TTIC 31210:
Advanced Natural Language Processing

Kevin Gimpel

Spring 2019

Lecture 6:
Finish Transformers;
Sequence-to-Sequence Modeling and Attention

Roadmap

- intro (1 lecture)
- **deep learning for NLP (5 lectures)**
- structured prediction: sequence labeling, syntactic and semantic parsing, dynamic programming (4 lectures)
- generative models, latent variables, unsupervised learning, variational autoencoders (2 lectures)
- Bayesian methods in NLP (2 lectures)
- Bayesian nonparametrics in NLP (2 lectures)
- review & other topics (1 lecture)

Assignments

- Any last-minute questions about Assignment 1?

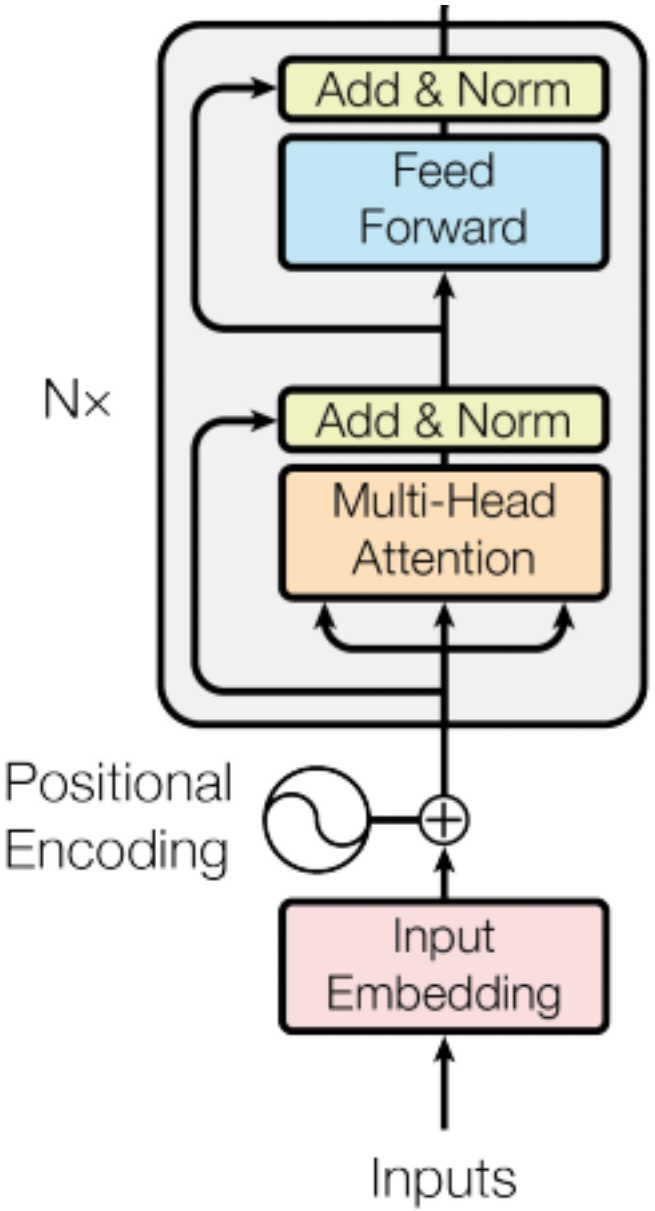
- Assignment 2 will be posted tonight, due in 2 weeks

Today

- finish transformers
- sequence-to-sequence modeling and attention

Transformer

- effective encoder for text sequences (and other data)
- no recurrent/convolutional modules
- only attention (various forms)



Vaswani et al. (2017): *Attention Is All You Need*

Attention

- attention is a useful generic tool
- often used to replace a sum or average with an attention-weighted sum

Attention

- e.g., for a word averaging encoder:

$$\mathbf{f}_{\text{avg}}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} emb(x_i)$$

$$\mathbf{f}_{\text{att}}(\boldsymbol{x}) = \sum_{i=1}^{n} att(x_i, i, \boldsymbol{x})\, emb(x_i)$$

"attention" function, returns a scalar

Example Attention Function

$$\mathbf{f}_{\text{att}}(\boldsymbol{x}) = \sum_{i=1}^{n} att(x_i, i, \boldsymbol{x}) emb(x_i)$$

$$att(x_i, i, \boldsymbol{x}) \propto \exp\{\mathbf{w}^\top emb(x_i)\}$$

- introduces a new parameter vector **w** which is learned along with the word embeddings
- attention is normalized over the sentence length

Queries, Keys, and Values

- we can often think of attention functions in terms of these abstractions
- query = what you use to search
- key = the field that you're comparing to
- value = the field that you return

Vaswani et al. (2017): *Attention Is All You Need*

- considering attention as query/key/value suggests using different spaces for different roles
- e.g., we could use separate transformations of the embedding space for keys and values:

$$\mathbf{f}_{\text{att}}(\boldsymbol{x}) = \sum_{i=1}^{n} att(x_i, i, \boldsymbol{x}) \left(\mathbf{W}^{(v)} emb(x_i) \right)$$

value transformation matrix

$$att(x_i, i, \boldsymbol{x}) \propto \exp \left\{ \mathbf{w}^\top \left(\mathbf{W}^{(k)} emb(x_i) \right) \right\}$$

key transformation matrix

Multi-Head Attention

- we may want to learn multiple attention functions in parallel
- why? so that they can learn complementary functionality for the task

Vaswani et al. (2017): *Attention Is All You Need*

Multi-Head Attention

- in the transformer, each attention head uses projections to lower dimension, followed by concatenation of the outputs from each head

Vaswani et al. (2017): *Attention Is All You Need*

Transformer

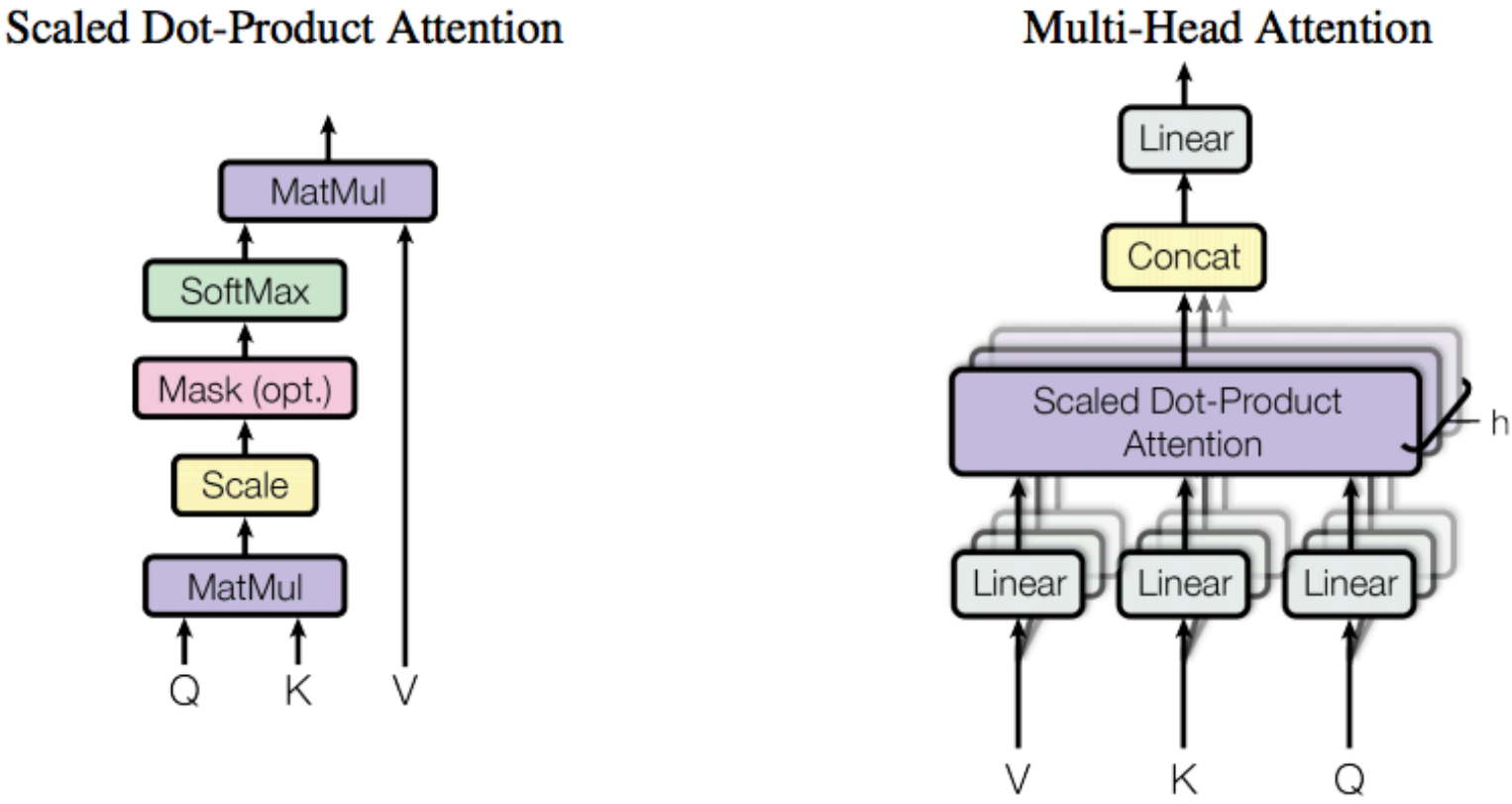


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Vaswani et al. (2017): *Attention Is All You Need*

Self-Attention

- many possibilities for self-attention functions
- intuitively, the following weights a word based on how similar it is to all other words in the sequence:

$$att(x_i, i, \boldsymbol{x}) \propto \exp\left\{\sum_{j=1}^{n} emb(x_i)^\top emb(x_j)\right\}$$

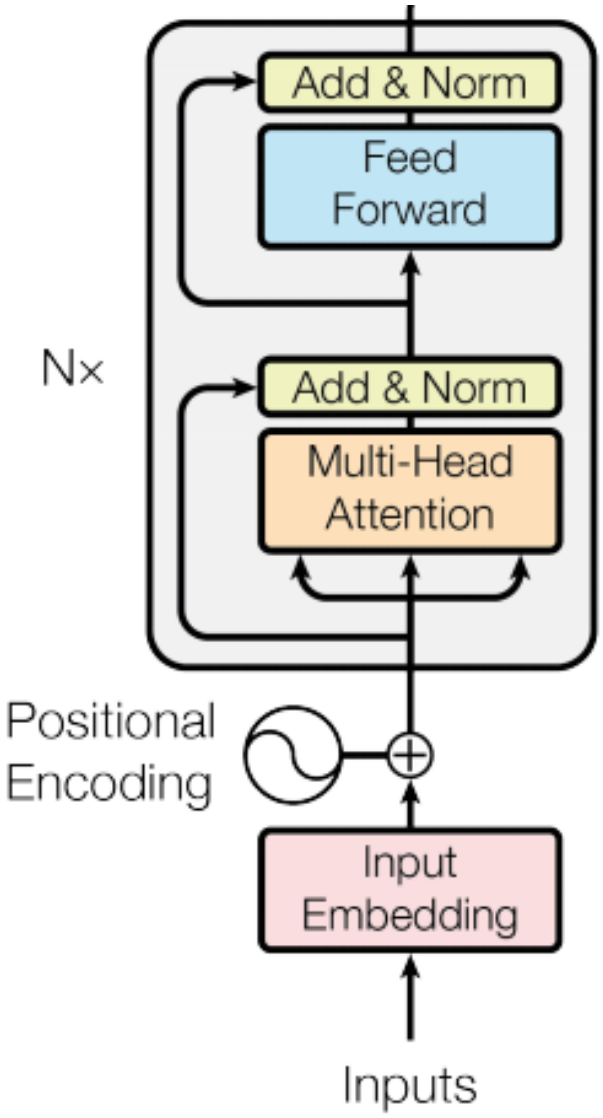
- can be combined with query/key/value-specific transformations and multiple heads

Vaswani et al. (2017): *Attention Is All You Need*

Word Position Information

- transformer adds embedding of position to word embedding in input

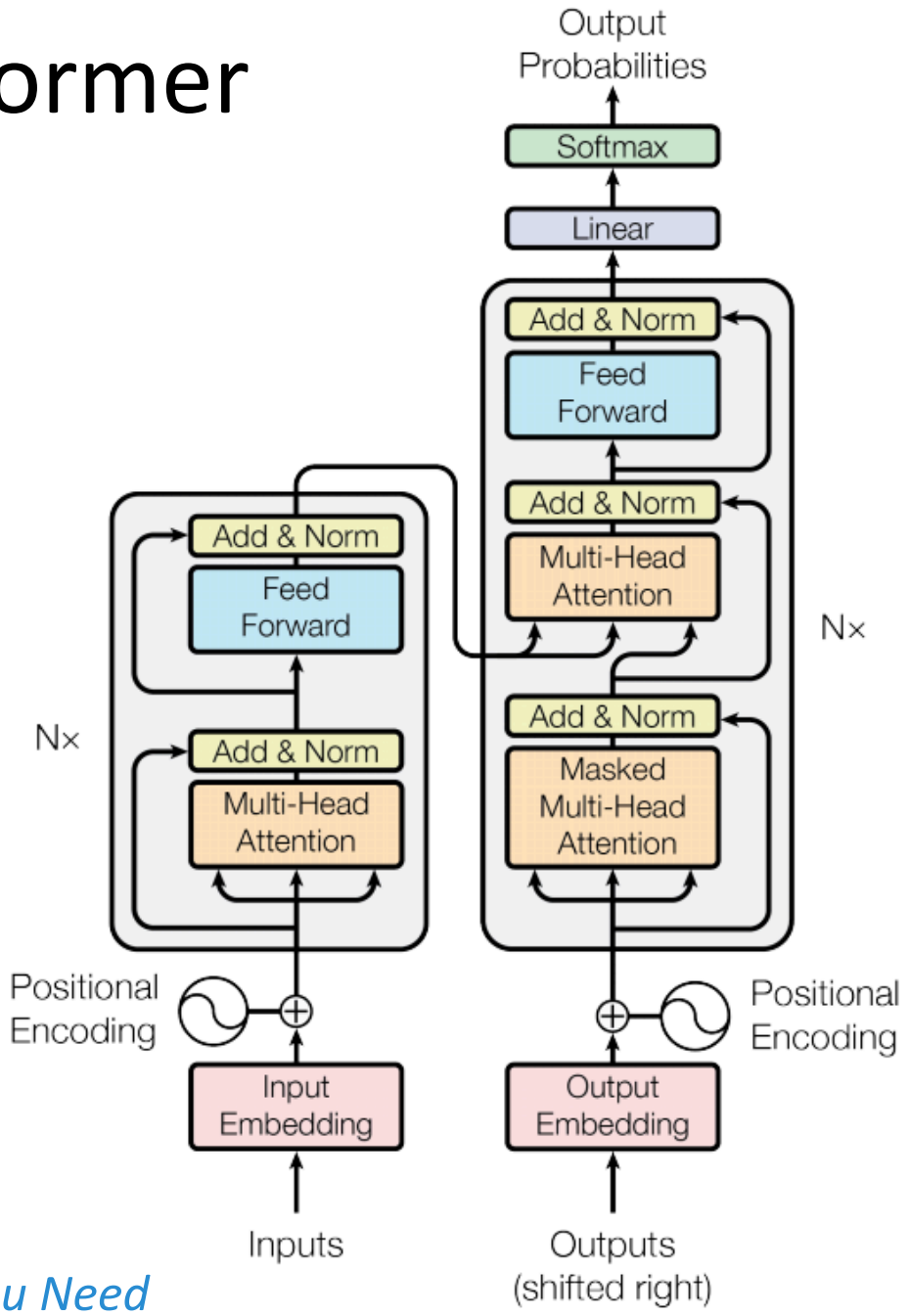
- predetermined/fixed sinusoidal embeddings or learned positional embeddings



Vaswani et al. (2017): *Attention Is All You Need*

Transformer

- developed for a setting with both encoding and decoding
- transformer decoder uses similar components to encoder, along with attention on encoder output



Vaswani et al. (2017): *Attention Is All You Need*

Attention Visualizations

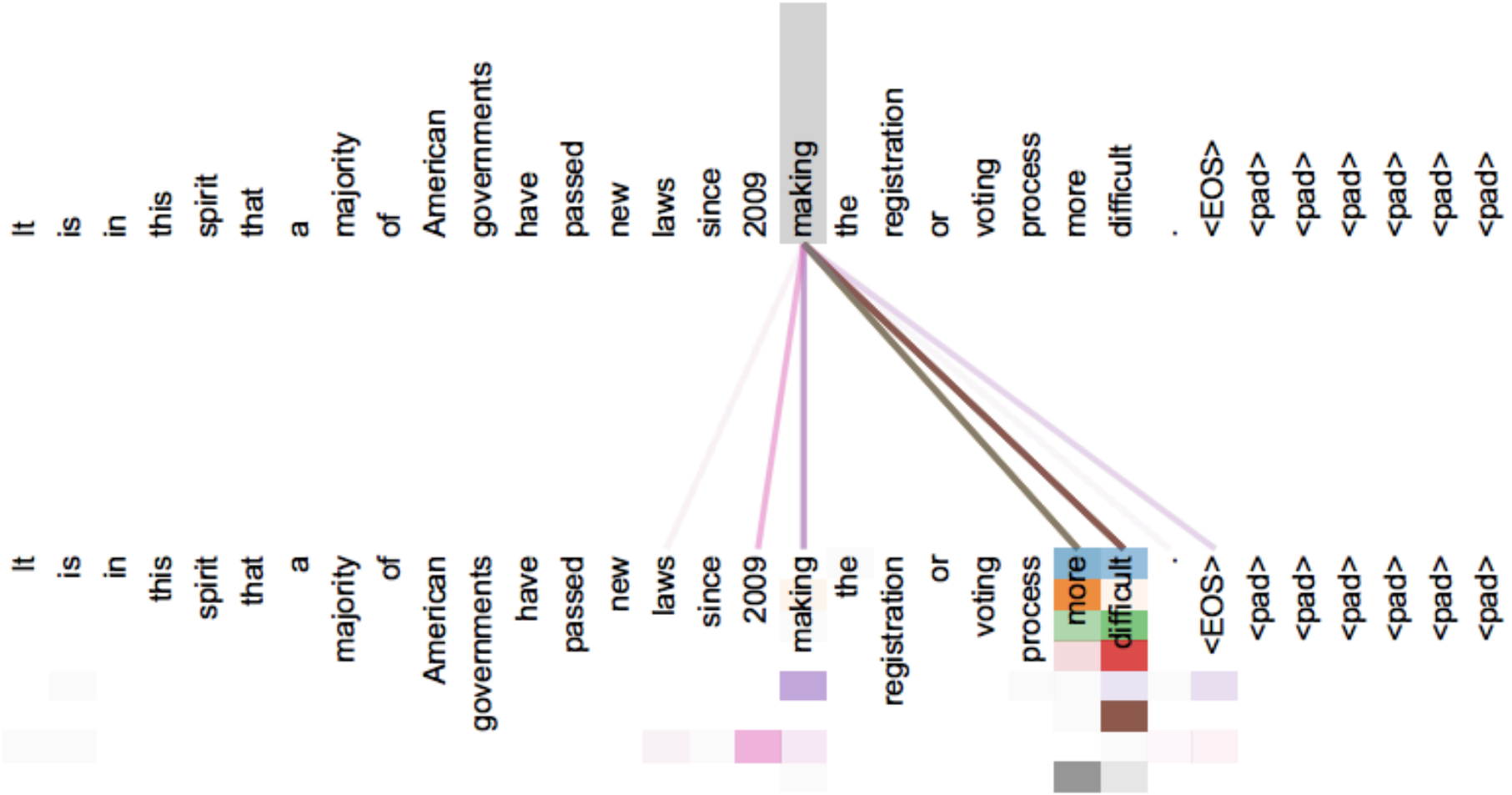
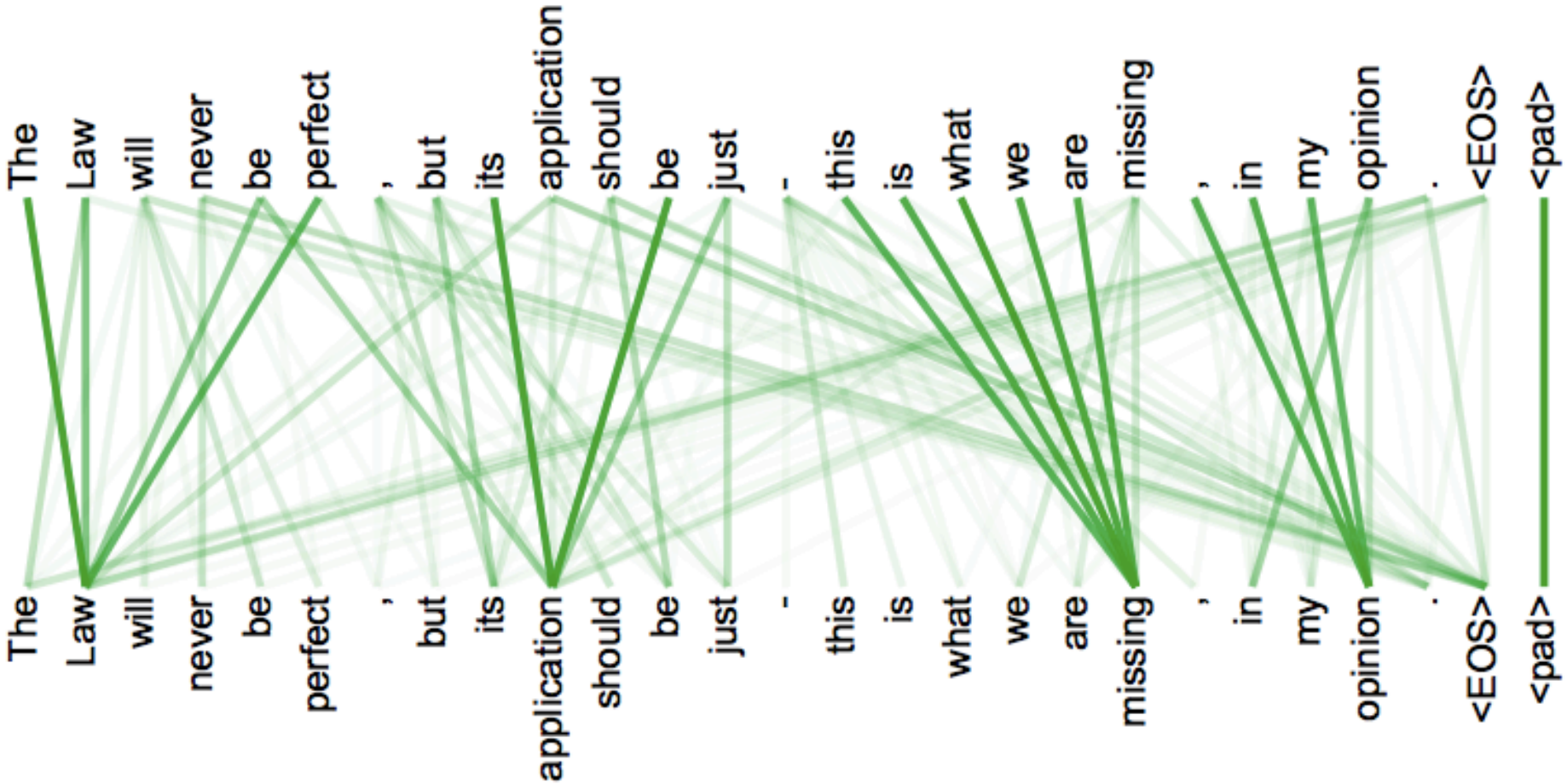


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

What do Transformers Learn?



What do Transformers Learn?

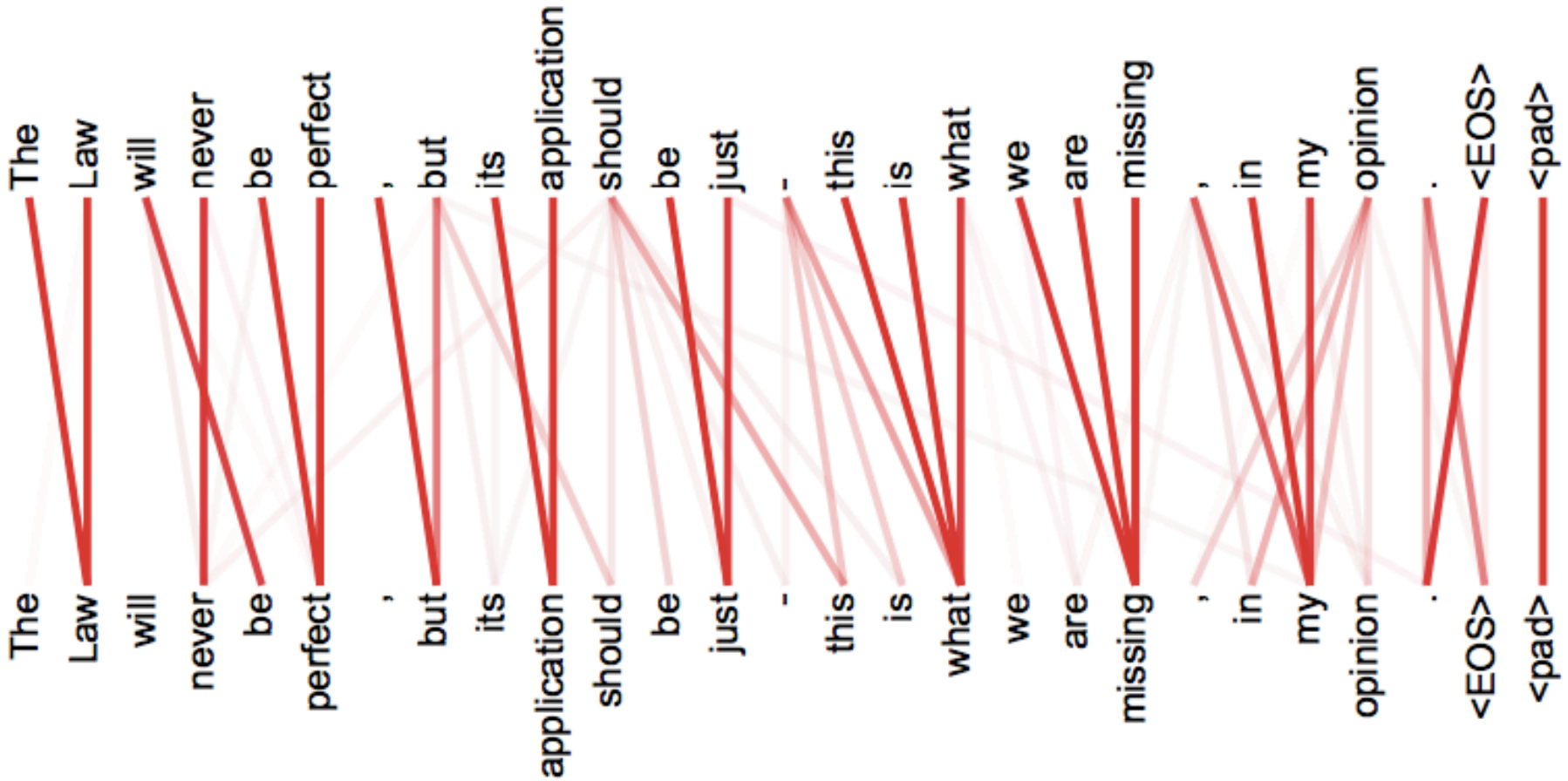
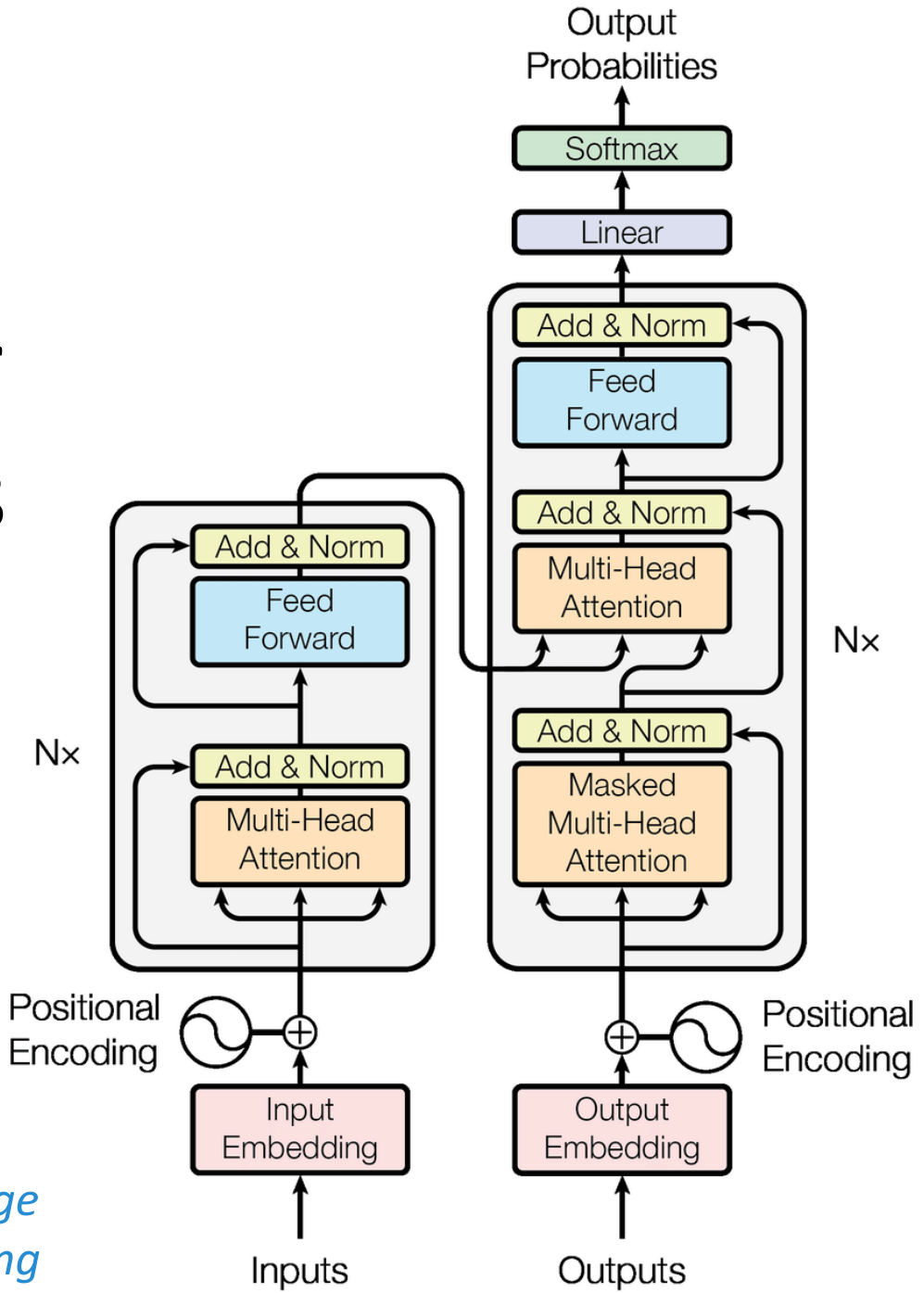


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

Applications

- transformer encoders have become widely used in NLP
- developed for sequence transduction tasks (e.g., machine translation)
- also highly effective for language modeling, parsing, etc.
- we'll briefly discuss two pretrained models based on transformers

Generative Pre-Training with Transformer Language Models



OpenAI

Radford et al. (2018): *Improving language understanding with unsupervised learning*

OpenAI GPT

- train a transformer language model (predict the next word)
- training data = passages from books
- applied to several NLP tasks that involve predicting things about sentences or filling in the next word
- transformer architecture (likely?) helps with capturing long-distance dependencies
- they then scaled it up to produce GPT-2

OpenAI GPT-2

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

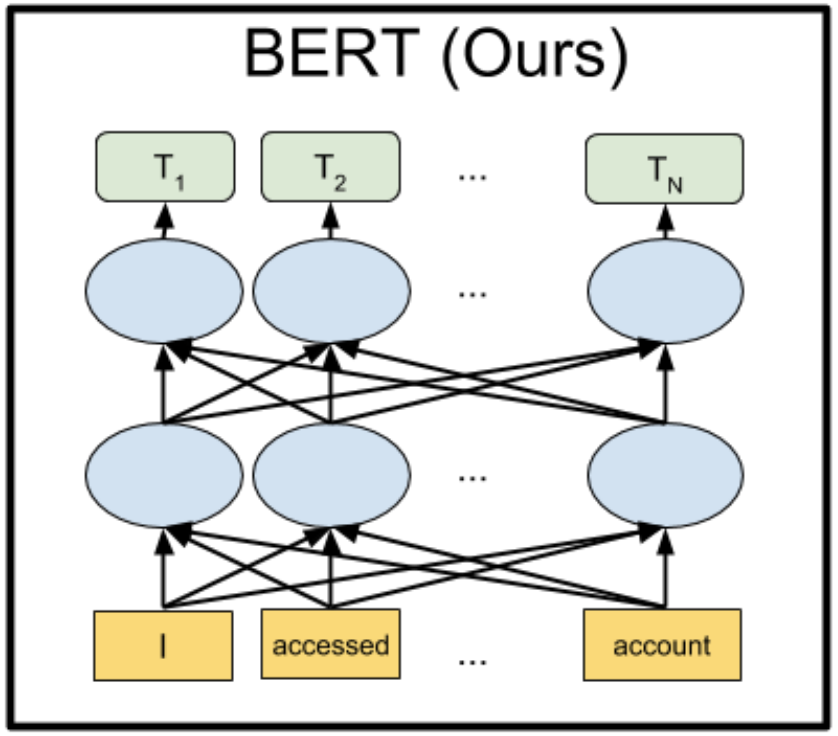
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

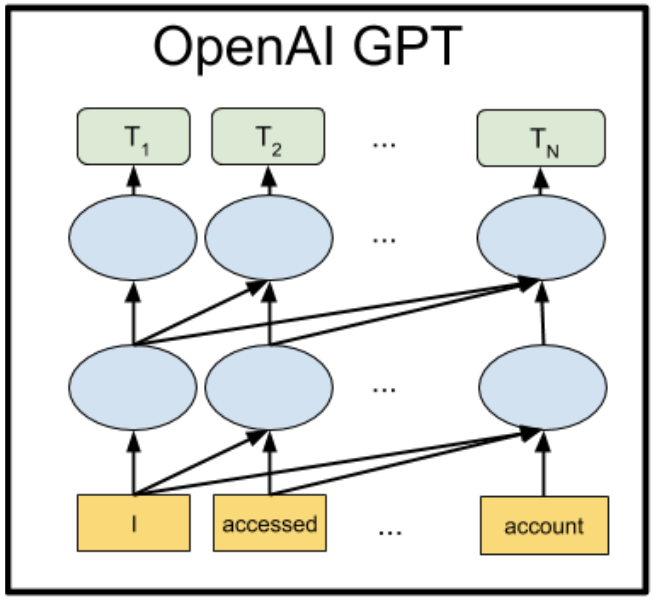
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

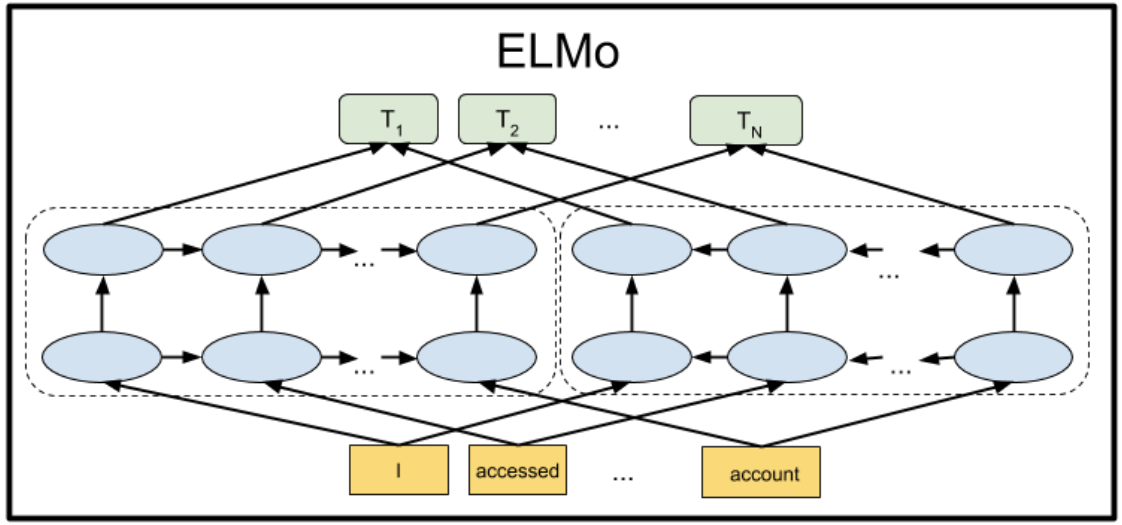
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Radford et al. (2019): *Language Models are Unsupervised Multitask Learners*

BERT







Google AI

Devlin et al. (2018): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

SQuAD1.1 Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph. How will your system compare to humans on this task?

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 20, 2019	BERT + DAE + AoA (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**87.147**	**89.474**
2 Mar 15, 2019	BERT + ConvLSTM + MTL + Verifier (ensemble) *Layer 6 AI*	86.730	89.286
3 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	86.673	89.147

General Language Understanding Evaluation (GLUE) Leaderboard

Rank	Model	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	QNLI	RTE
1	BERT: 24-layers, 1024-hidden, 16-heads	80.4	60.5	94.9	85.4/89.3	87.6/86.5	89.3/72.1	86.7	91.1	70.1
2	Singletask Pretrain Transformer	72.8	45.4	91.3	75.7/82.3	82.0/80.0	88.5/70.3	82.1	88.1	56.0
3	BiLSTM+ELMo+Attn	70.5	36.0	90.4	77.9/84.9	75.1/73.3	84.7/64.8	76.4	79.9	56.8

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
1	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
2	Microsoft D365 AI & MSR	MT-DNN++ (BigBird)		83.8	65.4	95.6	91.1/88.2	89.6/89.0	72.7/89.6
3	王玮	ALICE large (Alibaba DAMO NLP)		83.3	63.5	95.2	91.8/89.0	89.8/88.8	74.0/90.4
4	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
5	张倬胜	SemBERT		82.9	62.3	94.6	91.2/88.3	87.8/86.7	72.8/89.8
6	Anonymous Anonymous	BERT + BAM		82.3	61.5	95.2	91.3/88.3	88.6/87.9	72.5/89.7
7	Jason Phang	BERT on STILTs		82.0	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4
8	Jacob Devlin	BERT: 24-layers, 16-heads, 1024-		80.5	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3
9	Neil Houlsby	BERT + Single-task Adapters		80.2	59.2	94.3	88.7/84.3	87.3/86.1	71.5/89.4
10	Alec Radford	Singletask Pretrain Transformer		72.8	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5
11	GLUE Baselines	BiLSTM+ELMo+Attn		70.0	33.6	90.4	84.4/78.0	74.2/72.3	63.1/84.3

BERT Overview

- transformer encoder with 24 layers, hidden size 1024, 16 attention heads; 340M parameters
- learned positional embeddings (up to 512)
- word prediction softmax layers are attached to the output layer (at every position)
- two training tasks:
  - masked LM: mask certain input words, predict them
  - next-sentence classification: encode two sentences, classify whether second follows first in a corpus

Devlin et al. (2018): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT Input Representation

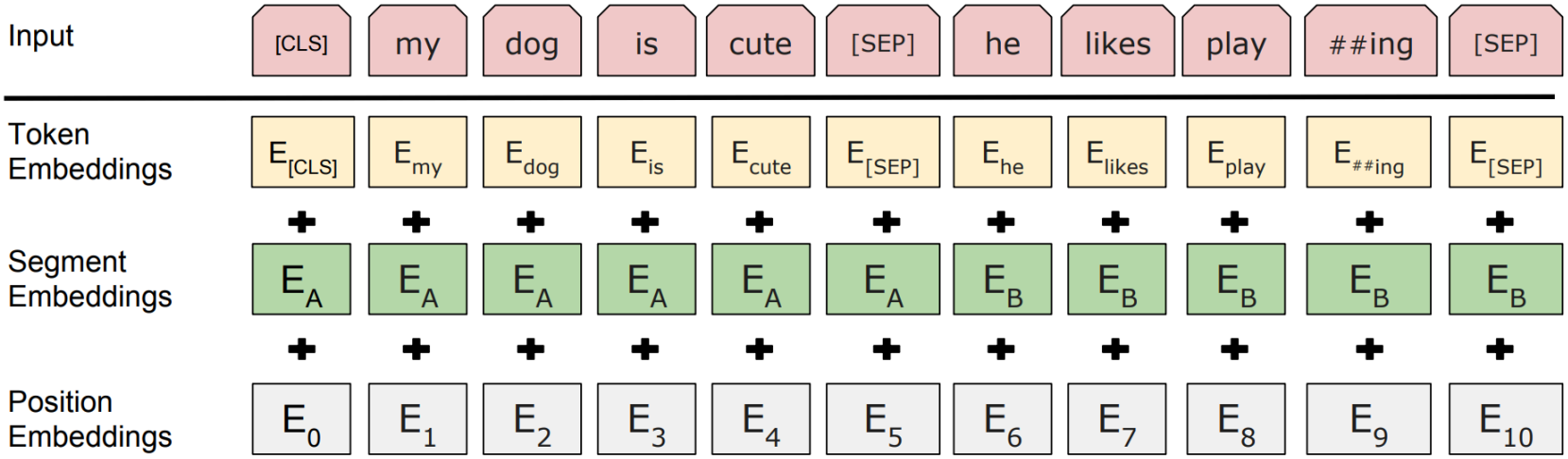


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- input uses WordPieces (vocabulary of size 30,000)
- first token is [CLS]
- special token [SEP] separates sentences

Devlin et al. (2018): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Pretraining Task 1: Masked Word Prediction

training data generator chooses 15% of tokens at random, e.g., in the sentence `my dog is hairy` it chooses `hairy`. It then performs the following procedure:

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Devlin et al. (2018): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Pretraining Task 2: Next Sentence Classification

Input = `[CLS] the man went to [MASK] store [SEP]`

`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`

`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

Devlin et al. (2018): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Today

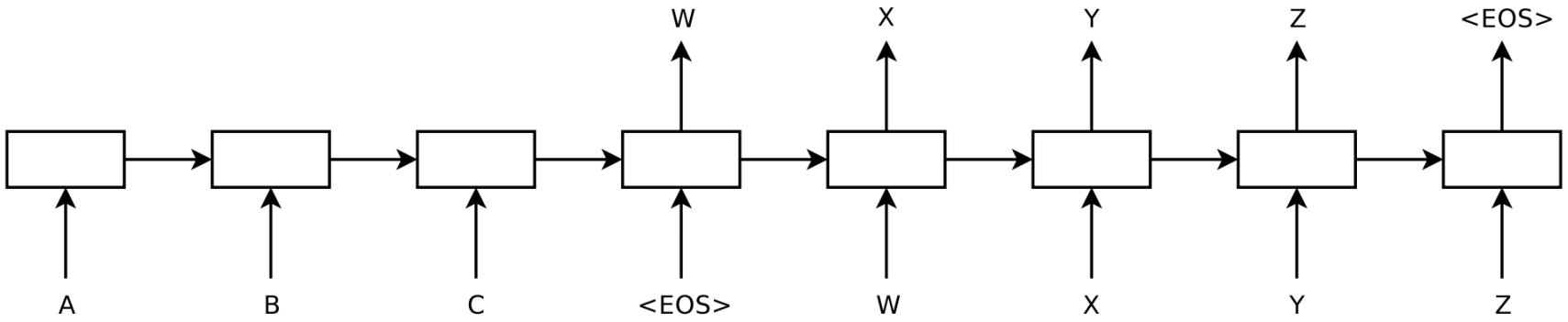
- finish transformers
- sequence-to-sequence modeling and attention

Sequence-to-Sequence Modeling

- data: <input sequence, output sequence> pairs
- use one RNN to encode input sequence
- use another RNN to decode the output sequence from the input sequence vector
- often more generally called “encoder-decoder” models

Application: Machine Translation

- encode source sentence into a vector, decode translation



Kalchbrenner & Blunsom (2013): *Recurrent Continuous Translation Models*

Cho et al. (2014): *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*

Sutskever et al. (2014): *Sequence to Sequence Learning with Neural Networks*

Application: Chatbots

Human: *what is the purpose of life ?*
Machine: *to serve the greater good .*
Human: *what is the purpose of living ?*
Machine: *to live forever .*
Human: *what is the purpose of existence ?*
Machine: *to find out what happens when we get to the planet earth .*
Human: *where are you now ?*
Machine: *i 'm in the middle of nowhere .*
Human: *what is the purpose of dying ?*
Machine: *to have a life .*
Human: *what is the purpose of being intelligent ?*
Machine: *to find out what it is .*
Human: *what is the purpose of emotions ?*
Machine: *i don 't know .*

Vinyals & Le (2015): *A Neural Conversational Model*

Application: Parsing

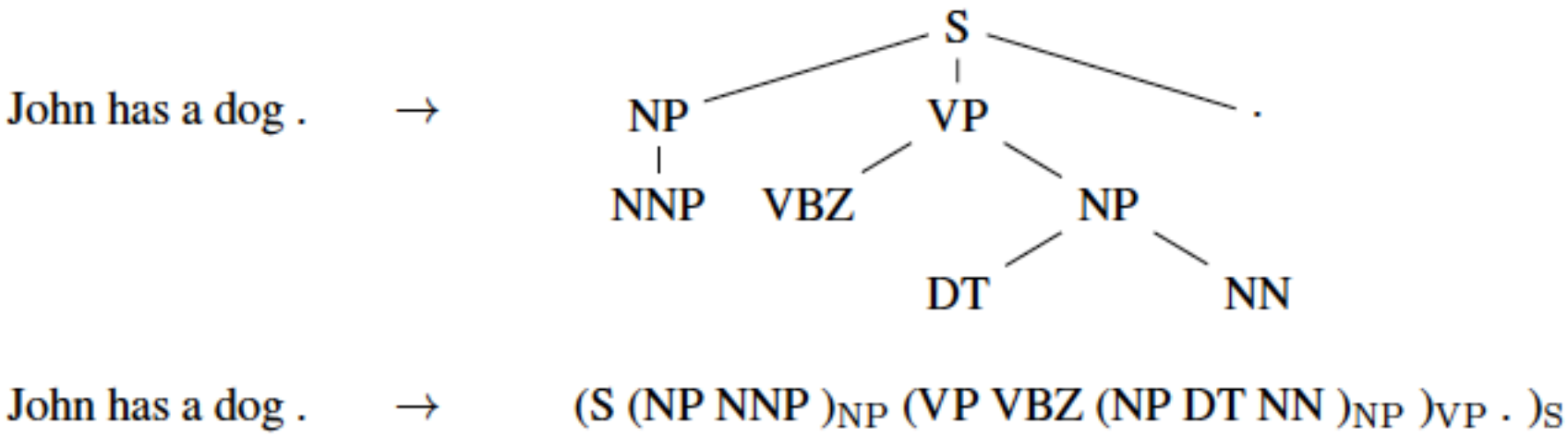
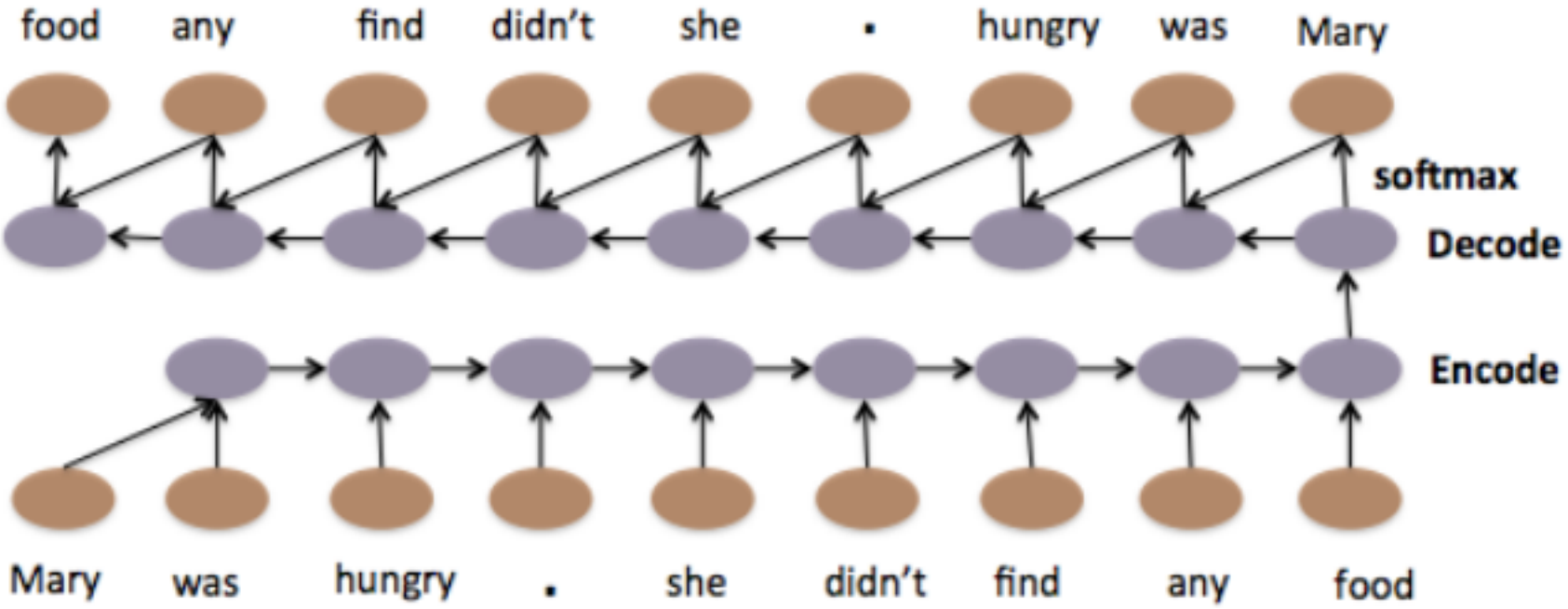


Figure 2: Example parsing task and its linearization.

{Vinyals, Kaiser} et al. (2015): *Grammar as a Foreign Language*

RNN Autoencoders

- encode sentence, decode sentence
- can use final hidden vector of sentence encoder as a sentence embedding

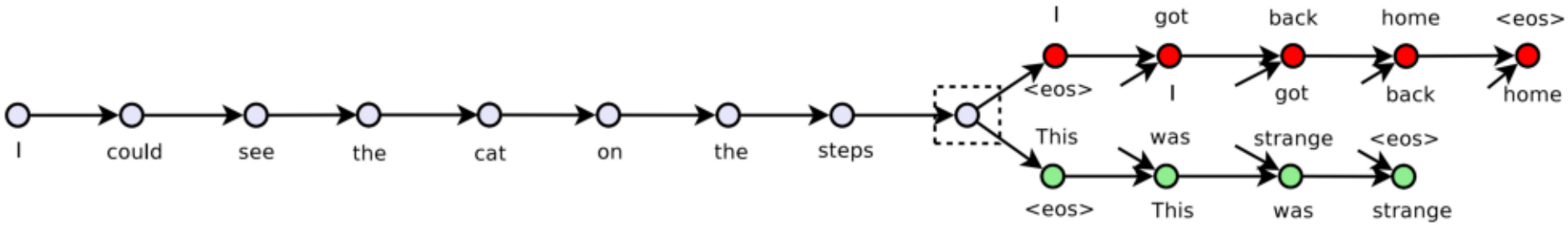


Li et al. (2015): *A Hierarchical Neural Autoencoder for Paragraphs and Documents*

Skip-Thoughts

- encode sentence using an RNN
- decode two neighboring sentences
- use different RNNs for previous and next sentences
- also pass encoder sentence vector on each decoding step

…I got back home I could see the cat on the steps This was strange …



Kiros et al. (2015): *Skip-Thought Vectors*

Skip-Thoughts

query sentence:

im sure youll have a glamorous evening , she said , giving an exaggerated wink .

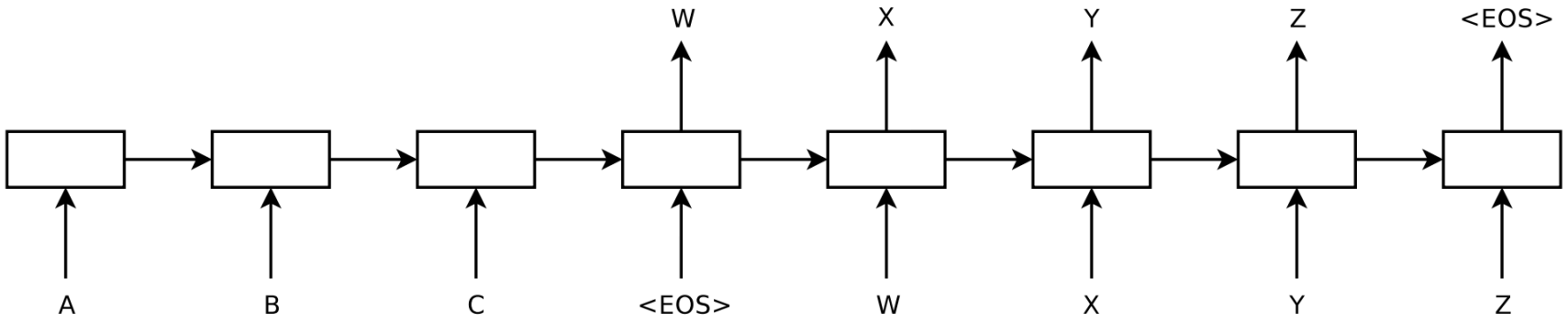
nearest neighbor:

im really glad you came to the party tonight , he said , turning to her .

Kiros, Zhu, Salakhutdinov, Zemel, Torralba, Urtasun, Fidler (2015)

Encoder-Decoder Models for Neural Machine Translation

- encode source sentence into a vector, decode translation



Kalchbrenner & Blunsom (2013): *Recurrent Continuous Translation Models*

Cho et al. (2014): *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*

Sutskever et al. (2014): *Sequence to Sequence Learning with Neural Networks*

Encoder-Decoder Models for Neural Machine Translation

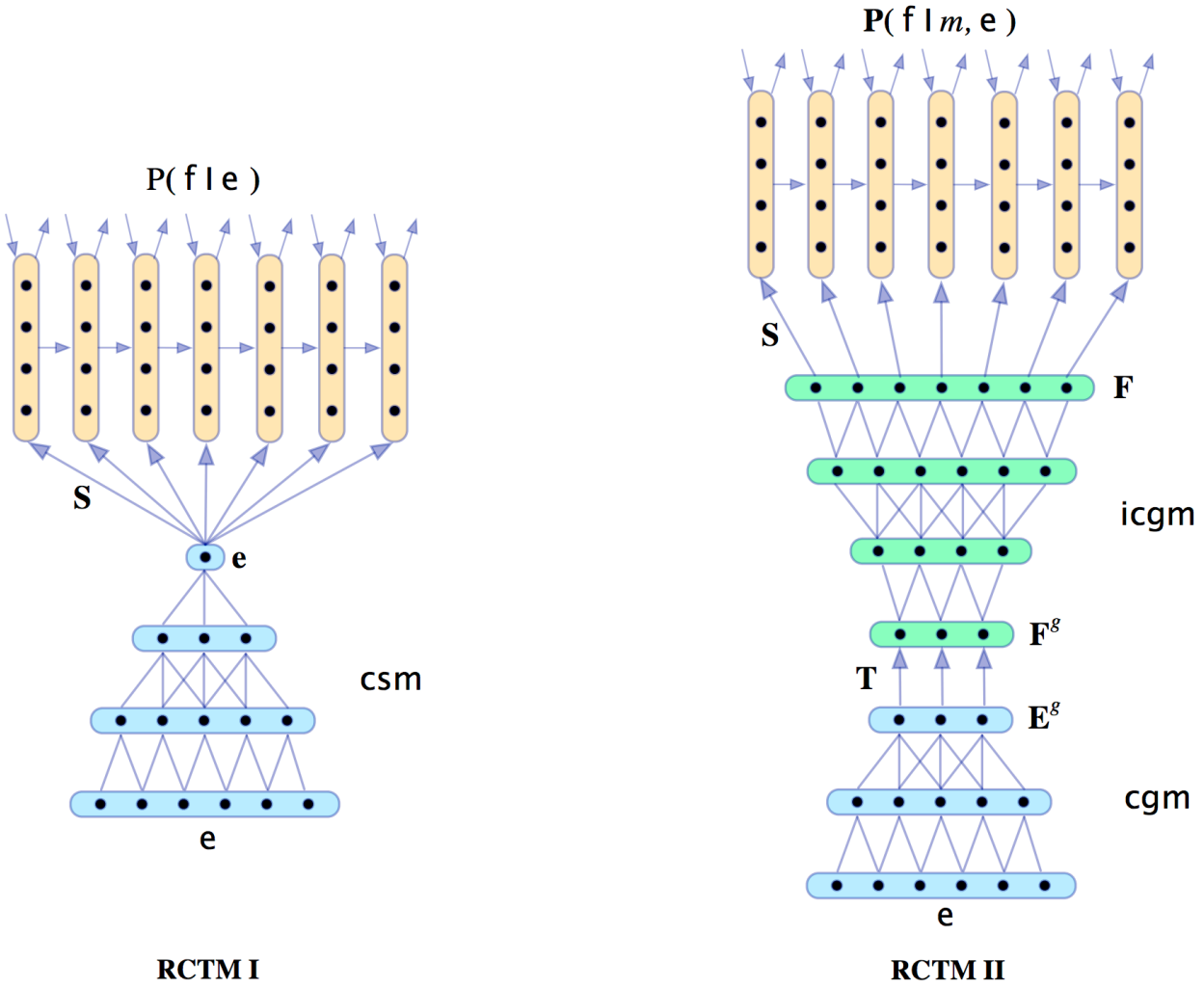


Figure 3: A graphical depiction of the two RCTMs. Arrows represent full matrix transformations while lines are vector transformations corresponding to columns of weight matrices.

Kalchbrenner & Blunsom (2013): *Recurrent Continuous Translation Models*

Encoder-Decoder Models for Neural Machine Translation

- encode source sentence into a vector, decode translation

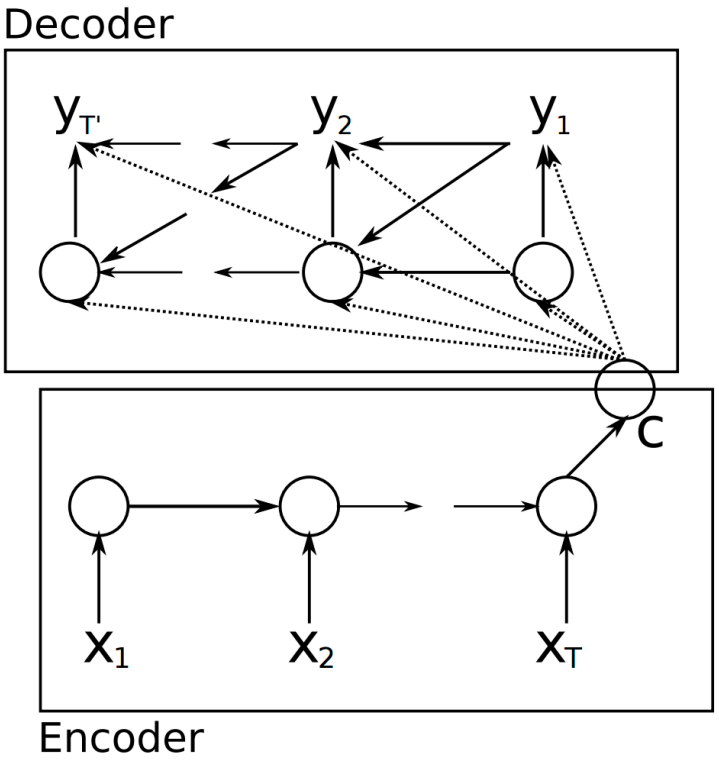
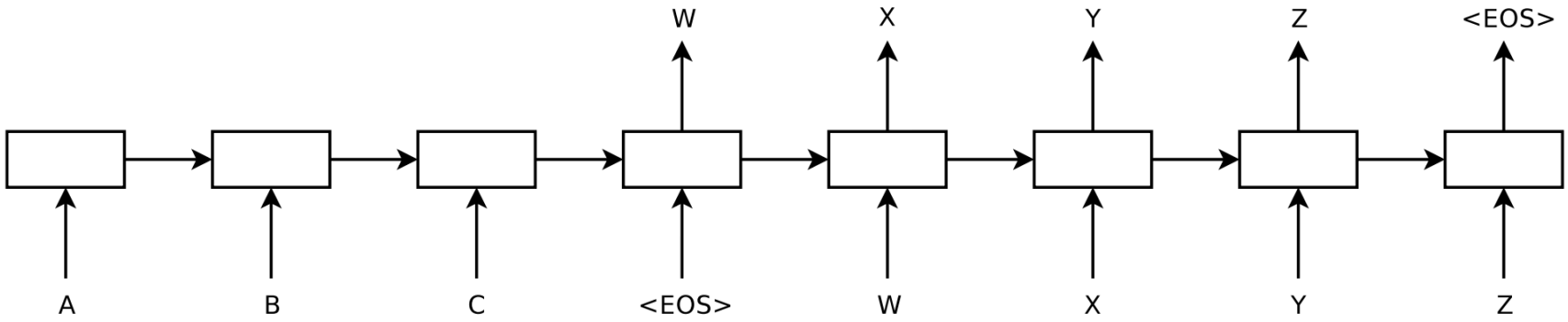


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

Cho et al. (2014): *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*

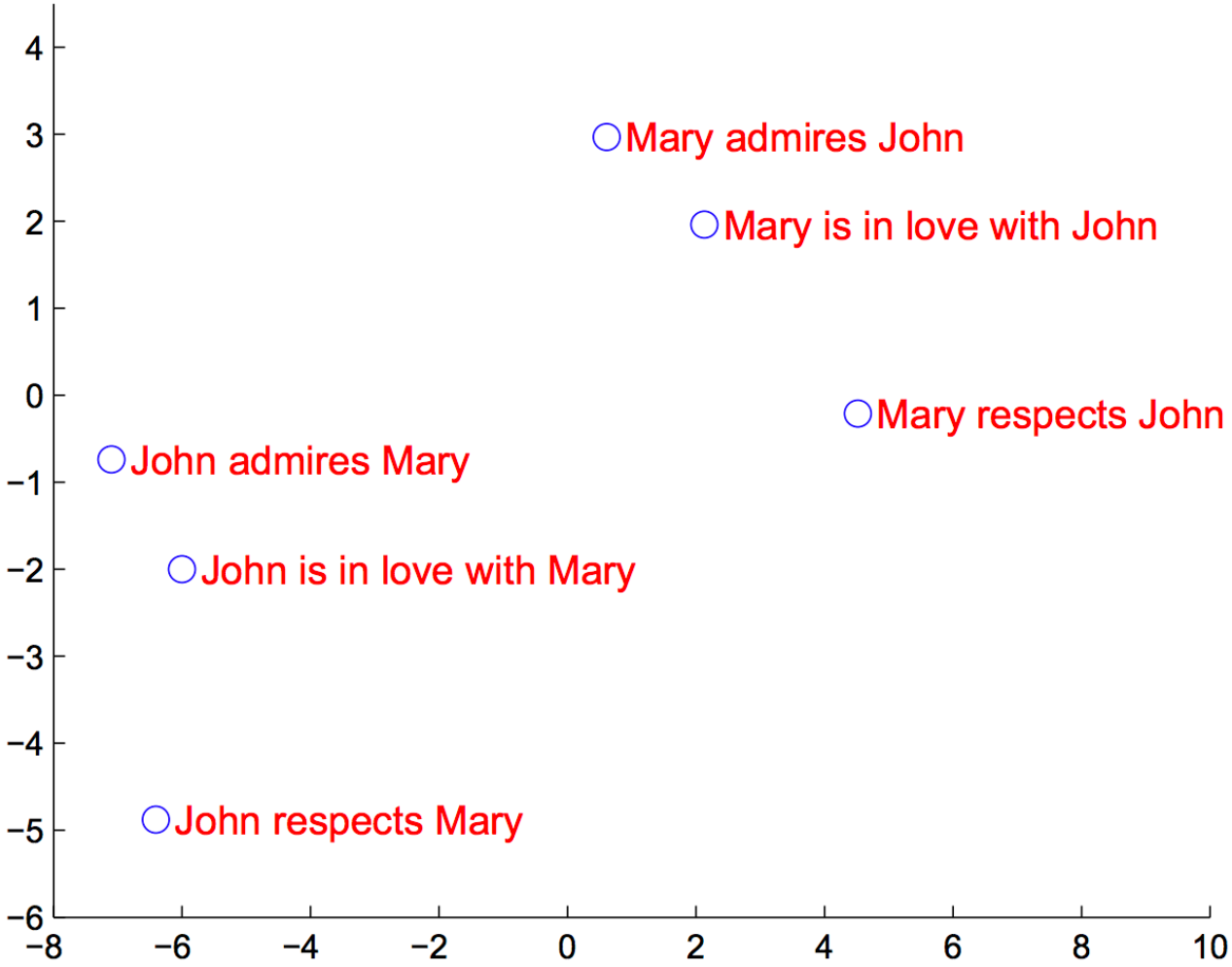
Encoder-Decoder Models for Neural Machine Translation

- encode source sentence into a vector, decode translation



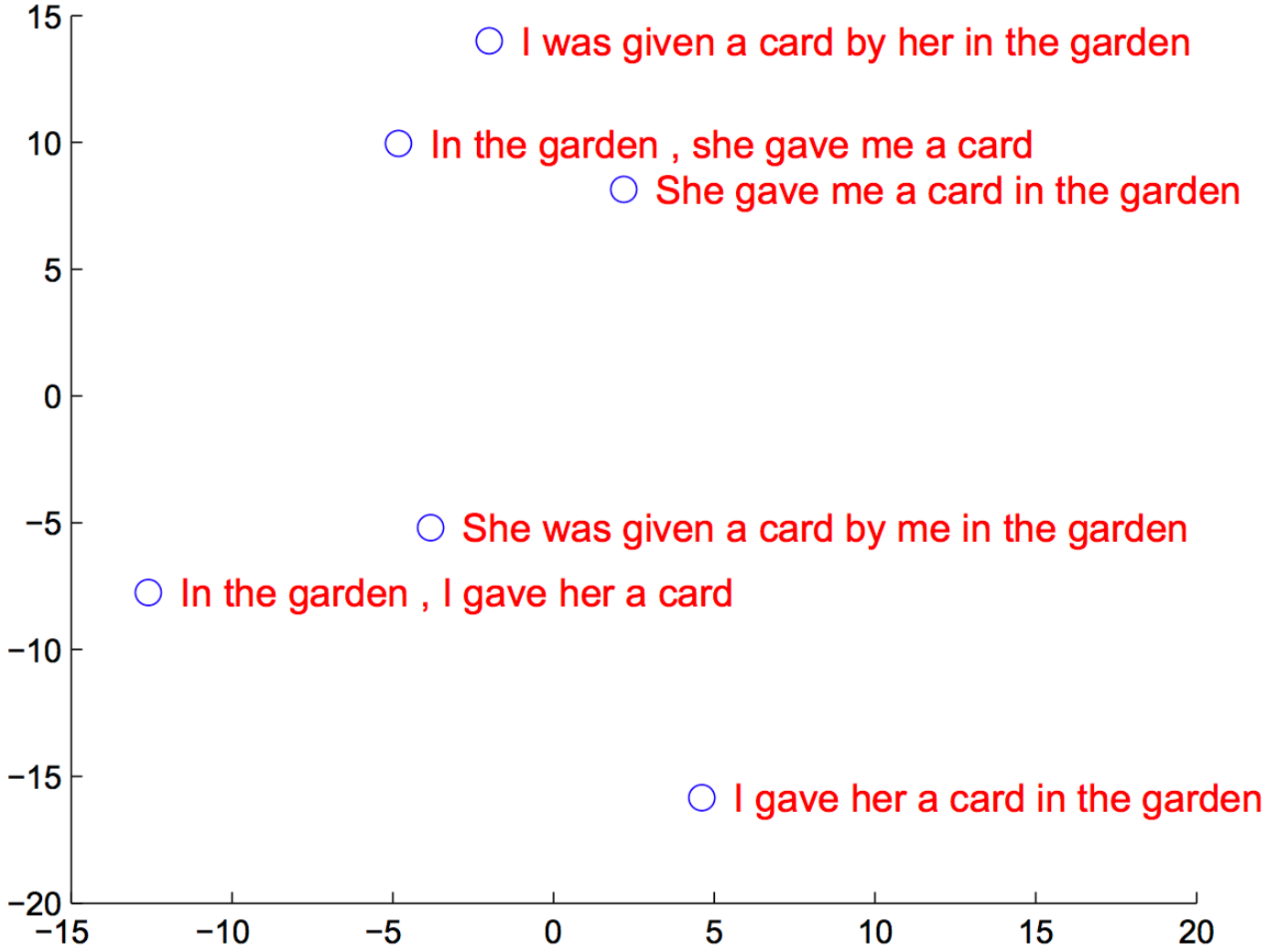
Sutskever et al. (2014): *Sequence to Sequence Learning with Neural Networks*

Encoder as a Sentence Embedding Model?

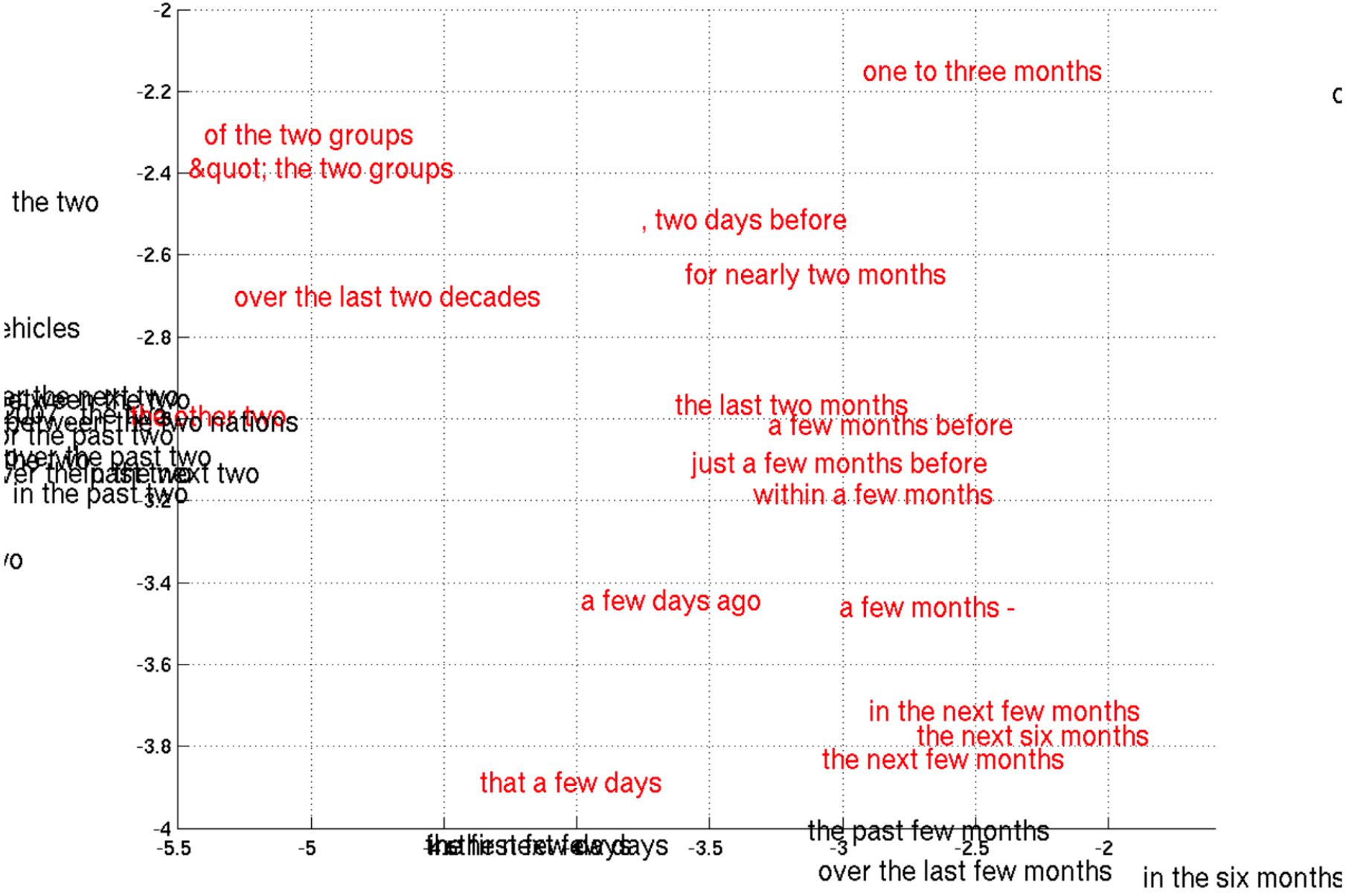


Sutskever et al. (2014): *Sequence to Sequence Learning with Neural Networks*

Encoder as a Sentence Embedding Model?



Sutskever et al. (2014): *Sequence to Sequence Learning with Neural Networks*

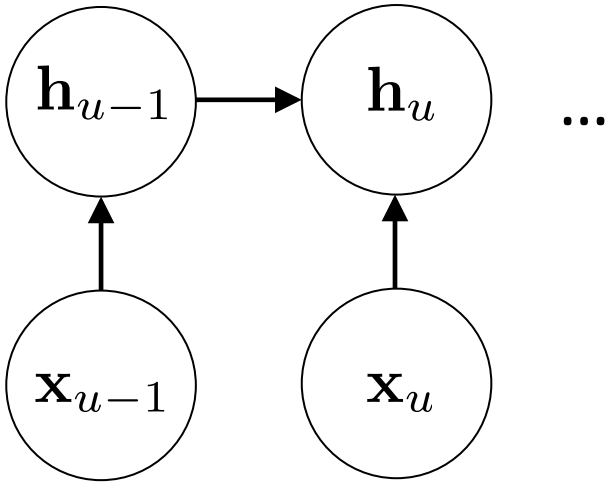


Cho et al. (2014): *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*

Sequence-to-Sequence Model

encoding input sequence:

$$\mathbf{h}_u = \tanh\left(\mathbf{W}^{(x)}\mathbf{x}_u + \mathbf{W}^{(h)}\mathbf{h}_{u-1} + \mathbf{b}^{(h)}\right)$$



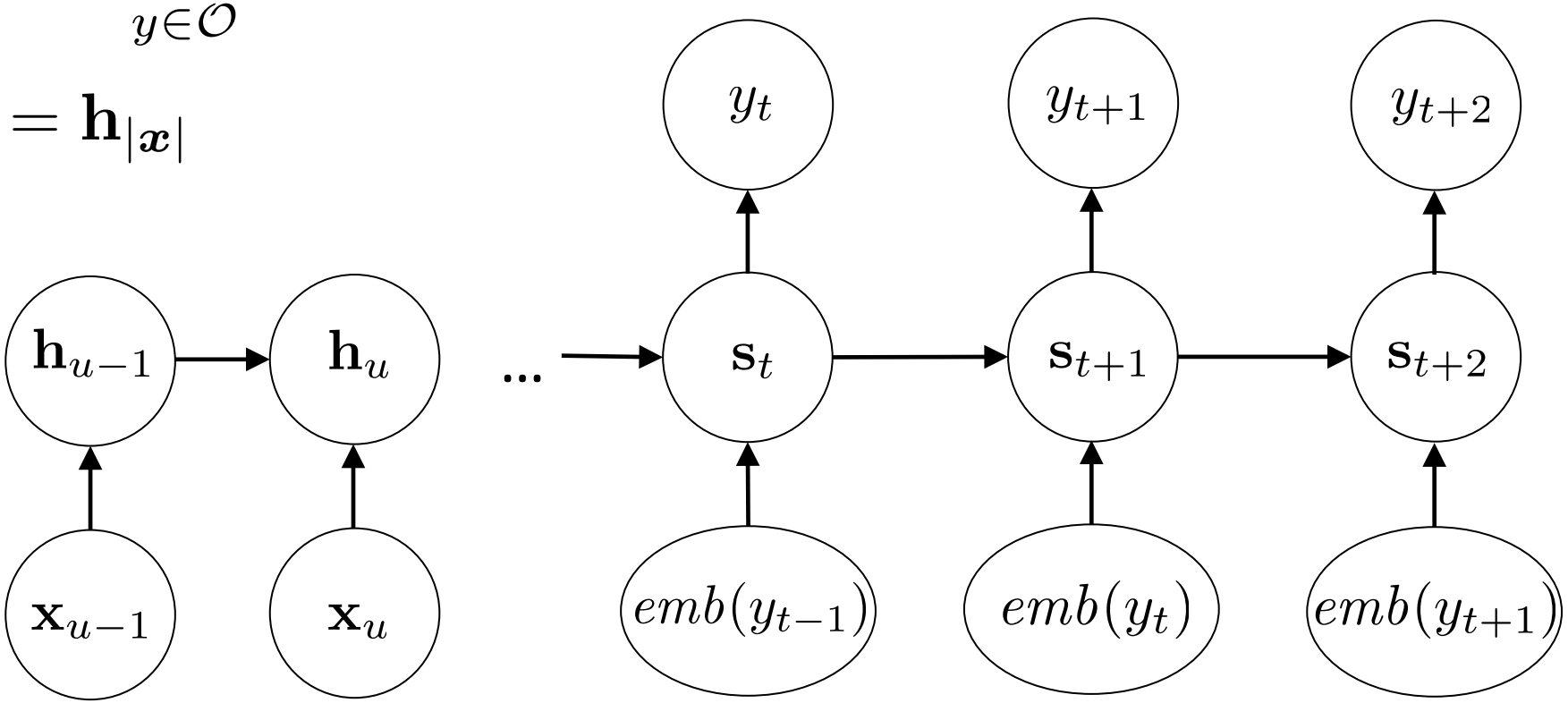
Sequence-to-Sequence Model

encoder: $\mathbf{h}_u = \tanh\left(\mathbf{W}^{(x)}\mathbf{x}_u + \mathbf{W}^{(h)}\mathbf{h}_{u-1} + \mathbf{b}^{(h)}\right)$

decoder: $\mathbf{s}_t = \tanh\left(\mathbf{W}^{(y)}\,emb(y_{t-1}) + \mathbf{W}^{(s)}\mathbf{s}_{t-1} + \mathbf{b}^{(s)}\right)$

$$y_t = \underset{y \in \mathcal{O}}{\operatorname{argmax}} \;\; \mathbf{s}_t^\top emb(y)$$

$$\mathbf{s}_0 = \mathbf{h}_{|\boldsymbol{x}|}$$



Extension: Attention

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Extension: Attention

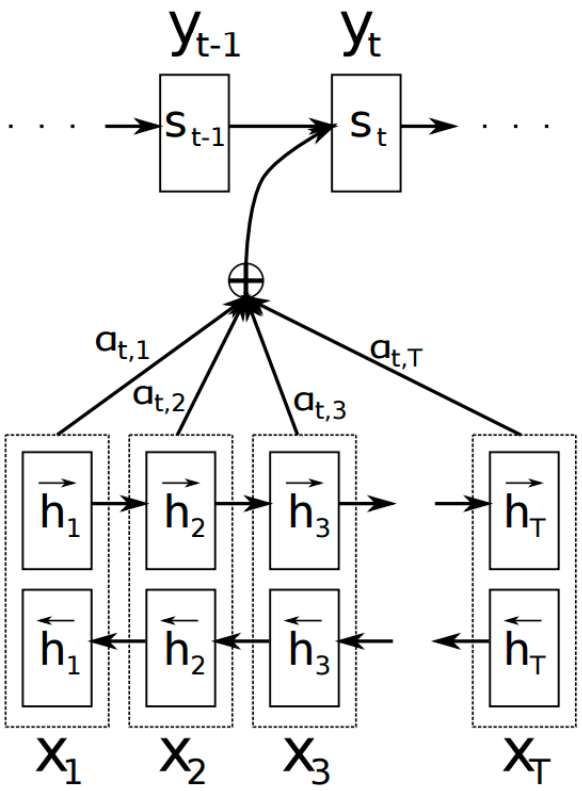
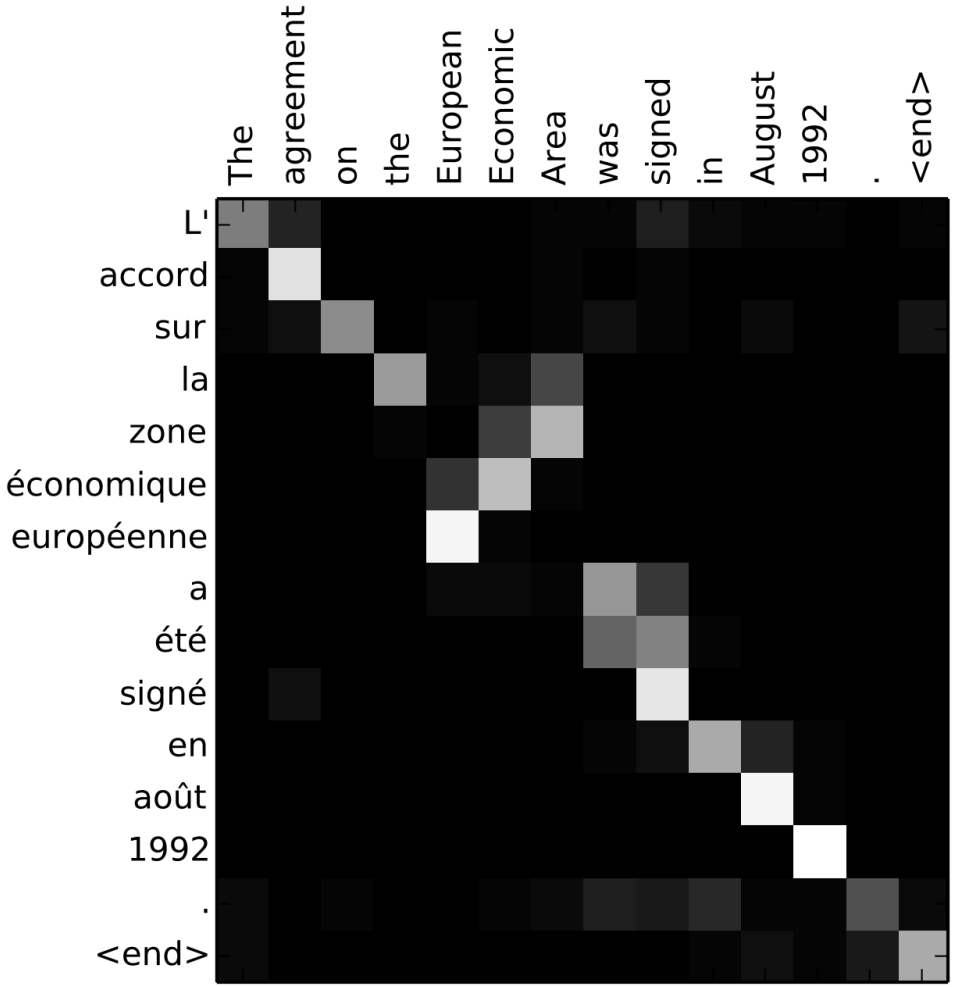


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Bahdanau et al. (2015): *Neural Machine Translation by Jointly Learning to Align and Translate*

Extension: Attention



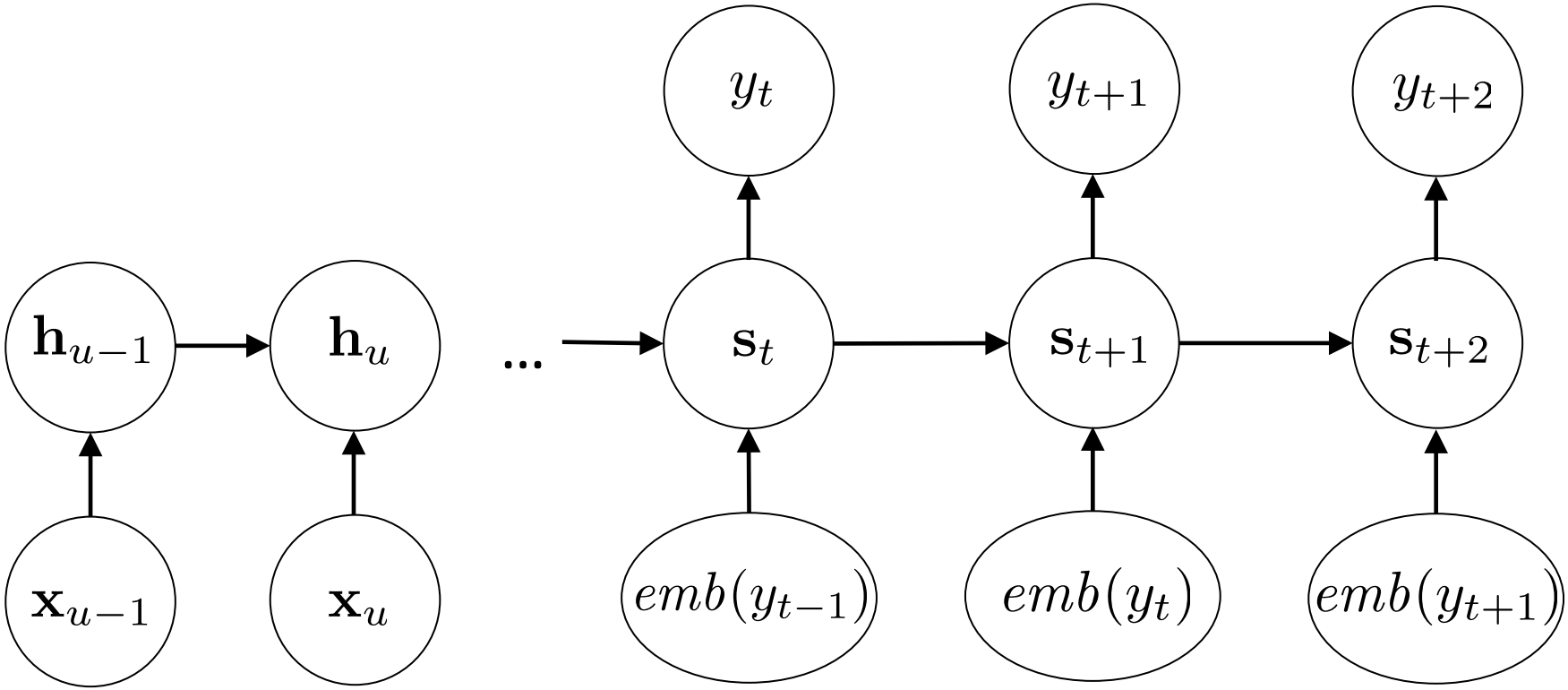
Bahdanau et al. (2015): *Neural Machine Translation by Jointly Learning to Align and Translate*

- disclaimer: the version I will present is a little simpler than the Bahdanau et al version but retains the key ideas
- see the paper for more details

Adding Attention

$$\alpha_{t,u} \propto \exp\{att(\mathbf{s}_{t-1}, \mathbf{h}_u)\}$$

att function models association between all pairs of hidden vectors in encoder and decoder



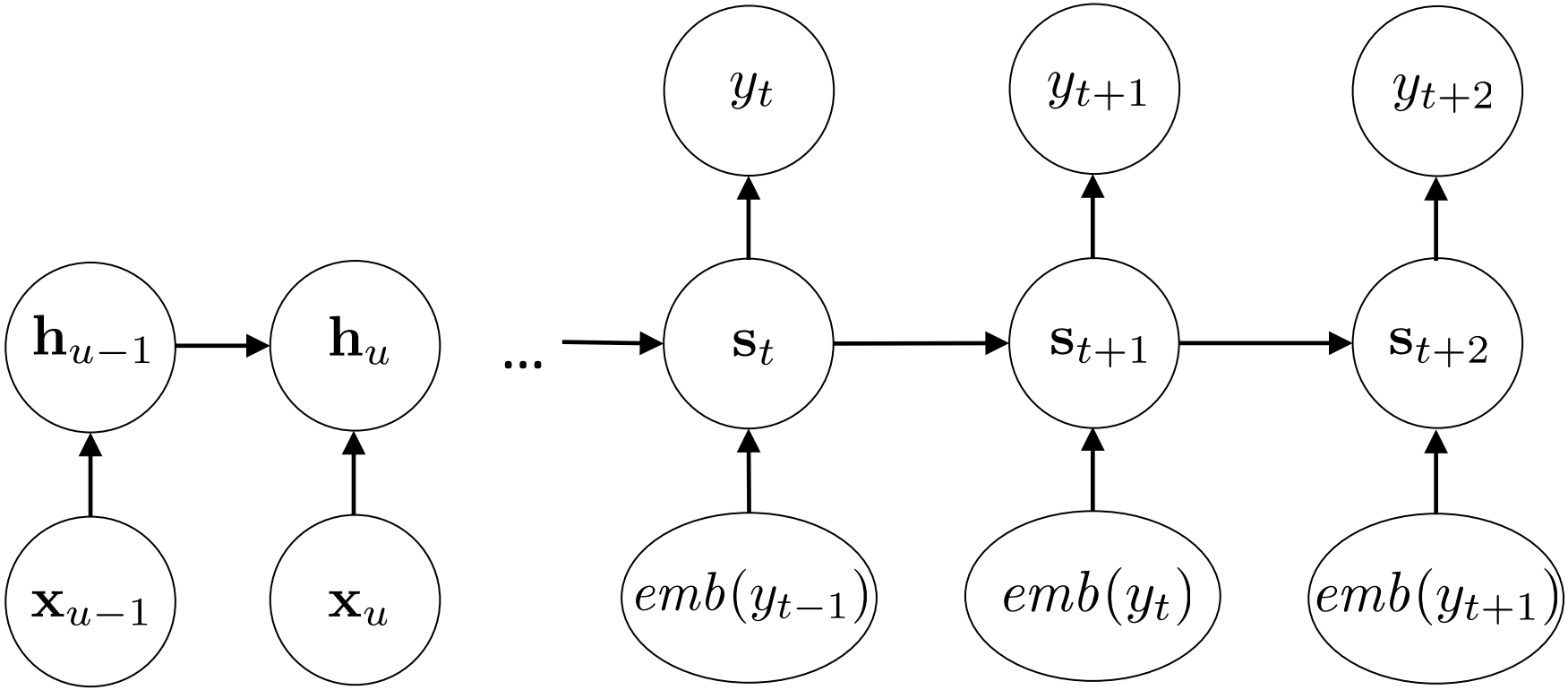
Adding Attention

$$\alpha_{t,u} \propto \exp\{att(\mathbf{s}_{t-1}, \mathbf{h}_u)\}$$

How is the *att* function defined?

Bahdanau et al. used a feed-forward network:

$$att(\mathbf{s}_{t-1}, \mathbf{h}_u) = \mathbf{v}_a^\top \tanh\left(\mathbf{W}^{(as)}\mathbf{s}_{t-1} + \mathbf{W}^{(ah)}\mathbf{h}_u\right)$$

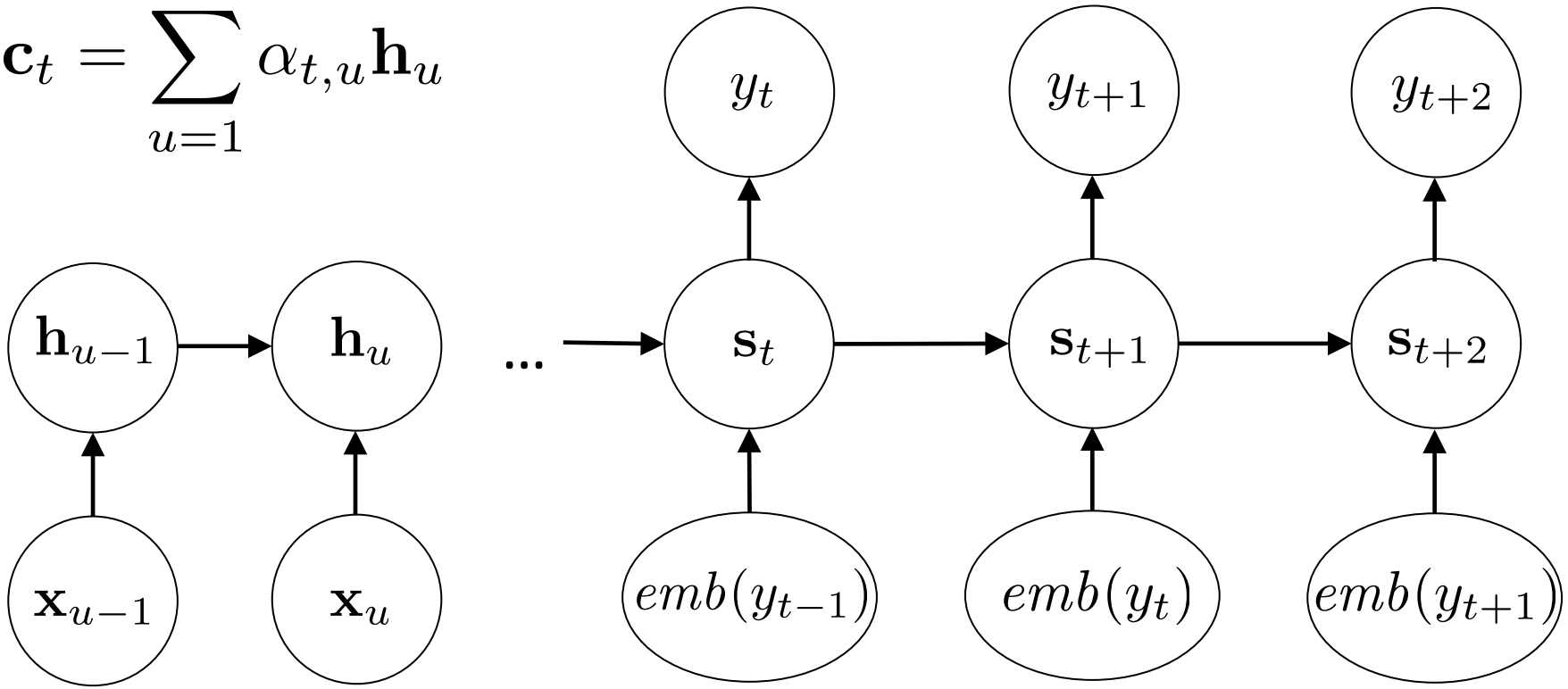


Adding Attention

$$\alpha_{t,u} \propto \exp\{att(\mathbf{s}_{t-1}, \mathbf{h}_u)\}$$

create new decoder hidden vector based on attention-weighted sum of encoder hidden vectors:

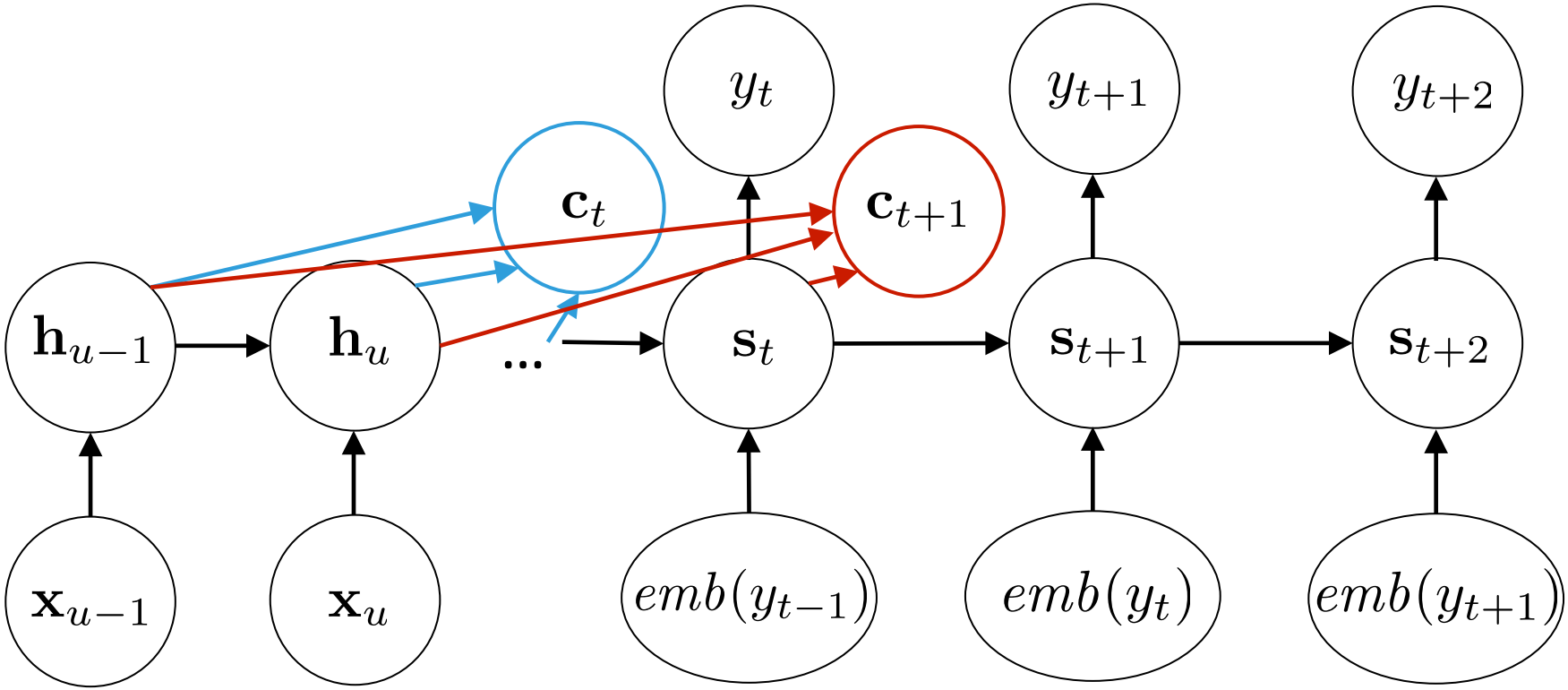
$$\mathbf{c}_t = \sum_{u=1}^{|\boldsymbol{x}|} \alpha_{t,u} \mathbf{h}_u$$



$$\alpha_{t,u} \propto \exp\{att(\mathbf{s}_{t-1}, \mathbf{h}_u)\}$$

$$\mathbf{c}_t = \sum_{u=1}^{|\boldsymbol{x}|} \alpha_{t,u} \mathbf{h}_u$$

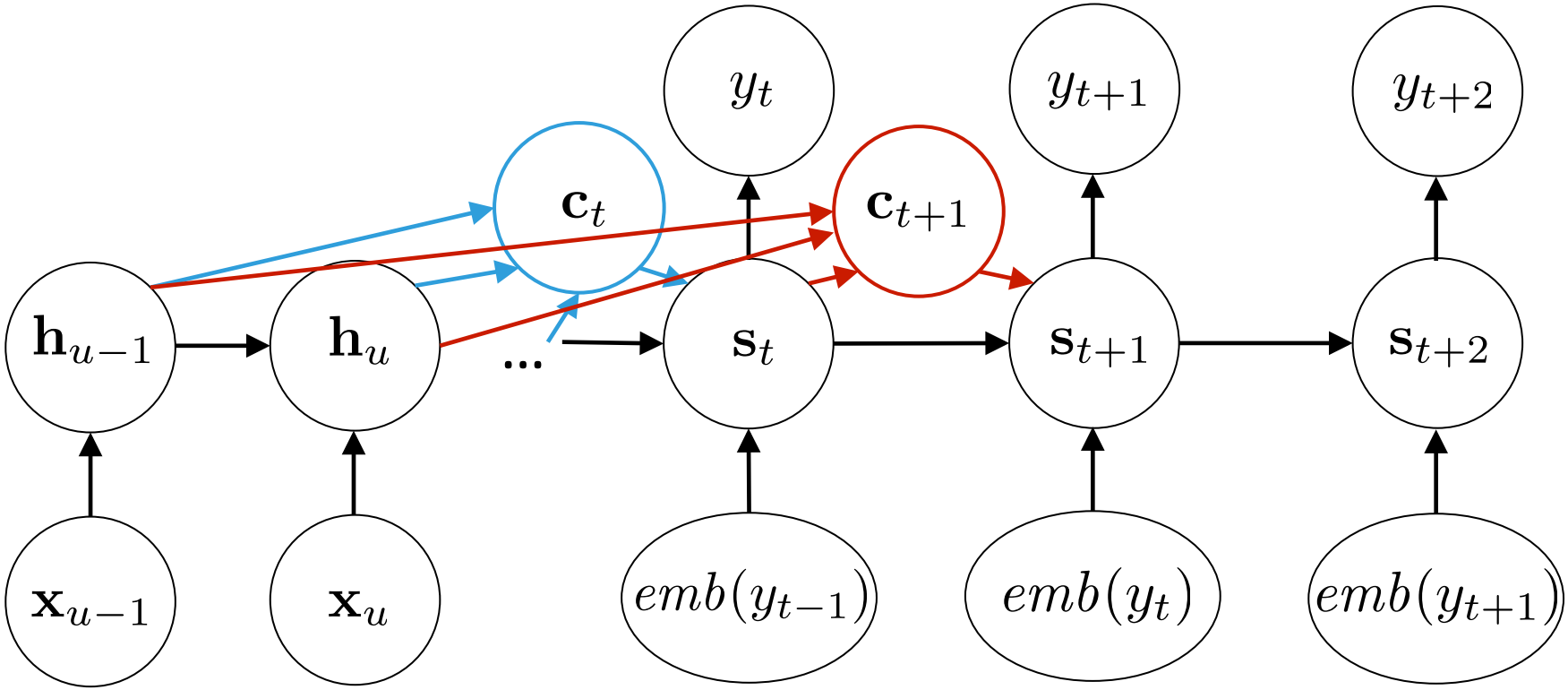
create new decoder hidden vector based on attention-weighted sum of encoder hidden vectors:



$$\mathbf{c}_t = \sum_{u=1}^{|\boldsymbol{x}|} \alpha_{t,u} \mathbf{h}_u$$

new RNN decoder hidden state update equation:

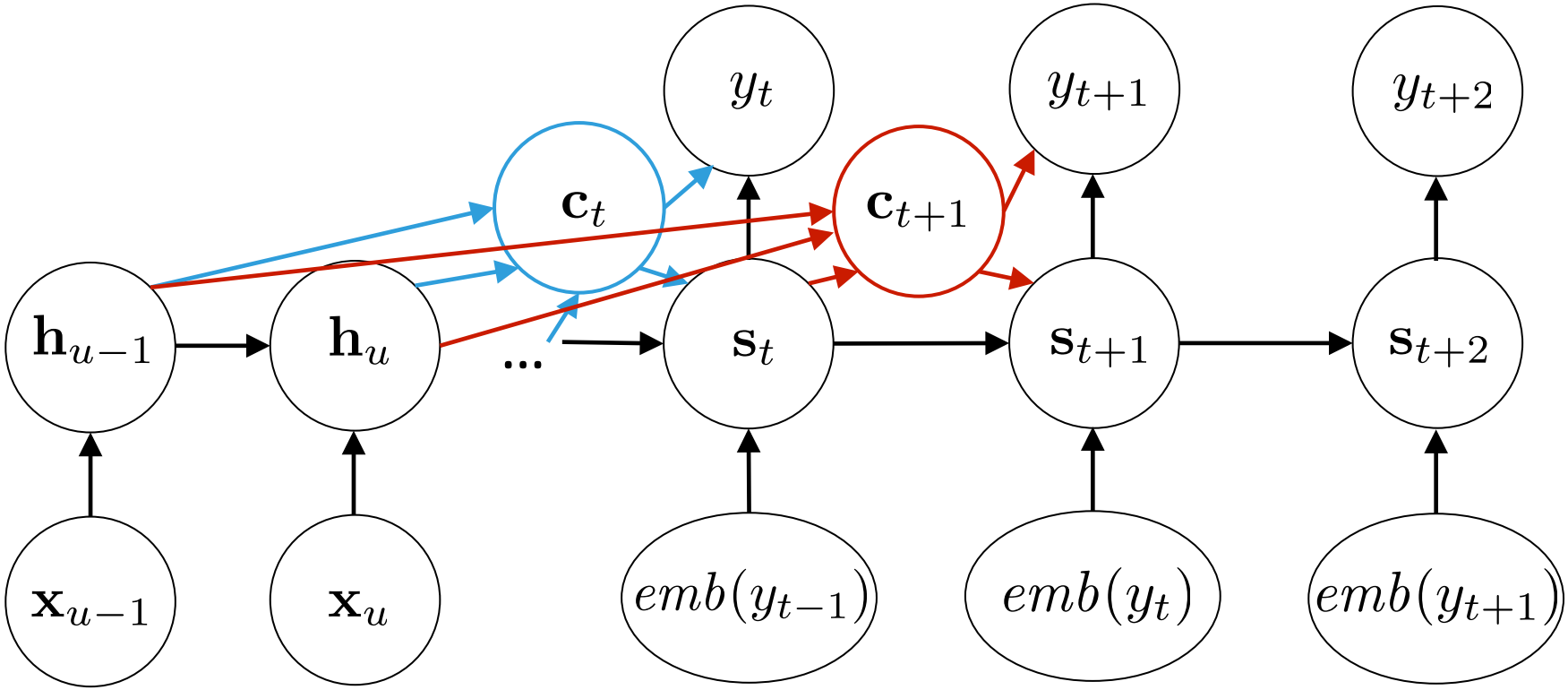
$$\mathbf{s}_t = \tanh\left(\mathbf{W}^{(y)} emb(y_{t-1}) + \mathbf{W}^{(s)} \mathbf{s}_{t-1} + \mathbf{W}^{(c)} \mathbf{c}_t + \mathbf{b}^{(s)}\right)$$



$$\mathbf{s}_t = \tanh\left(\mathbf{W}^{(y)} emb(y_{t-1}) + \mathbf{W}^{(s)}\mathbf{s}_{t-1} + \mathbf{W}^{(c)}\mathbf{c}_t + \mathbf{b}^{(s)}\right)$$

new prediction equation:

$$y_t = \underset{y \in \mathcal{O}}{\text{argmax}} \ \text{score}(emb(y), \mathbf{s}_t, \mathbf{c}_t)$$

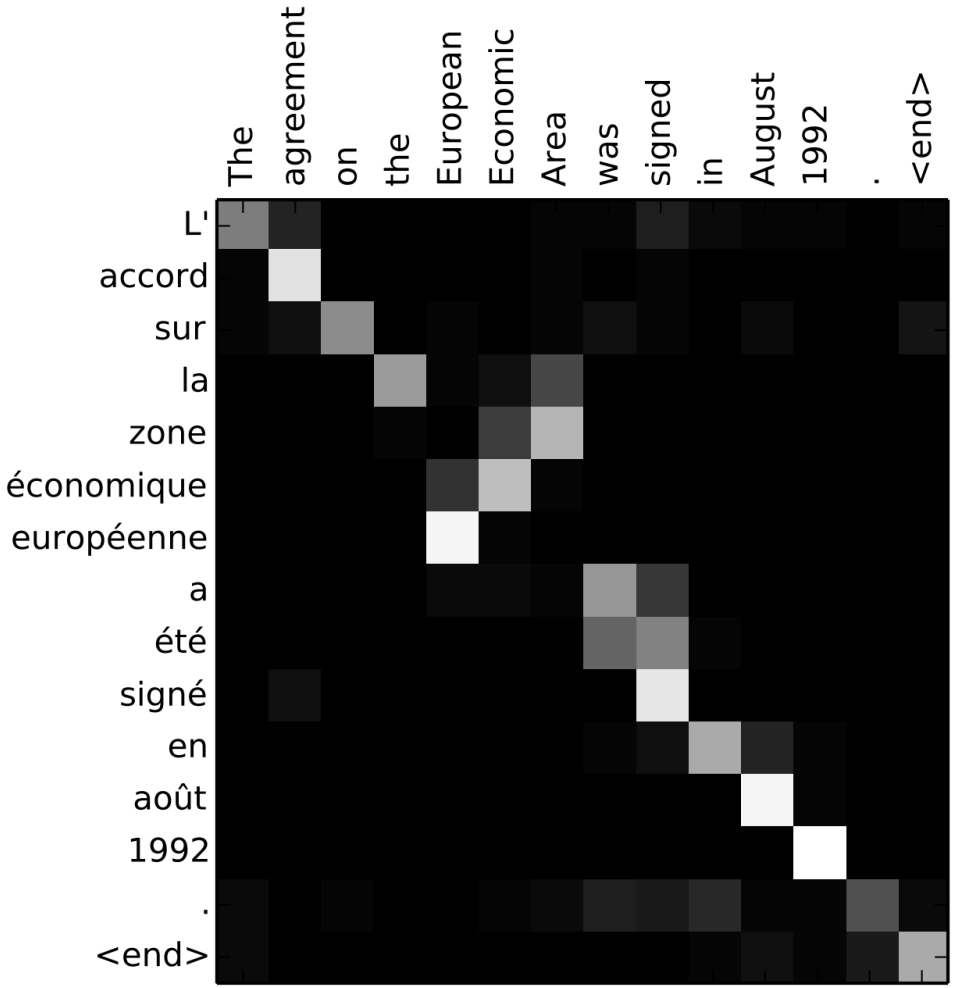


Notes

- while initially developed for machine translation, attention is useful for many other applications
  - speech recognition
  - image captioning
  - summarization
  - data-to-text generation
- many different kinds of attention; see Luong et al. (2015) for other variations and comparisons
- sometimes attention seems interpretable, but it can be misleading

Luong et al. (2015): *Effective Approaches to Attention-based Neural Machine Translation*

Attention in Machine Translation



Bahdanau et al. (2015): *Neural Machine Translation by Jointly Learning to Align and Translate*

Sentence Summarization

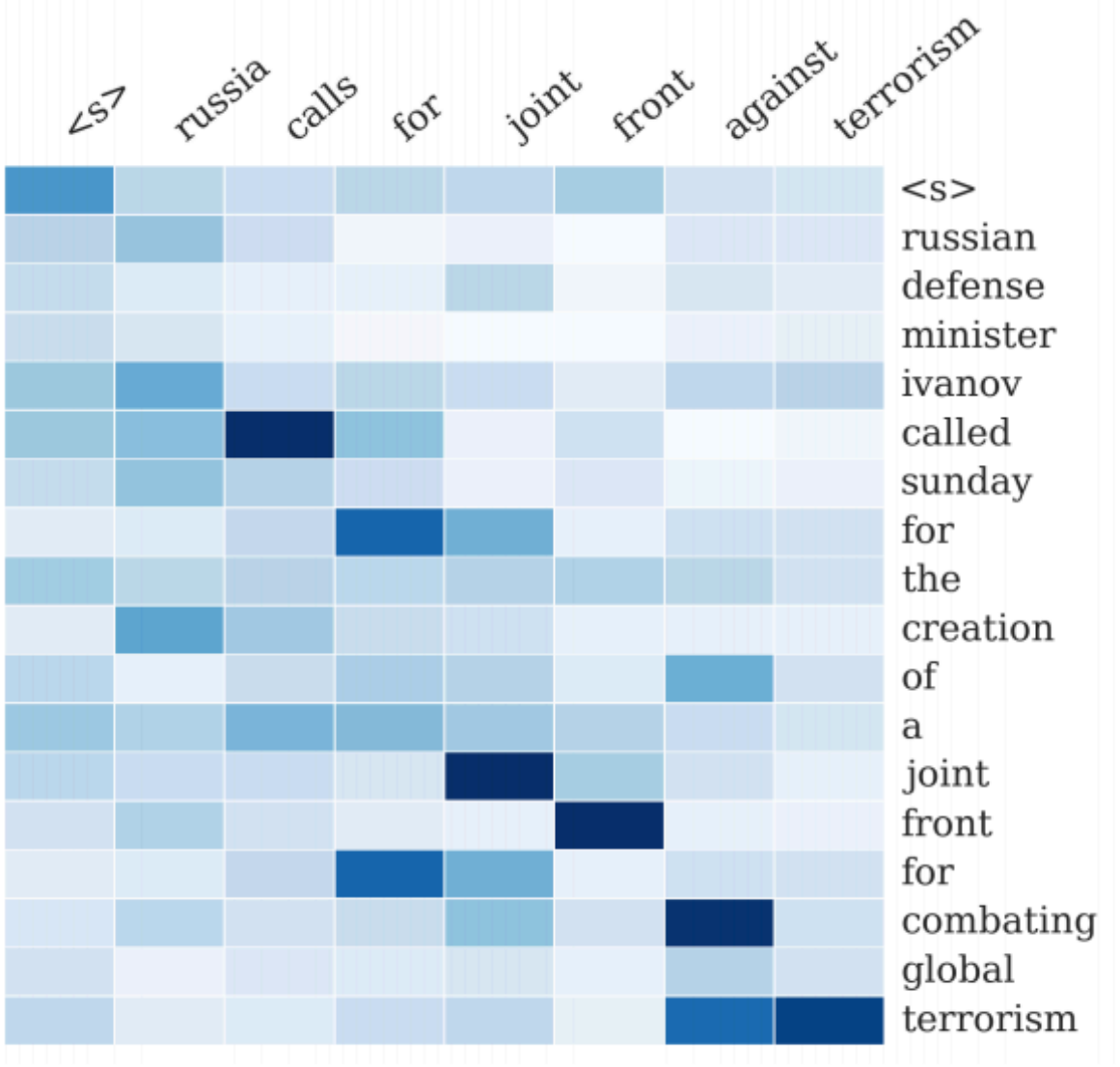


Figure 1: Example output of the attention-based summarization (ABS) system. The heatmap represents a soft alignment between the input (right) and the generated summary (top). The columns represent the distribution over the input after generating each word.

Rush et al. (2015): *A Neural Attention Model for Abstractive Sentence Summarization*

Natural Language Inference

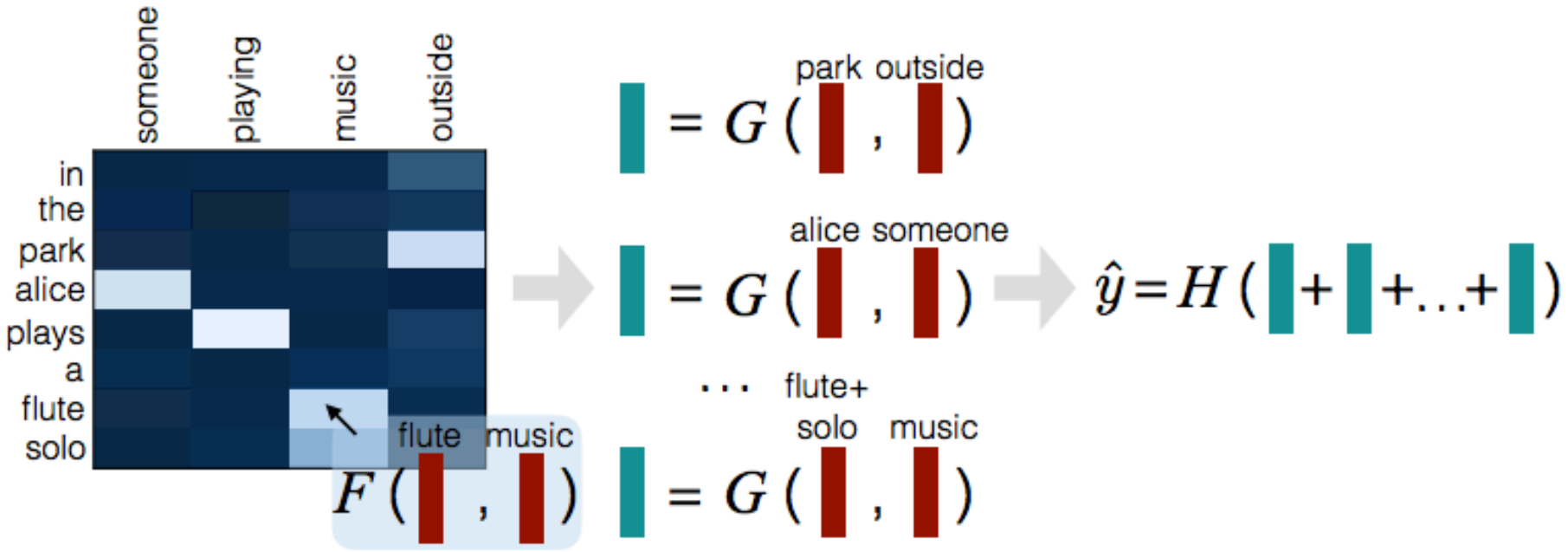
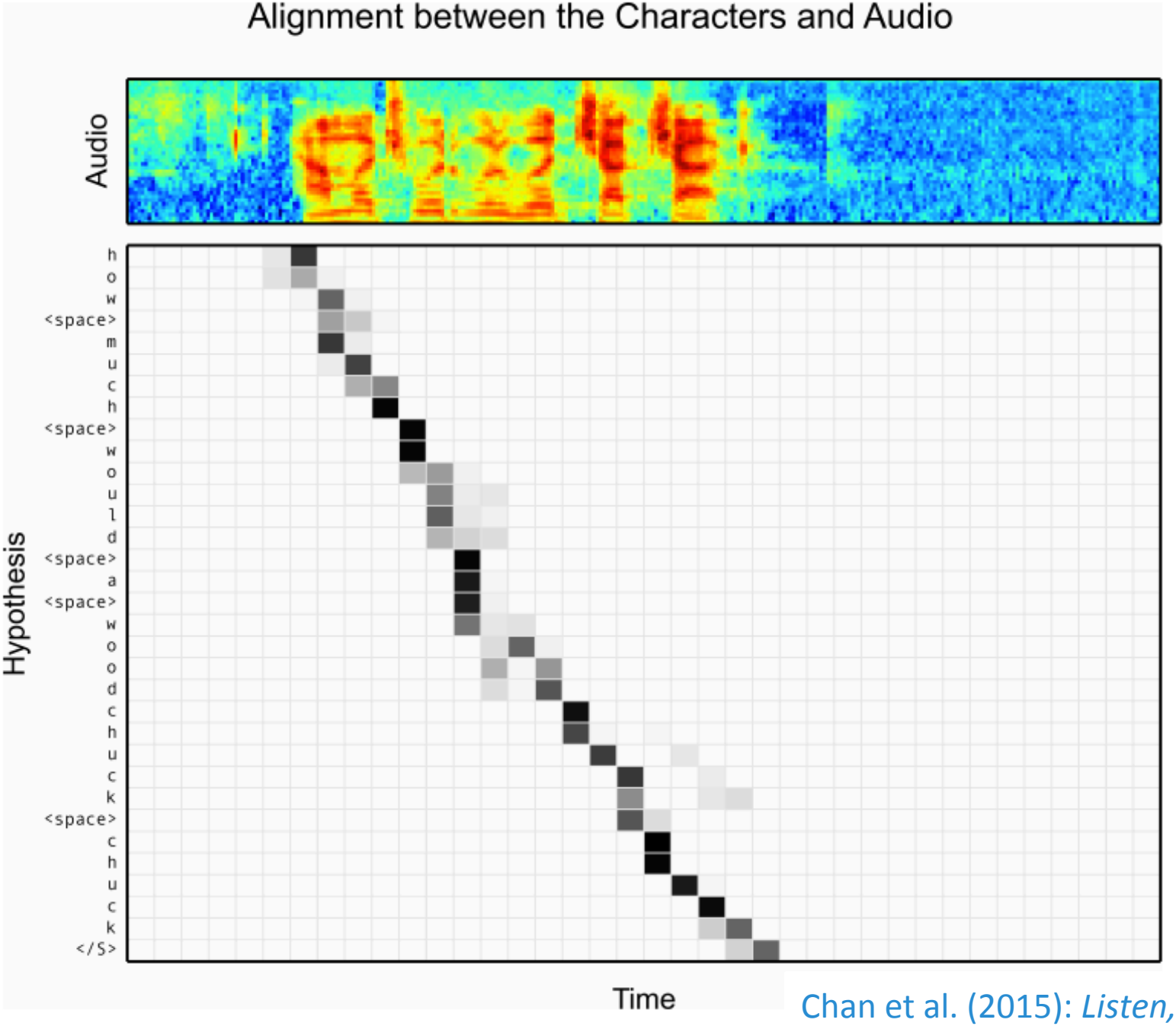


Figure 1: Pictoral overview of the approach, showing the *Attend* (left), *Compare* (center) and *Aggregate* (right) steps.

Parikh et al. (2016): *A Decomposable Attention Model for Natural Language Inference*

Speech Recognition



Chan et al. (2015): *Listen, Attend and Spell*

Image Captioning

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

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.

Xu et al. (2015): *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*