10707
Deep Learning

Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

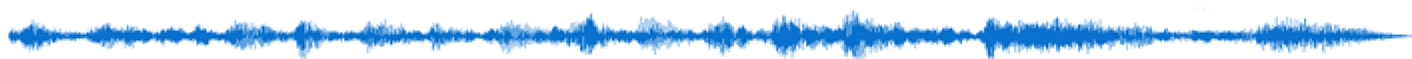
Sequence Models

Sequences

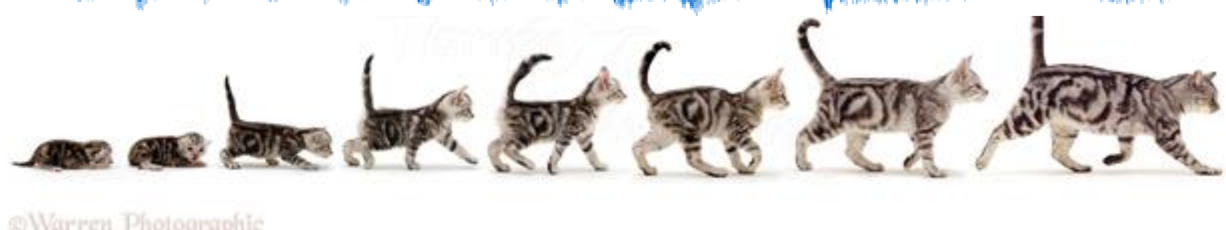
- Words, Letters

> *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



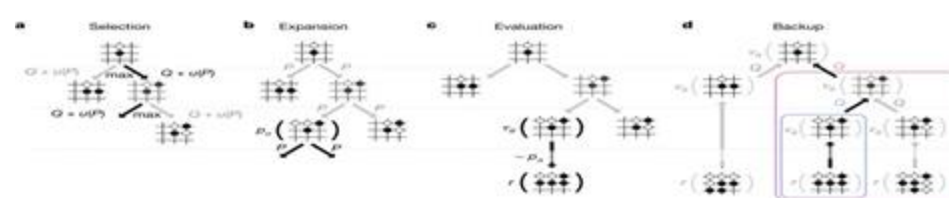
- Images, Videos



- 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

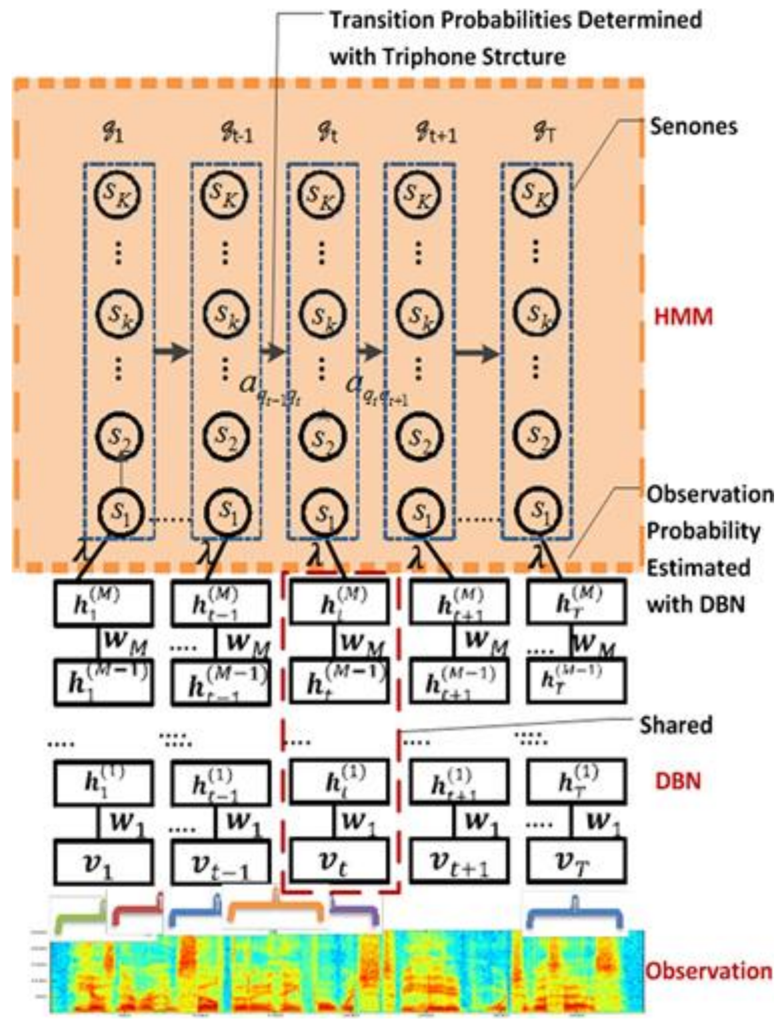
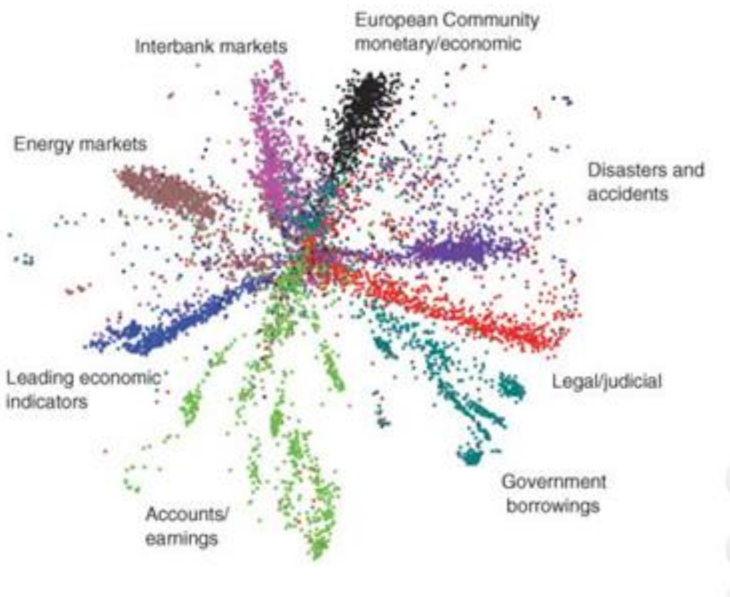
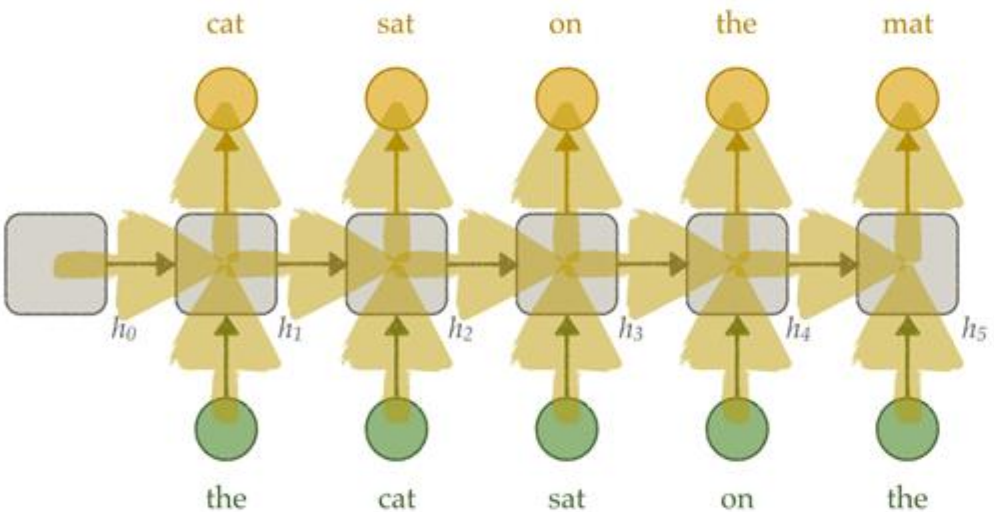


Figure credit: Li Deng

Two Key Ingredients

Neural Embeddings + Recurrent Language Models





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

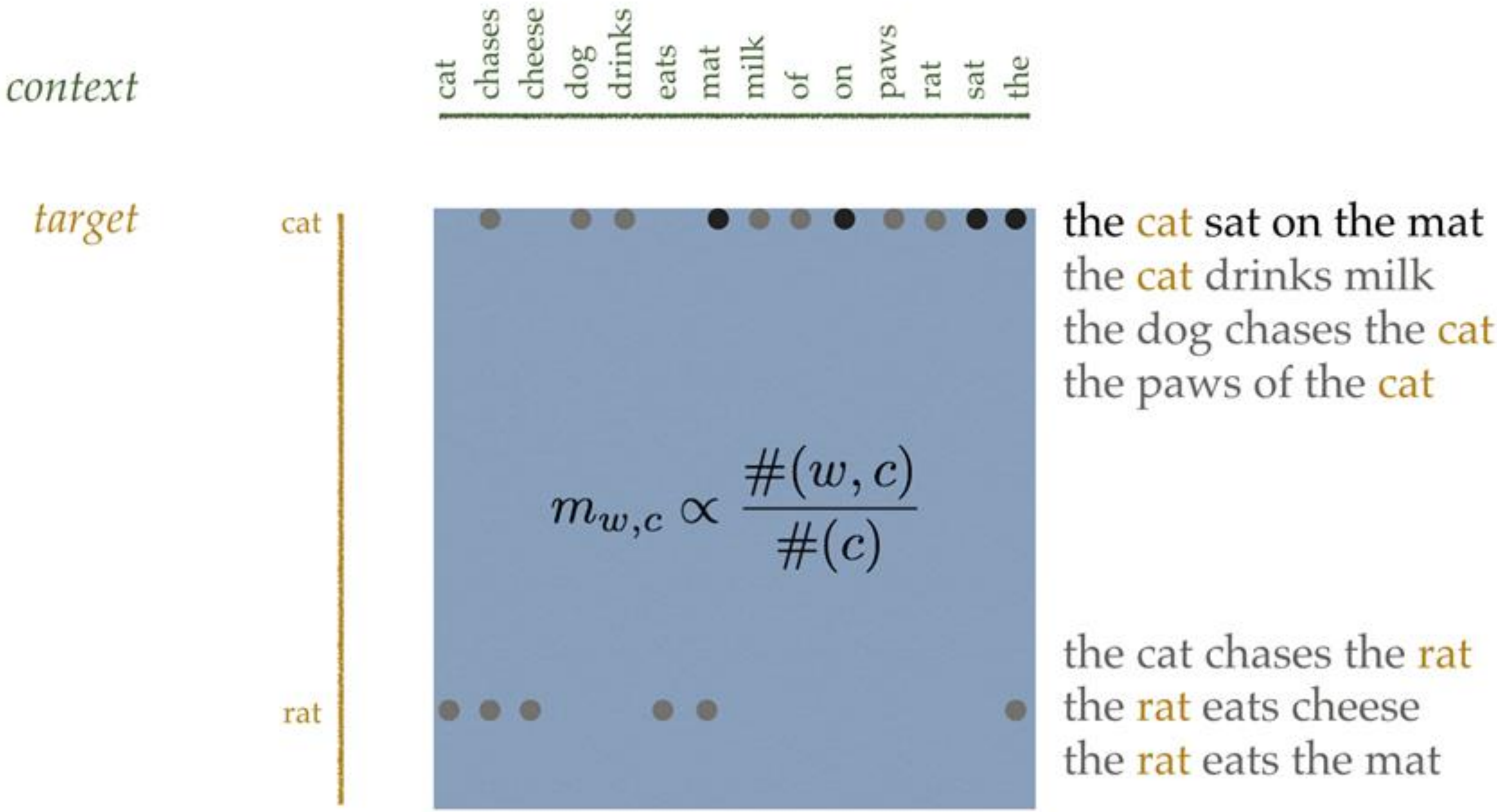
Mikolov, T., et al. "Recurrent neural network based language model." *Interspeech (2010)*

Language Models

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

N-grams

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

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

Chain Rule

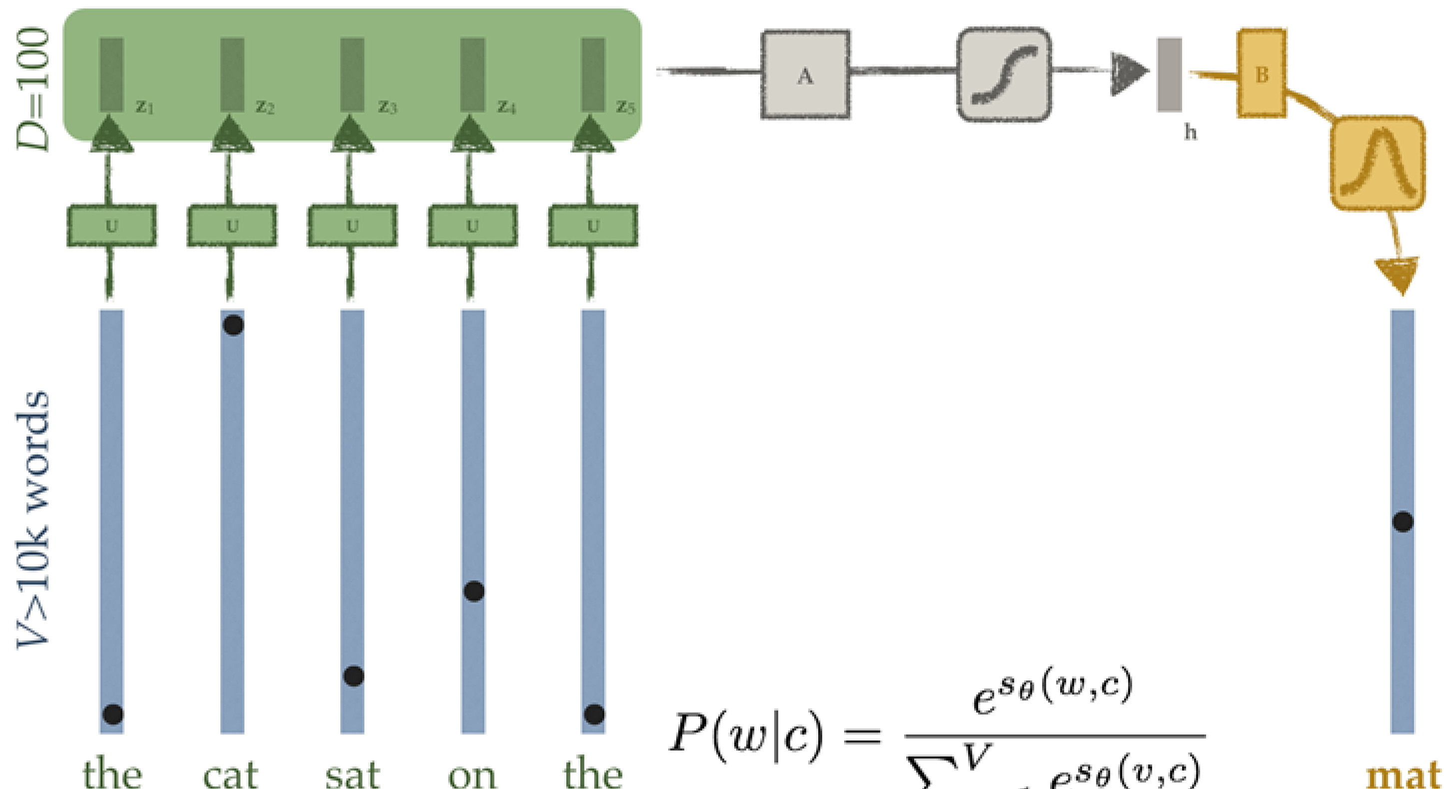
$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

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

Key Insight: Vectorizing Context

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



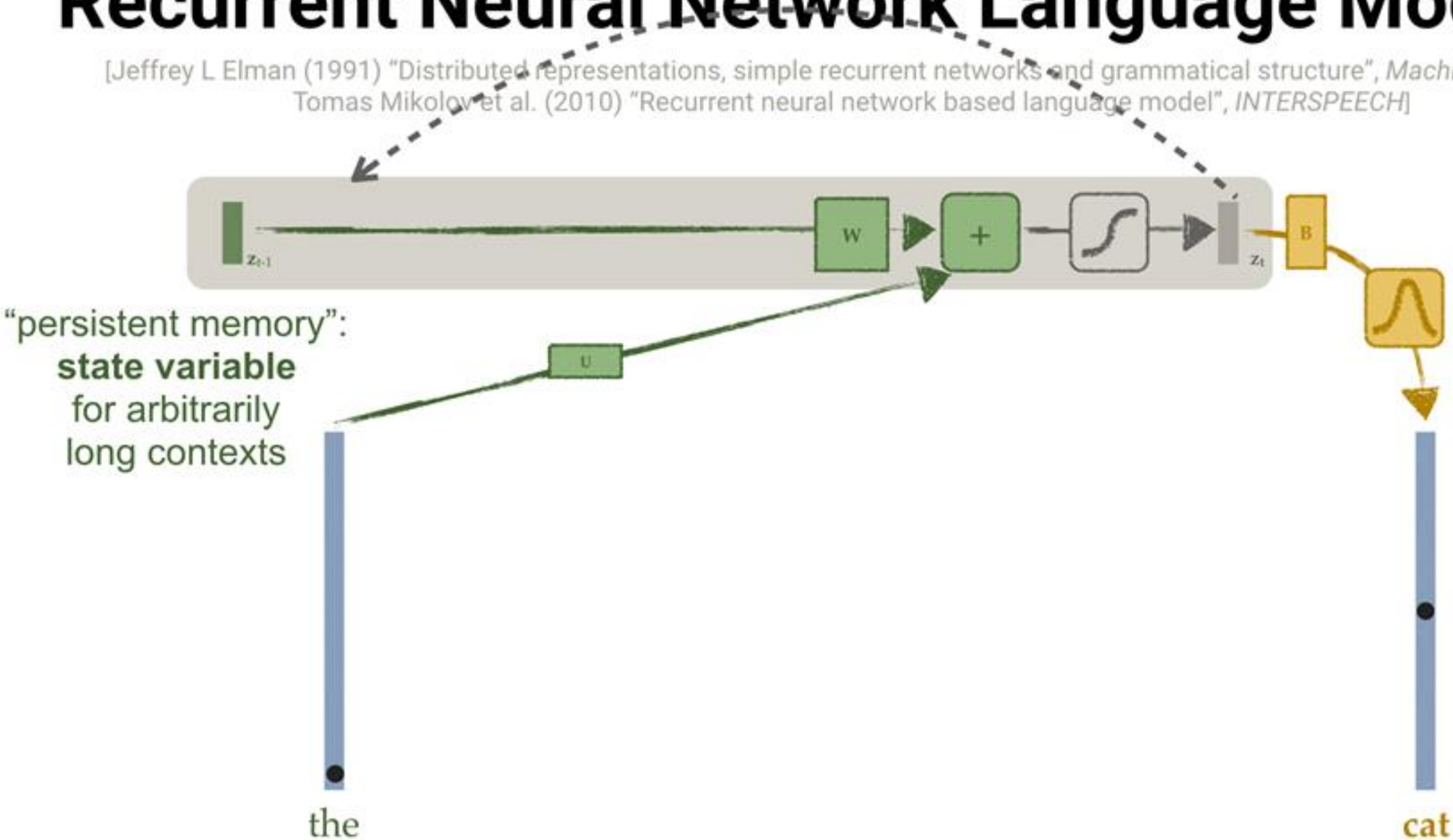
$$P(w|c) = \frac{e^{s_\theta(w,c)}}{\sum_{v=1}^{V} e^{s_\theta(v,c)}}$$

Bengio, Y. et al., "A Neural Probabilistic Language Model", *JMLR (2001, 2003)*
Mnih, A., Hinton, G., "Three new graphical models for statistical language modeling", *ICML 2007*

Slide Credit: Piotr Mirowski

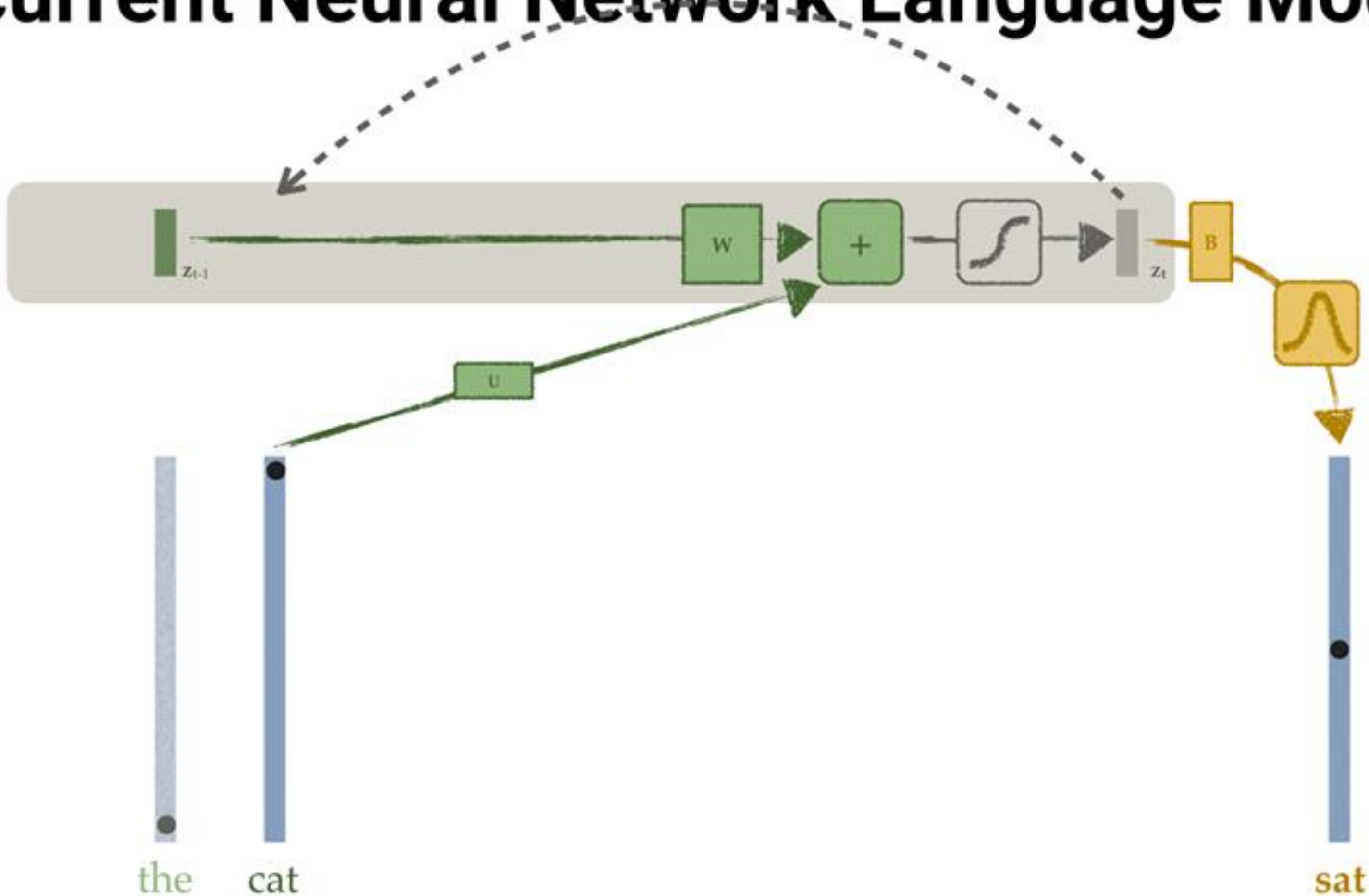
Recurrent Neural Network Language Models

[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*]



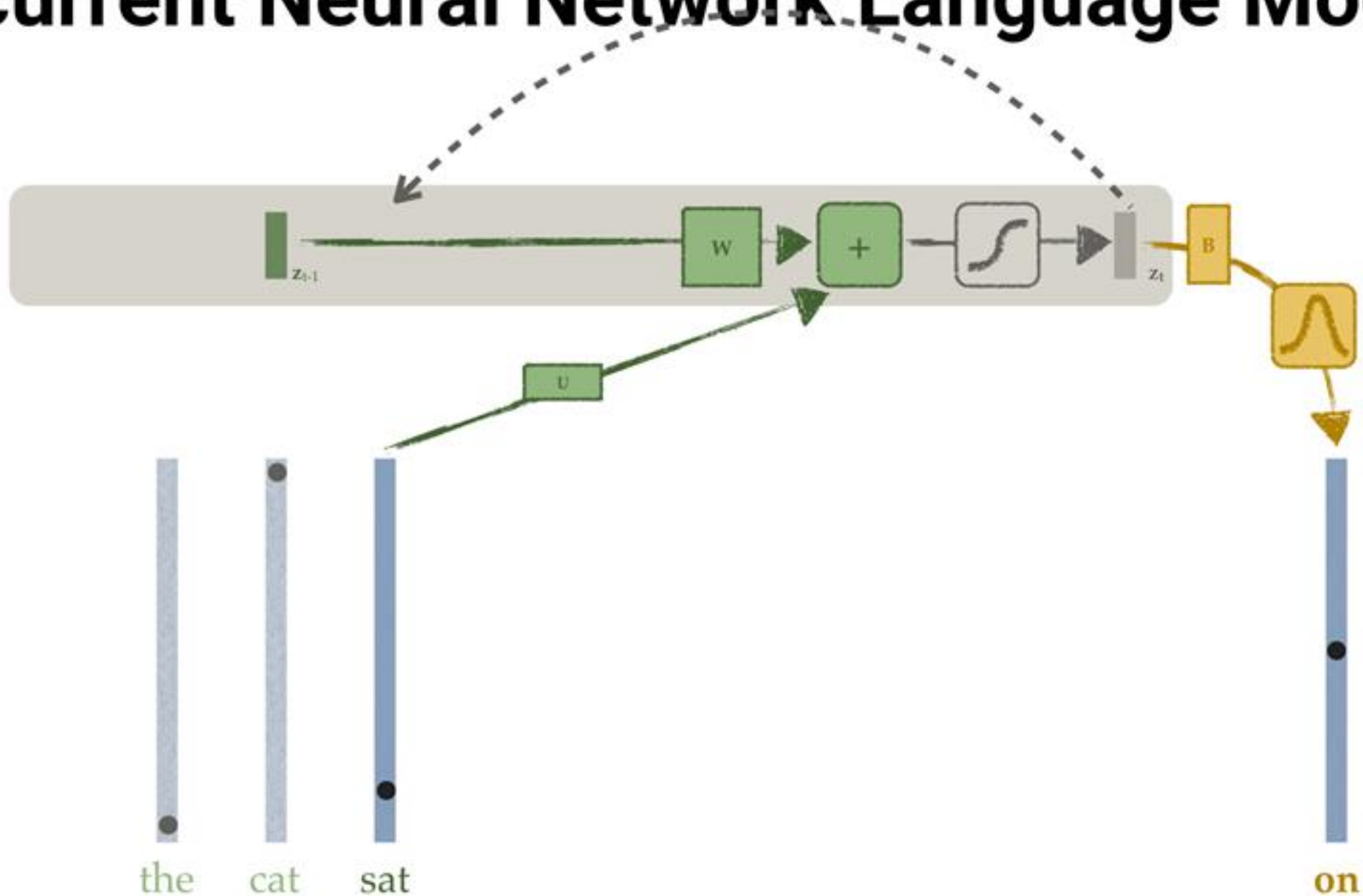
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



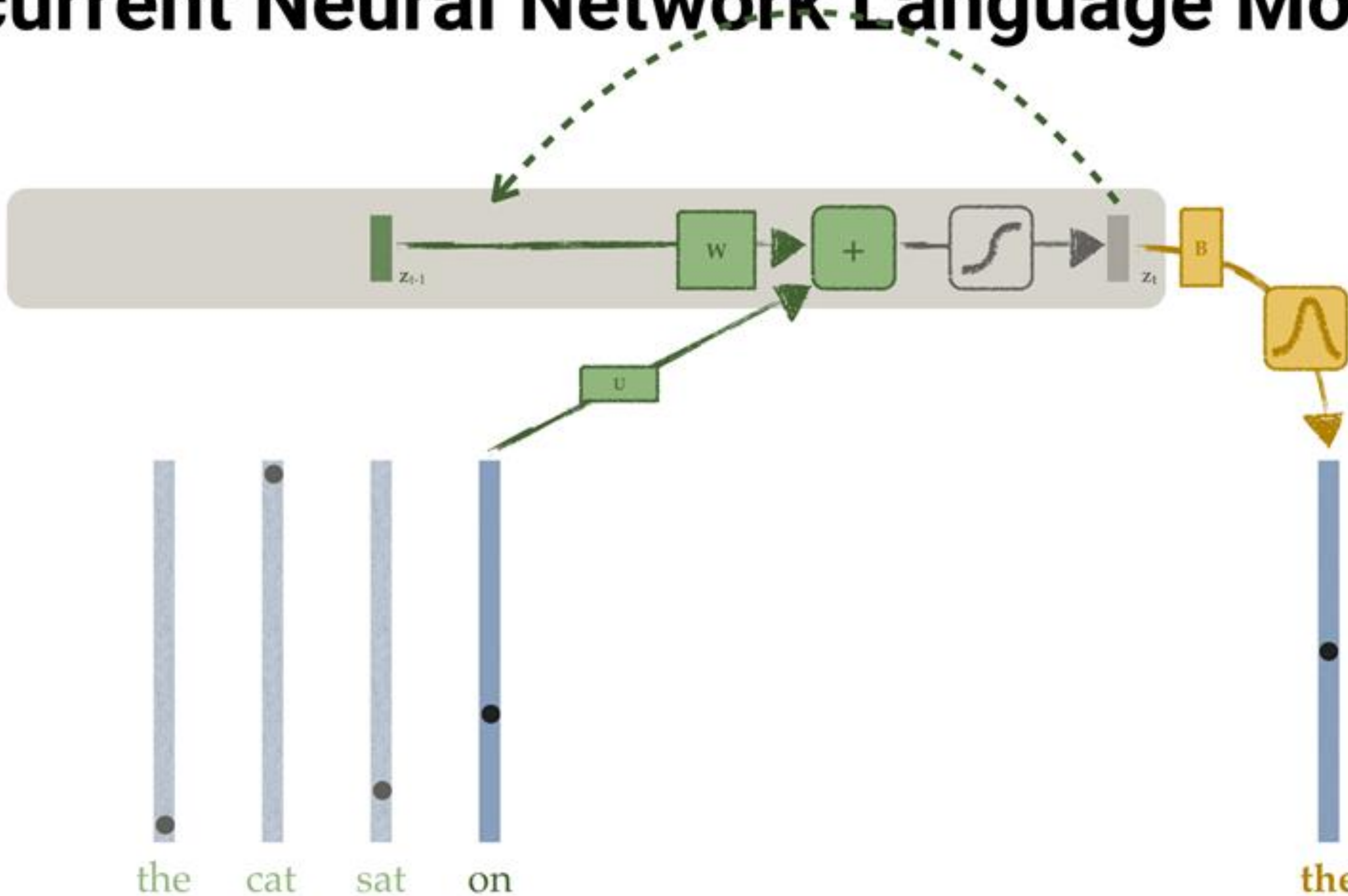
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



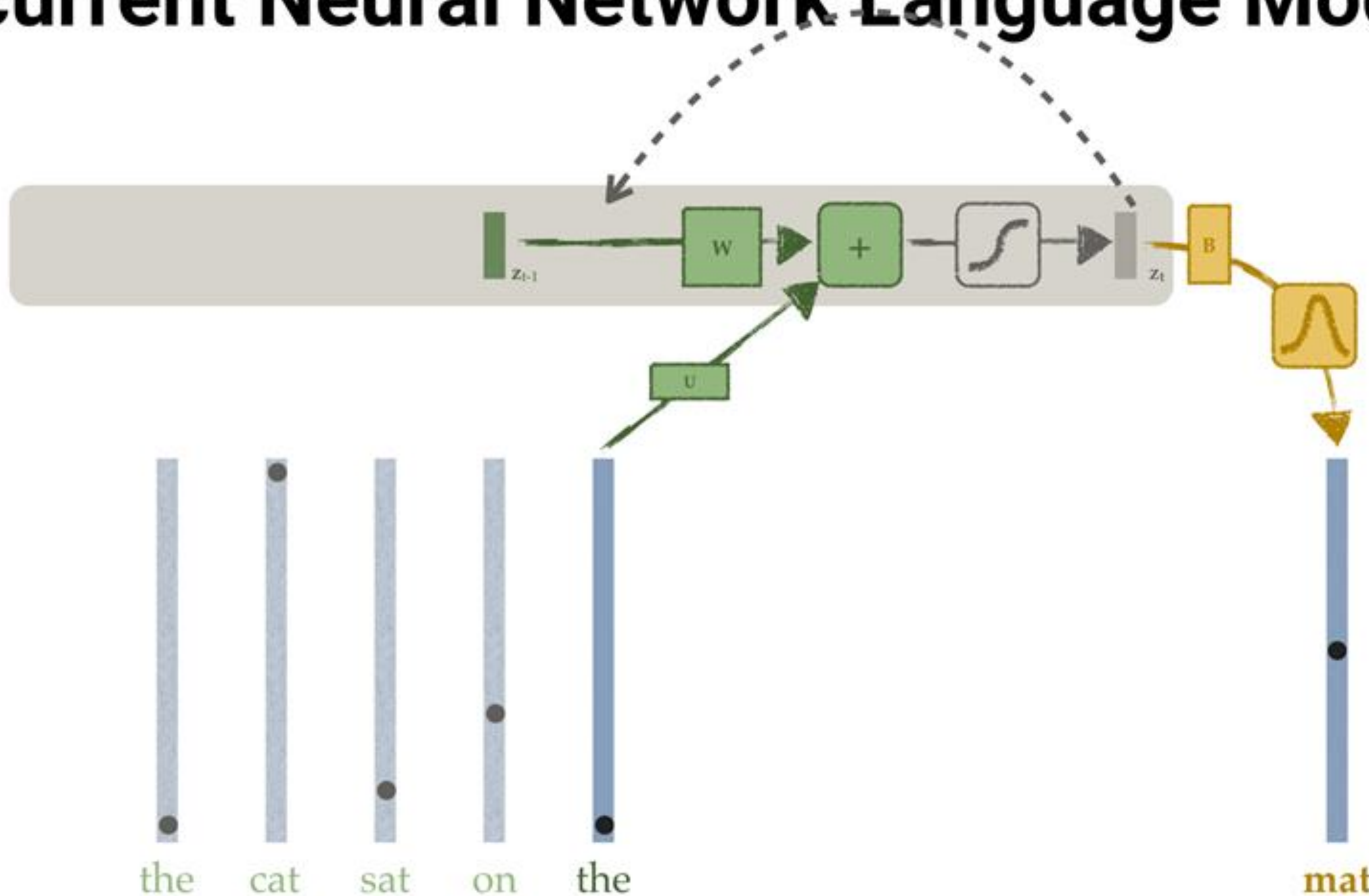
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

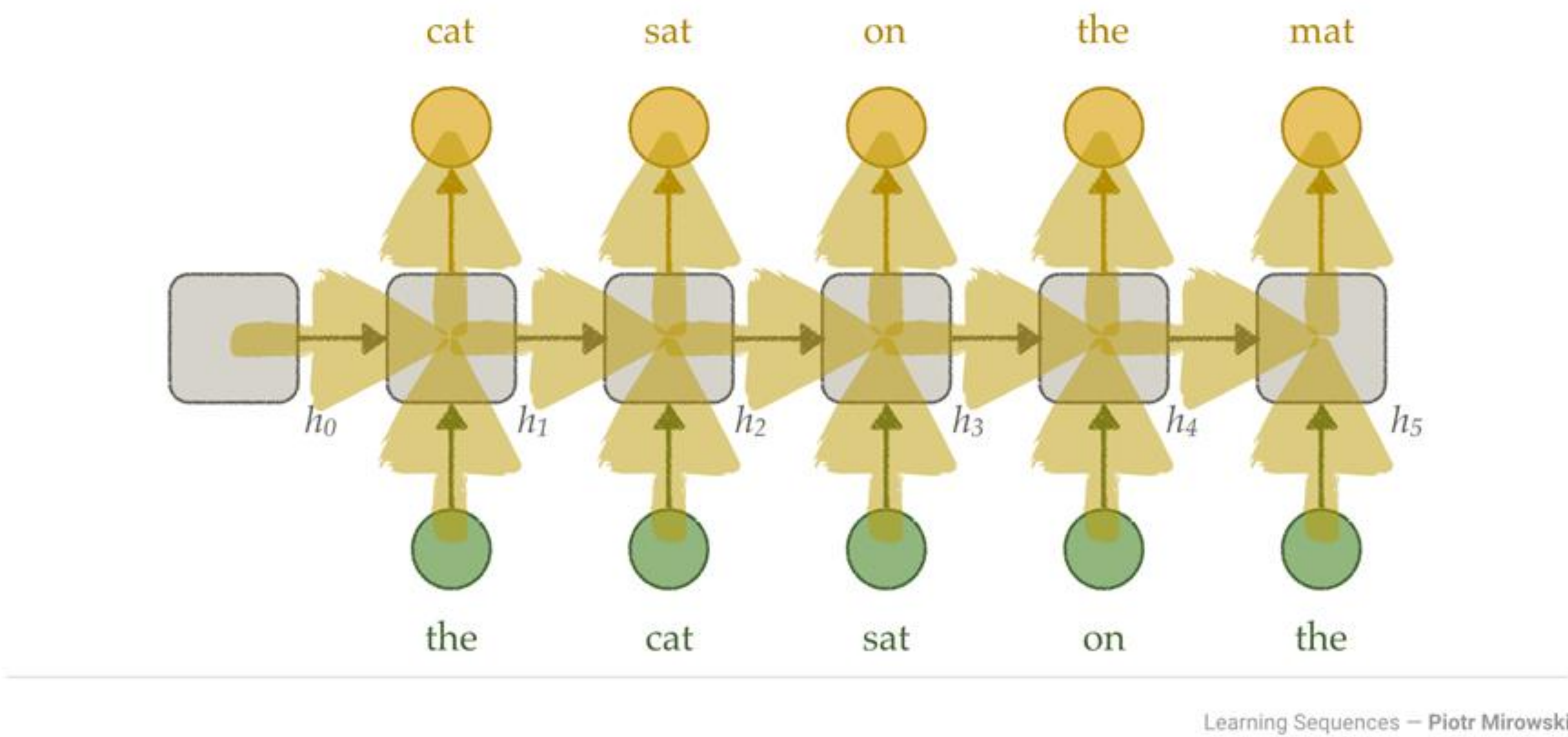


Slide Credit: Piotr Mirowski

What do we Optimize?

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

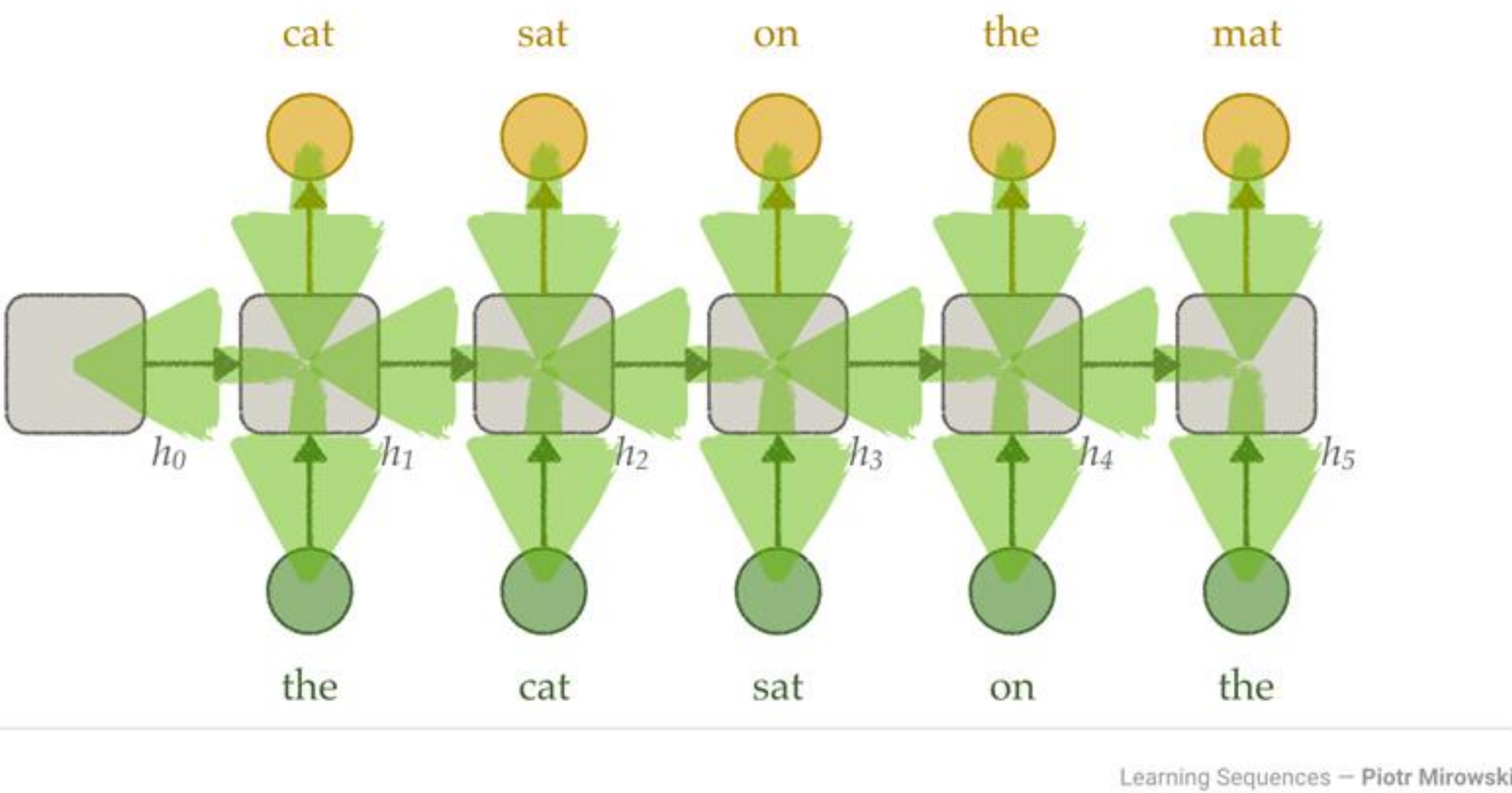
Recurrent Neural Network Language Models



- Forward Pass

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



- Backward Pass

Slide Credit: Piotr Mirowski

Seq2Seq

Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig

Recurrent Continuous Translation Models

Nal Kalchbrenner, Phil Blunsom

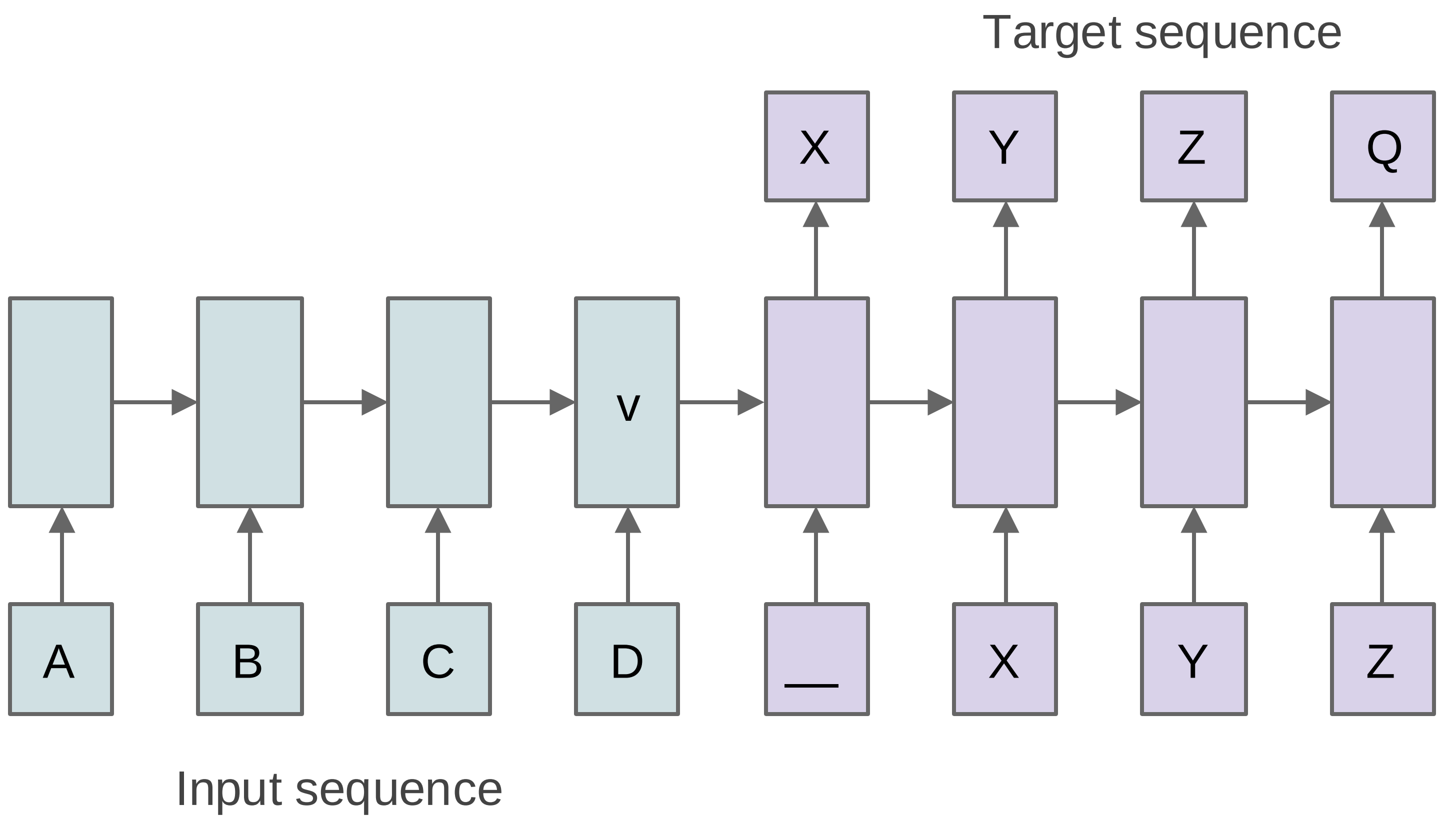
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever, Oriol Vinyals, Quoc V. Le

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)*

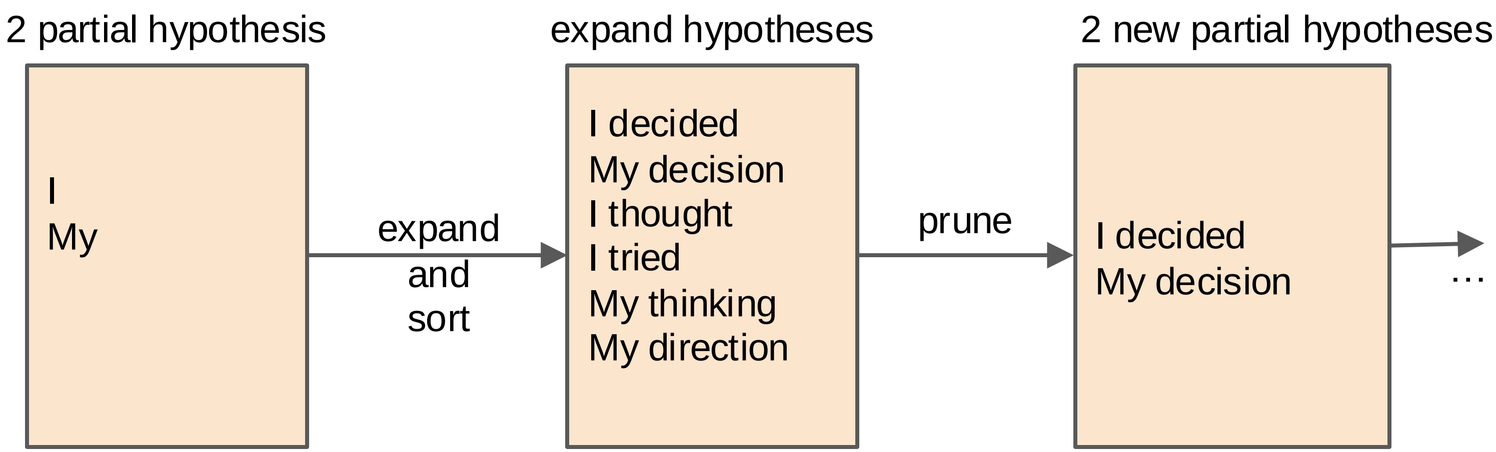
Seq2Seq



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

Decoding in a Nutshell (Beam Size 2)

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



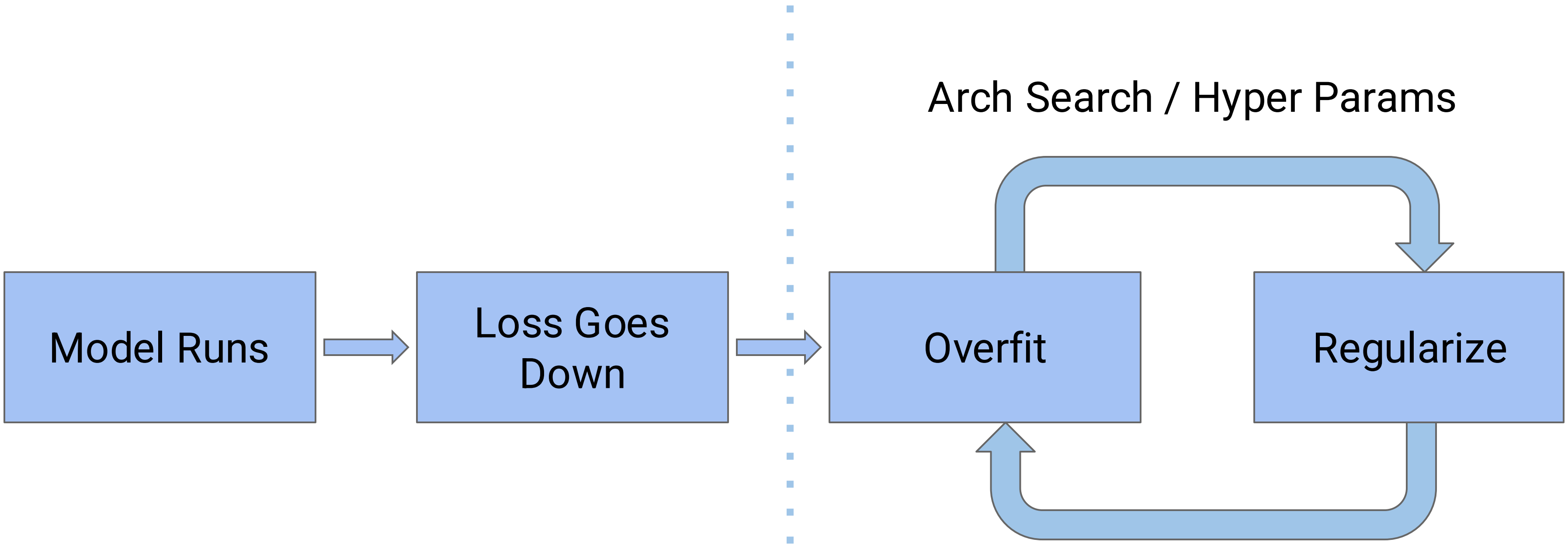
Code

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



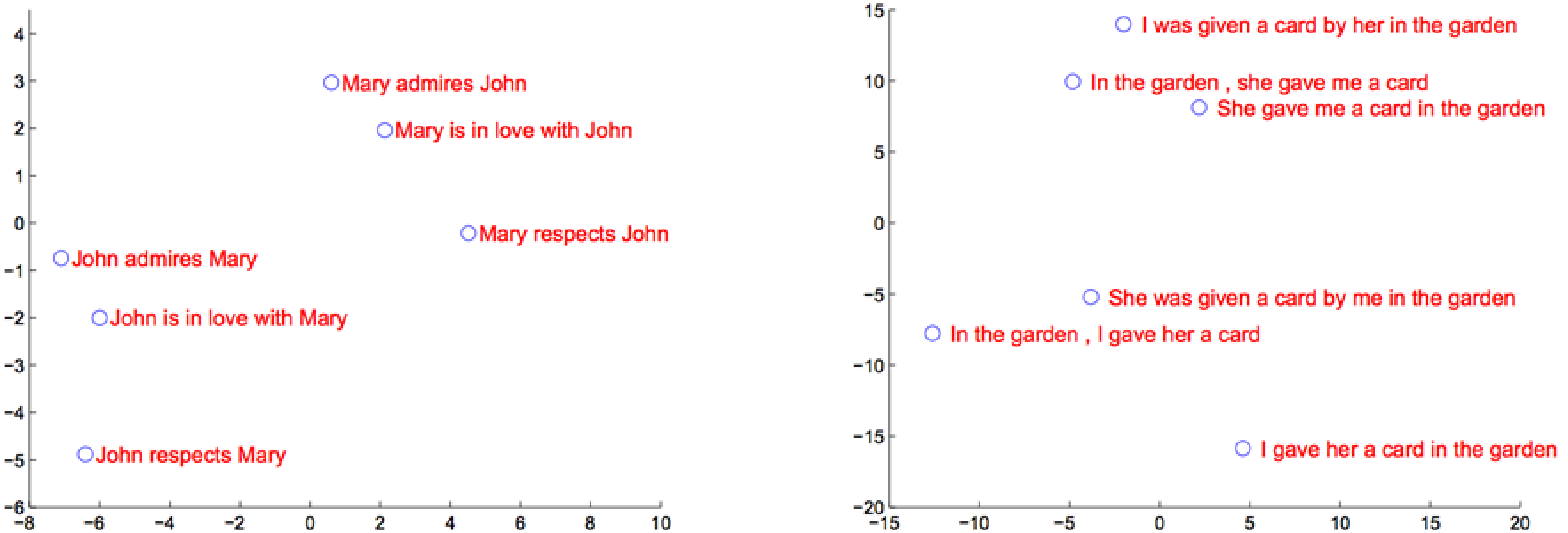
(Some) Tricks of the Trade

- 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

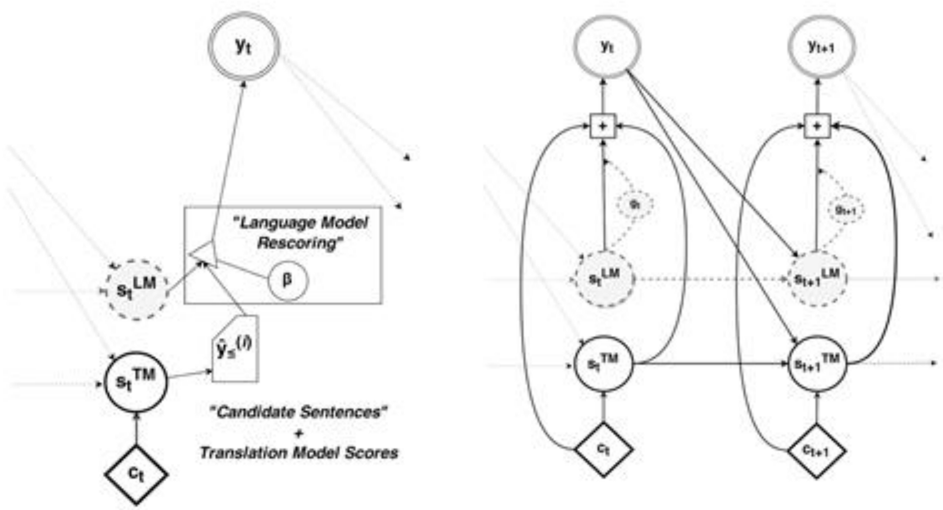
Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size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



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

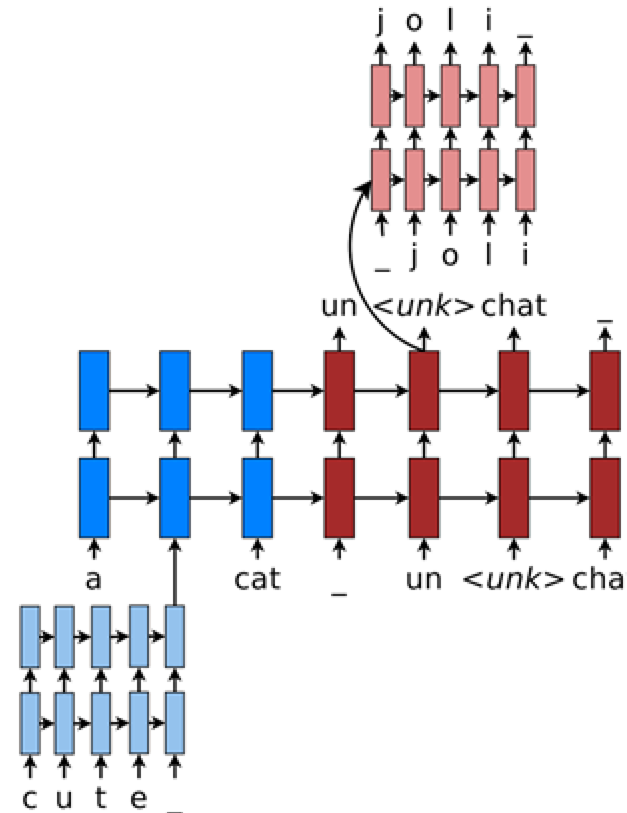
Machine Translation: Concerns

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



(a) Shallow Fusion (Sec. 4.1)

(b) Deep Fusion (Sec. 4.2)



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

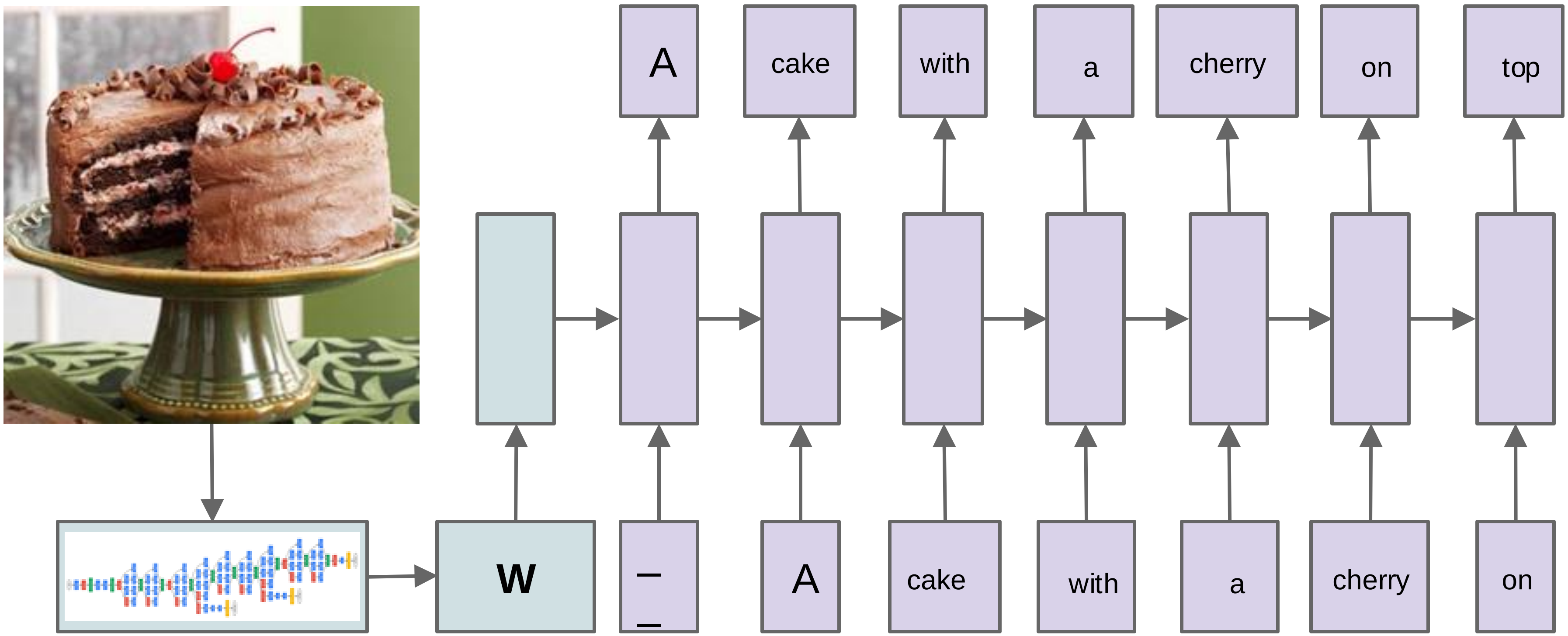
p(English | French)

$\downarrow$

p(English | Image)

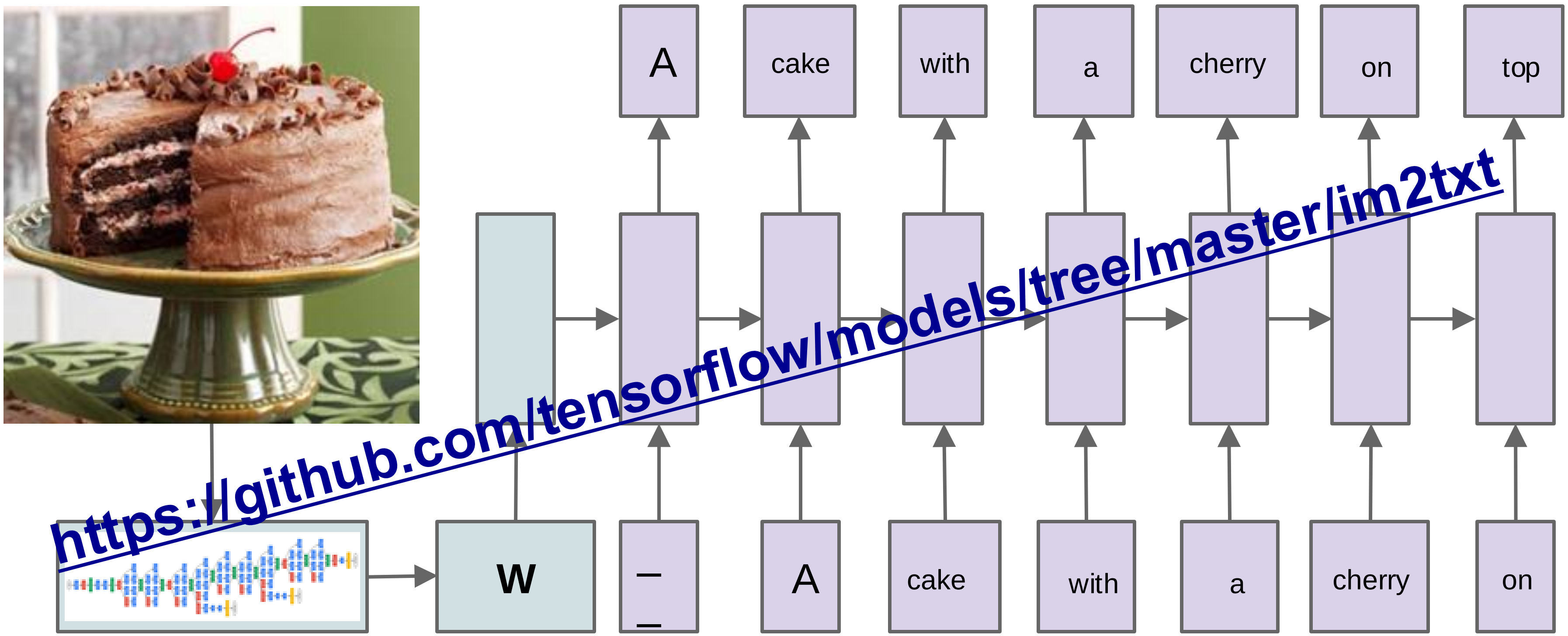
1. Vinyals, O., et al. "Show and Tell: A Neural Image Caption Generator." *CVPR* (2015).
2. Mao, J., et al. "Deep captioning with multimodal recurrent neural networks (m-rnn)." *ICLR (2015).*
3. Karpathy, A., Li, F., "Deep visual-semantic alignments for generating image descriptions." *CVPR (2015)*
4.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

Image Captioning



$$\theta^\star = \arg\max_\theta p(S|I)$$

Image Captioning



$$\theta^\star = \arg\max_\theta p(S|I)$$

Image Captioning

LZ

a car is parked in
the middle of nowhere .

a wooden table and chairs
arranged in a room .

there is a cat sitting on a shelf .

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

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:

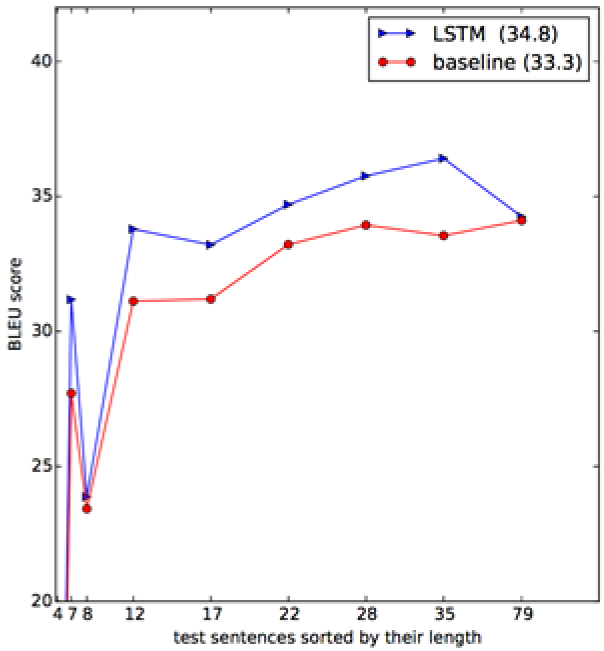
```
vqppkn
sqdvfljmnc
y2vxdddsepnimcbvubkomhrpliibtwztbljipcc
```

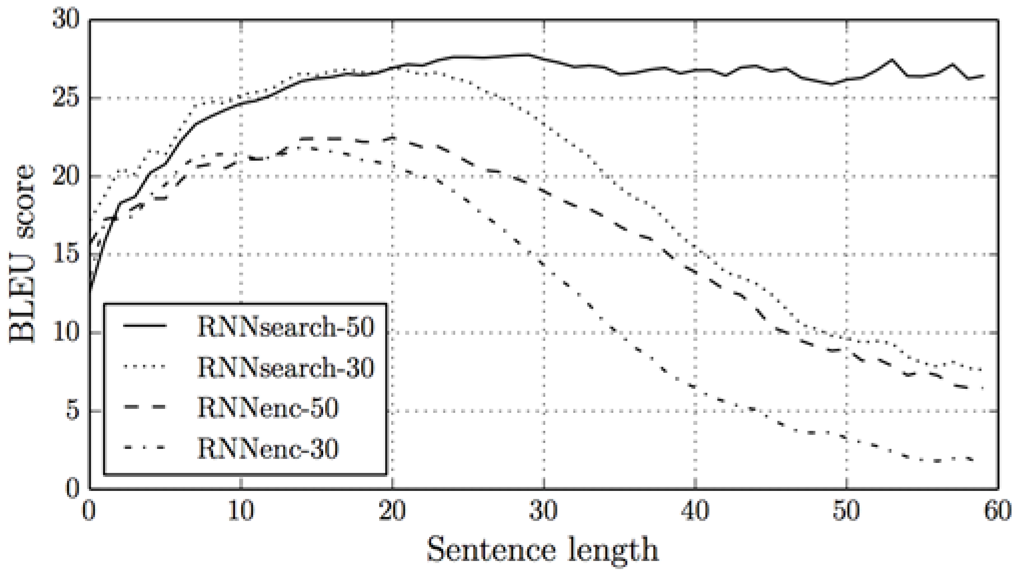
Target: hkhpg

Zaremba, W., Sutskever, I. “Learning To Execute.” *arxiv (2014)*.

Seq2Seq - Limitations

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





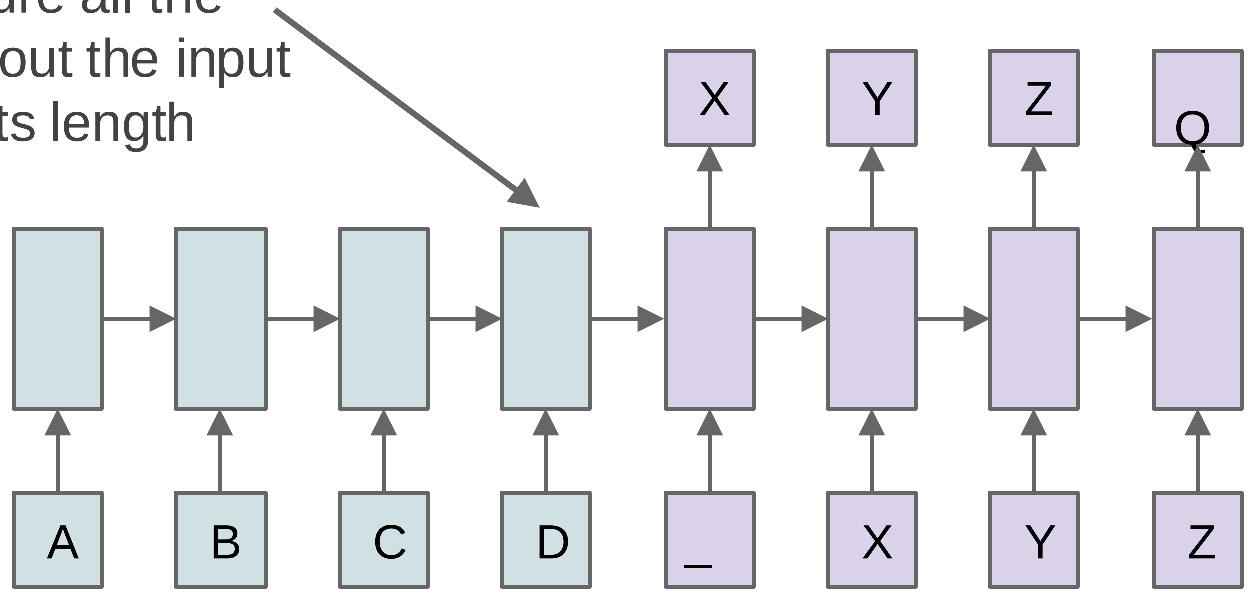
Sutskever, I., et al. “Sequence to Sequence Learning with Neural Networks.” *NIPS (2014)*

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

Attention

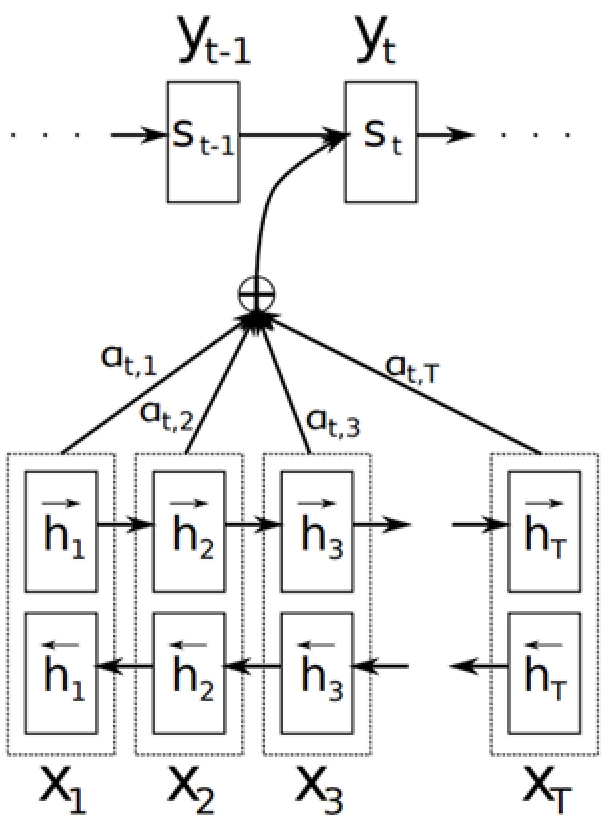
Seq2Seq - The issue with long inputs

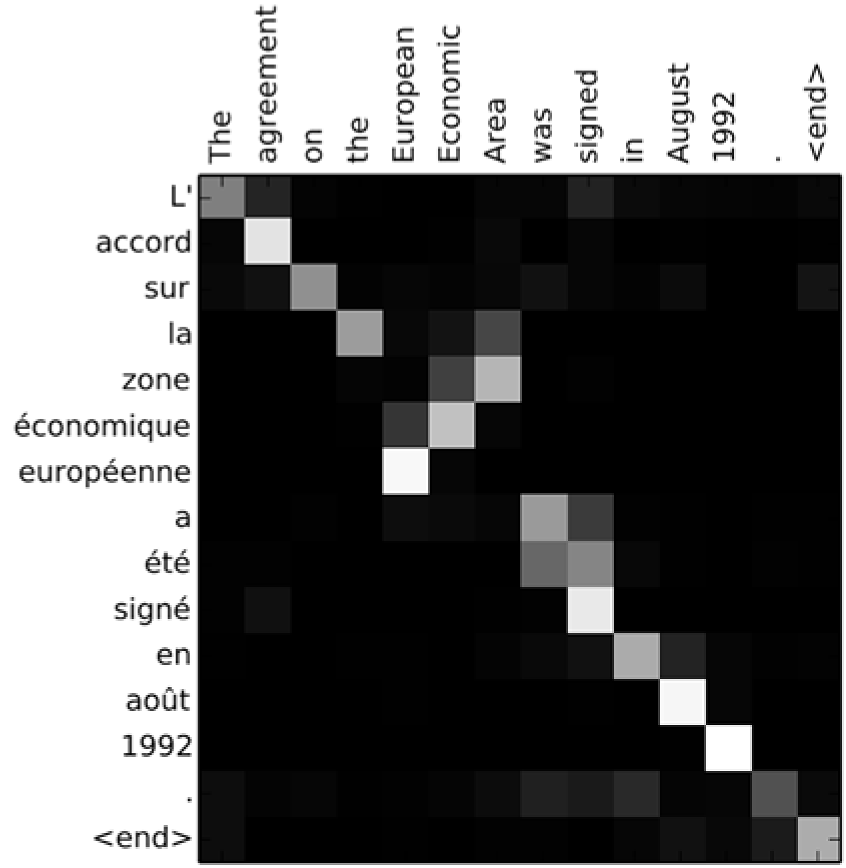
- Same embedding informs the entire output
- Needs to capture all the information about the input regardless of its length



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

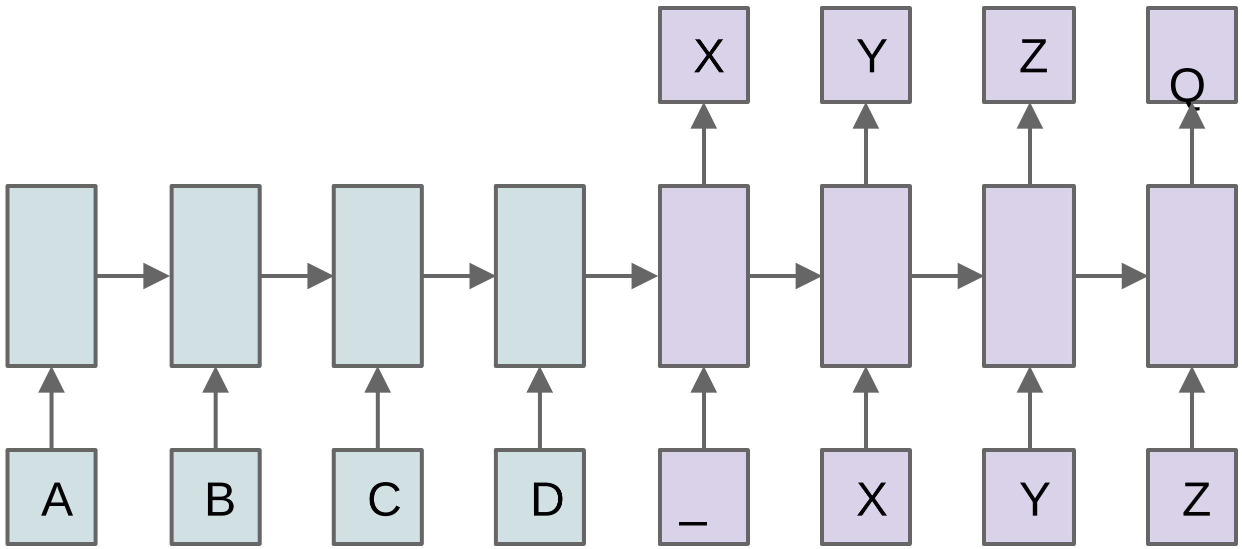
Seq2Seq with Attention





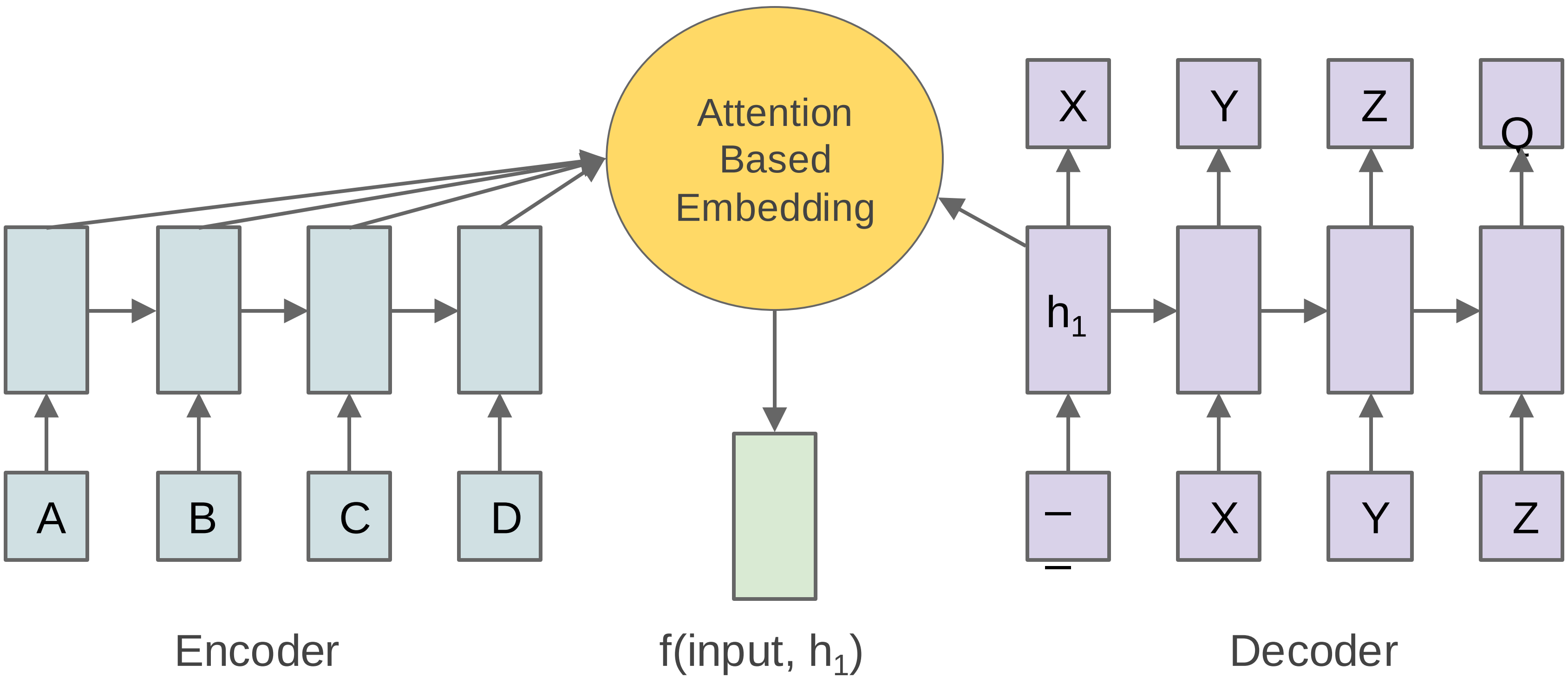
Bahdanau, D., et al. "Neural Machine Translation by Jointly Learning to Align and Translate." *ICLR (2015)*

Seq2Seq



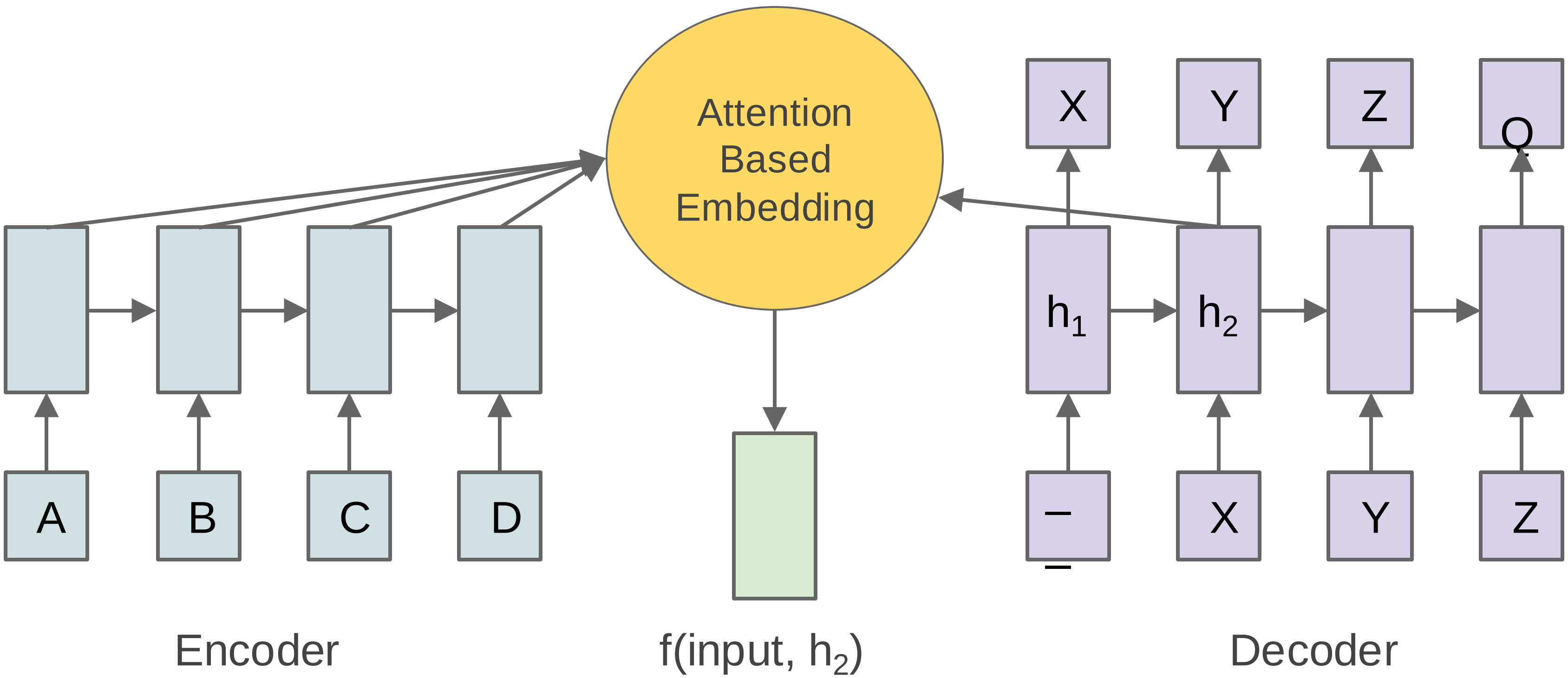
Seq2Seq with Attention

- A different embedding computed for every output step



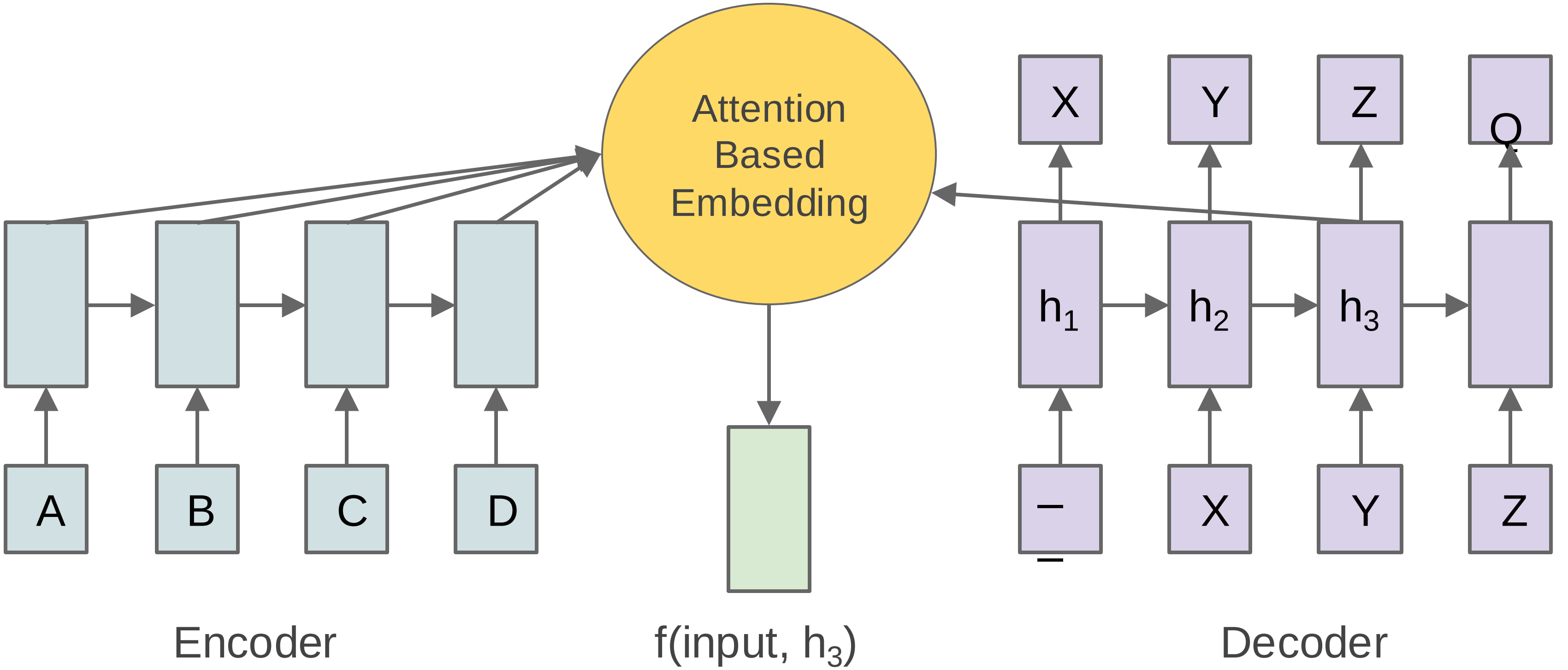
Seq2Seq with Attention

- A different embedding computed for every output step



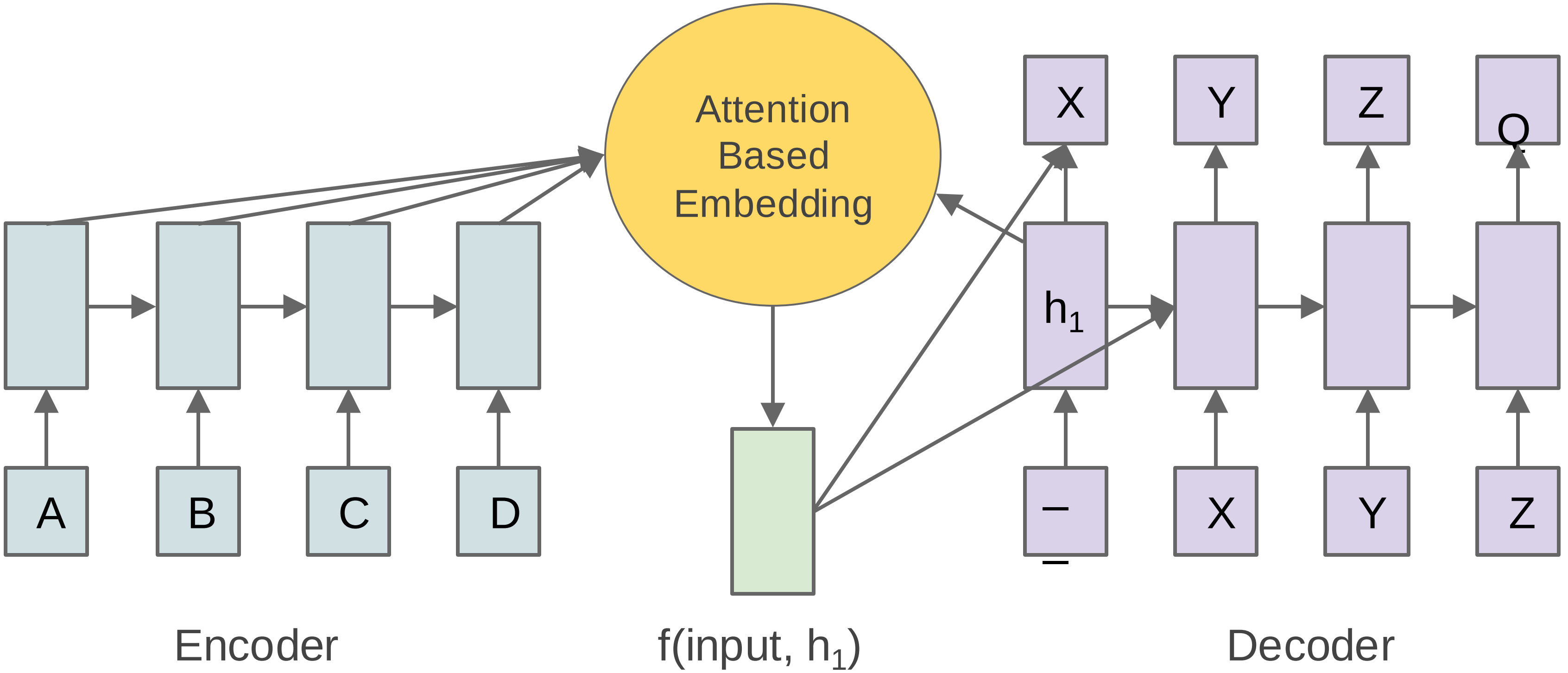
Seq2Seq with Attention

- A different embedding computed for every output step



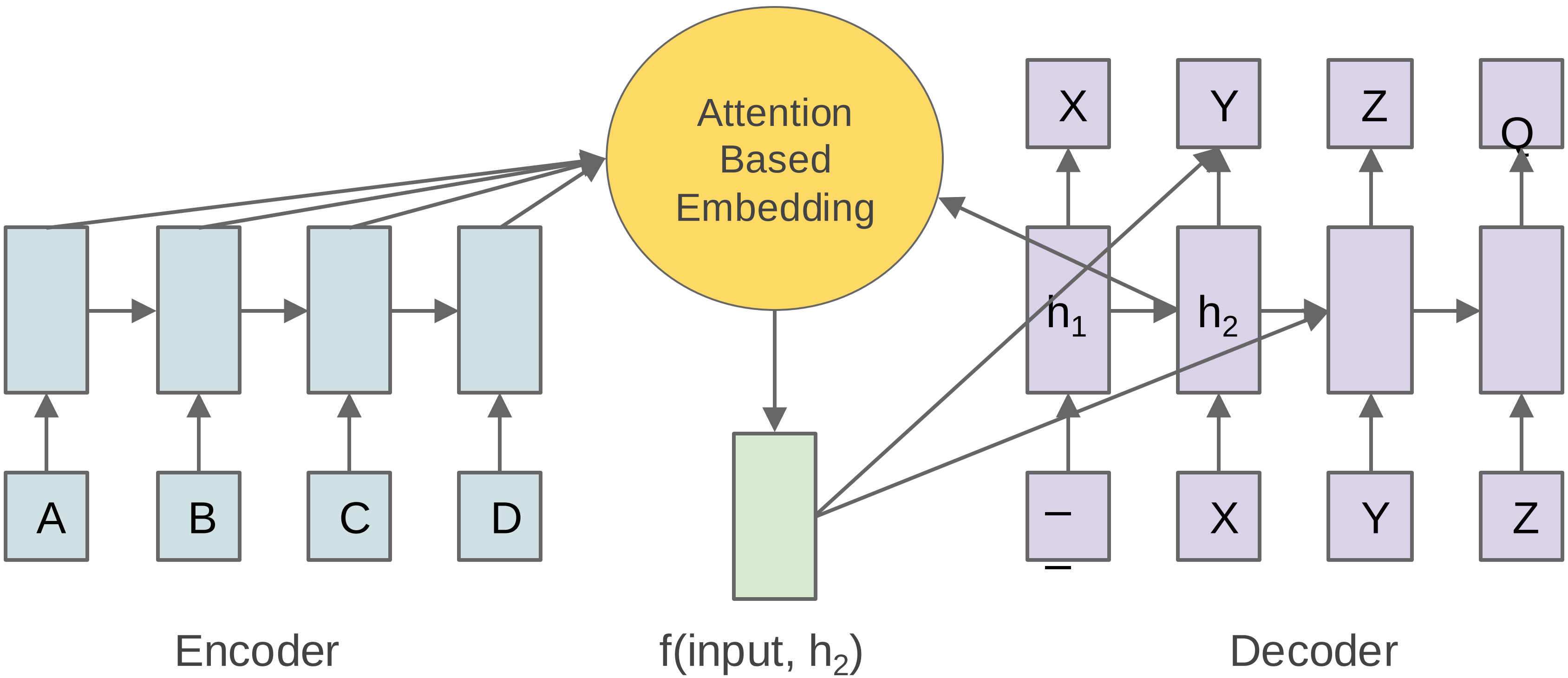
Seq2Seq with Attention

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



Seq2Seq with Attention

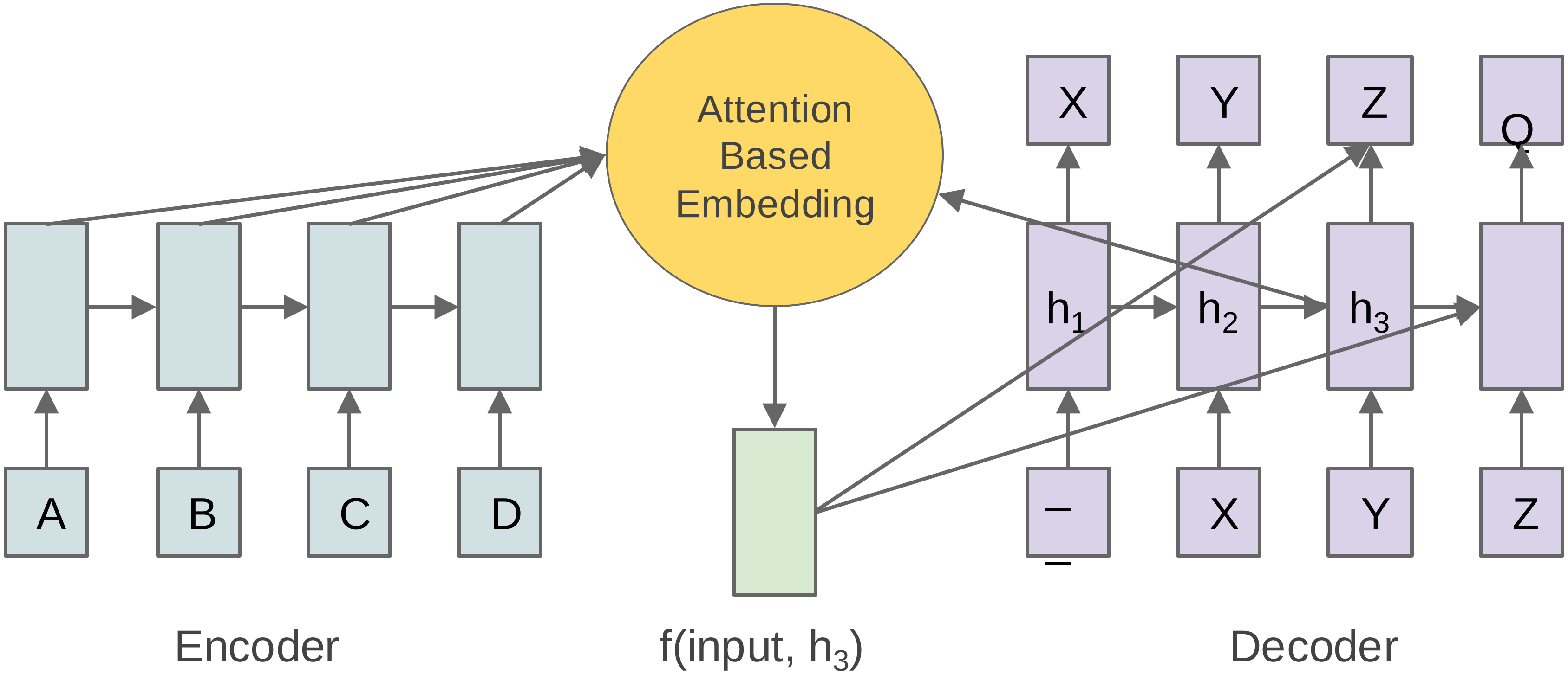
- Embedding used to predict output, and compute next hidden state



- Attention arrows for step 1 omitted

Seq2Seq with Attention

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



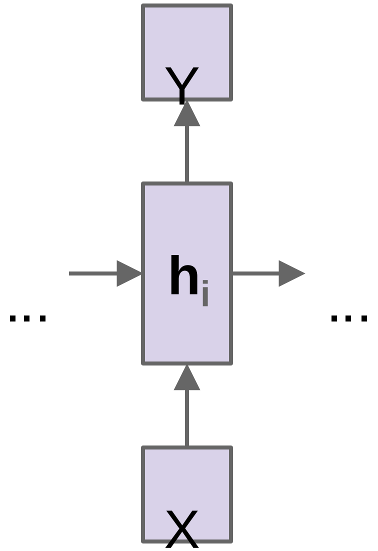
- Attention arrows for steps 1 and 2 omitted

Attention Based Embedding

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

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 $\mathbf{h_i^T e_1, h_i^T e_2, h_i^T e_3, h_i^T e_4}$ representing similarity / relevance between encoder steps and query.
- Normalize $\mathbf{[h_i^T e_1, h_i^T e_2, h_i^T e_3, h_i^T e_4\,]}$ 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]

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

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

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

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