best lists of translations.

1 Introduction

In most statistical approaches to machine translation the basic units of translation are phrases that are composed of one or more words. A major component of the phrase-based approach is the use of language labels, significantly beyond the word level and to accurately predict labels for a highly diverse range of unseen phrases and sentences (Grefenstette et al., 2011; Socher et al., 2011; Socher et al., 2012; Hermann and Blunsom, 2013; Kalchbrenner and Blunsom, 2013).

Part-of-speech continuous models were first proposed by Ishizuka, Ishizuka, 2009 and later refined by Ishizuka, Ishizuka, 2013.

100

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1700–1709,
Seattle, Washington, USA, 18–21 October 2013. ©2013 Association for Computational Linguistics.

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

Kyungyun Cho	Dzmitry Bahdanau
Bart van Merriënboer	Jacobs University, Germany
Caglar Gulcehre	d.bahdanau@jacobs-university.de
Université de Montréal	
firstname.lastname@umontreal.ca	
Fethi Bougares	Yousha Bengio
Holger Schwenk	
Université du Maine, France	
Université de Montréal, CIFAR Senior	
firstname.lastname@univ-lille1.fr	find.me.on/the.web

Abstract

In this paper, we propose a novel neural network model called RNN Encoder-Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the sequence into another sequence of symbols. The parameters and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the proposed model.

additional feature in the existing log-linear model. Qualitatively, we show that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases.

1. Introduction

Deep neural networks have shown great success in various applications such as object recognition (see, e.g., Krizhevsky et al., 2012) and speech recognition (see, e.g., Dahl et al., 2012). Furthermore, many recent works showed that neural networks can be successfully used in a number of NLP tasks (see, e.g., Socher et al., 2013; Potts, These include, but are not limited to, language modeling (Bengio et al., 2003), paraphrase detection (Socher et al., 2011) and word embedding extraction (Mikolov et al., 2013). In the field of statistical machine translation (SMT), deep neural networks have begun to show promising results. Schwenk (2012) summarizes a successful use of feedforward neural networks in the framework of phrase-based SMT systems.

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever **Oriol Vinyals** **Quoc V. Le**
Google Google Google
ilyasu@google.com vinyals@google.com qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence-to-sequence learning. Our approach uses a deep neural network architecture that uses a multi-layered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, whereas the LSTM's BLEU score on the first 100 hypotheses is 33.0, which is close to the baseline and may have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 35.8, which is closer to previous best result on this task. The LSTM also handles sentence-level and semantic dependencies that are sometimes orders of magnitude closer to the native language than the massive corpus. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

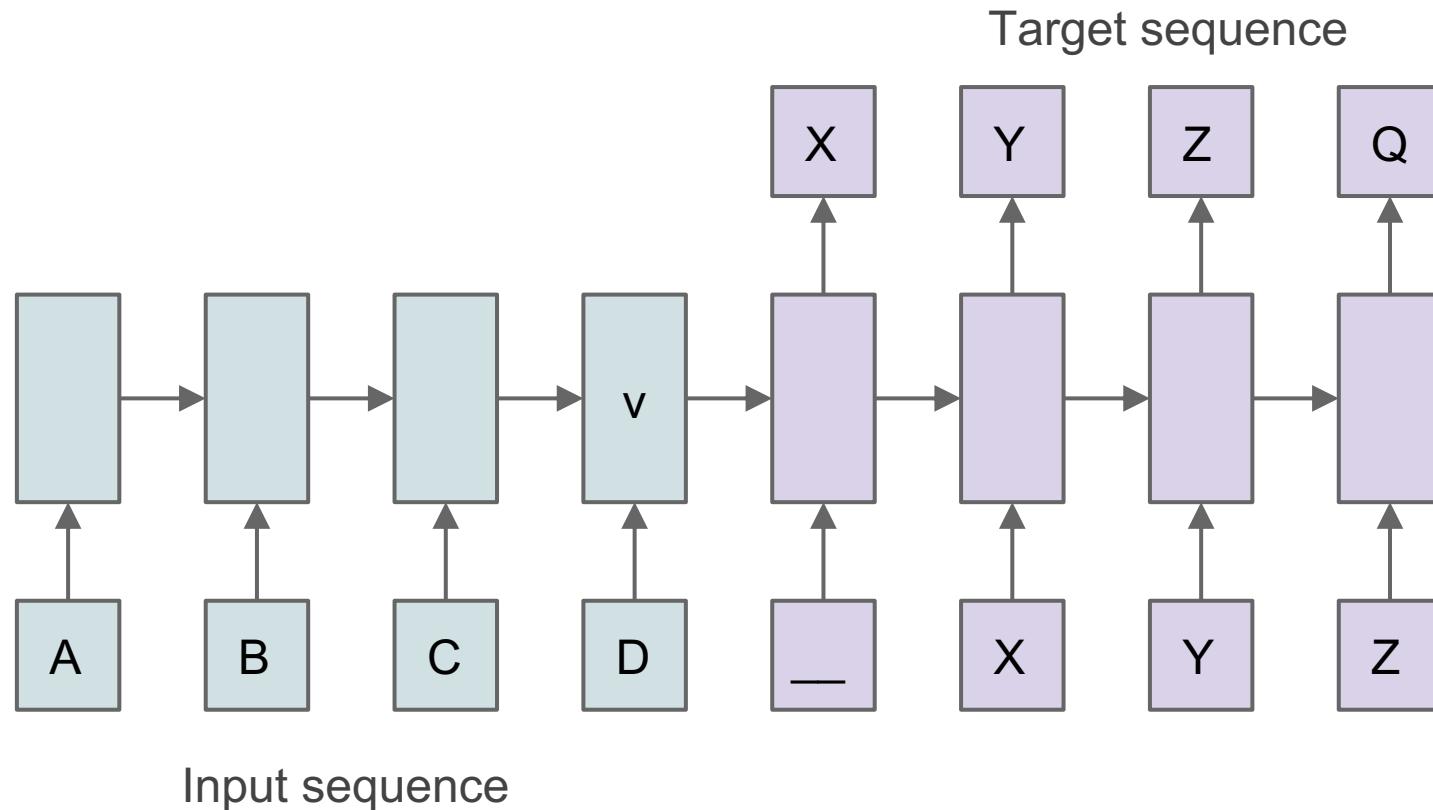
1 Introduction

Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. They are especially powerful because they can perform arbitrary parallel computations by dividing the computation of steps into a sequence of parallel layers. The parallelity of DNNs is related to their depth. An N -bit numbers uses at most $2N$ layers of computation [27]. So, while traditional neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network's parameters. Thus, if there are n parameters in a large DNN that are specified by ground truth labels, we can train the network in the task very rapidly, using supervised backpropagation to find these parameters and solve the problem.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be easily encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering and document classification are sequential problems.

1. Auli, M., et al. "Joint Language and Translation Modeling with Recurrent Neural Networks." *EMNLP* (2013)
 2. Kalchbrenner, N., et al. "Recurrent Continuous Translation Models." *EMNLP* (2013)
 3. Cho, K., et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical MT." *EMNLP* (2014)
 4. Sutskever, I., et al. "Sequence to Sequence Learning with Neural Networks." *NIPS* (2014)

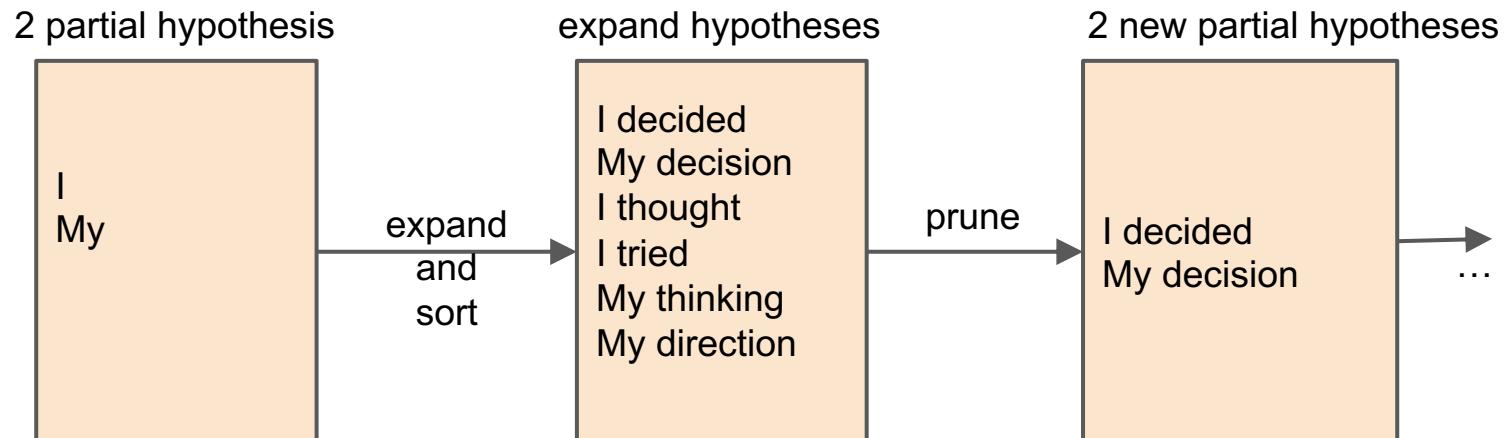
Seq2Seq



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

Decoding in a Nutshell (Beam Size 2)

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



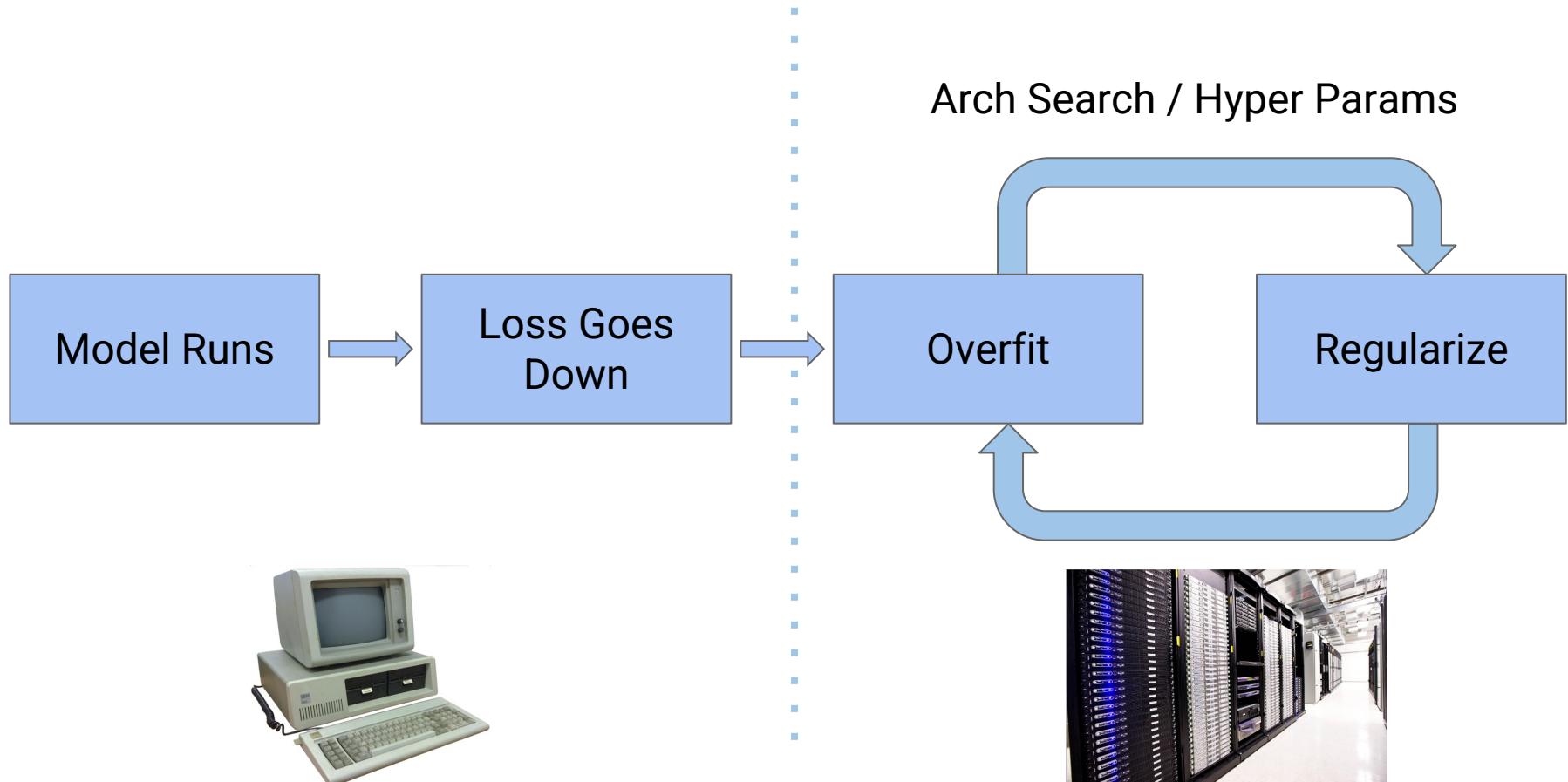
Code

Source:

<https://github.com/keveman/tensorflow-tutorial/blob/master/PTB%20Word%20Language%20Modeling.ipynb>

```
class LSTMCell(object):
    def __init__(self, state_size):
        self.state_size = state_size
        self.W_f = tf.Variable(self.initializer())
        self.W_i = tf.Variable(self.initializer())
        self.W_o = tf.Variable(self.initializer())
        self.W_C = tf.Variable(self.initializer())
        self.b_f = tf.Variable(tf.zeros([state_size]))
        self.b_i = tf.Variable(tf.zeros([state_size]))
        self.b_o = tf.Variable(tf.zeros([state_size]))
        self.b_C = tf.Variable(tf.zeros([state_size]))
    def __call__(self, x_t, h_t1, C_t1):
        X = tf.concat(1, [h_t1, x_t])
        f_t = tf.sigmoid(tf.matmul(X, self.W_f) + self.b_f)
        i_t = tf.sigmoid(tf.matmul(X, self.W_i) + self.b_i)
        o_t = tf.sigmoid(tf.matmul(X, self.W_o) + self.b_o)
        Ctilde_t = tf.tanh(tf.matmul(X, self.W_C) + self.b_C)
        C_t = f_t * C_t1 + i_t * Ctilde_t
        h_t = o_t * tf.tanh(C_t)
        return h_t, C_t
    def initializer(self):
        return tf.random_uniform([2*self.state_size, self.state_size],
                               -0.1, 0.1)
```

Vicious Cycle



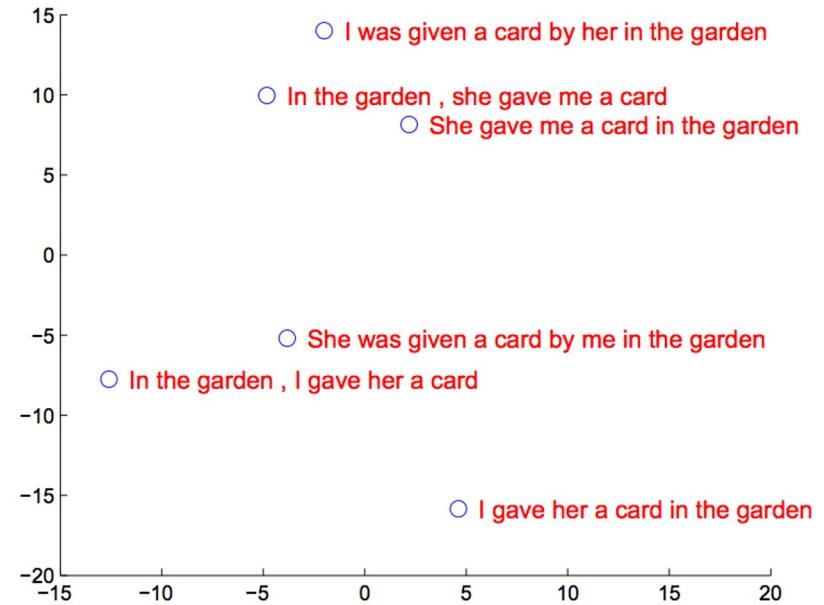
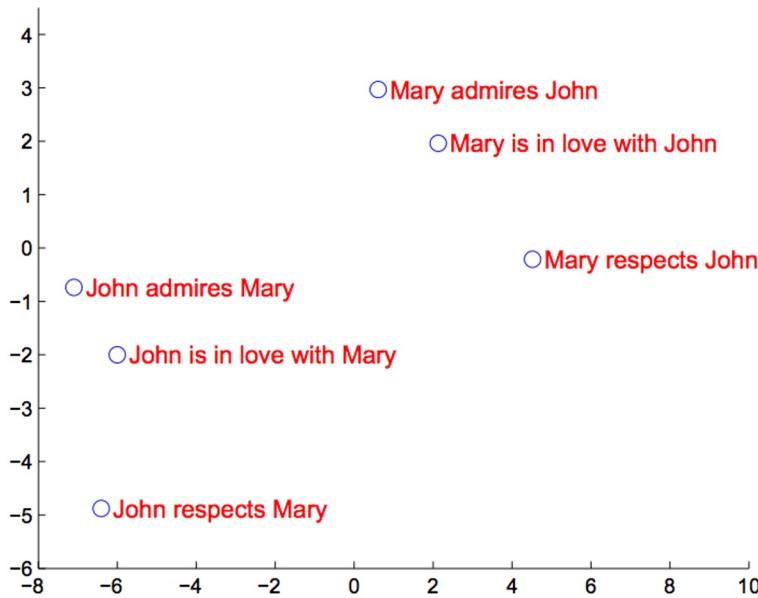
(Some) Tricks of the Trade

- Long sequences?
 - Attention
 - Bigger state
- Can't overfit?
 - Bigger hidden state
 - Deep LSTM + Skip Connections
- Overfit?
 - Dropout + Ensembles
- Tuning
 - Keep calm and decrease your learning rate
 - Initialization of parameters is critical (in seq2seq we used $U(-0.05, 0.05)$)
 - Clip the gradients!
 - E.g. if $\|grad\| > 5$: $grad = grad / \|grad\| * 5$

Applications

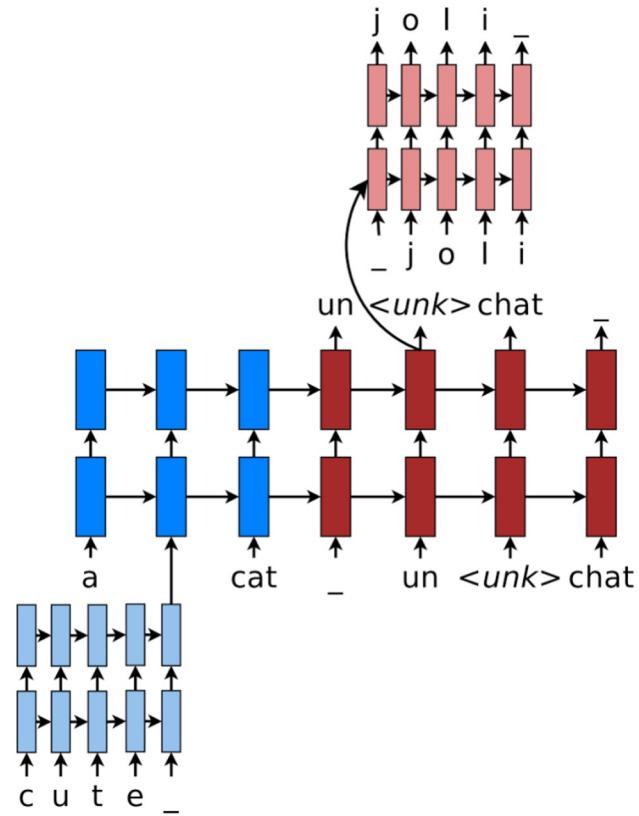
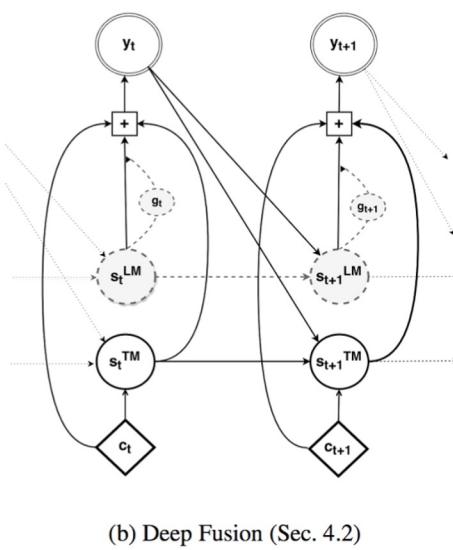
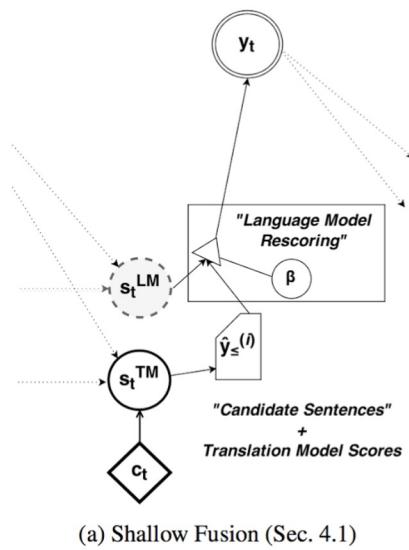
Machine Translation

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



Machine Translation: Concerns

- Using Language Models [1]
- OOV words [2]
- Sequence length



1. Gulcehre, C., et al. "On using monolingual corpora in neural machine translation." *arXiv* (2015).
2. Luong, T., and Manning, C. "Achieving open vocabulary neural MT with hybrid word-character models." *arXiv* (2016).

Image Captioning

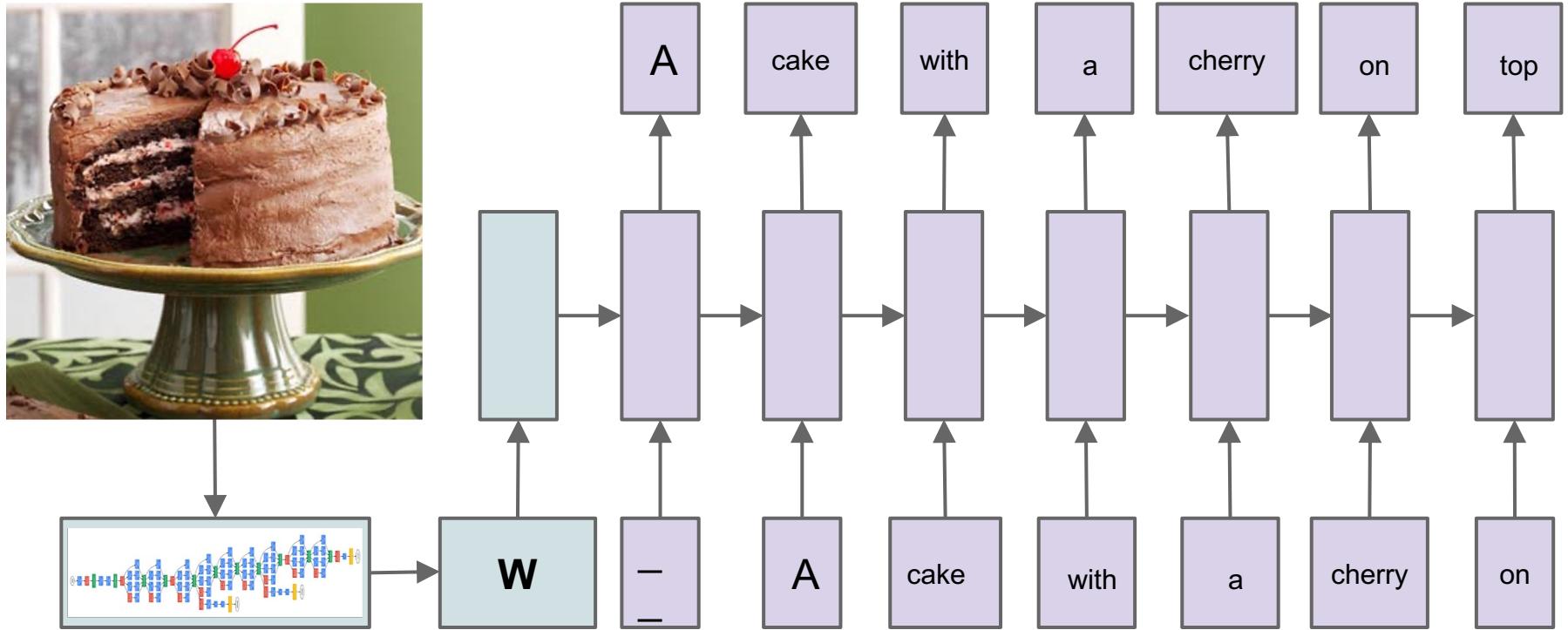
$p(\text{English} \mid \text{French})$



$p(\text{English} \mid \text{Image})$

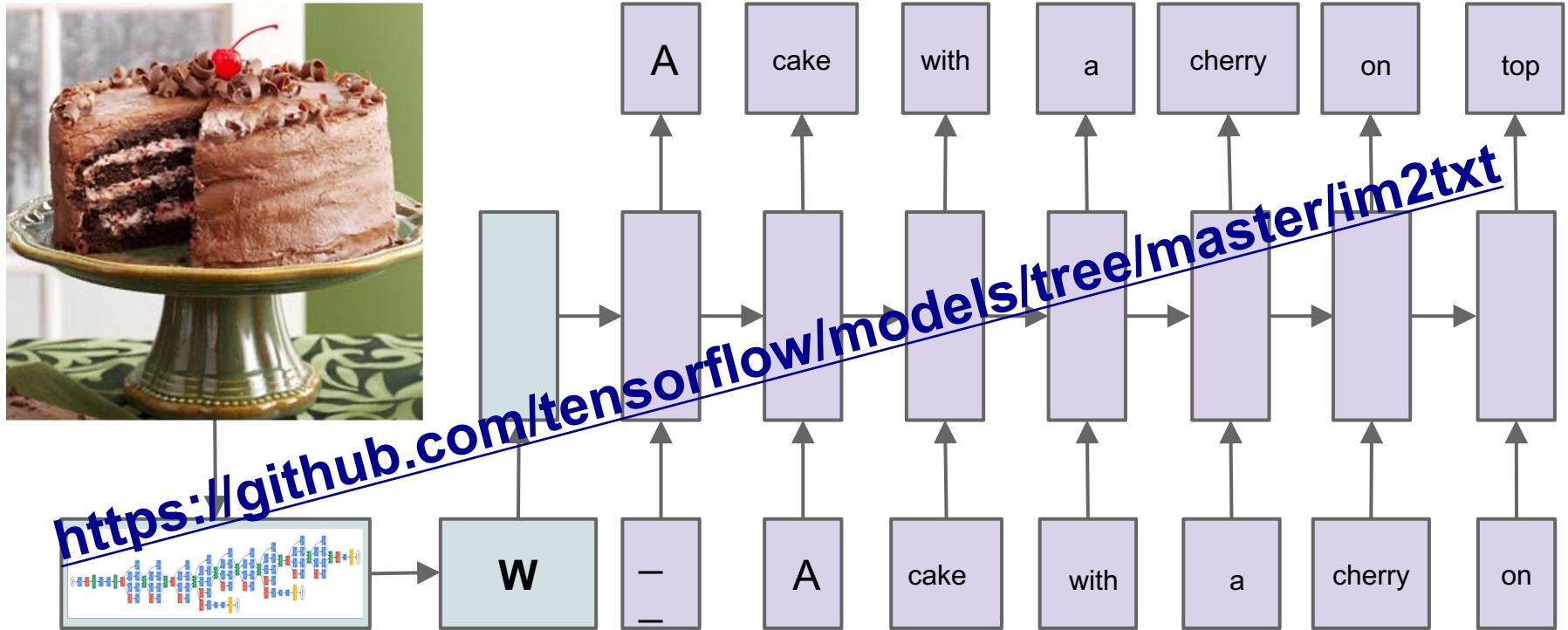
1. Vinyals, O., et al. "Show and Tell: A Neural Image Caption Generator." *CVPR* (2015).
2. Mao, J., et al. "Deep captioning with multimodal recurrent neural networks (m-rnn)." *ICLR* (2015).
3. Karpathy, A., Li, F., "Deep visual-semantic alignments for generating image descriptions." *CVPR* (2015)
4. Kiros, Zemel, Salakhutdinov, "Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models", *TACL 2015*

Image Captioning



$$\theta^* = \arg \max_{\theta} p(S|I)$$

Image Captioning



$$\theta^* = \arg \max_{\theta} p(S|I)$$

Image Captioning



a car is parked in
the middle of nowhere .



a wooden table and chairs
arranged in a room .



a ferry boat on a marina
with a group of people .



there is a cat sitting on a shelf .



a little boy with a bunch
of friends on the street .

Image Captioning



Human: A close up of two bananas with bottles in the background.

BestModel: A bunch of bananas and a bottle of wine.

Image Captioning



Human: A woman holding up a yellow banana to her face.

BestModel: A woman holding a banana up to her face.

Image Captioning



Human: A man outside cooking with a sub in his hand.

BestModel: A man is holding a sandwich in his hand.

Image Captioning



Human: Someone is using a small grill to melt his sandwich.

BestModel: A person is cooking some food on a grill.

Image Captioning



Human: A blue , yellow and red train travels across the tracks near a depot.

BestModel: A blue and yellow train traveling down train tracks.

Learning to Execute

- One of the first (modern) examples of learning algorithms
- 2014--??? “era of discovery” → Apply seq2seq to *everything*

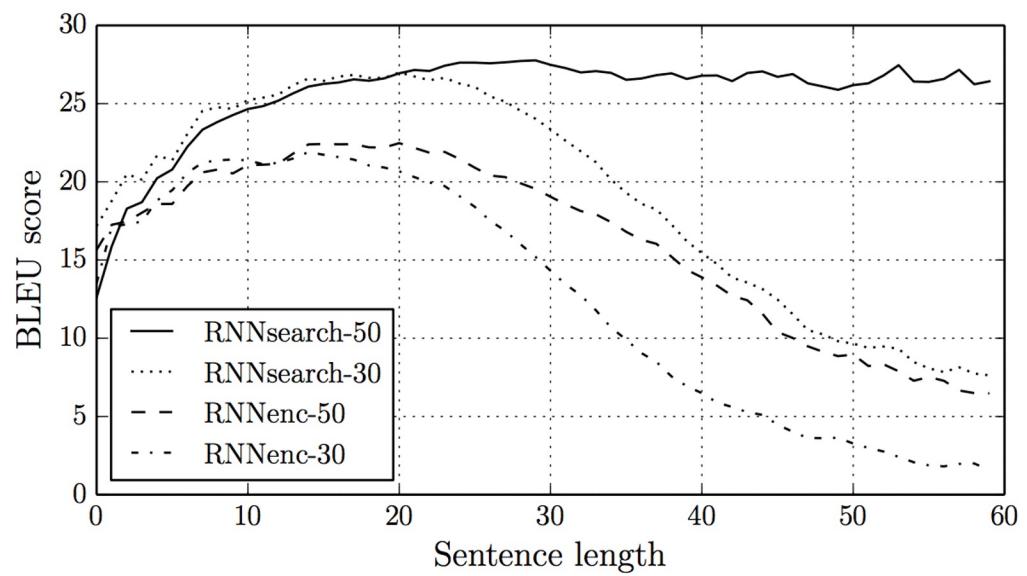
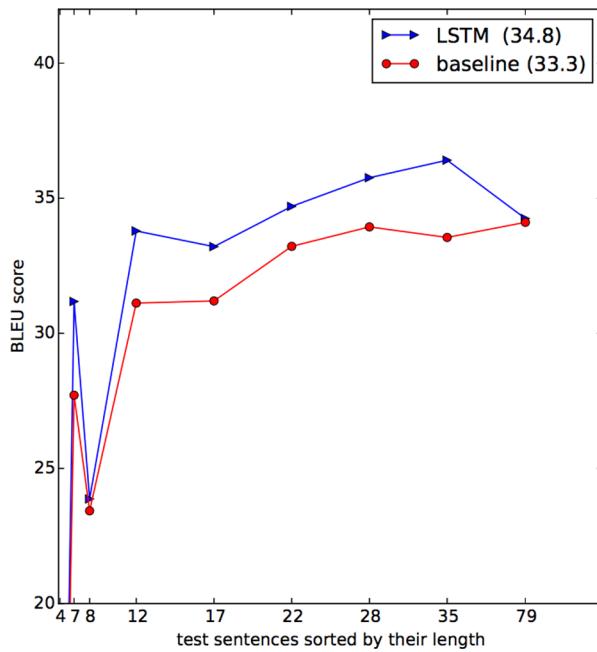
Input:
j=8584
for x in range(8):
 j+=920
 b=(1500+j)
 print((b+7567))
Target: 25011.

Input:
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
Target: 12184.

Input:
vqppkn
sqdvljmnc
y2vxdddsepnimcbvubkomhrpliibtwztbljipcc
Target: hkhpg

Seq2Seq - Limitations

- Fixed Size Embeddings are easily overwhelmed by long inputs or long outputs



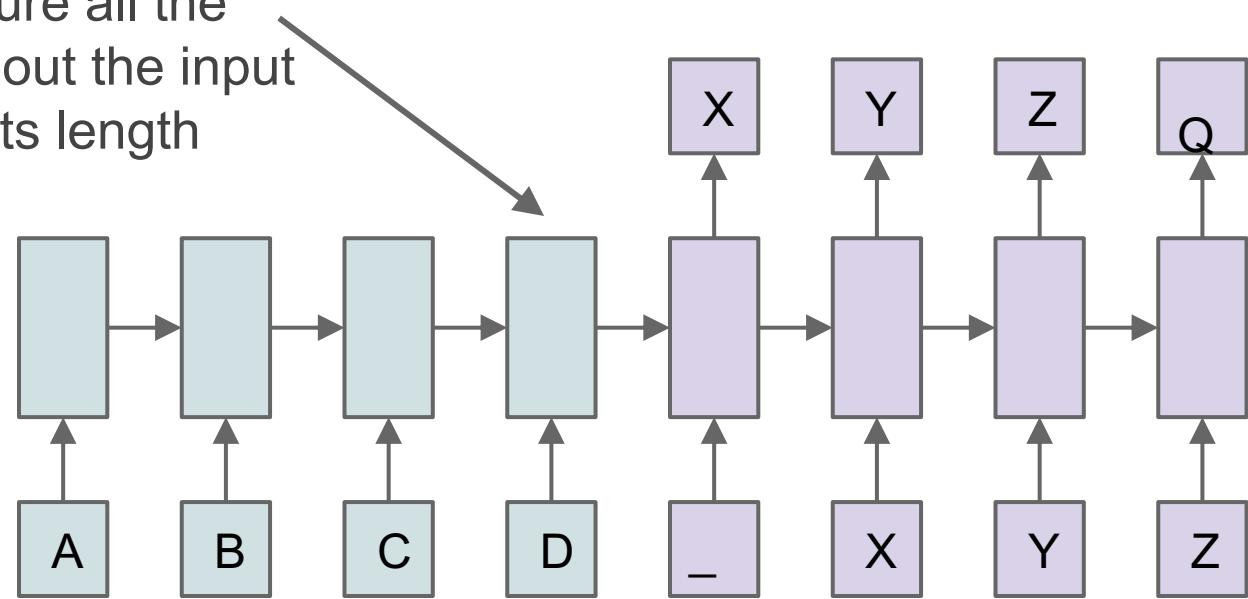
Sutskever, I., et al. "Sequence to Sequence Learning with Neural Networks." *NIPS* (2014)

Bahdanau, D., et al. "Neural Machine Translation by Jointly Learning to Align and Translate." *ICLR* (2015)

Attention

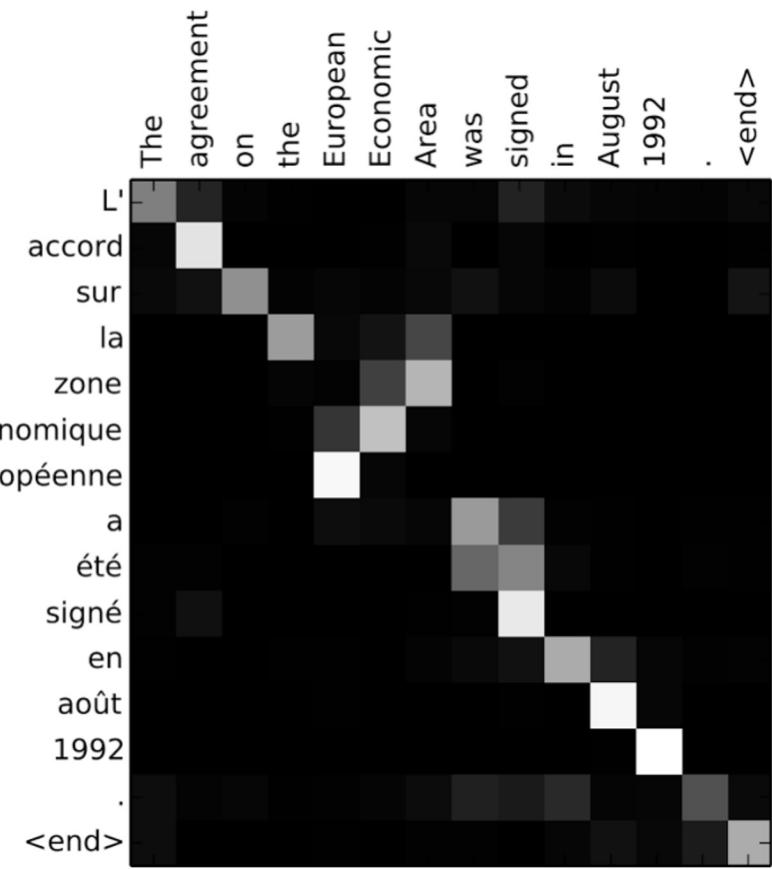
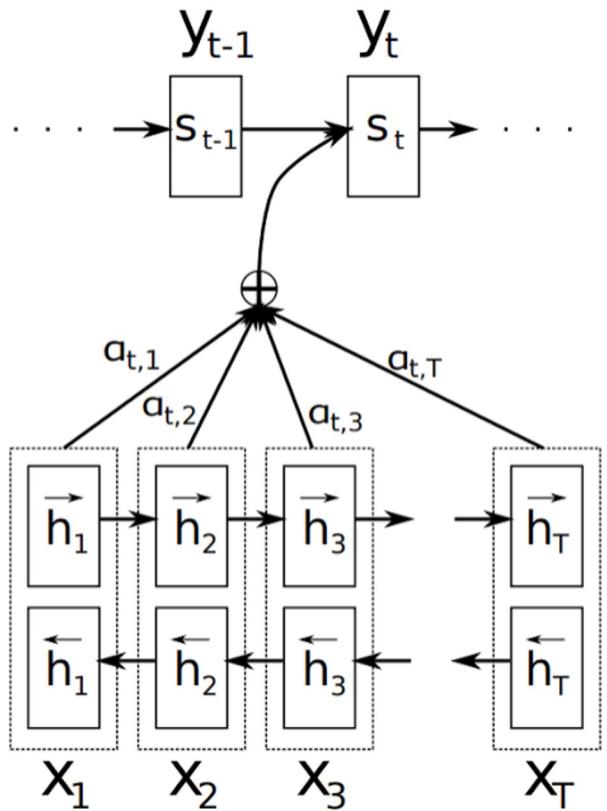
Seq2Seq - The issue with long inputs

- Same embedding informs the entire output
- Needs to capture all the information about the input regardless of its length

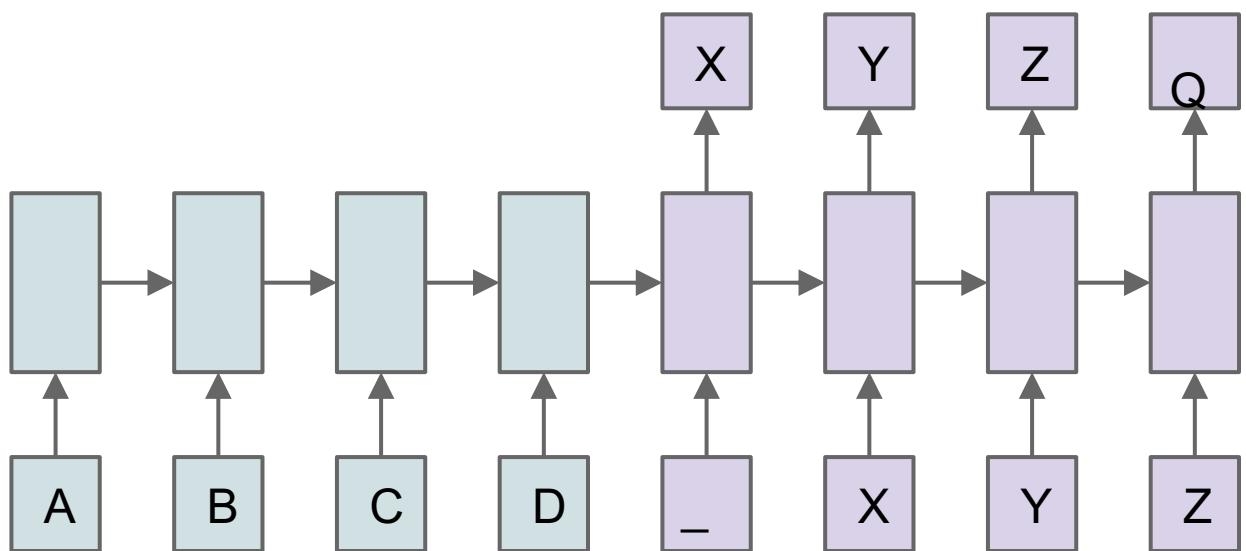


Is there a better way to pass the information from encoder to the decoder ?

Seq2Seq with Attention

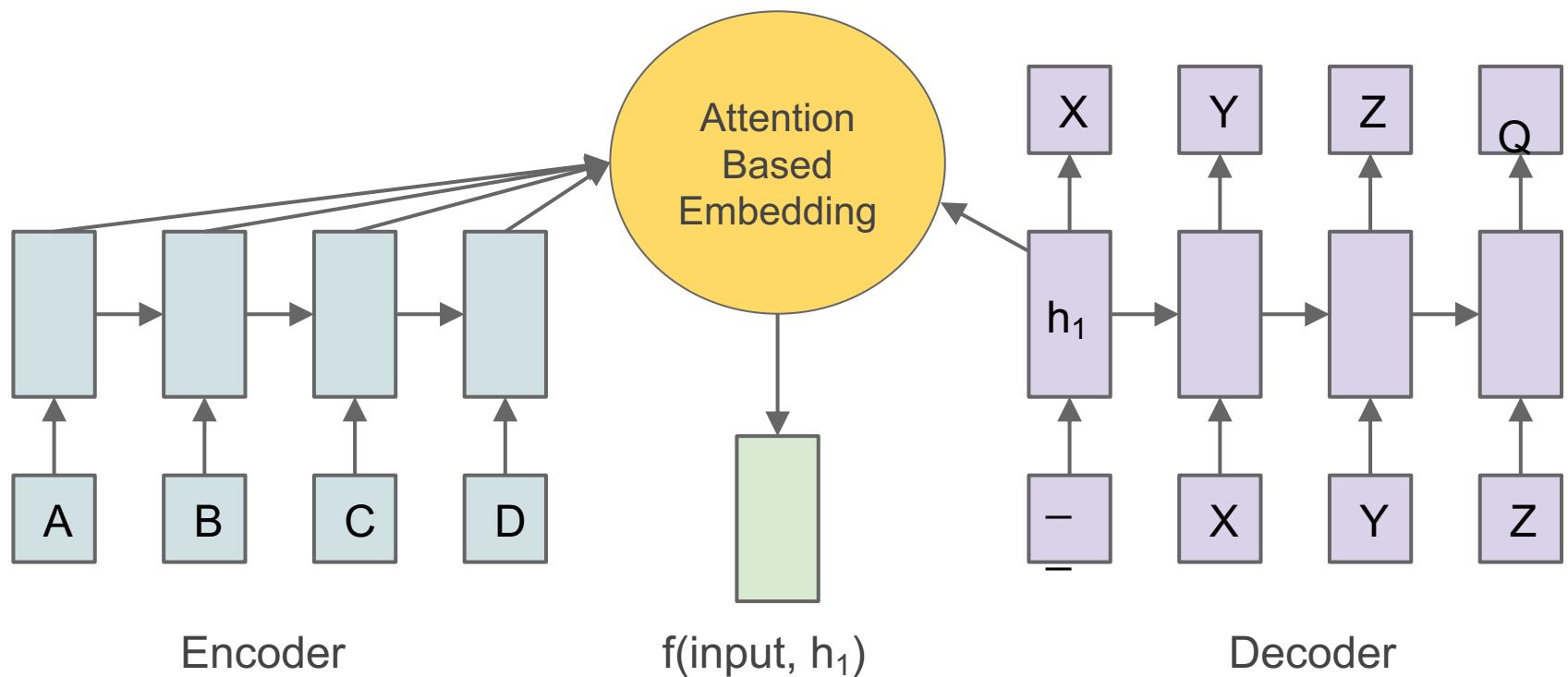


Seq2Seq



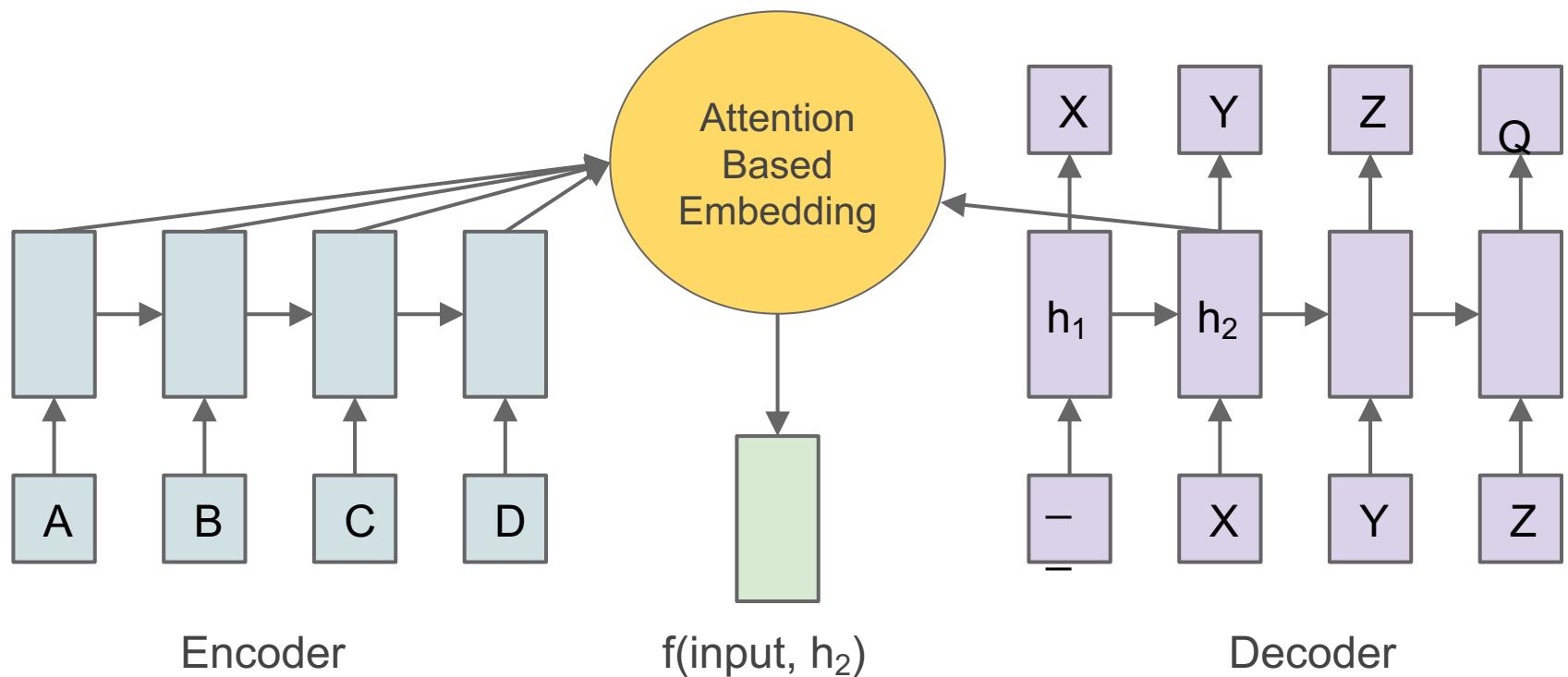
Seq2Seq with Attention

- A different embedding computed for every output step



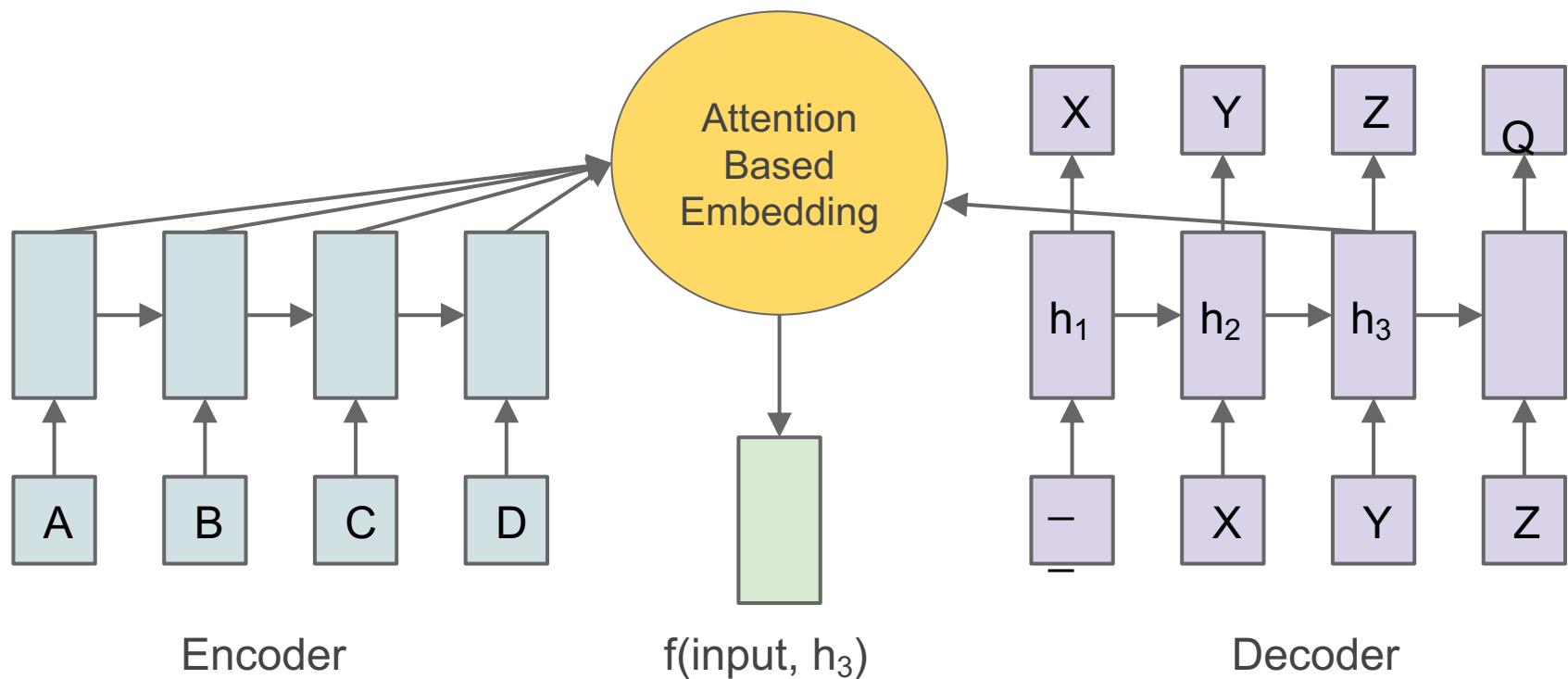
Seq2Seq with Attention

- A different embedding computed for every output step



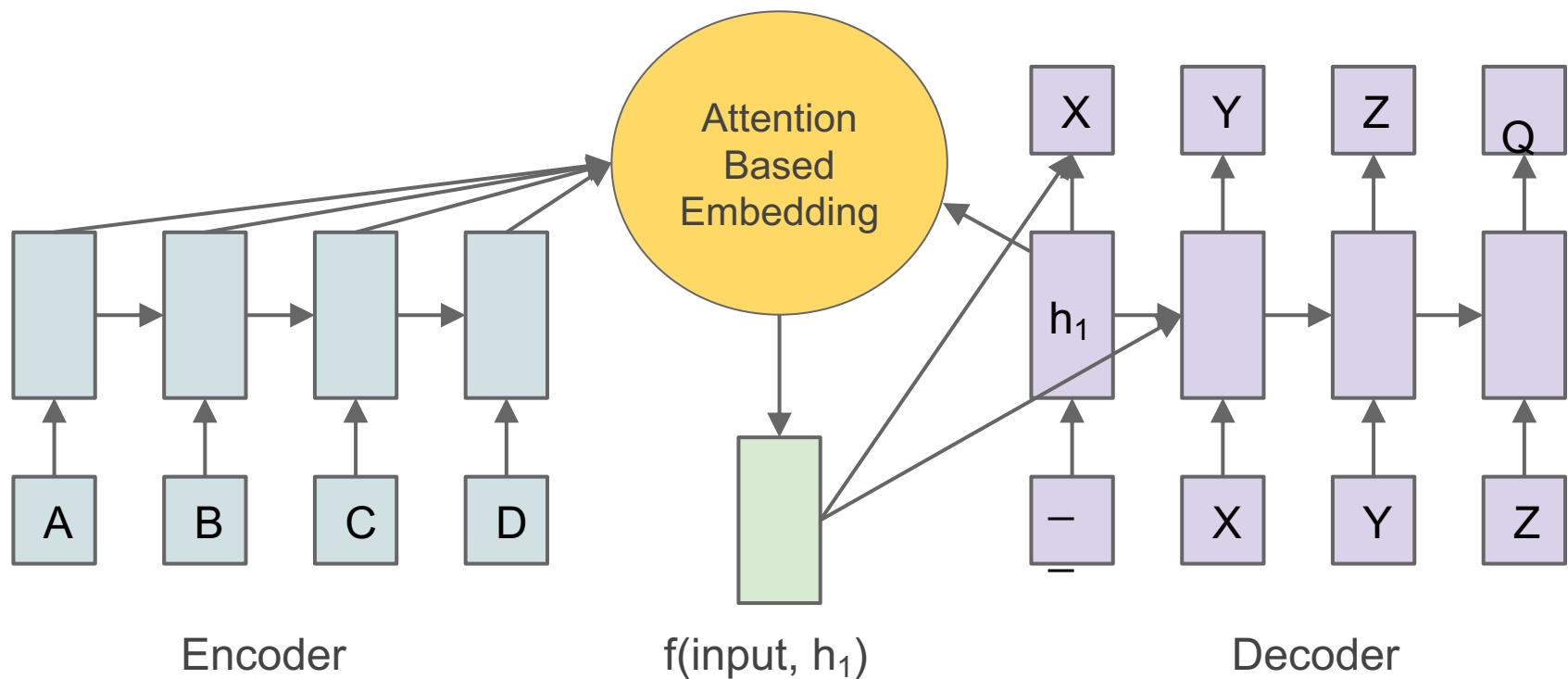
Seq2Seq with Attention

- A different embedding computed for every output step



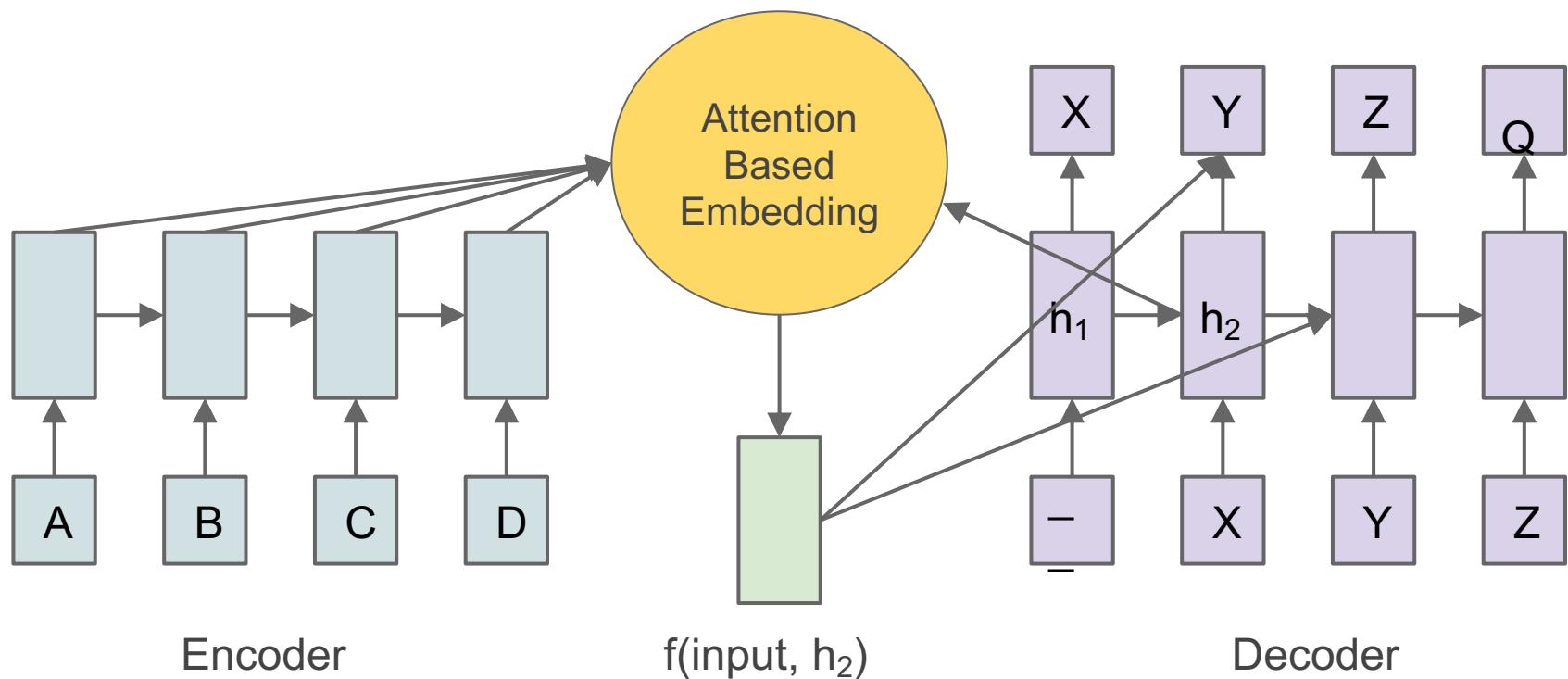
Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state



Seq2Seq with Attention

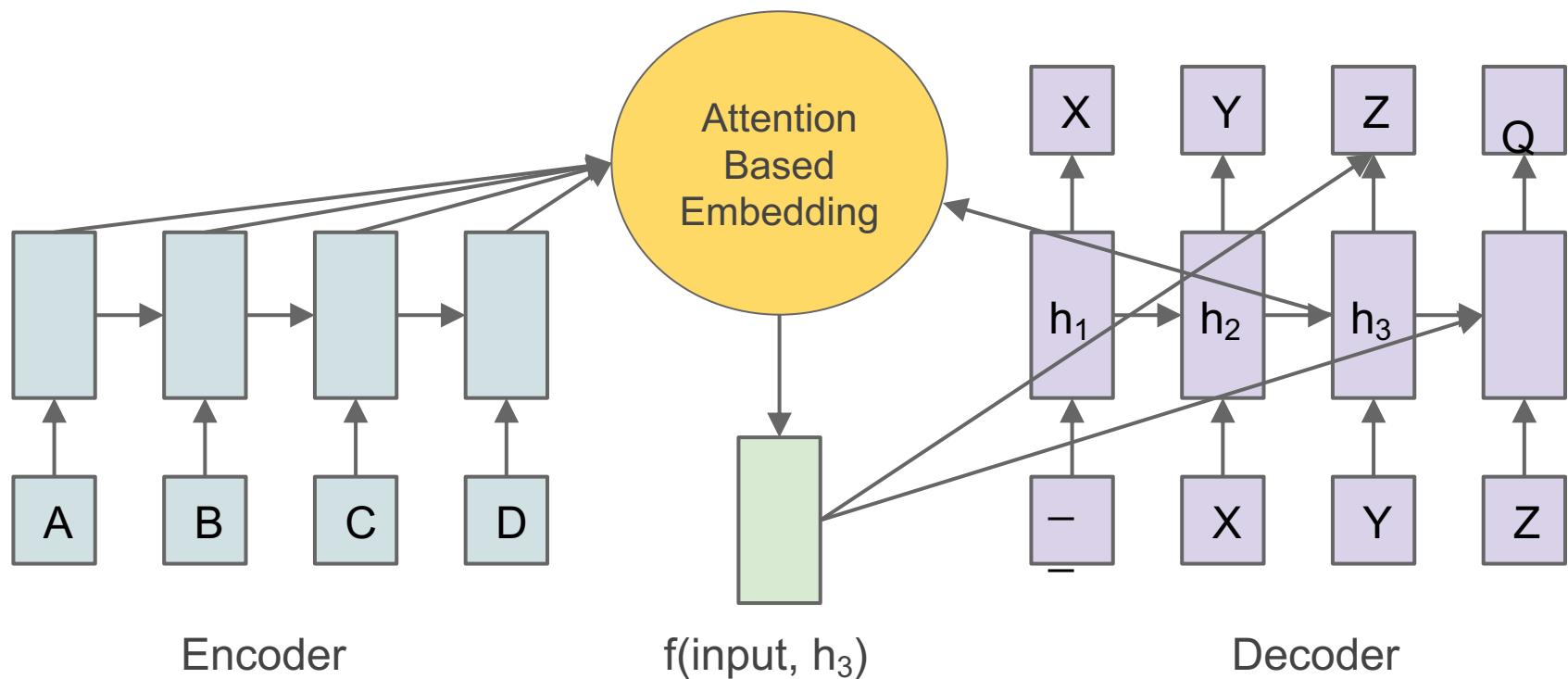
- Embedding used to predict output, and compute next hidden state



- Attention arrows for step 1 omitted

Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state



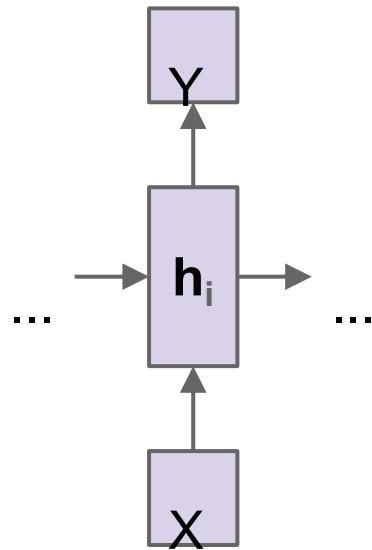
- Attention arrows for steps 1 and 2 omitted

Attention Based Embedding

- Linear blending of embedding RNN states $e_1 e_2 e_3 e_4$ is a natural choice
- How to produce the coefficients (attention vector) for blending ?
 - Content based coefficients based on query state h_i and embedding RNN states $e_1 e_2 e_3 e_4$

Dot product Attention

- Inputs: “I am a cat.”
- Input RNN states: $\mathbf{e}_1 \mathbf{e}_2 \mathbf{e}_3 \mathbf{e}_4$
- Decoder RNN state at step i (query): \mathbf{h}_i
- Compute scalars $\mathbf{h}_i^T \mathbf{e}_1, \mathbf{h}_i^T \mathbf{e}_2, \mathbf{h}_i^T \mathbf{e}_3, \mathbf{h}_i^T \mathbf{e}_4$ representing similarity / relevance between encoder steps and query.
- Normalize $[\mathbf{h}_i^T \mathbf{e}_1, \mathbf{h}_i^T \mathbf{e}_2, \mathbf{h}_i^T \mathbf{e}_3, \mathbf{h}_i^T \mathbf{e}_4]$ with softmax to produce attention weights, e.g. [0.0 0.05 0.9 0.05]



Content Based Attention

Attention [Bahdanau, Cho and Bengio, 2014]

$$u_j = v^T \tanh(W_1 e_j + W_2 d) \quad j \in (1, \dots, n)$$

$$a_j = \text{softmax}(u_j) \quad j \in (1, \dots, n)$$

$$d' = \sum_{j=1}^n a_j e_j$$

Graves, A., et al. "Neural Turing Machines." *arxiv* (2014)

Weston, J., et al. "Memory Networks." *arxiv* (2014)

Other strategies for attention models

- Tensored attention
 - Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation.” EMNLP’15.
- Multiple heads
- Pyramidal encoders
 - William Chan, Navdeep Jaitly, Quoc Le, Oriol Vinyals. “Listen Attend and Spell”. ICASSP 2015.
- Hierarchical Attention
 - Andrychowicz, Marcin, and Karol Kurach. "Learning efficient algorithms with hierarchical attentive memory." *arXiv preprint arXiv:1602.03218* (2016).
- Hard Attention
 - Xu, Kelvin, et al. “Show, attend and tell: Neural image caption generation with visual attention.” ICML 2015