RNN sequence models and attention

Lecture 13, 10/26/17

David Sontag

6.867, Machine Learning

Massachusetts
Institute of
Technology

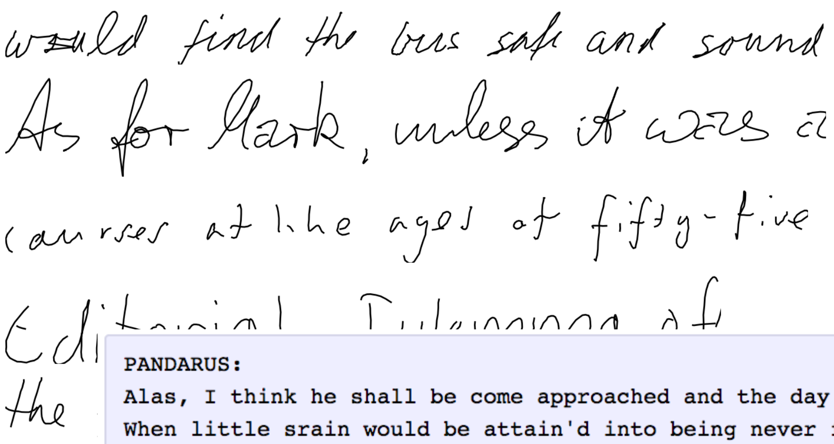
Course announcements

- Project Milestone 3 due tonight
- HW3 released this week, due 11/14
- References for today's class
  - *Deep Learning* (Sections 10 and 12), Goodfellow, Bengio, Courville. MIT Press, 2016
  - *Attention and Augmented Recurrent Neural Networks*, Chris Olah and Shan Carter. Distill 2016
  - **Available for free online**

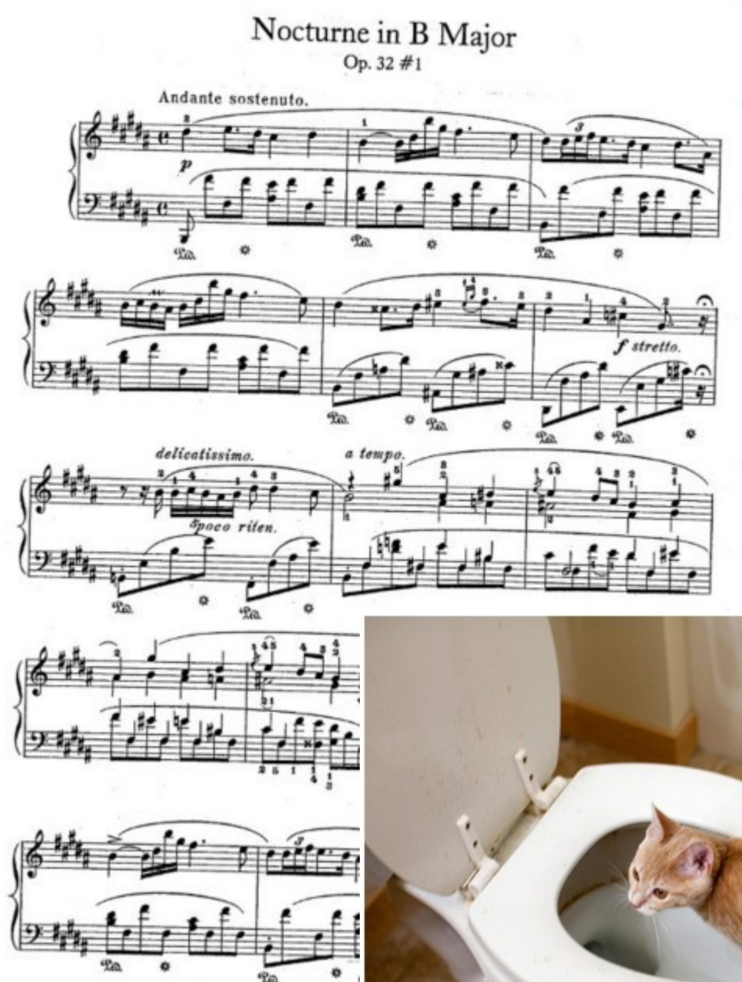
Today's lecture: Outline

- Using RNNs for modeling sequences
  - Modeling outputs
  - Modeling inputs (e.g., word embeddings)
  - Inference
- Attention mechanisms
  - Within sequences (e.g., for machine translation)
  - Within images (e.g., visual attention)
  - One-shot learning
  - Memory networks

Using RNNs for modeling sequences




```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```



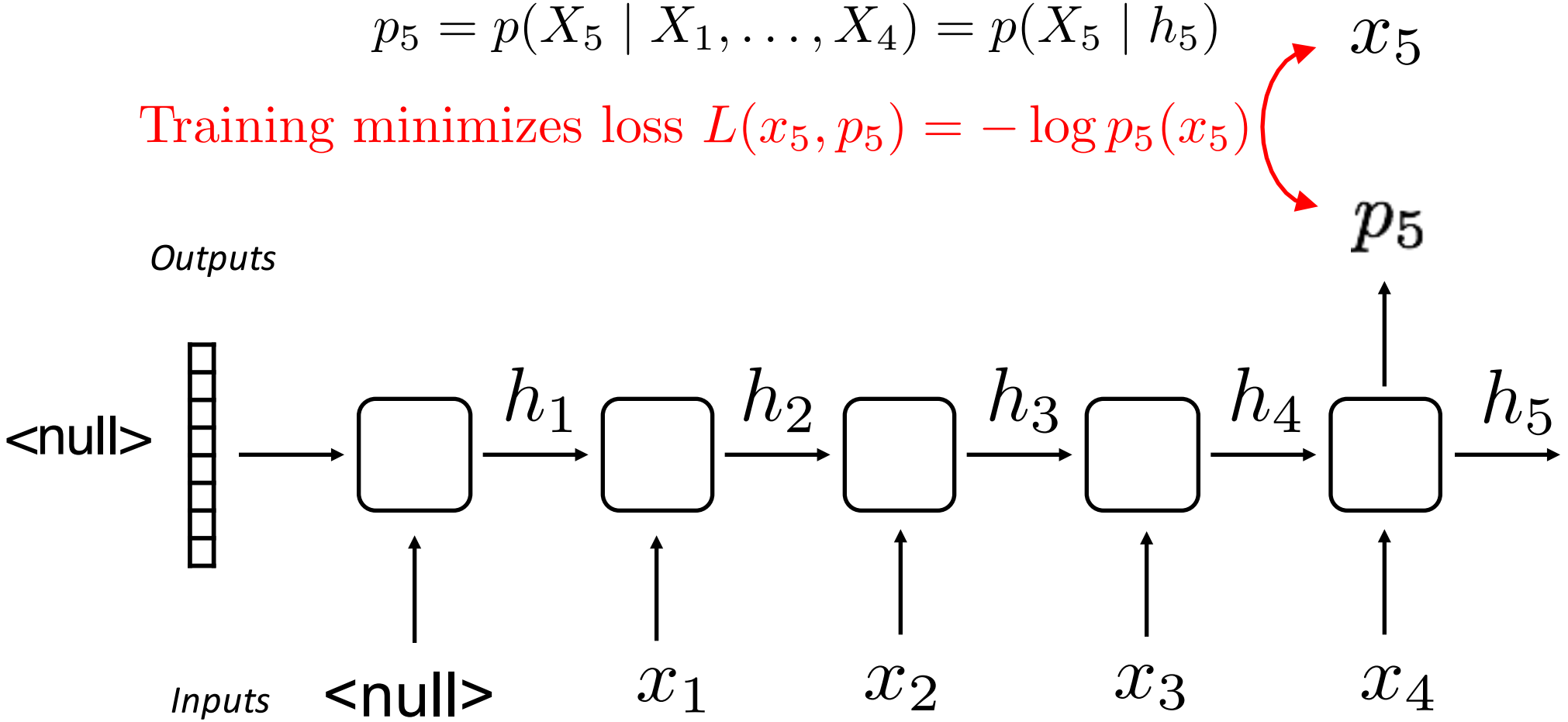
a cat is sitting on a toilet seat
logprob: -7.79

Using RNNs ~~for modeling sequences~~ (training)

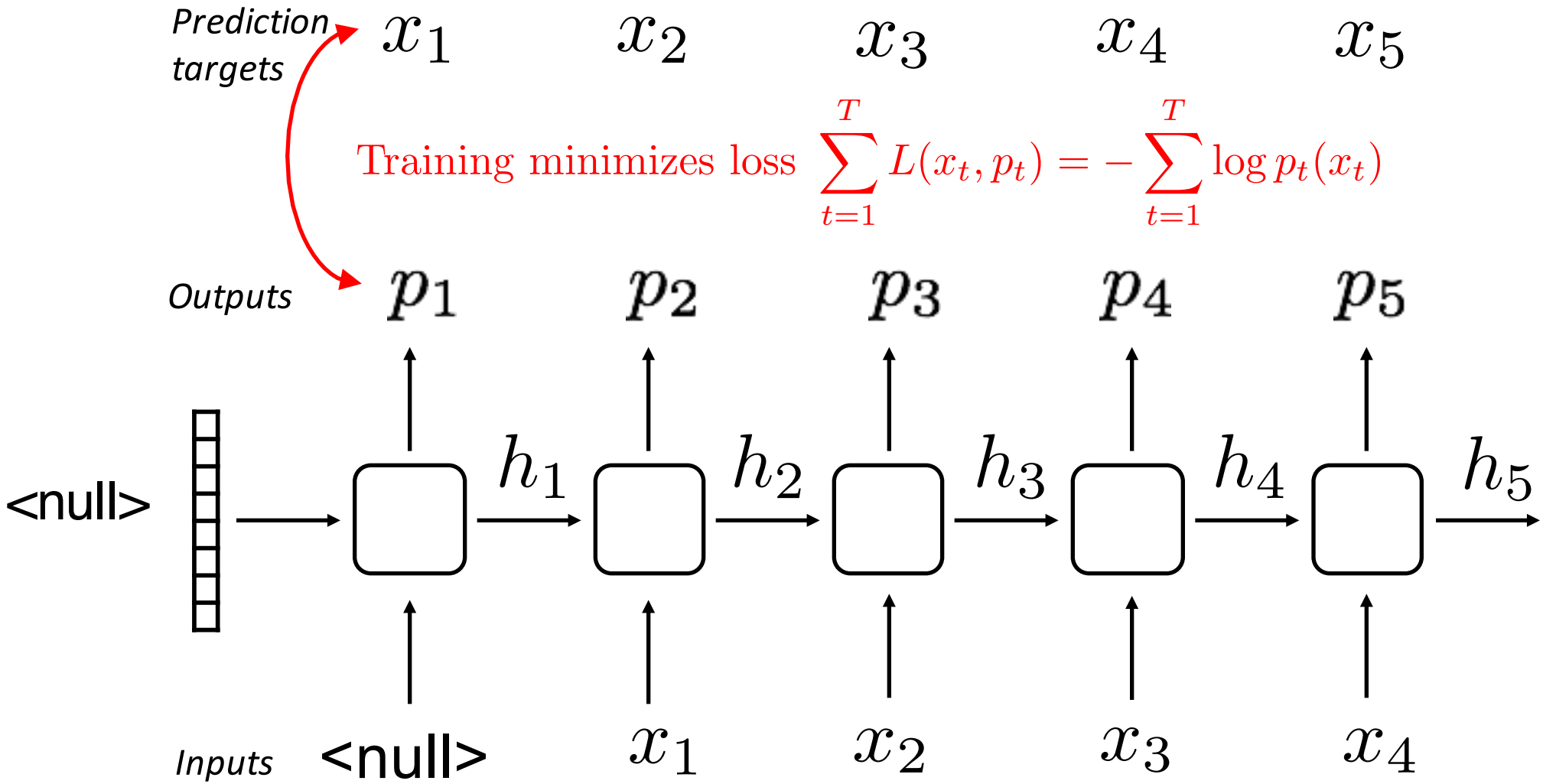
Given inputs x_1, … x_4, make a prediction using the hidden state at the last time step,

$$p_5 = p(X_5 \mid X_1, \ldots, X_4) = p(X_5 \mid h_5)$$

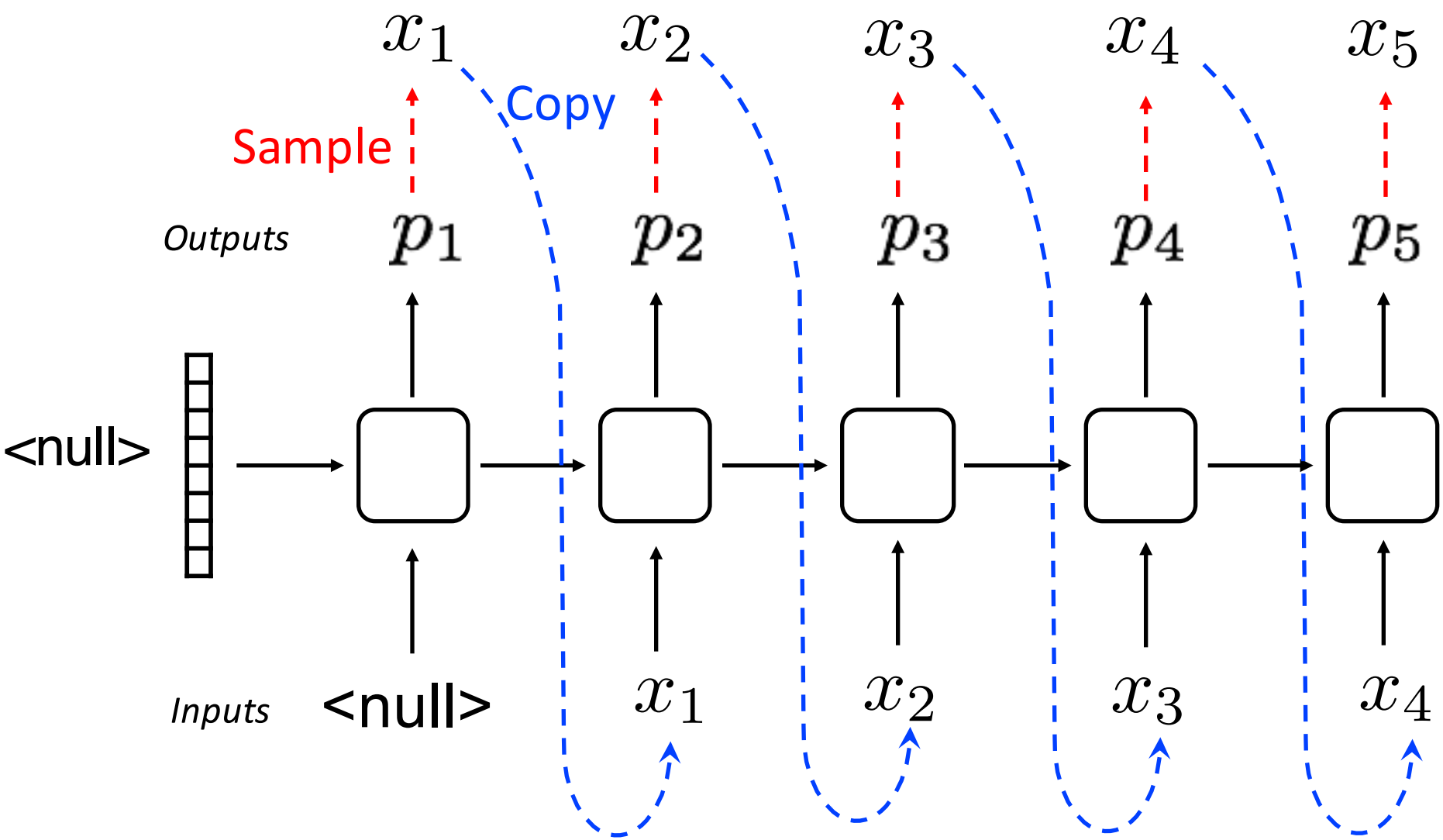
Training minimizes loss $L(x_5, p_5) = -\log p_5(x_5)$



Using RNNs for modeling sequences (training)



Using RNNs for modeling sequences (test)



Modeling outputs

- **Language modeling**: At each time step, output distribution of support |V| (vocabulary size), which could be in the millions

$$p(x_t = i \mid h_t) = \frac{\exp(h_t \cdot w_i)}{\sum_{j \in V} \exp(h_t \cdot w_j)}$$

- Computing gradient of *log* $p(x_t|h_t)$ takes running time |V| because of the denominator: extremely expensive
- One solution is to instead use a *hierarchical softmax*:

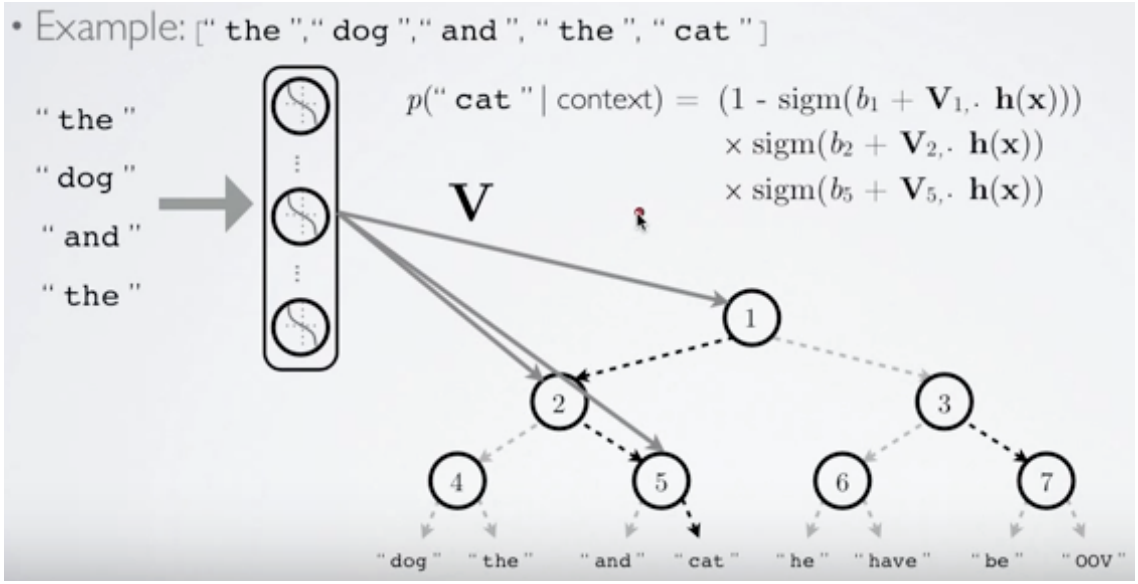
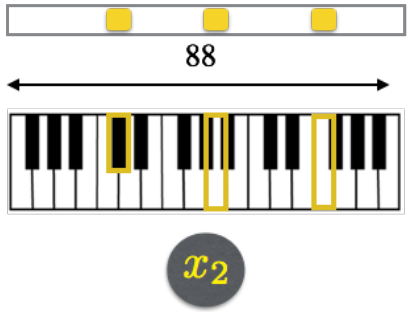


Figure: https://www.youtube.com/watch?v=B95LTf2rVWM

Modeling outputs

- **Polyphonic music**: At each time step, output distribution over a 88-dimensional binary vector, $x_{t,1}, x_{t,2}, \ldots, x_{t,88}$, conditioned on hidden state h_t



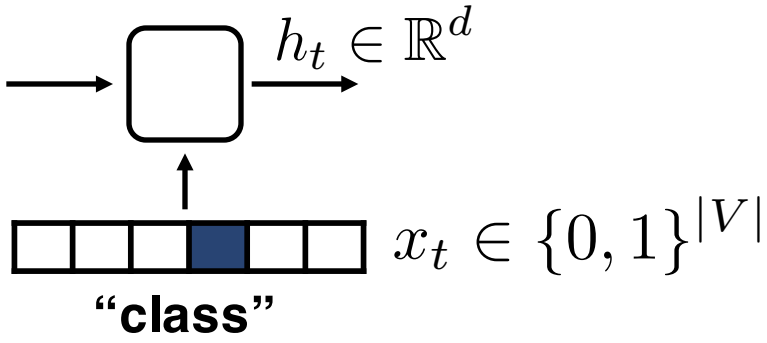
- Can't treat as multi-class and use softmax, as x_t has 2^{88} states!
- Common approach is to model as *independent* outputs, i.e.

$$\Pr(x_t \mid h_t) = \prod_{k=1}^{88} \Pr(x_{t,k} \mid h_t)$$

  using e.g. a logistic function for each (if continuous, Gaussian)
- A more powerful approach would be to use a NADE, or neural autoregressive density estimator (Uria, et al. ICML '14)

Modeling inputs: word embeddings

- Recall that we originally suggested representing words using a one-hot encoding:



- Also recall the equations defining the LSTM unit:

$$f_t = \text{sigmoid}(W^{f,h} h_{t-1} + W^{f,x} x_t)$$
$$i_t = \text{sigmoid}(W^{i,h} h_{t-1} + W^{i,x} x_t)$$
$$o_t = \text{sigmoid}(W^{o,h} h_{t-1} + W^{o,x} x_t)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{c,h} h_{t-1} + W^{c,x} x_t)$$
$$h_t = o_t \odot \tanh(c_t)$$

forget gate

input gate

output gate

memory cell

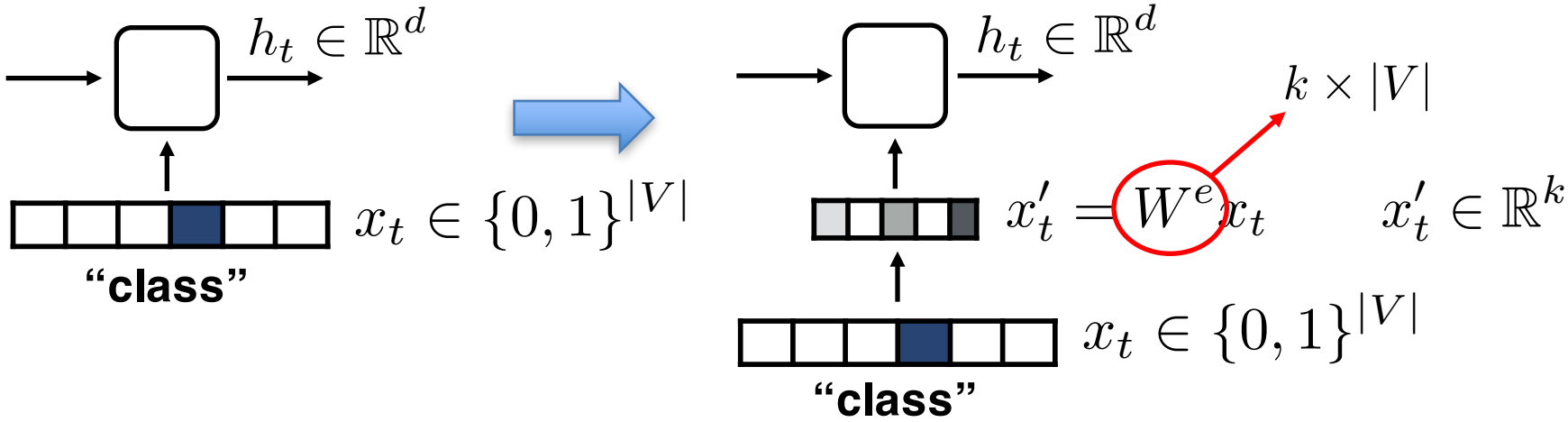
visible state

dimension $d \times |V|$

- **Challenge:** how do we make *d* large, yet not overfit with rare words?

Modeling inputs: word embeddings

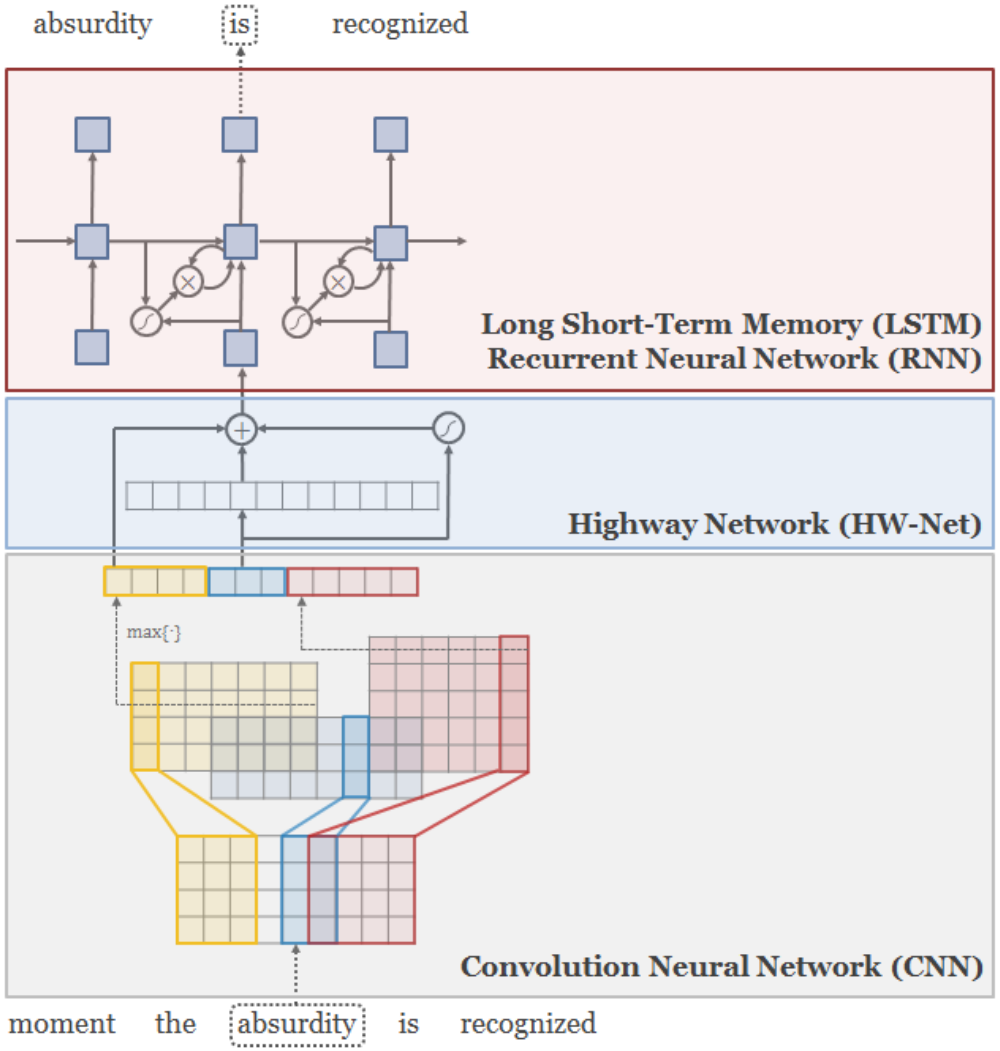
- Instead, do *linear transformation* of words prior to feeding to RNN



- Each column of W^e can be thought of as a *word embedding*, which can be trained end-to-end
- These parameters are now shared across all gates of the LSTM
- It is very common to use *pre-trained* word embeddings, coming from learning a language model (or a simpler objective, such as word2vec), on a much larger dataset

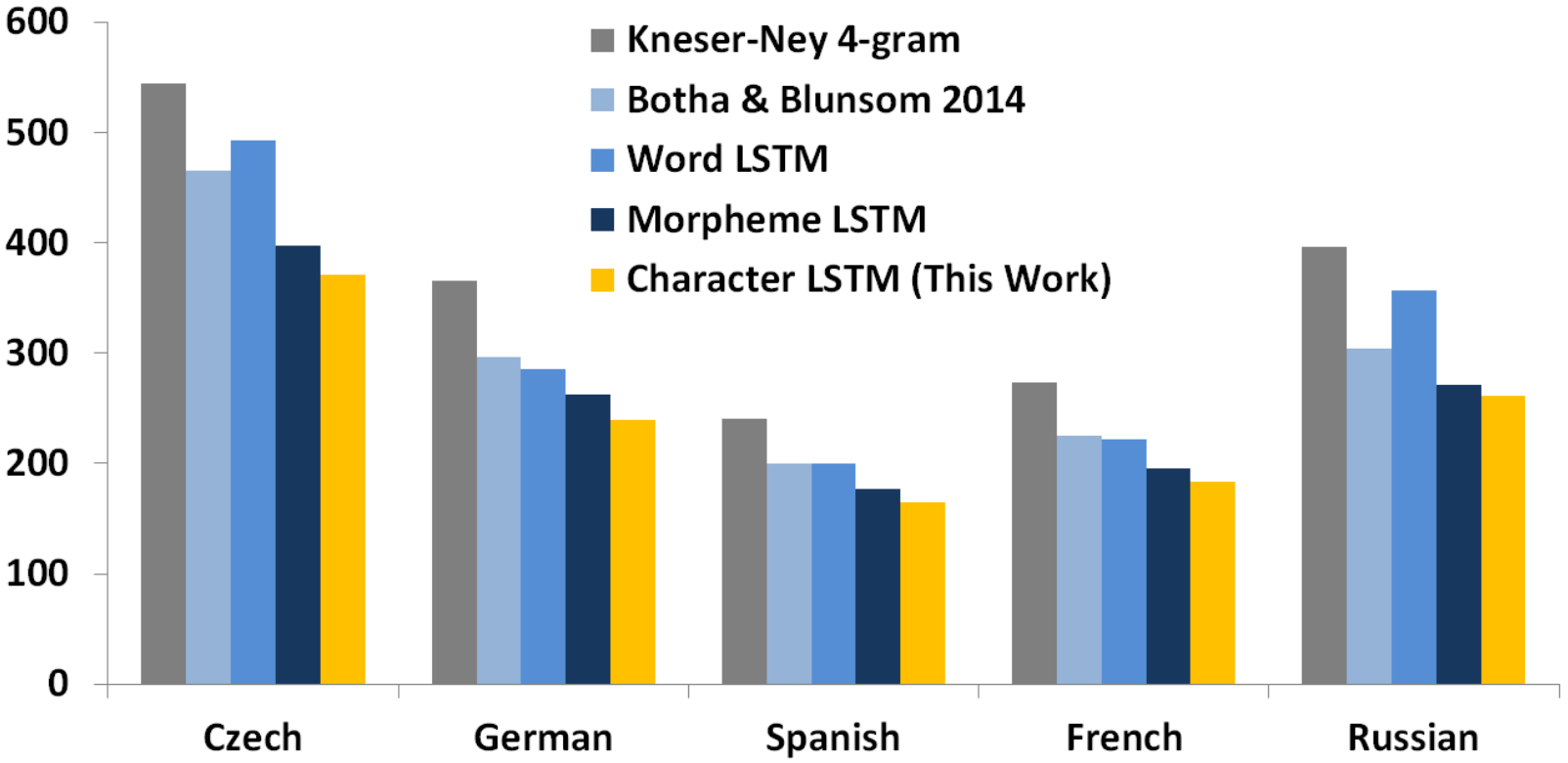
Modeling inputs: word embeddings

- Kim et al. AAAI '16 suggest an alternative – use a convolutional neural network to ***construct*** word embedding directly from the words' characters (CharCNN)

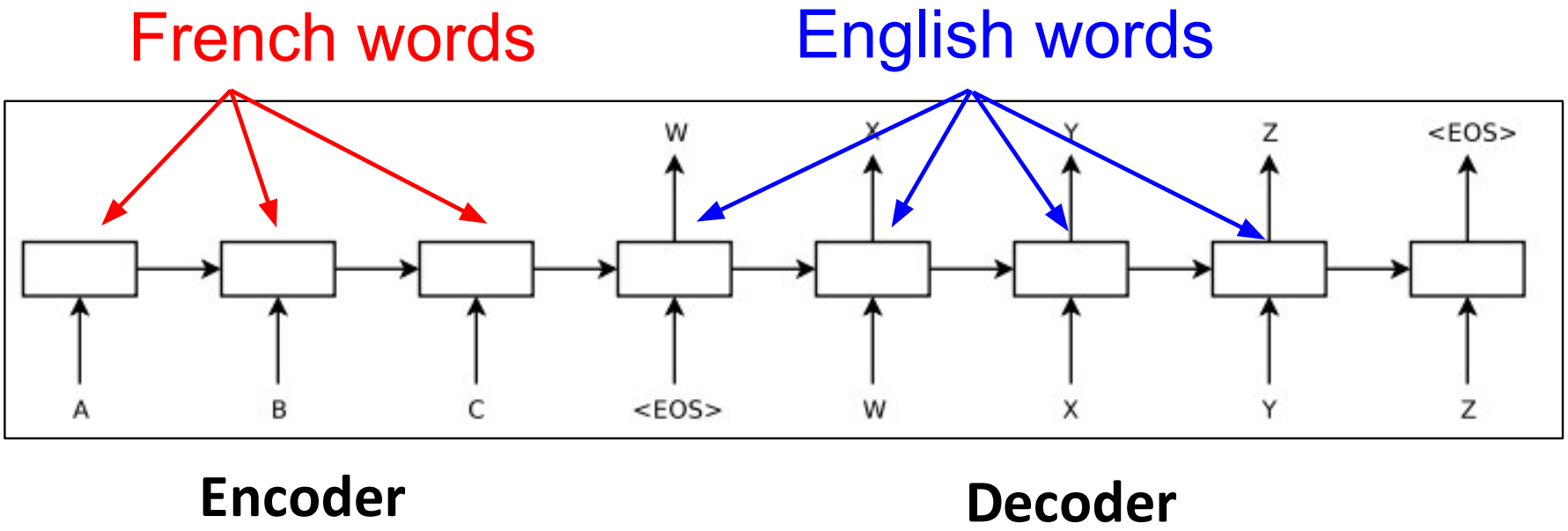


Modeling inputs: word embeddings

- Kim et al. AAAI '16 suggest an alternative – use a convolutional neural network to ***construct*** word embedding directly from the words' characters (CharCNN)
- Gives better results for morphologically rich languages with rare words
- Uses fewer parameters!



Basic seq2seq model for machine translation



Probabilistic inference

- We have already seen that (forward) sampling from these sequence models is easy
- However, for tasks such as sentence completion and machine translation, we want to find the *most likely sequence*:

$$\max_{x_{\text{english}}} p(x_{\text{english}} \mid x_{\text{french}})$$

- This is called "MAP inference", for *maximum a posteriori*
- Doing this exactly is NP-hard for RNN sequence models
- Many heuristics exist, such as beam search

Attention

- How can we improve complex encoder-decoder RNN architectures?
- One key insight is that summarizing *all* of the source information into a vector may be asking for too much
- For example:
  - Translations between e.g. English and German are pretty well aligned
  - Had we translated word by word, we would have been close!
  - Instead, the simple sequence-to-sequence (seq2seq) model has to memorize the full word order to translate properly

Attention

- Can we design an architecture that, at least informally, incorporates the idea of alignment?

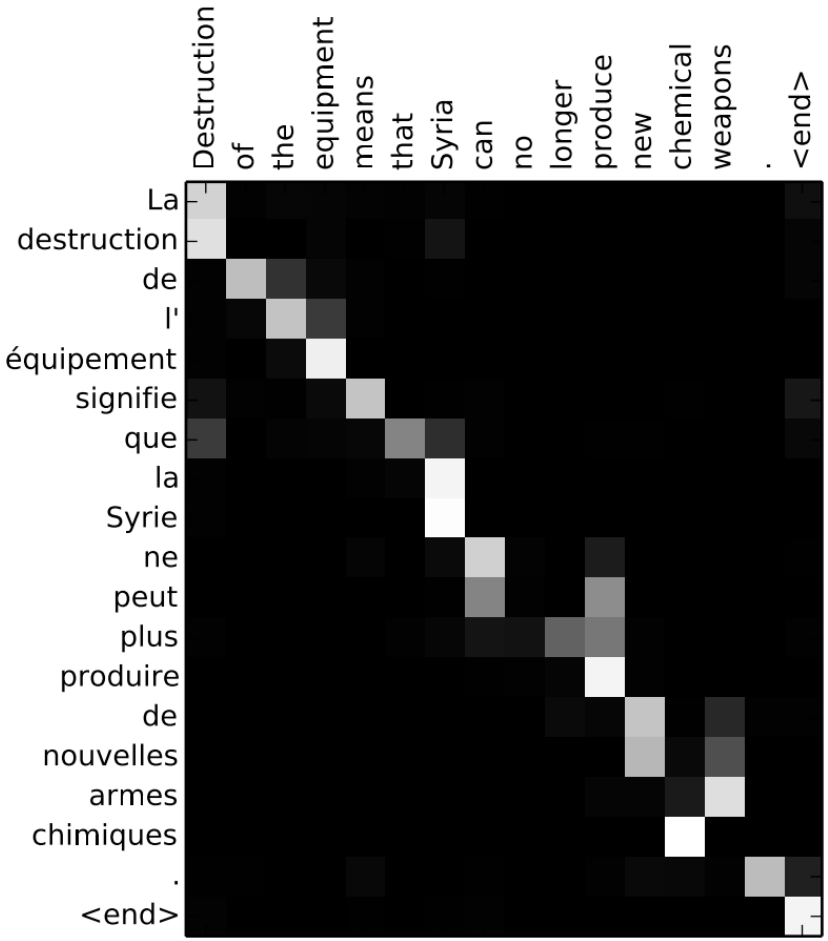
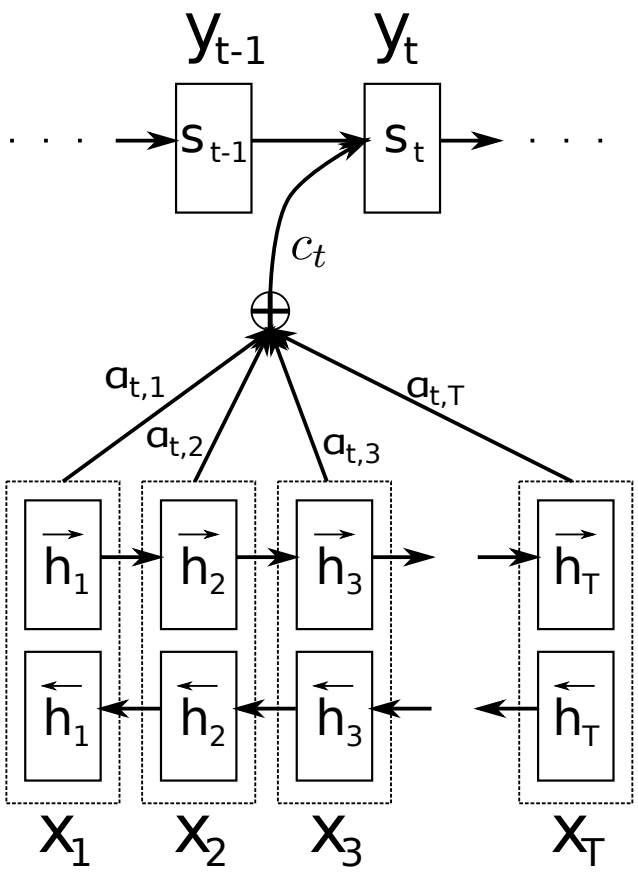


Figure: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

Learning to align and translate

- Bahdanau et al. ICLR '15 suggest precisely such a method:



Attention mechanism:

$$\alpha_{t,j} = \frac{\exp(a(s_{t-1}, h_j))}{\sum_{k=1}^{T_x} \exp(a(s_{t-1}, h_k))}$$

Then, feed this into the decoder:

$$c_t = \sum_{j=1}^{T_x} \alpha_{t,j} [\overleftarrow{h}_j, \overrightarrow{h}_j]$$

Bi-directional RNN encoder

- Note that this significantly increases *computation time*

Attention within images

- The same idea can be applied to improve image caption generation by using *visual attention* (Xu et al., arXiv:1502.03044, '16)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

One-shot learning

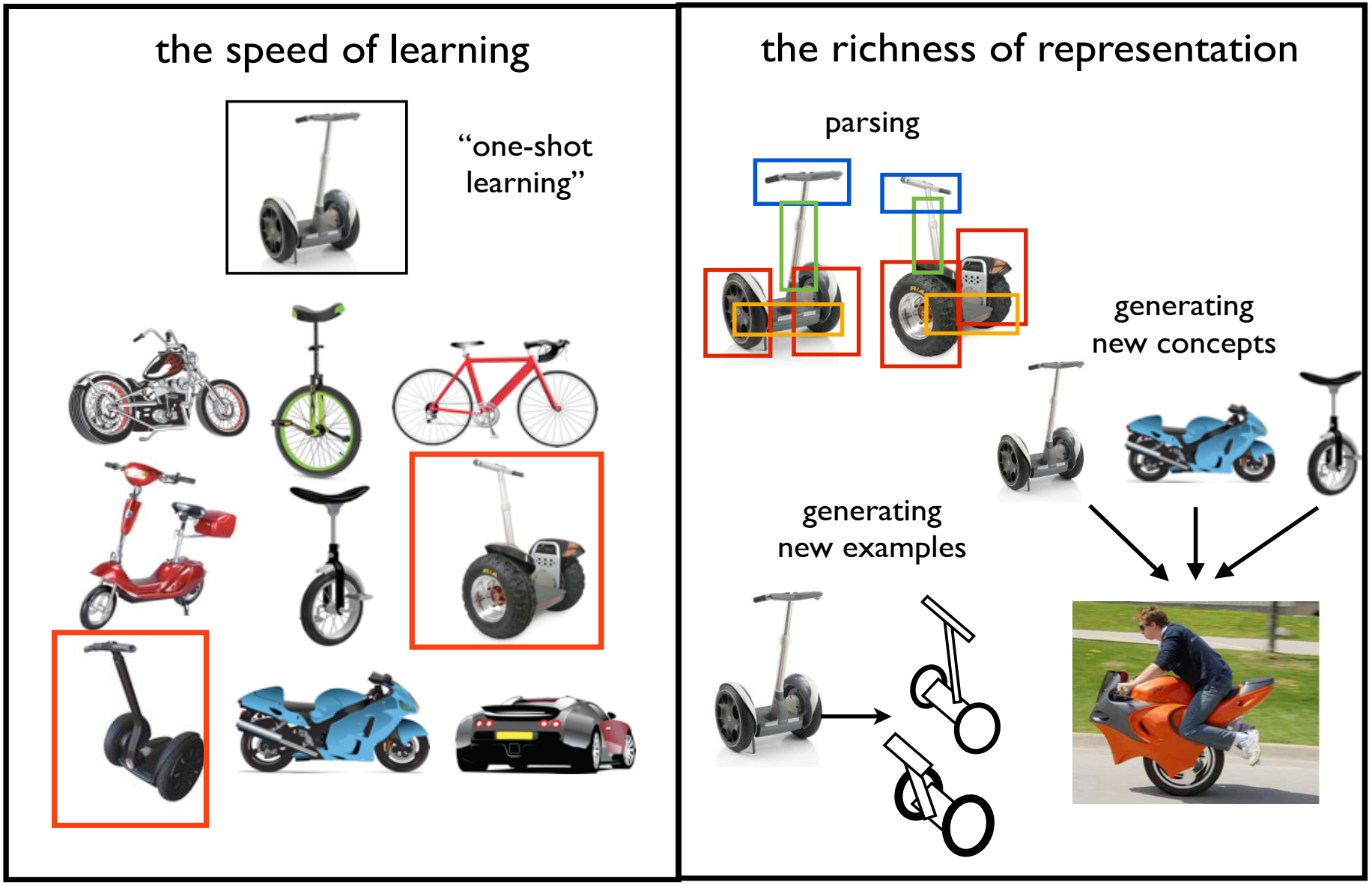


Figure acknowledgements: Brenden Lake

One-shot learning: Omniglot dataset

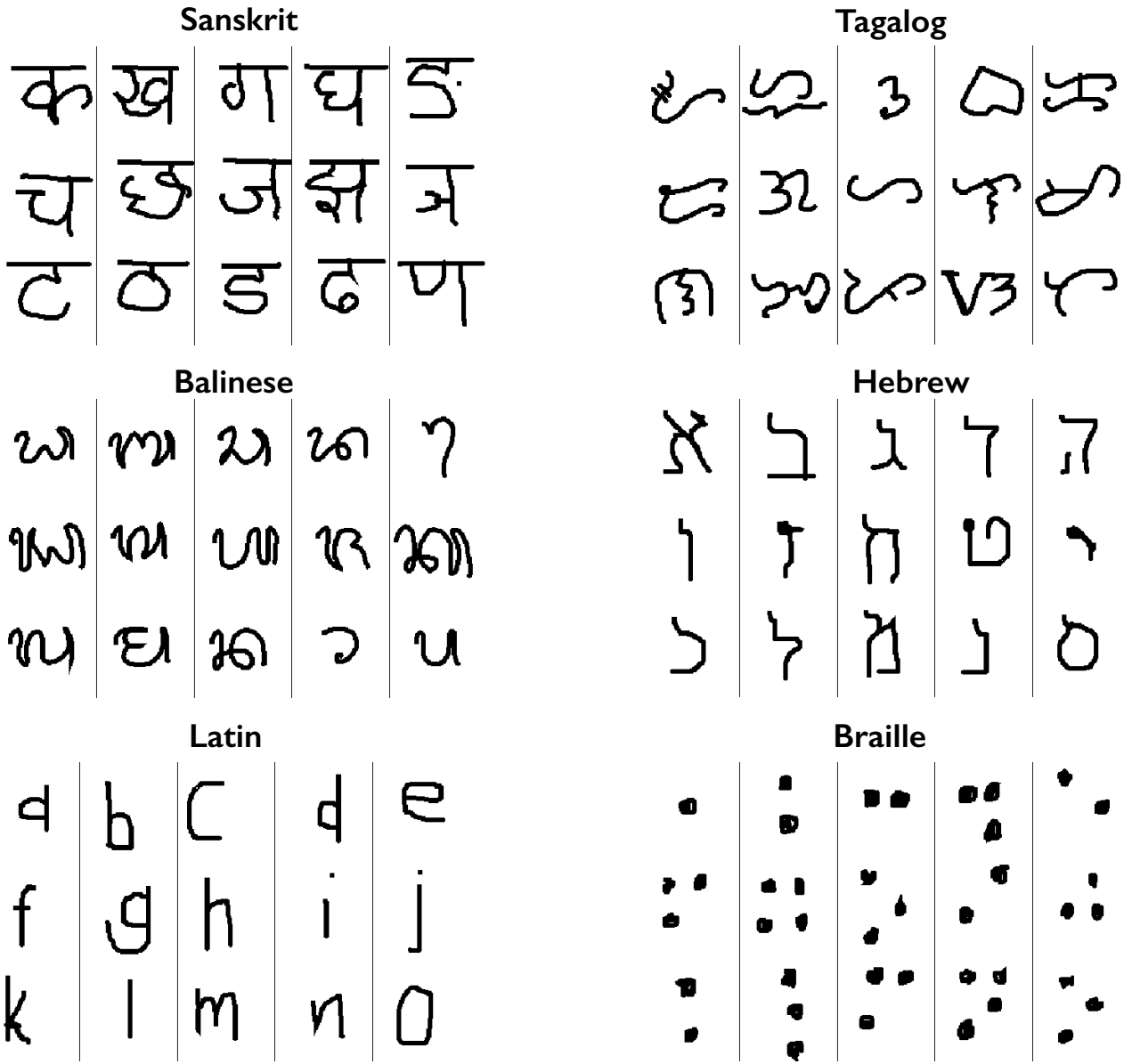


Figure acknowledgements: Brenden Lake

One-shot learning using attention

- **Matching networks**: non-parametric model which predicts using attention to the training dataset:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

- Attention mechanism (*c* is cosine distance) given by

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$

- One batch of stochastic training chooses a subset of labels examples from each label. Learning maximizes:

$$\arg\max_{\theta} E_{L \sim T} \left[E_{S \sim L, B \sim L} \left[\sum_{(x,y) \in B} \log P_{\theta}(y|x, S) \right] \right]$$

[Vinyals et al., 2016]

One-shot learning using attention

- Matching network architecture:

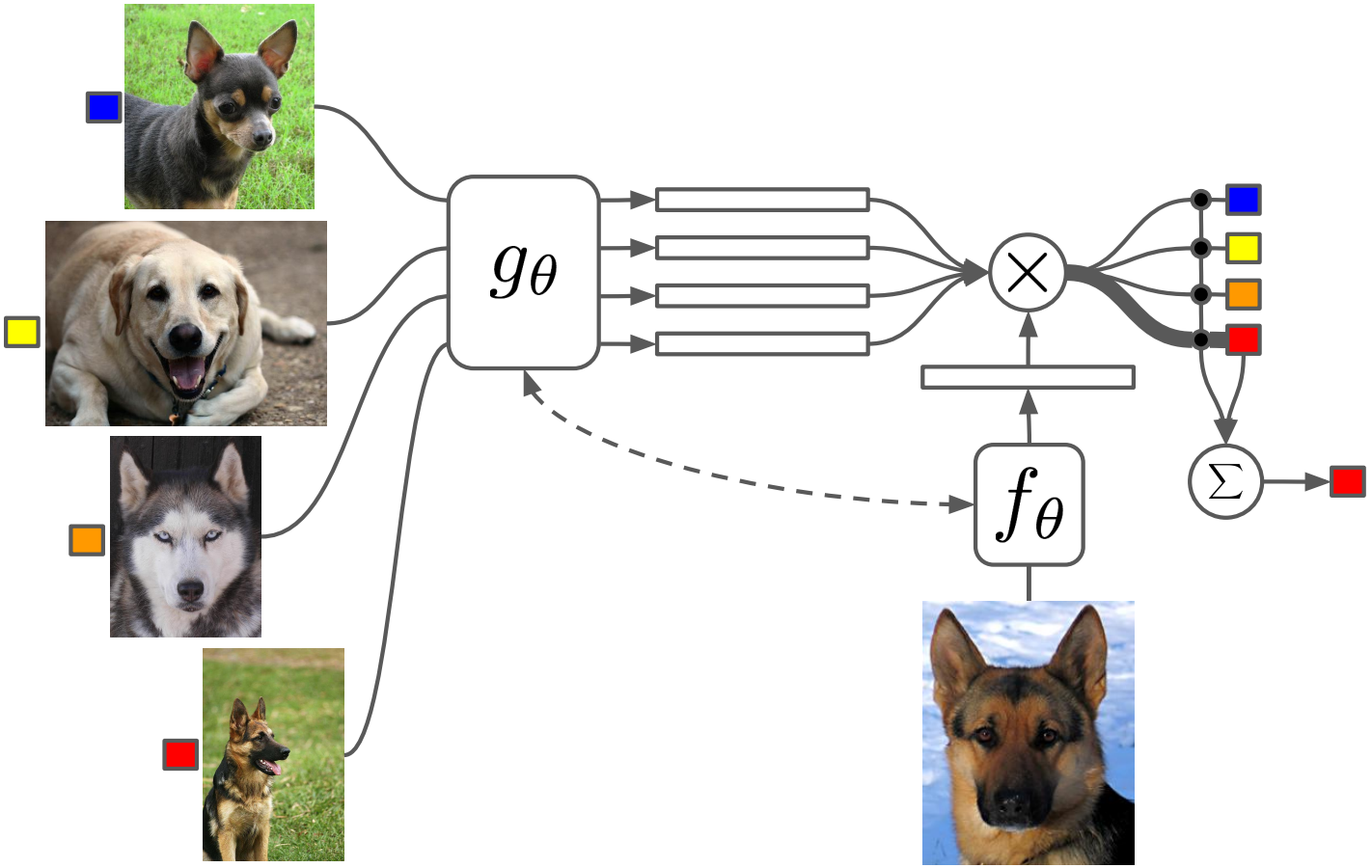


Figure 1: Matching Networks architecture

[Vinyals et al., 2016]

One-shot learning using attention

- Matching network results on Omniglot dataset:

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	–	–
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	**98.1%**	**98.9%**	**93.8%**	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	**98.7%**

[Vinyals et al., 2016]

Neural architectures with memory

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk. Joe travelled to his office. Joe left the milk. Joe went to the bathroom.

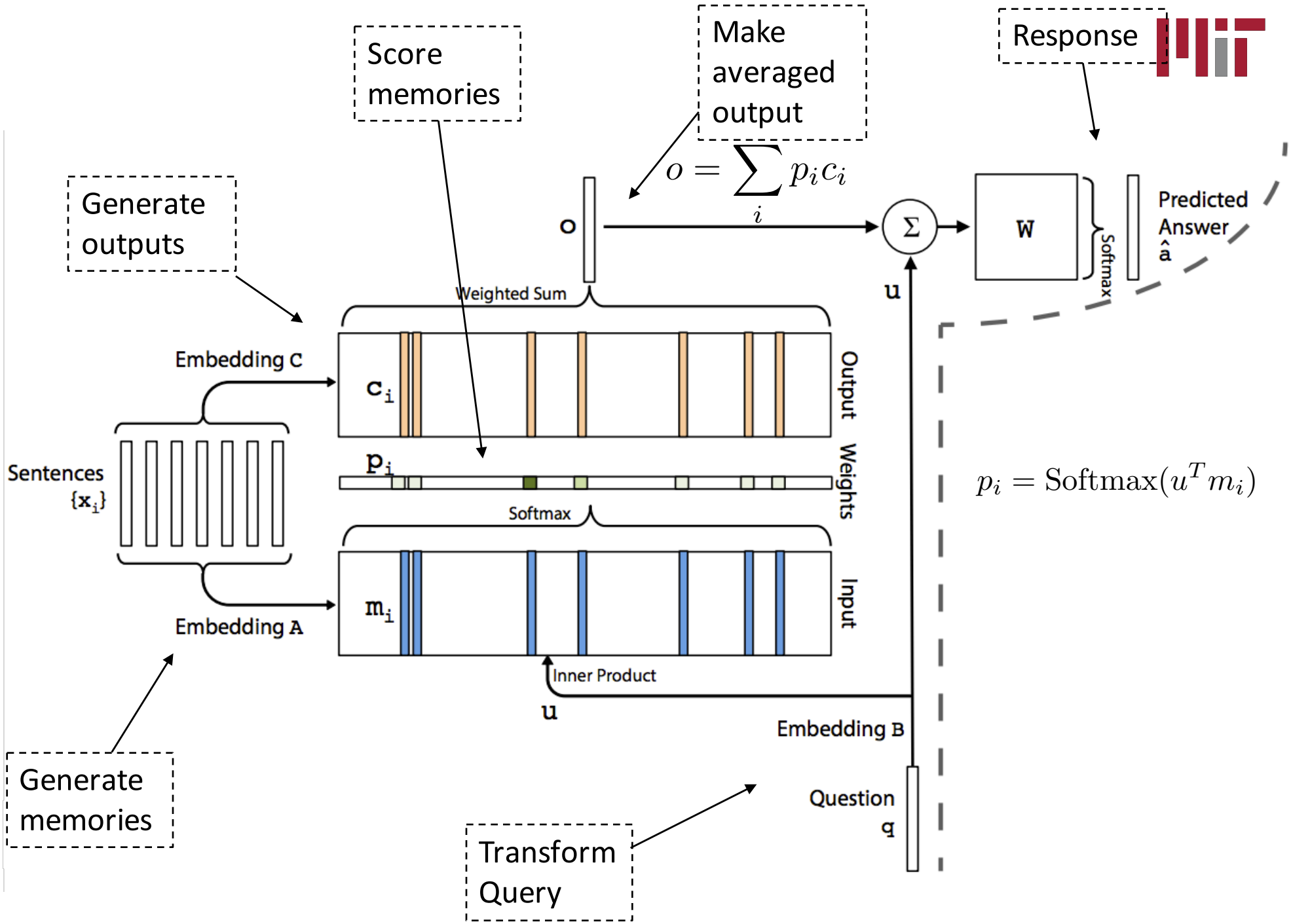
Questions from Joe's angry mother:

Q1 : Where is Joe?

Q2 : Where is the milk now?

Q3 : Where was Joe before the office?

Slide acknowledgement: Nitish Gupta, Shreya Rajpal



End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015