CS 109b

LSTMs and Translation

Alexander Rush (@harvardnlp)

Where we are going.

페루정부는유색인종과흑인의출입을달가와하지않는레스토랑에대한여러불만을들은뒤수도리마에있는유명레스토랑을닫았고, 추후통보가있을때까지임시휴업을명했다.
NMT: The government has closed a number of restaurants in the capital, Lima, after hearing complaints about a famous restaurant that does not welcome the entrance of people of color and blacks.
V8: The famous restaurant closed in Lima after hearing several complaints to the restaurant, which is not a happy access for people of color, black, Peru has ordered extra holiday until further notice.

Last Class:

Language Modeling, Embeddings, and RNNs

Aim: model the probability of

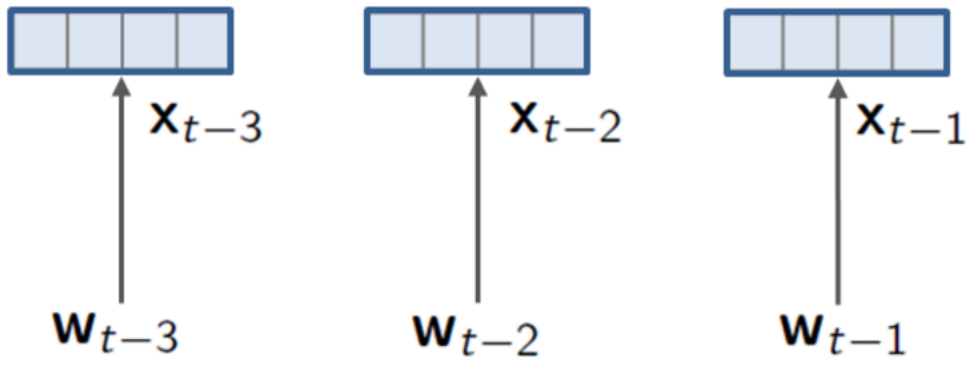
$$p(w_t|w_1, \ldots, w_{t-1})$$

Approach: utilize a neural network to compute probabilities.

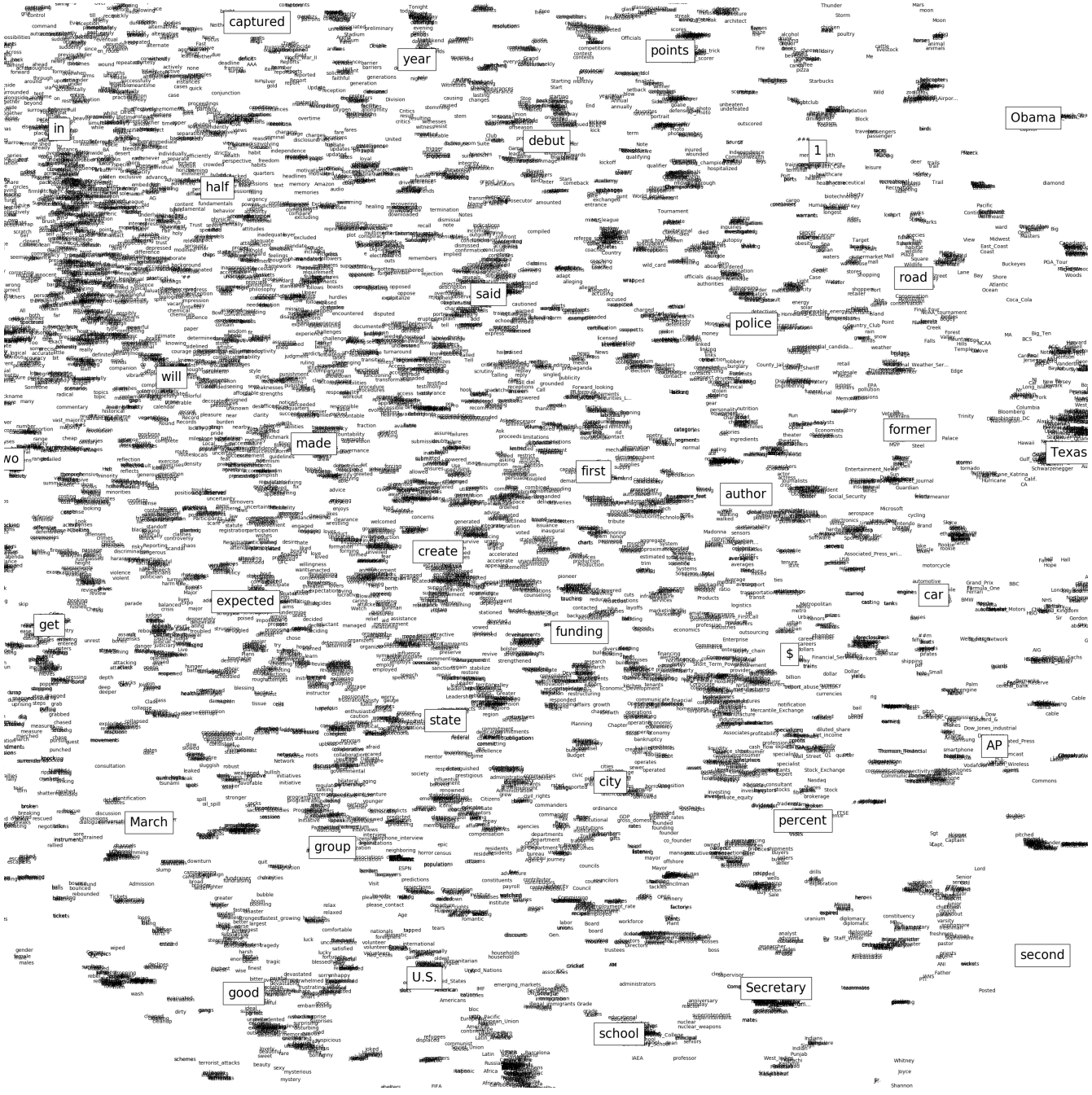
RNN For Language Modeling

Embeddings	words	$\Rightarrow$	word vectors
RNNs	word vectors	$\Rightarrow$	hidden states
Softmax	hidden states	$\Rightarrow$	word prediction

Words Embeddings

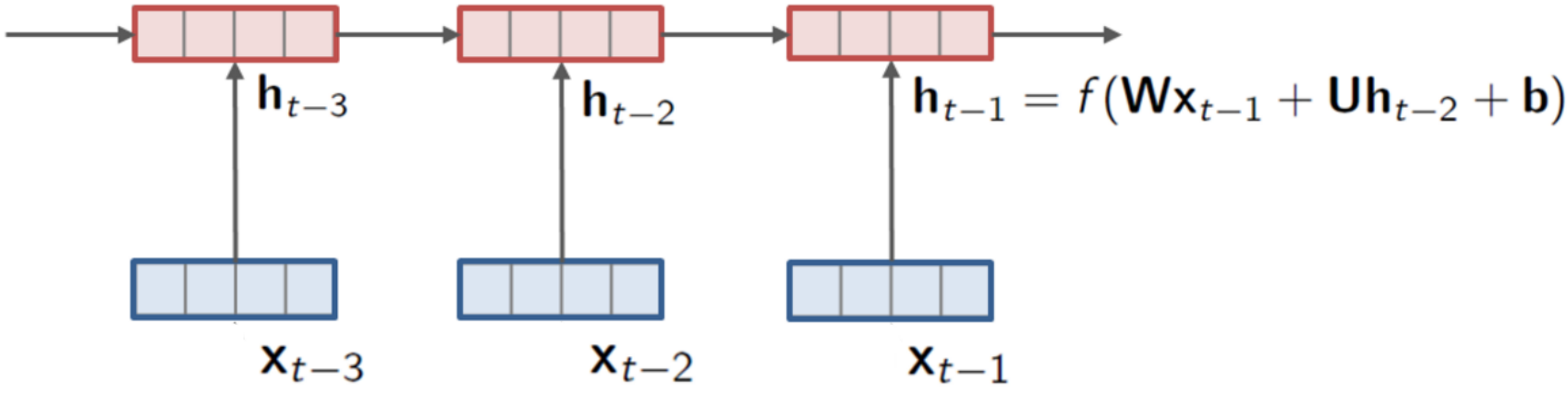


- Map words to vector space, just as before.



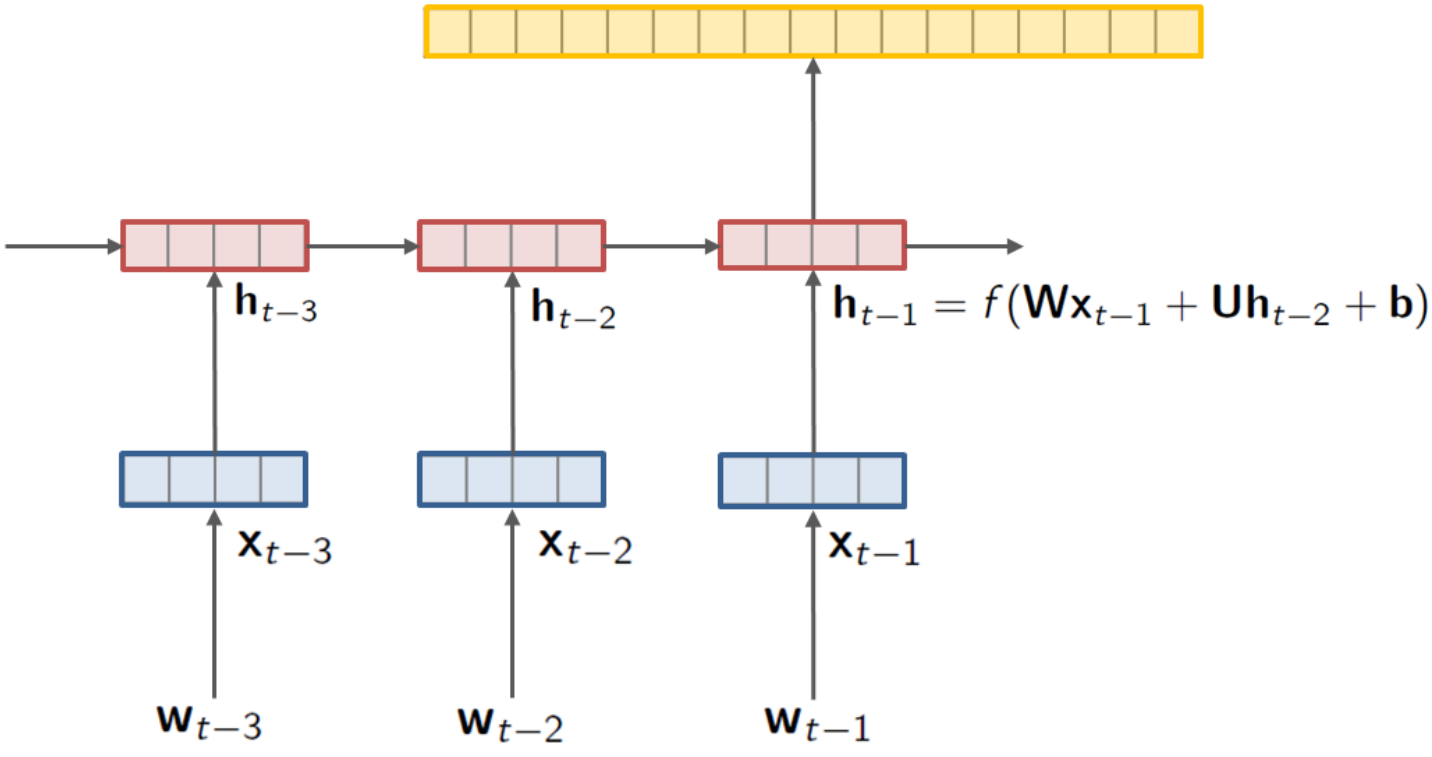
[Words Vectors]

RNN



- Apply recurrent neural network over vector space of words to create hidden states.

Softmax



- Compute softmax over all possible next words to predict.

$$p(w_t \mid w_1, \ldots, w_{t-1}) = \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{b})_w$$

Contents

RNN Mathematically

Let $h_0 \leftarrow 0$,

$$h_1 \leftarrow \textbf{RNN}(h_0, \mathbf{x}_1; \theta)$$

$$h_2 \leftarrow \textbf{RNN}(h_1, \mathbf{x}_2; \theta)$$

$$h_3 \leftarrow \textbf{RNN}(h_2, \mathbf{x}_3; \theta)$$

$$\vdots$$

$$h_n \leftarrow \textbf{RNN}(h_{n-1}, \mathbf{x}_n; \theta)$$

Where RNN is a neural network layer with the same weights θ applied at each time step, for instance:

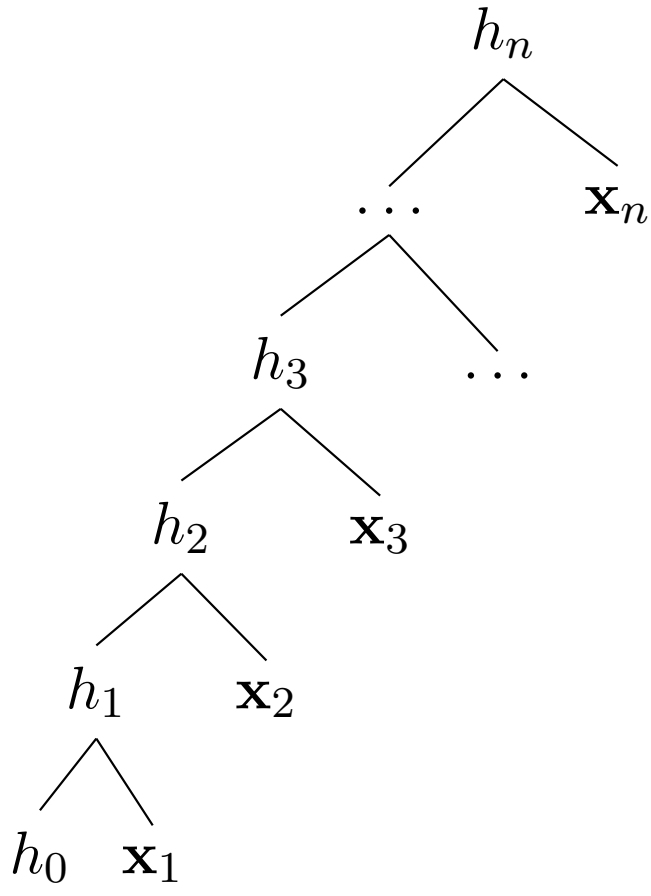
$$\textbf{RNN}(h, \mathbf{x}) = \tanh(\mathbf{W}h + \mathbf{U}\mathbf{x})$$

What types of relationships might the network to learn?

$$
\begin{aligned}
\textbf{ExpAvg}(h, \mathbf{x}) &= \tanh(\alpha \mathbf{x} + (1-\alpha)h) \\
\textbf{LastWord}(h, \mathbf{x}) &= \tanh(\mathbf{x}) \\
\textbf{ConcatLast2}(h, \mathbf{x}) &= \tanh([\mathbf{U}\mathbf{x}, \mathbf{W}\boldsymbol{h}])
\end{aligned}
$$

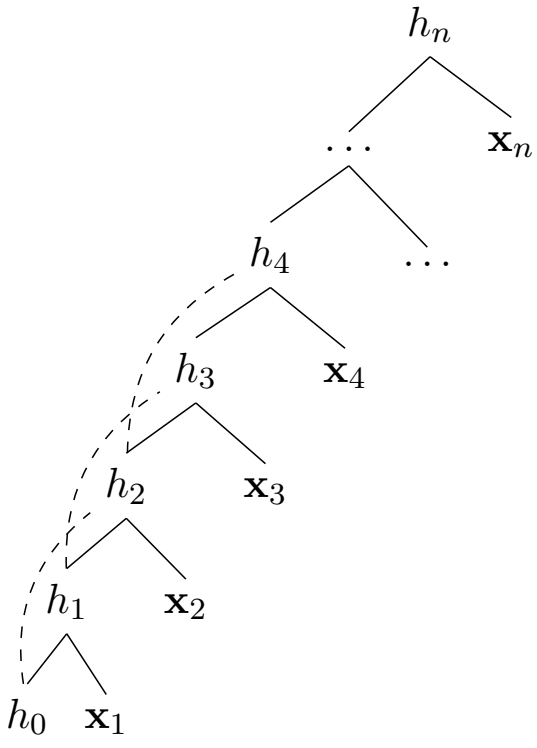
(Formally, a non-linear IIR transformation.)

Issues with Basic RNN



Very deep network. Can be hundreds of layers of non-linear transformations.

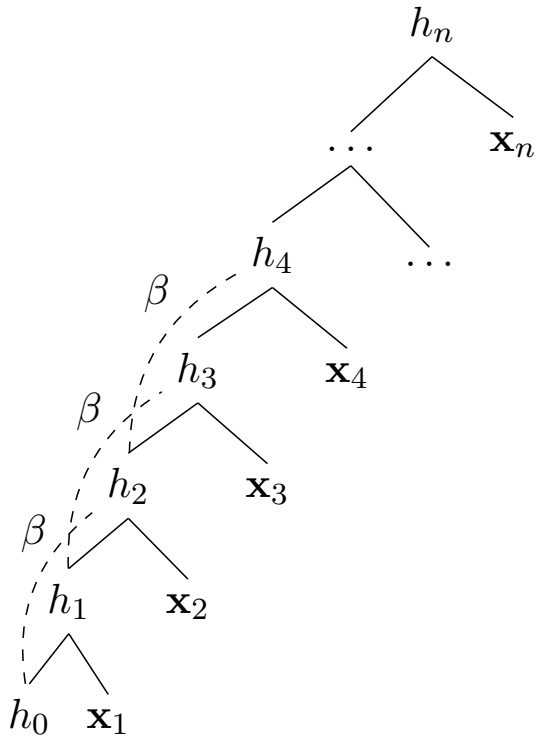
RNN with Skip-Connections



$$\textbf{RNN-SKIP}(h, \mathbf{x}) = \frac{1}{2} \times h + \frac{1}{2} \times \tanh(\mathbf{W}h + \mathbf{U}\mathbf{x})$$

- Skip-connections **short cut** the multiplicative transformation.

RNN with Dynamic Skip-Connections

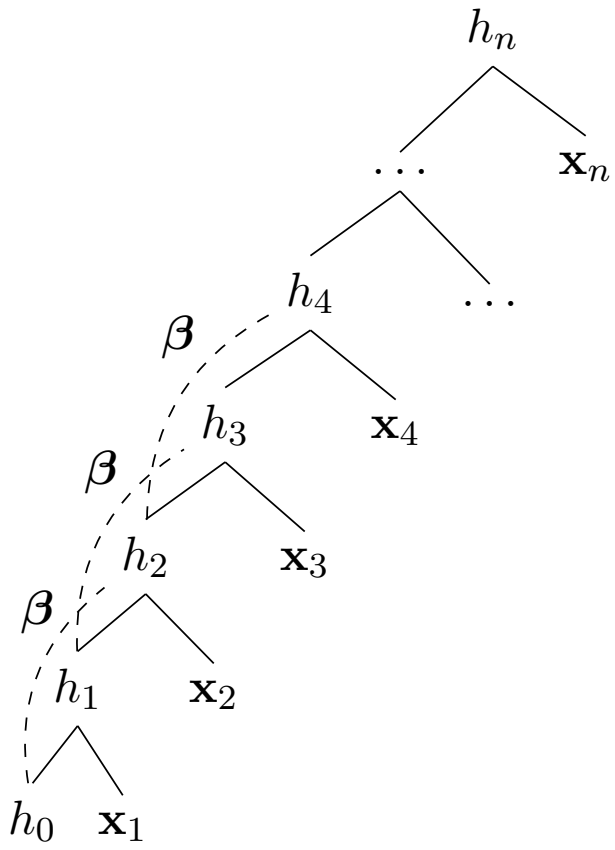


$$\textbf{RNN-DSKIP}(h, \mathbf{x}) = \beta \times h + (1 - \beta) \times \tanh(\mathbf{W}h + \mathbf{U}\mathbf{x})$$

$$\beta = NN(h, \mathbf{x})$$

- Dynamic skip-connections let the network to control **when** it short cuts.

RNN with Gated Skip-Connections

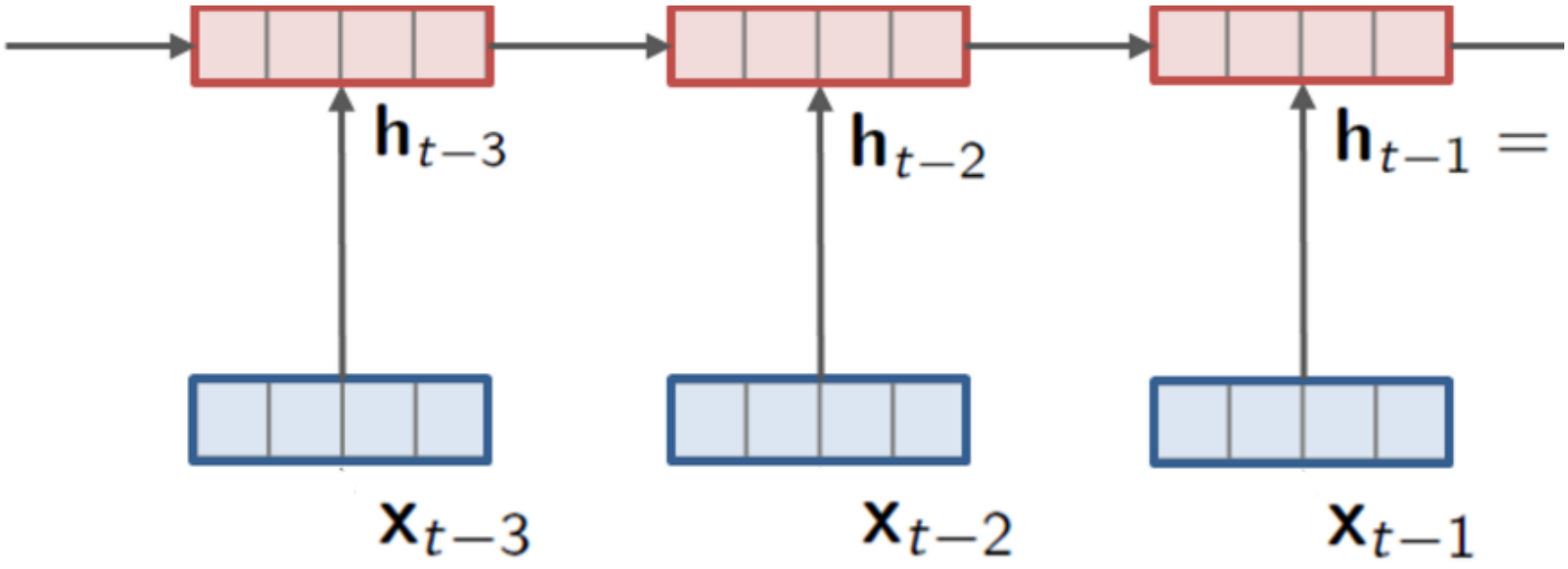


$$\textbf{RNN-GATE}(h, \mathbf{x}) = \boldsymbol{\beta} \odot h + (1 - \boldsymbol{\beta}) \odot \tanh(\mathbf{W}h + \mathbf{U}\mathbf{x})$$

$$\boldsymbol{\beta} = NN(h, \mathbf{x})$$

- Gates let the network control **when** and **what** it shortcuts.

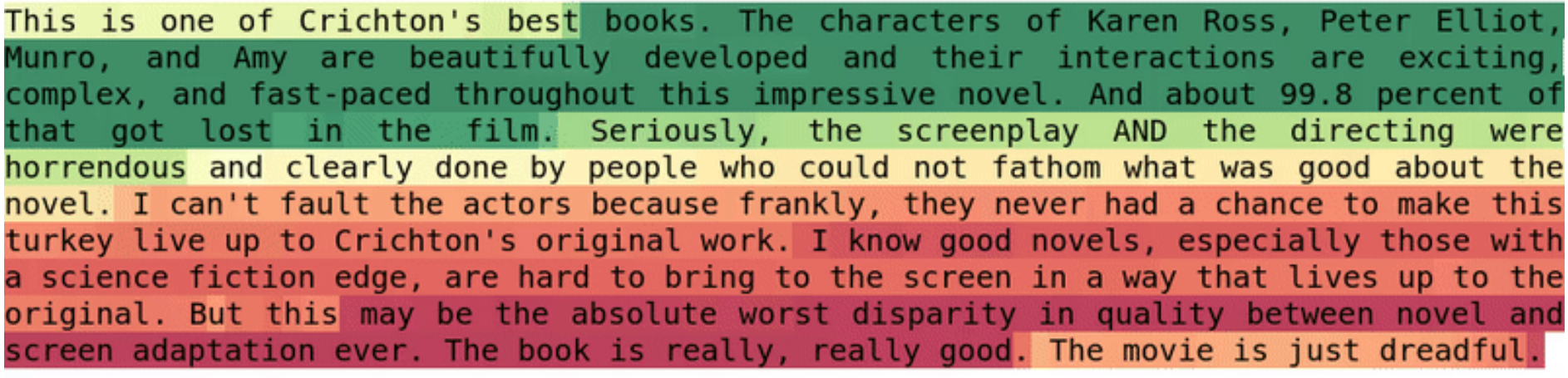
Why gating?



- Gated RNN can retain the value of important dimensions, and alter others.
- This allows the system to have "memory" states.

Sentiment Prediction

It has been observed that language models can pick up on sentiment properties. This image shows how one dimension retains sentiment information over time by not updating.



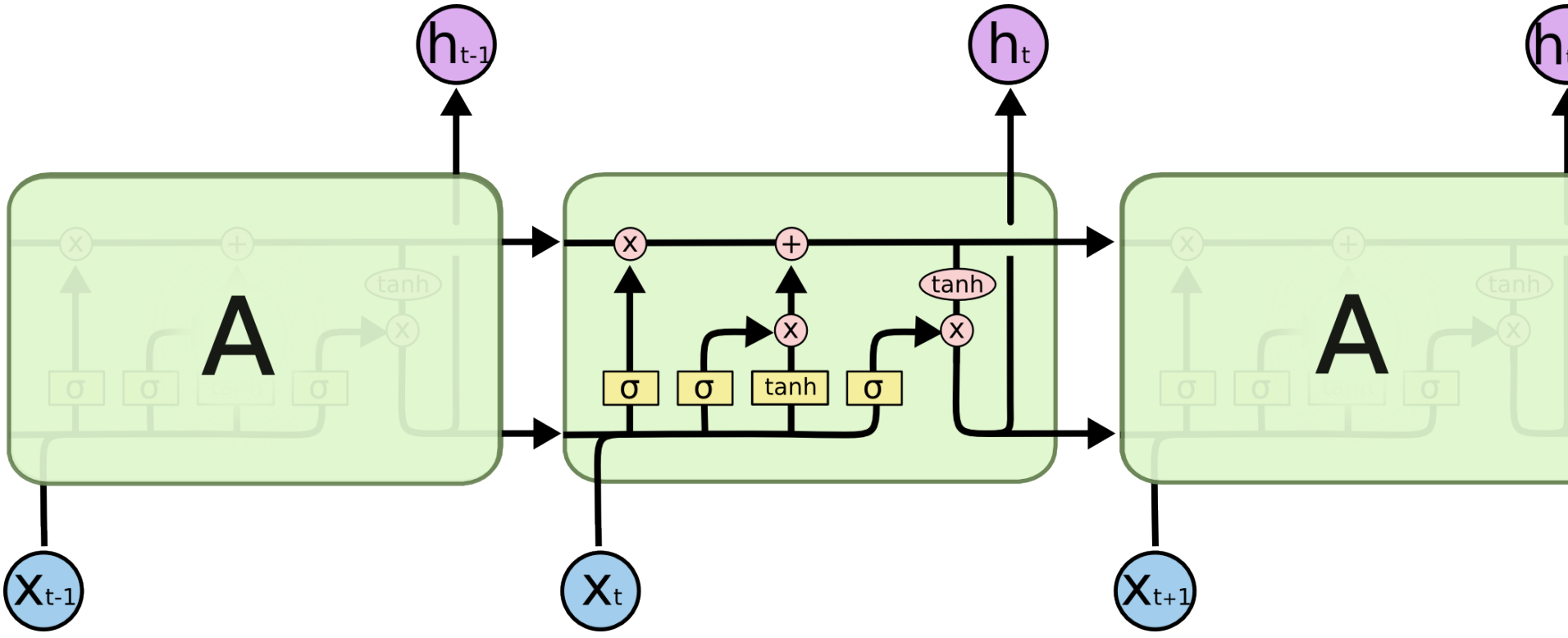
Long Short-Term Memory

The most famous gated RNN is the LSTM. which uses three different types of gates.

It is complicated to say the least:

$$
\begin{aligned}
\textbf{LSTM}(h_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, h_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \mathbf{i} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
h_i &= \tanh(\mathbf{c}_i) \odot \mathbf{o} \\
\mathbf{i} &= \tanh(\mathbf{W}^{xi}\mathbf{x} + \mathbf{W}^{hi}h_{i-1} + \mathbf{b}^i) \\
\mathbf{j} &= \sigma(\mathbf{W}^{xj}\mathbf{x} + \mathbf{W}^{hj}h_{i-1} + \mathbf{b}^j) \\
\mathbf{f} &= \sigma(\mathbf{W}^{xf}\mathbf{x} + \mathbf{W}^{hf}h_{i-1} + \mathbf{b}^f) \\
\mathbf{o} &= \tanh(\mathbf{W}^{xo}\mathbf{x} + \mathbf{W}^{ho}h_{i-1} + \mathbf{b}^o)
\end{aligned}
$$

Schematic View



What you should know about LSTMs.

- Very effective at language modeling.

Language Model	Perplexity
Count-Based	141
LSTM	78

- Seem to better capture long-term dependencies (when necessary) due to gating.
- If you need to use, details can be abstracted away.

```
model.add(LSTM(100))
```

Example 1: Synthetic (Finite-State) Language

alphabet: ( ) 0 1 2 3 4

corpus: (1 (2) ()) 0 (((3)) 1)

- Numbers are randomly generated, must match nesting level.
- Train a predict-next-word language model (decoder-only).

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

[Parens Example]

Example 2: Real Language

alphabet: all english words

corpus: Project Gutenberg Children's books

- Train a predict-next-word language model (decoder-only).

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

[LM Example]

Contents

LSTM Language Models

We can now estimate,

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

So what. This alone is not inherently interesting (even to me).

However, what if instead we compute a *conditional* language model.

$$p(w_t|w_1, \ldots, w_{t-1}, c)$$

Where c is the object that we *want to talk about*.

Machine Translation

- Supervised machine learning problem, given foreign sentence predict translation.
- Data consists of pairs of French/English data (or any other language pair)
- Classical problem, goes back to the 1950s.
- Modern incarnation is very data driven, tens of millions of sentence pairs.

Back to the Shannon Game

THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG READING LAMP ON THE DESK SHED GLOW ON POLISHED ___

But now also assume you know a Spanish version of the sentence:

la habitacin no era muy encender una pequea lmpara de lectura oblonga en el resplandor caseta de escritorio en vidrio pulido

Call this a *conditional* language modeling problem.

$$p(w_t|w_1, \ldots, w_{t-1}, c)$$

Back to the Shannon Game

THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG READING LAMP ON THE DESK SHED GLOW ON POLISHED ___

But now also assume you know a Spanish version of the sentence:

la habitacin no era muy encender una pequea lmpara de lectura oblonga en el resplandor caseta de escritorio en vidrio pulido

Call this a *conditional* language modeling problem.

$$p(w_t|w_1, \ldots, w_{t-1}, c)$$

Conditional LM

Need a neural network to compute the softmax of the next word.

$$p(w_t|w_1, \ldots, w_{t-1}, c)$$

But we now have a general purpose tool for computing a neural network on sequences, the LSTM.

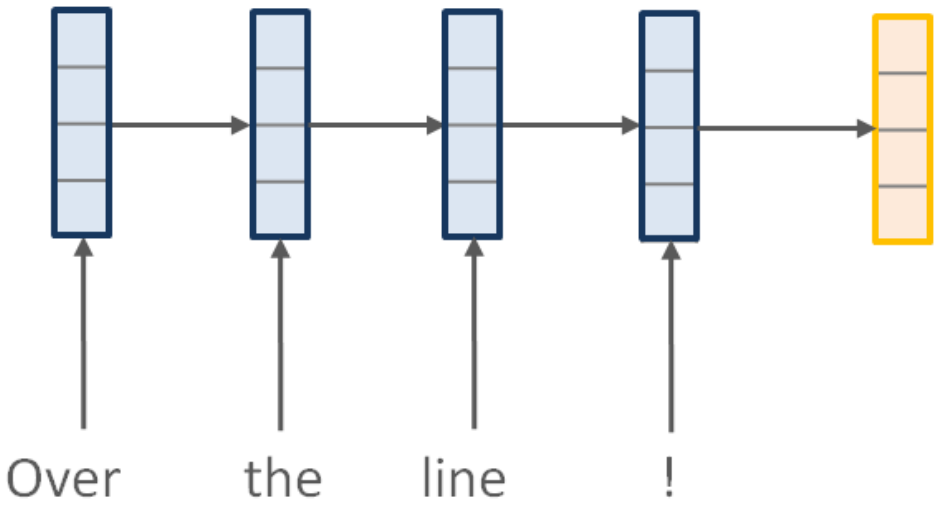
Sequence-to-Sequence

1. "Read" source language sentence with LSTM.
2. Construct a final source vector representation.
3. Utilize this vector as part of an LSTM language model.

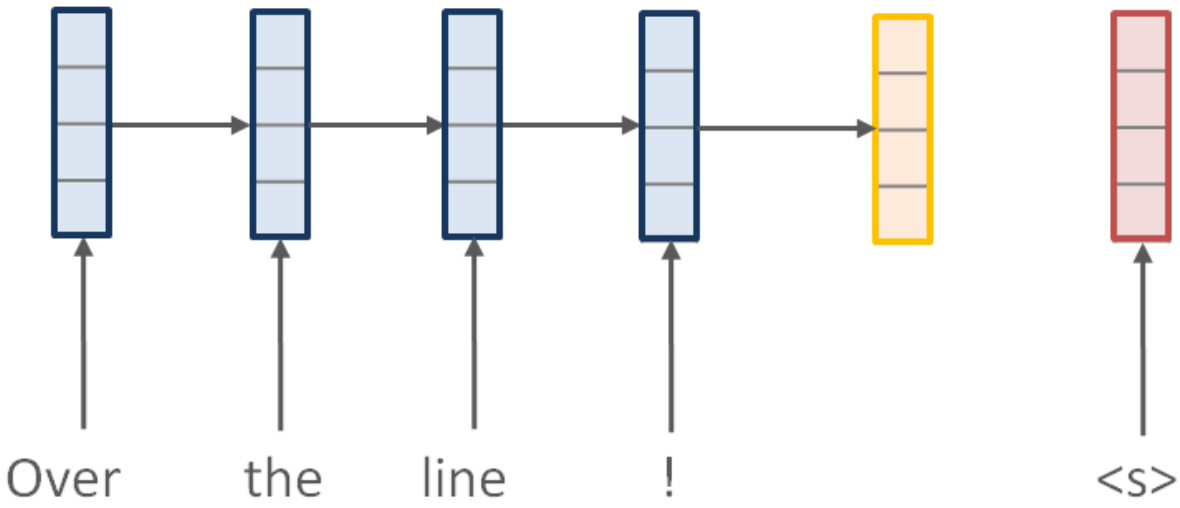
Example: Neural Machine Translation

Over the line !

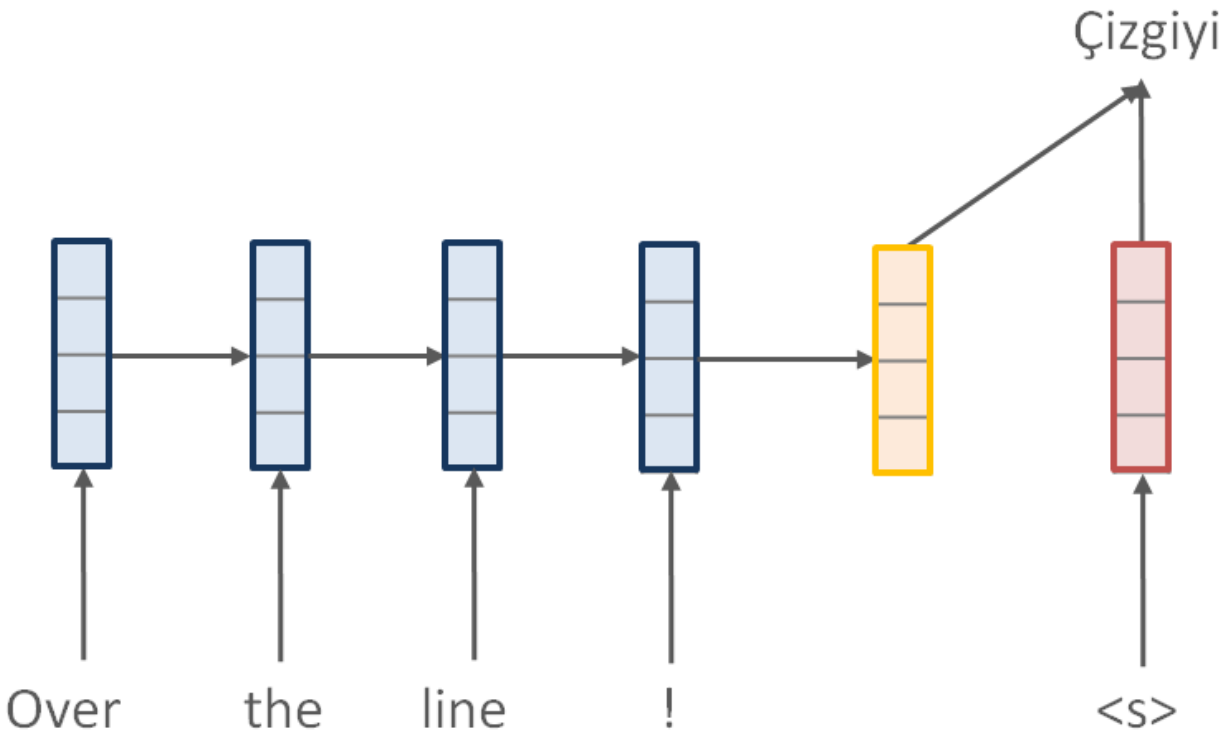
Example: Neural Machine Translation



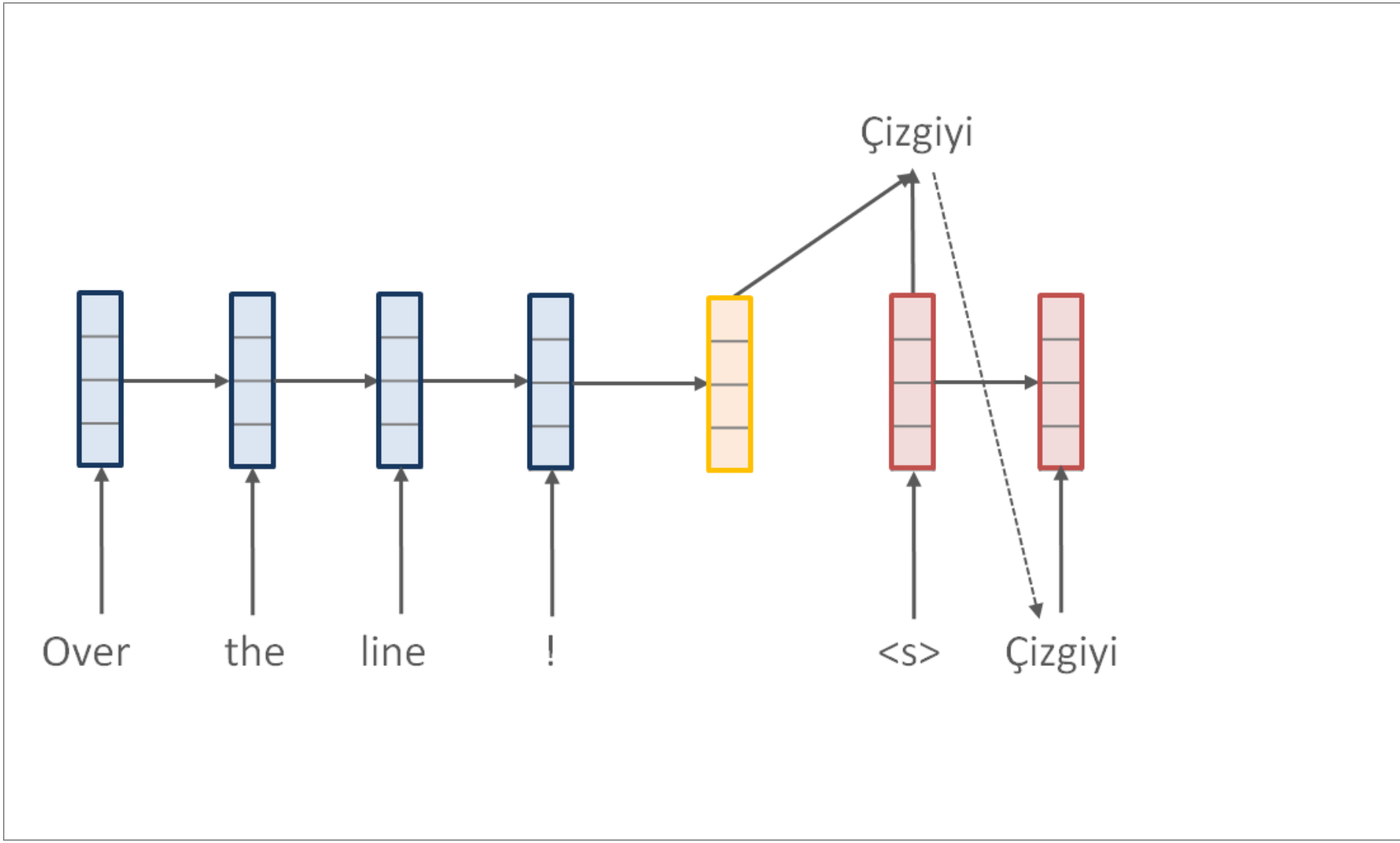
Example: Neural Machine Translation



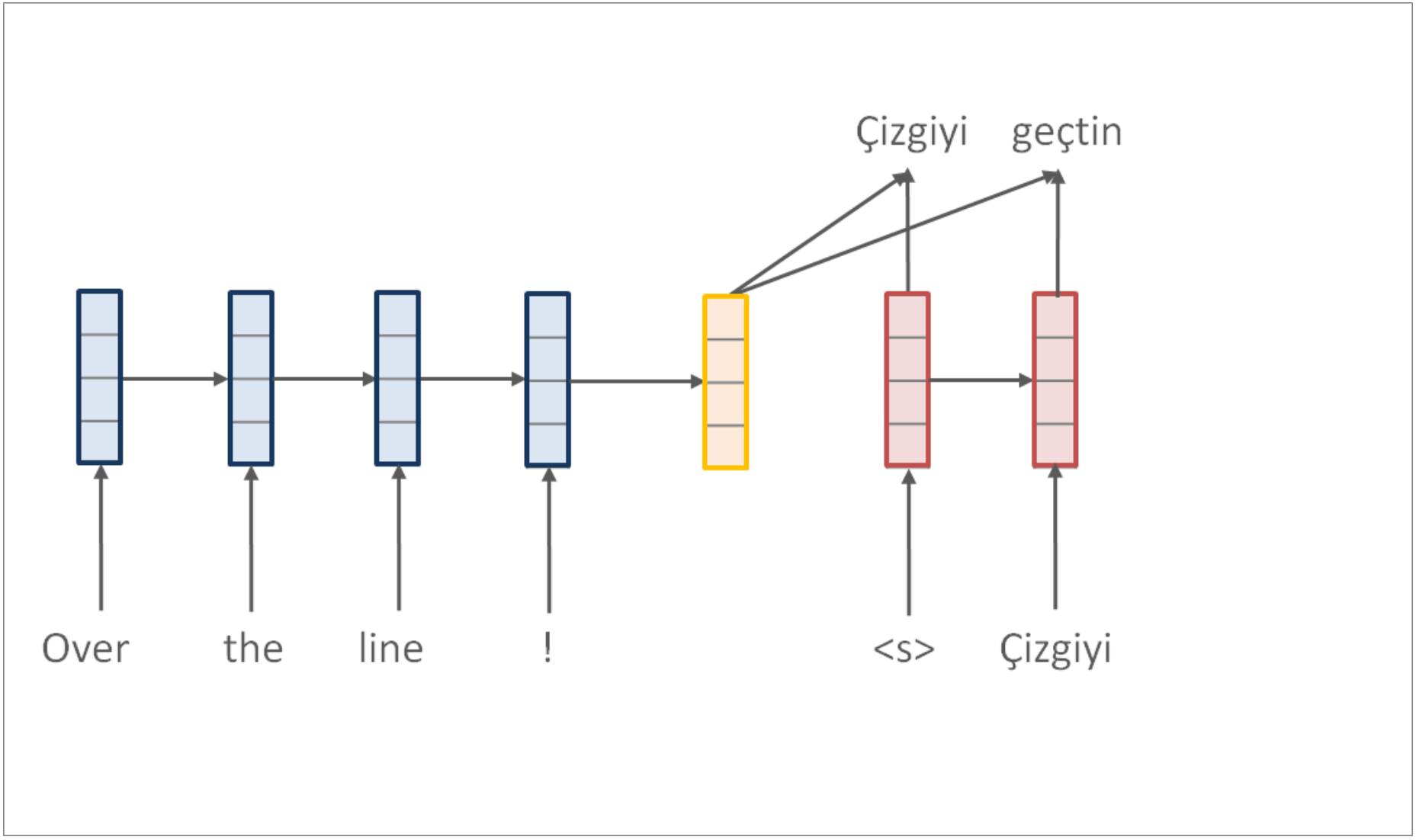
Example: Neural Machine Translation



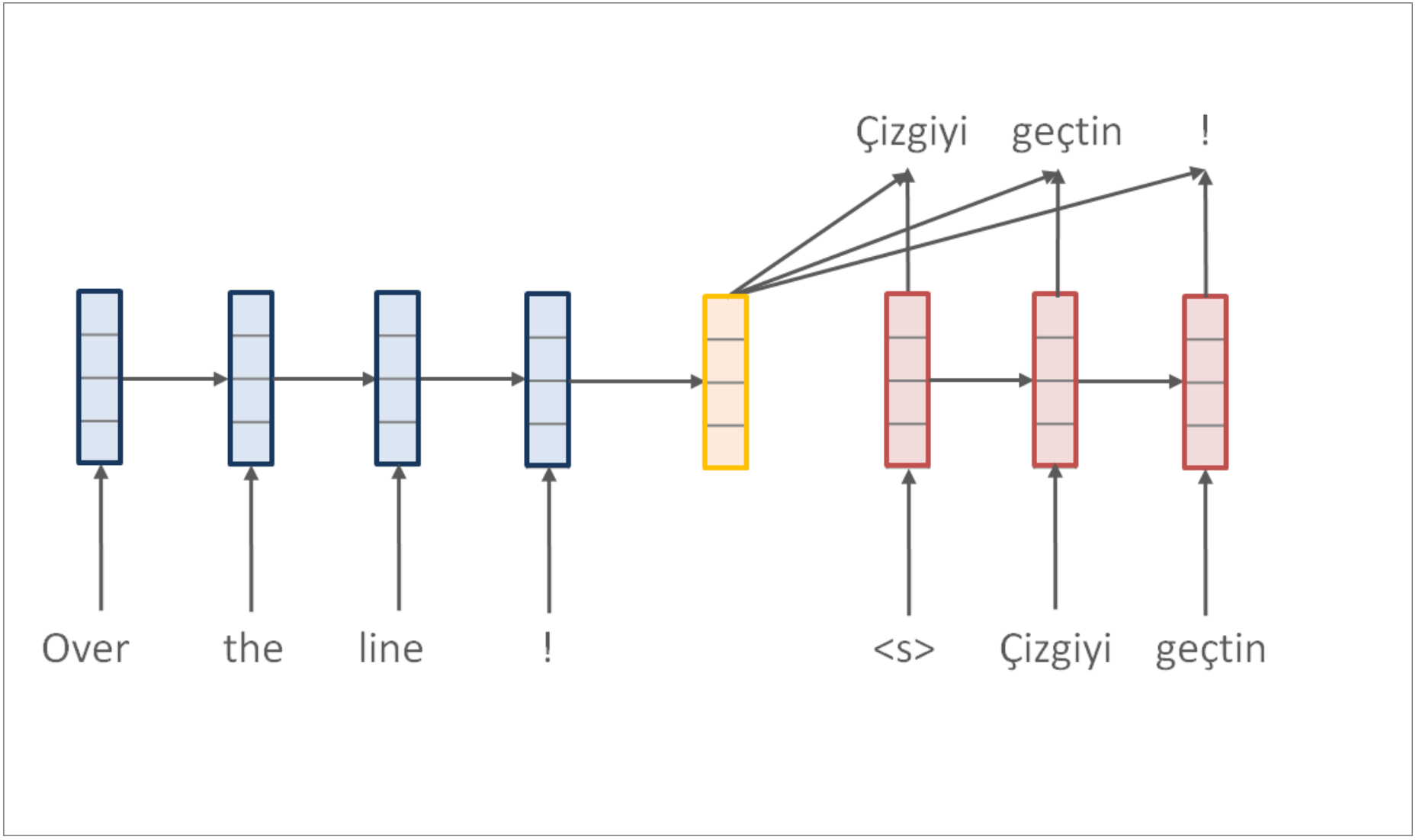
Example: Neural Machine Translation



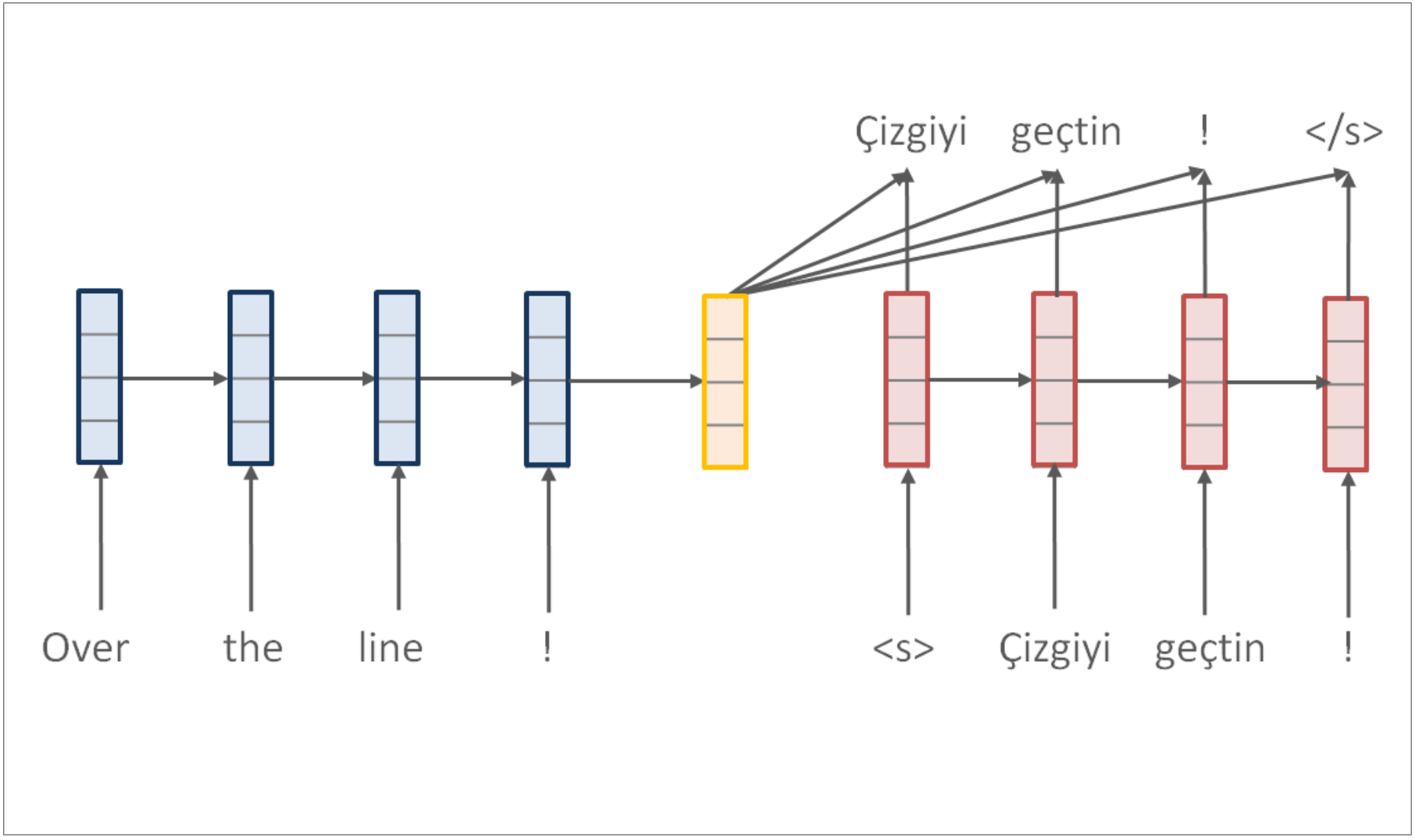
Example: Neural Machine Translation



Example: Neural Machine Translation



Example: Neural Machine Translation



Sequence-to-Sequence Translation

Simple idea, amazing it works at all.

Caveats:

- Needed lots of training data (millions of sentences).
- Needed very large models (1000 hidden dimensions, 4 LSTMs stacked).
- Results trailed traditional methods for translation.

Argument:

- Hard to fit entire sentence in a single large vector.

Contents

Neural Attention

Input (Foreign sentence)

Memory-Bank Encoder

Encoder(input) $= h_1, h_2, \ldots, h_T$

Attention Distribution
"where"

Context Vector
"what"

Decoder

Neural Attention

Input (Foreign sentence)

Memory-Bank Encoder

Encoder(input) $= h_1, h_2, \ldots, h_T$

Attention Distribution
"where"

Context Vector
"what"

Decoder

Neural Attention

Input (Foreign sentence)

Memory-Bank Encoder

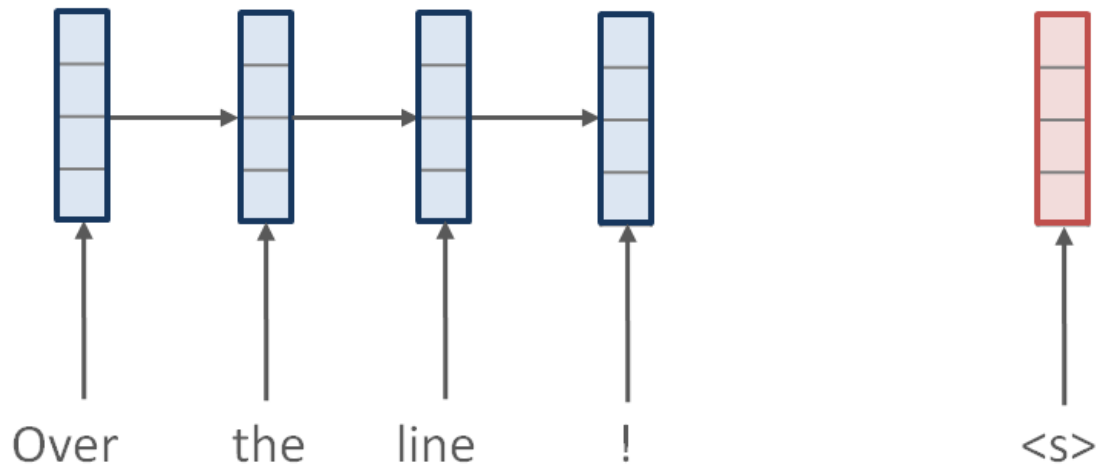
Encoder(input) $= h_1, h_2, \ldots, h_T$

Attention Distribution
"where"

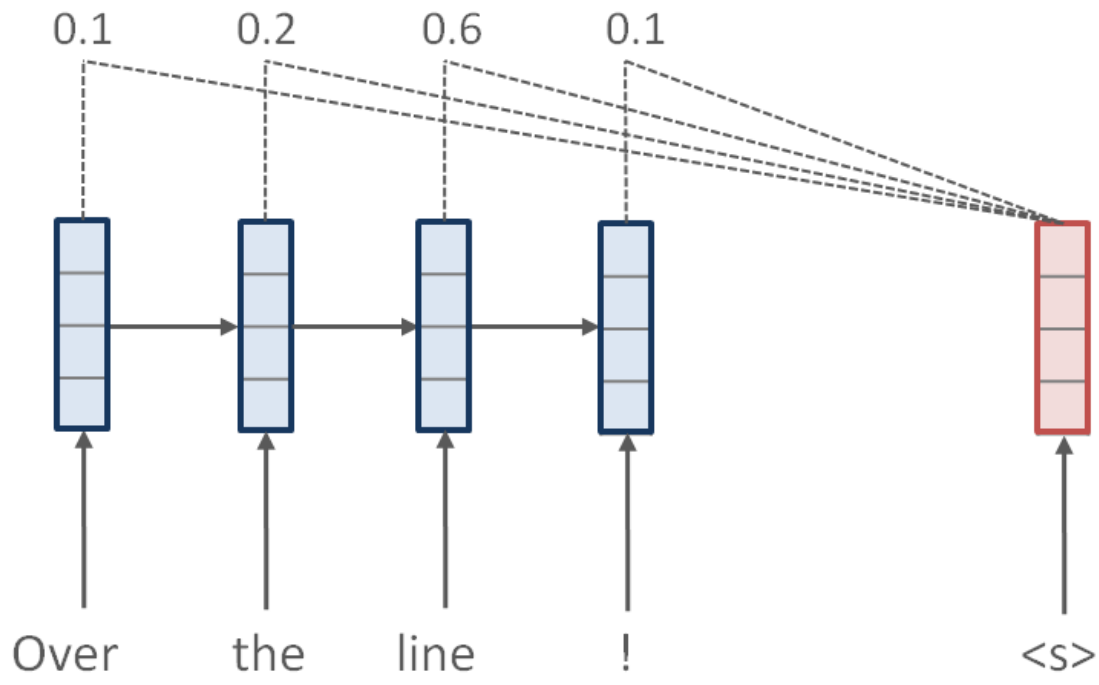
Context Vector
"what"

Decoder

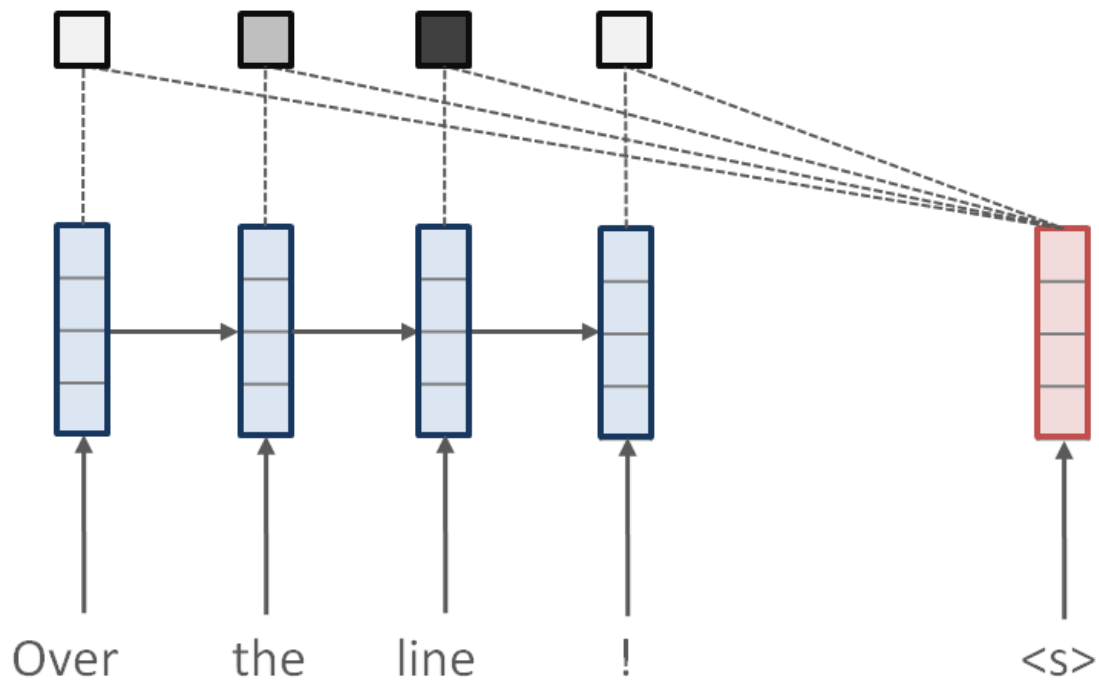
Attention-based Neural Machine Translation



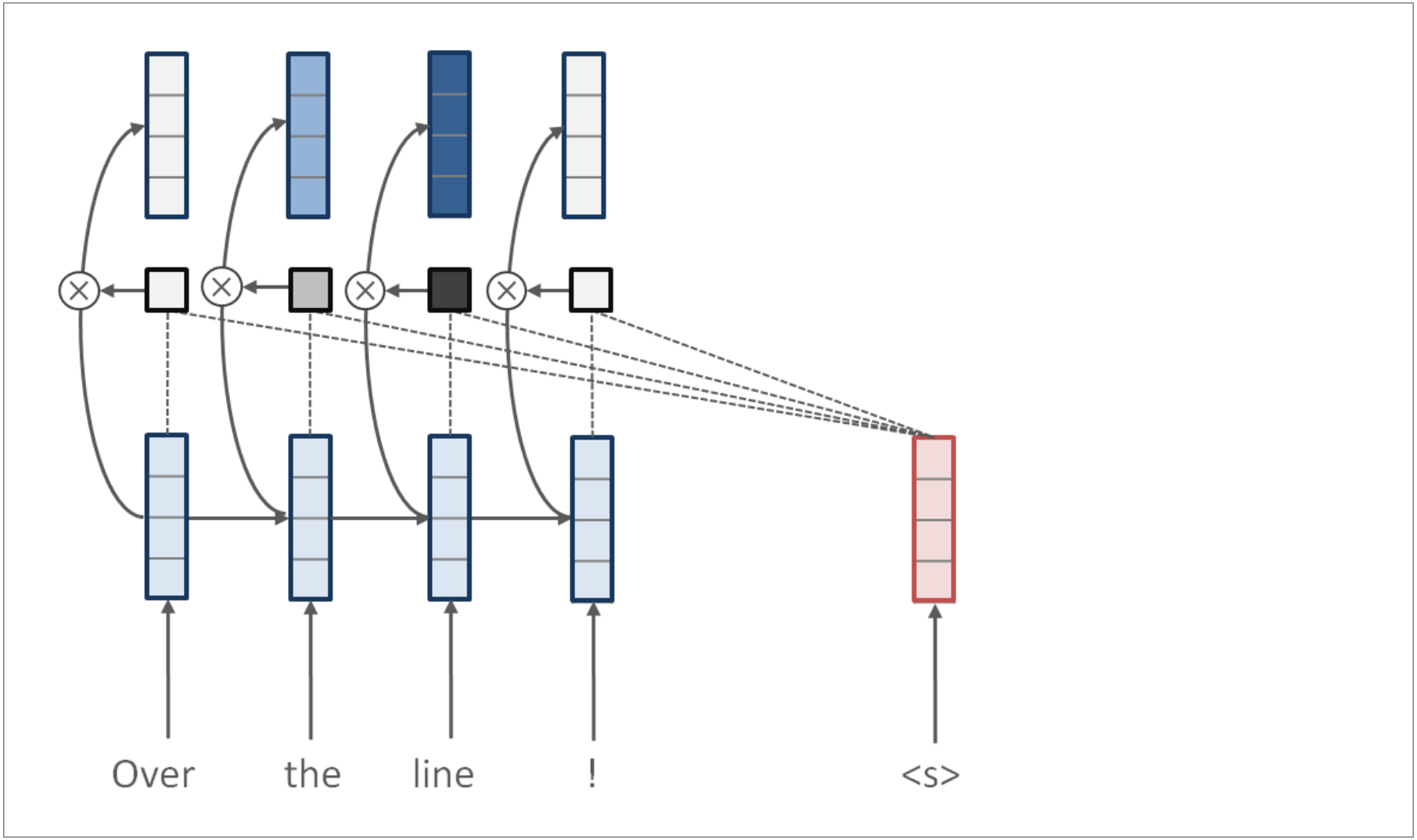
Attention-based Neural Machine Translation



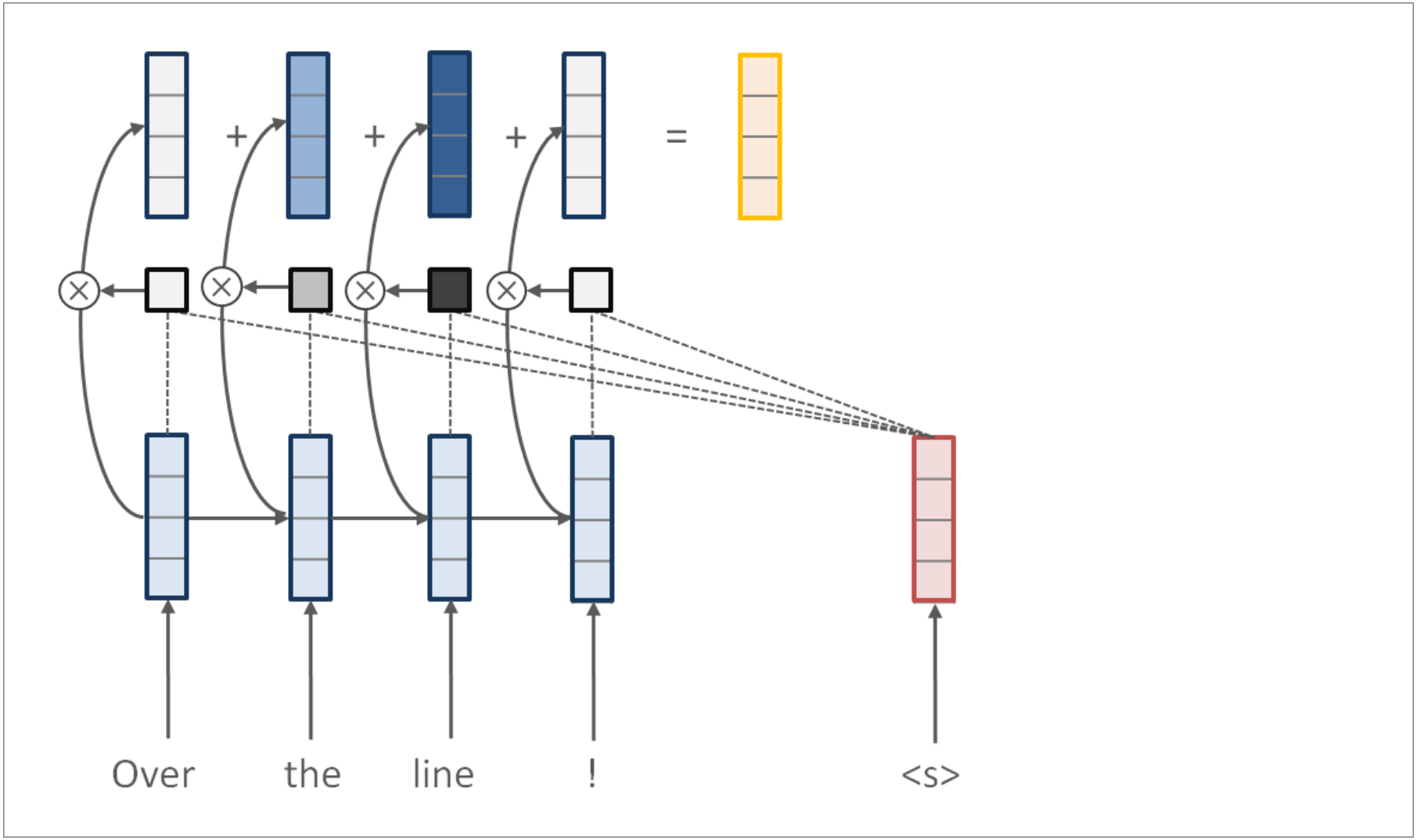
Attention-based Neural Machine Translation



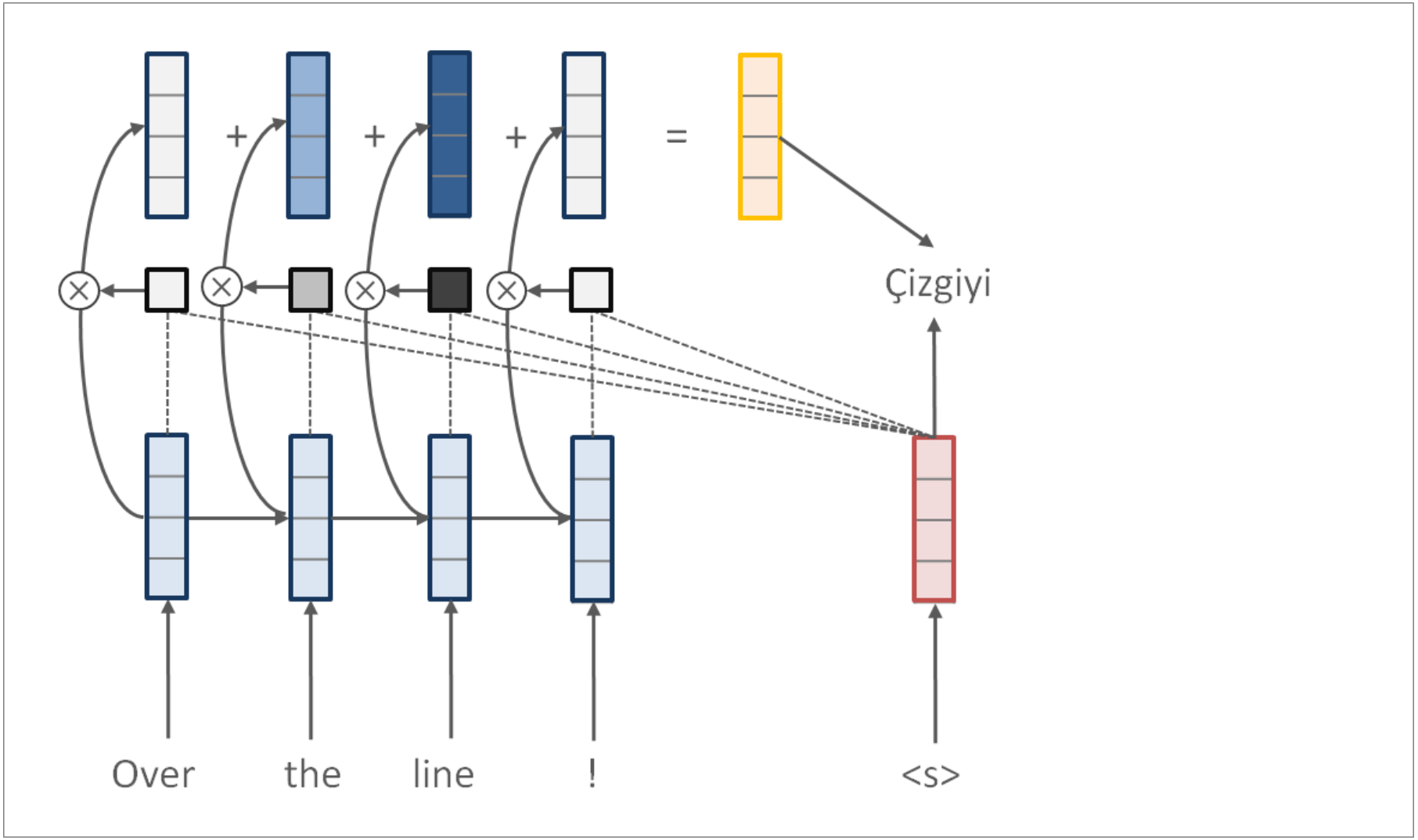
Attention-based Neural Machine Translation



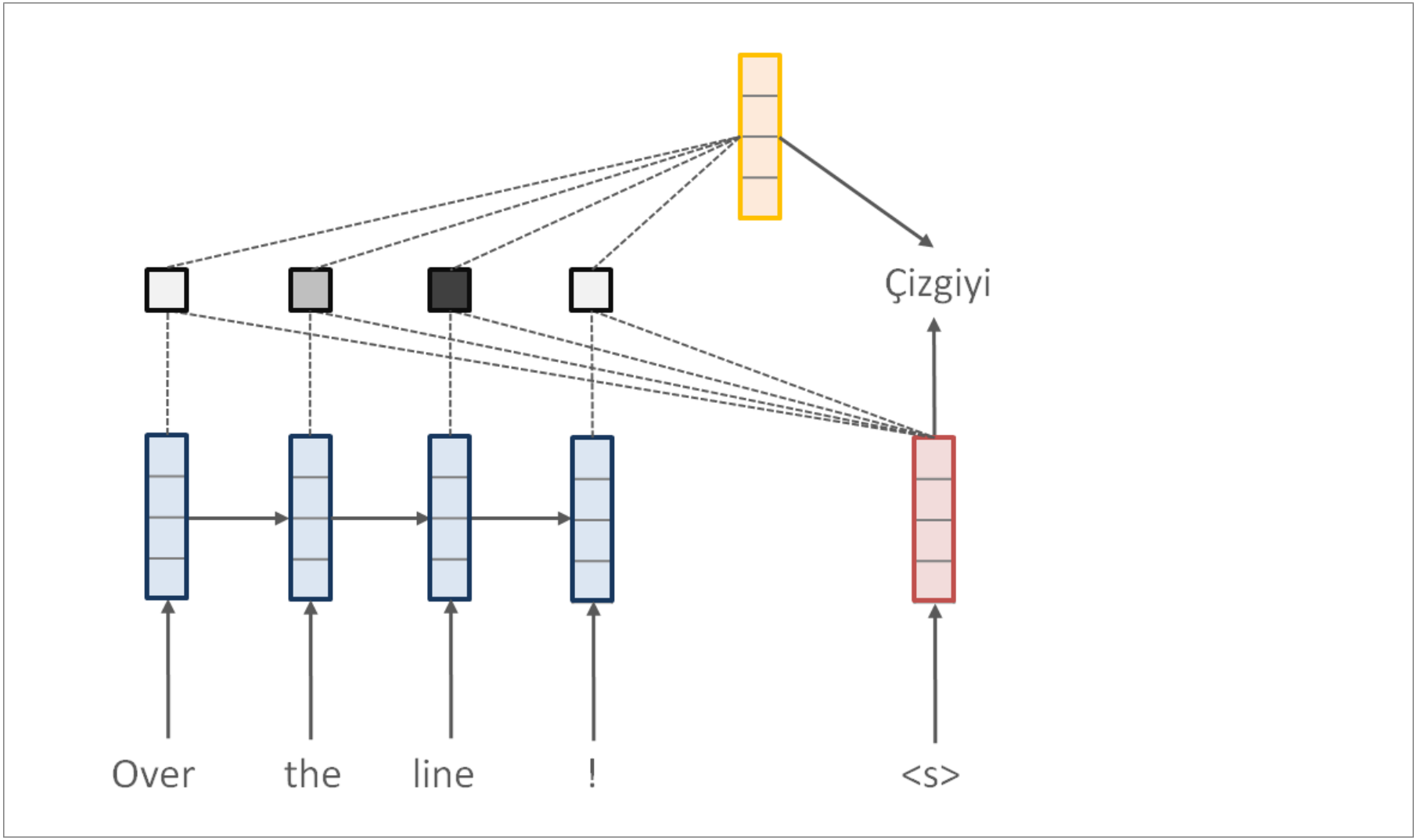
Attention-based Neural Machine Translation



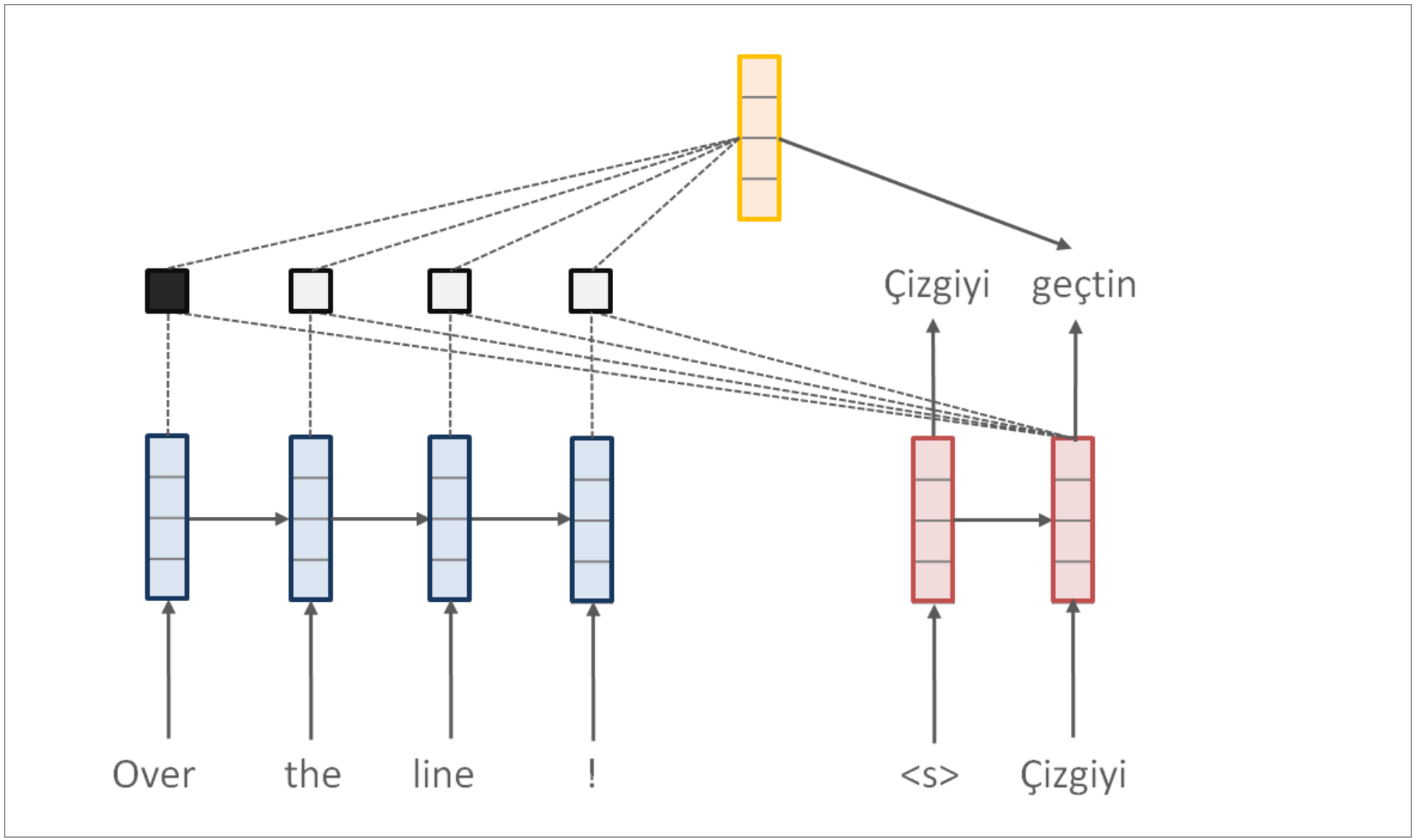
Attention-based Neural Machine Translation



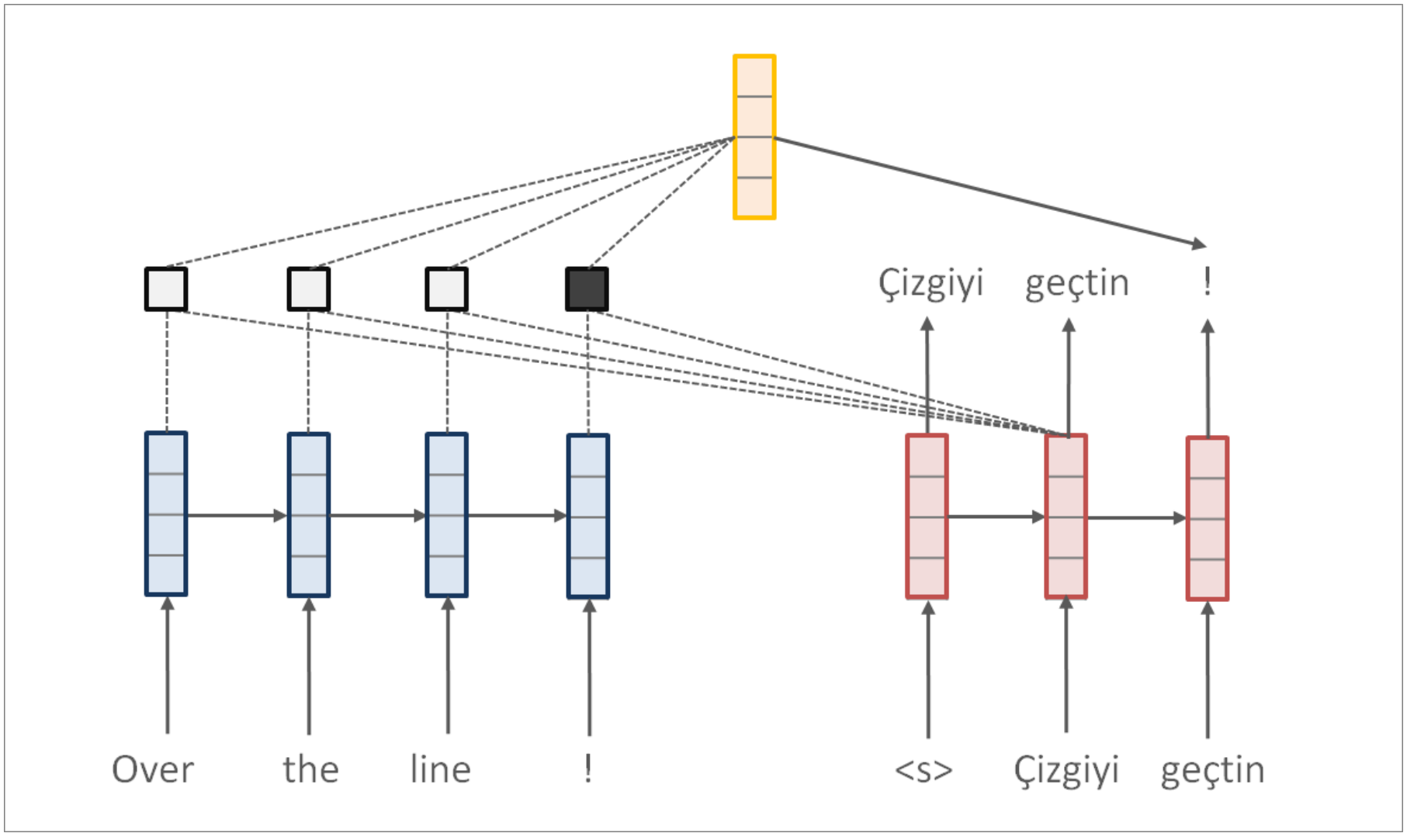
Attention-based Neural Machine Translation



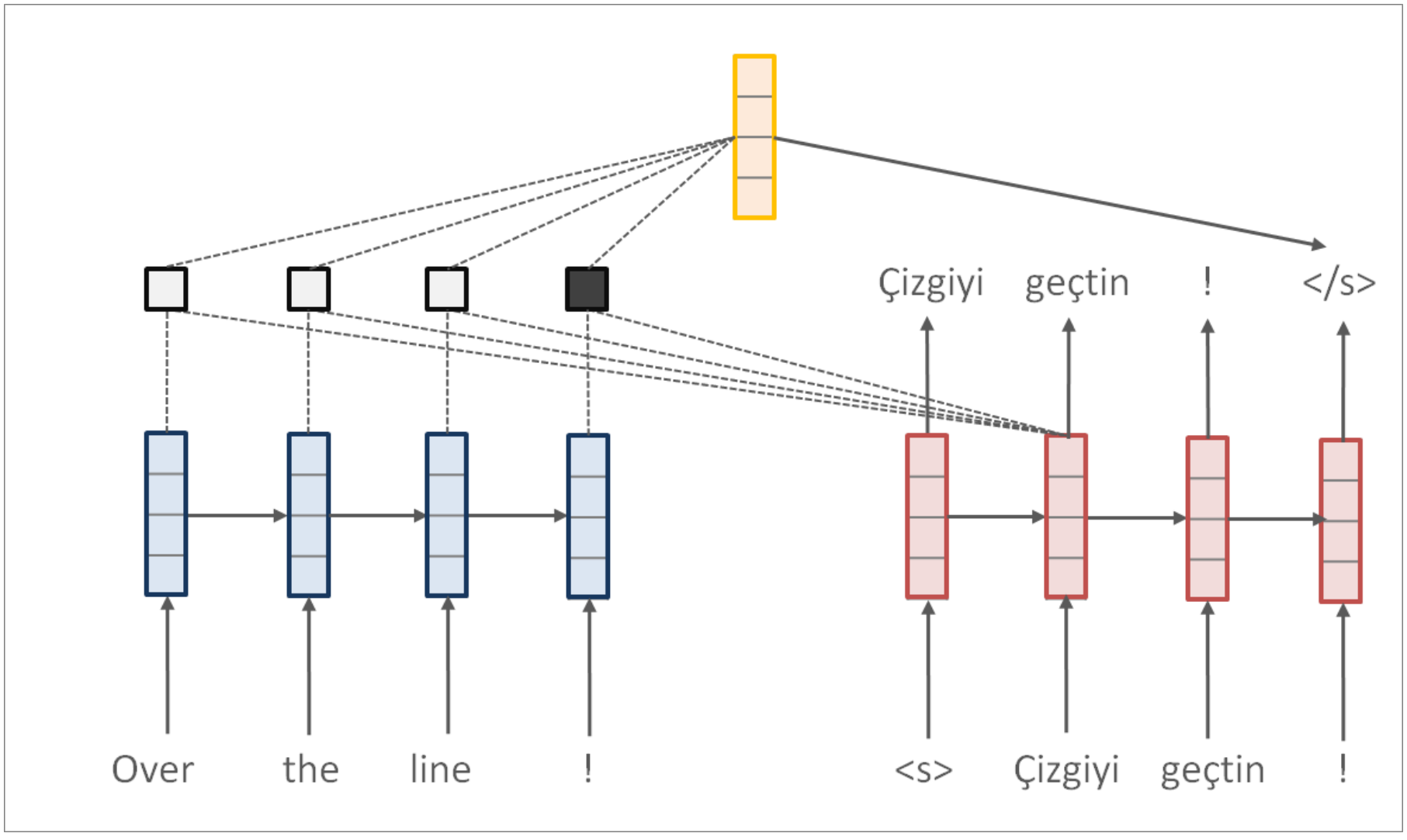
Attention-based Neural Machine Translation



Attention-based Neural Machine Translation



Attention-based Neural Machine Translation



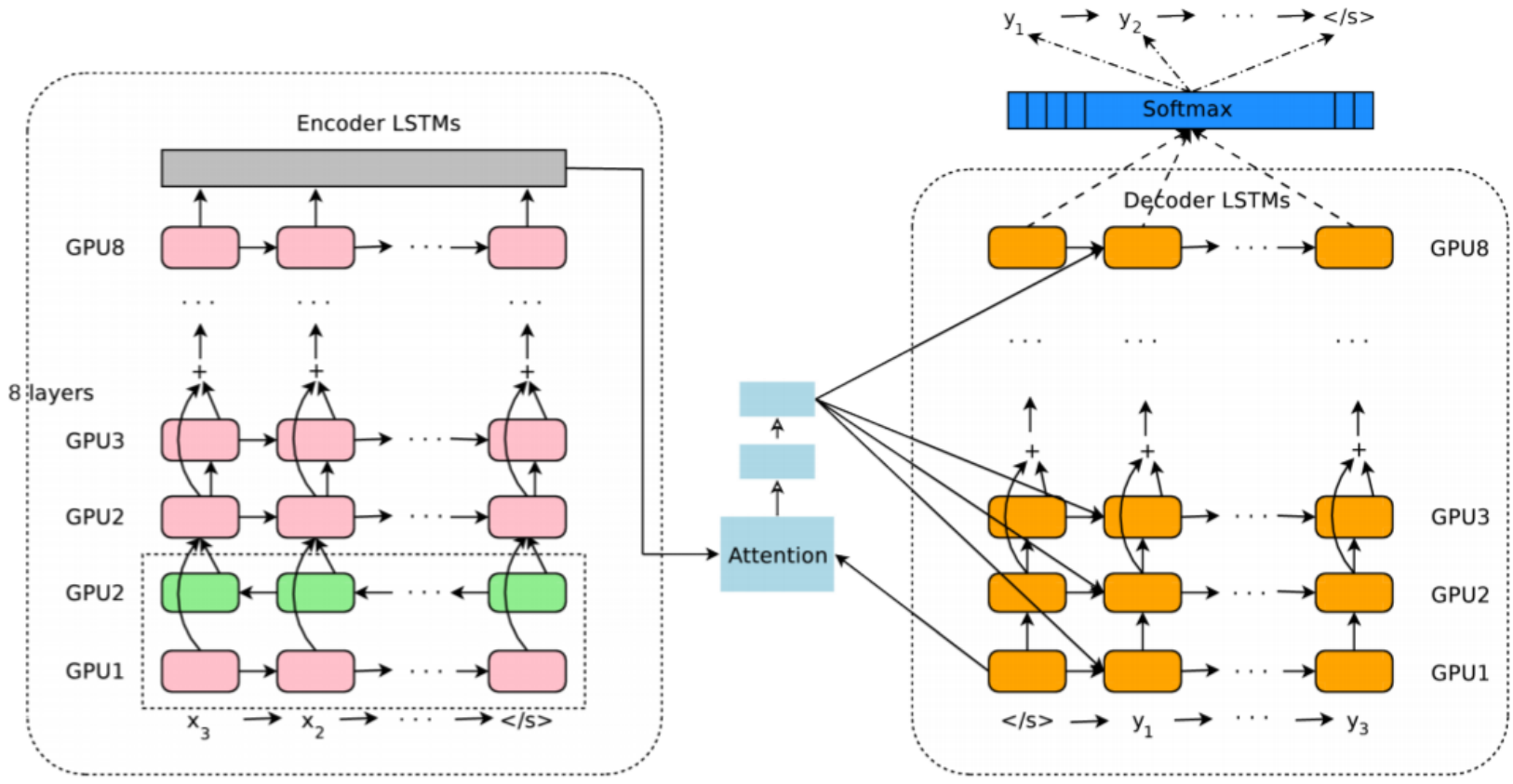
Attention Networks: Notation

$x_1, \ldots, x_T$	Memory bank ("what")
q	Query
z	Memory selection (random variable)
$p(z \mid x, q; \theta)$	Attention distribution ("where")
$c = \mathbb{E}_{z \mid x,q}[x_z]$	Context Vector

Attention Networks: Machine Translation

$x_1, \ldots, x_T$	Memory bank	Source RNN hidden states
q	Query	Decoder hidden state
z	Memory selection	Source position $\{1, \ldots, T\}$
$p(z = i \mid x, q; \theta)$	Attention distribution	$\mathrm{softmax}(x_i^\top q)$
$c = \mathbb{E}[x_z]$	Context Vector	

Google NMT


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

Contents

Encoder-Decoder with Attention

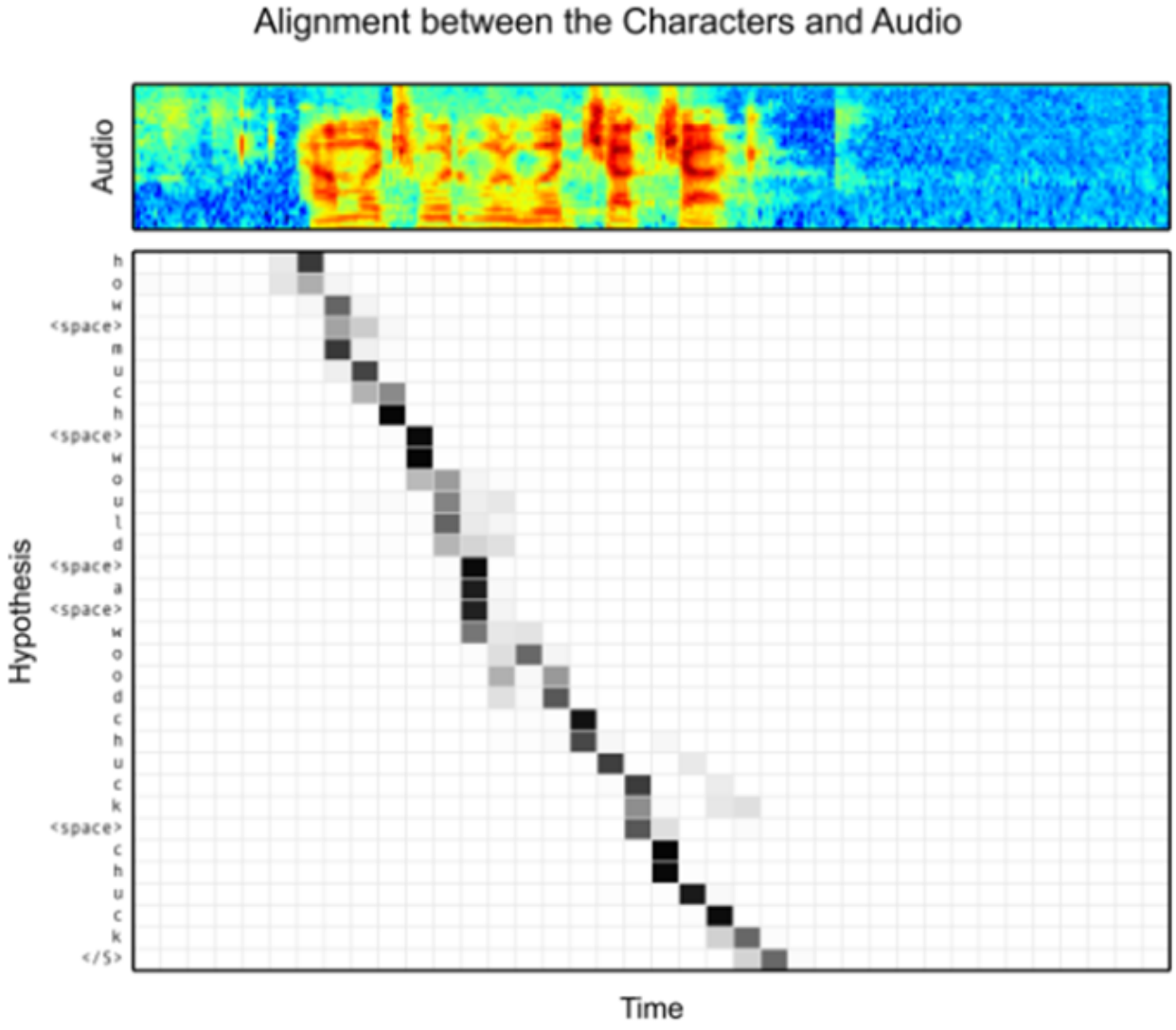
- Machine Translation

- Question Answering
- Natural Language Inference
- Algorithm Learning
- Parsing
- Speech Recognition
- Summarization
- Caption Generation
- and more...

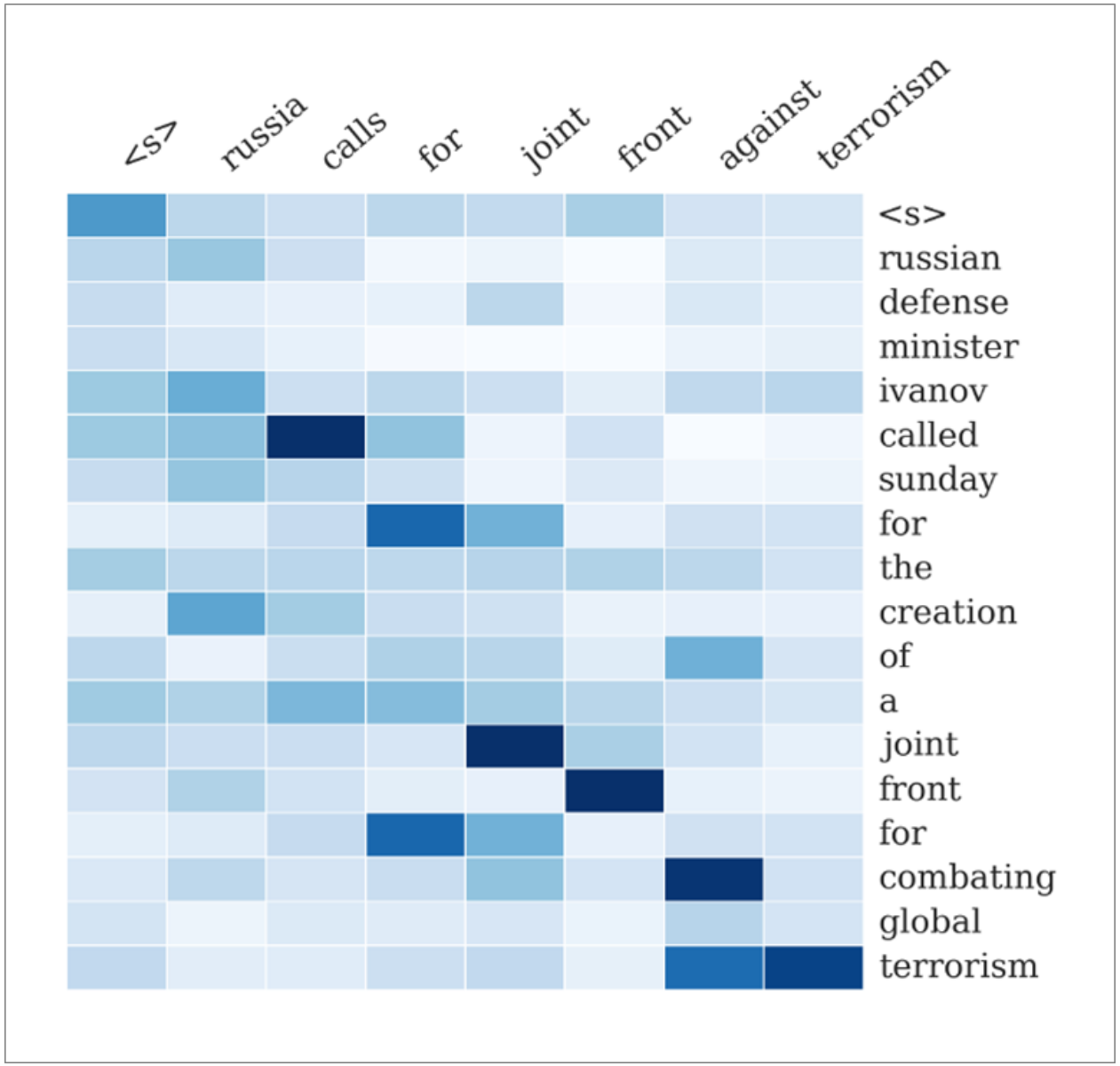
Other Applications: Image Captioning

(b) A woman is throwing a frisbee in a park.

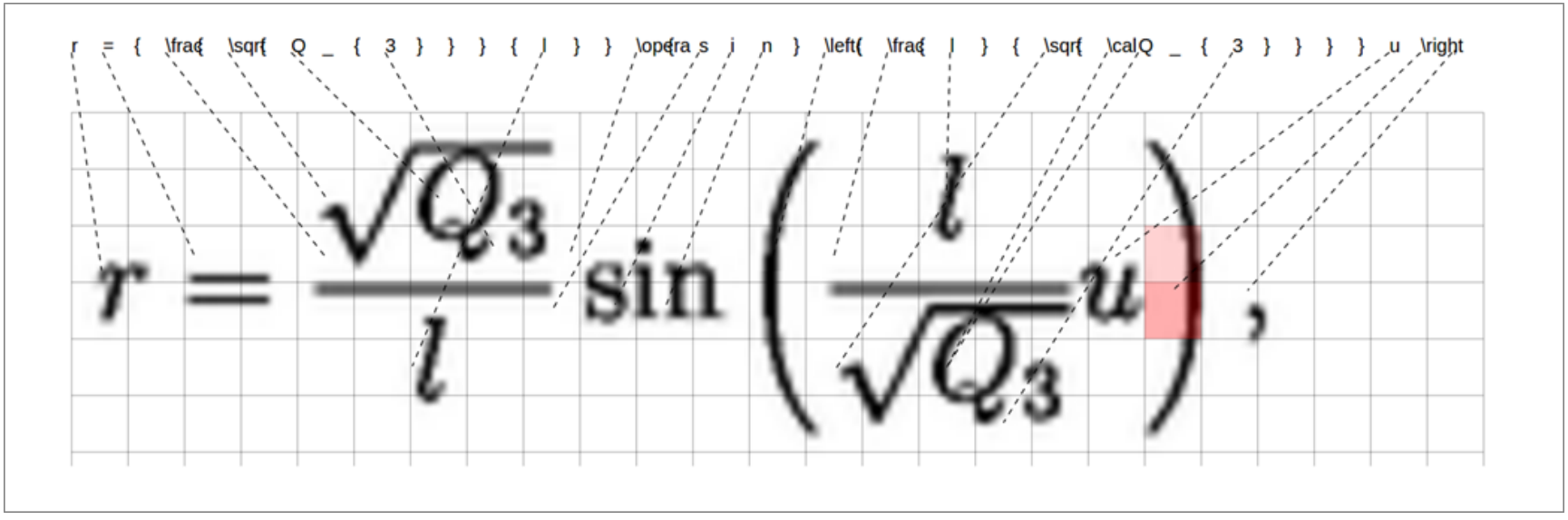
Other Applications: Speech Recognition

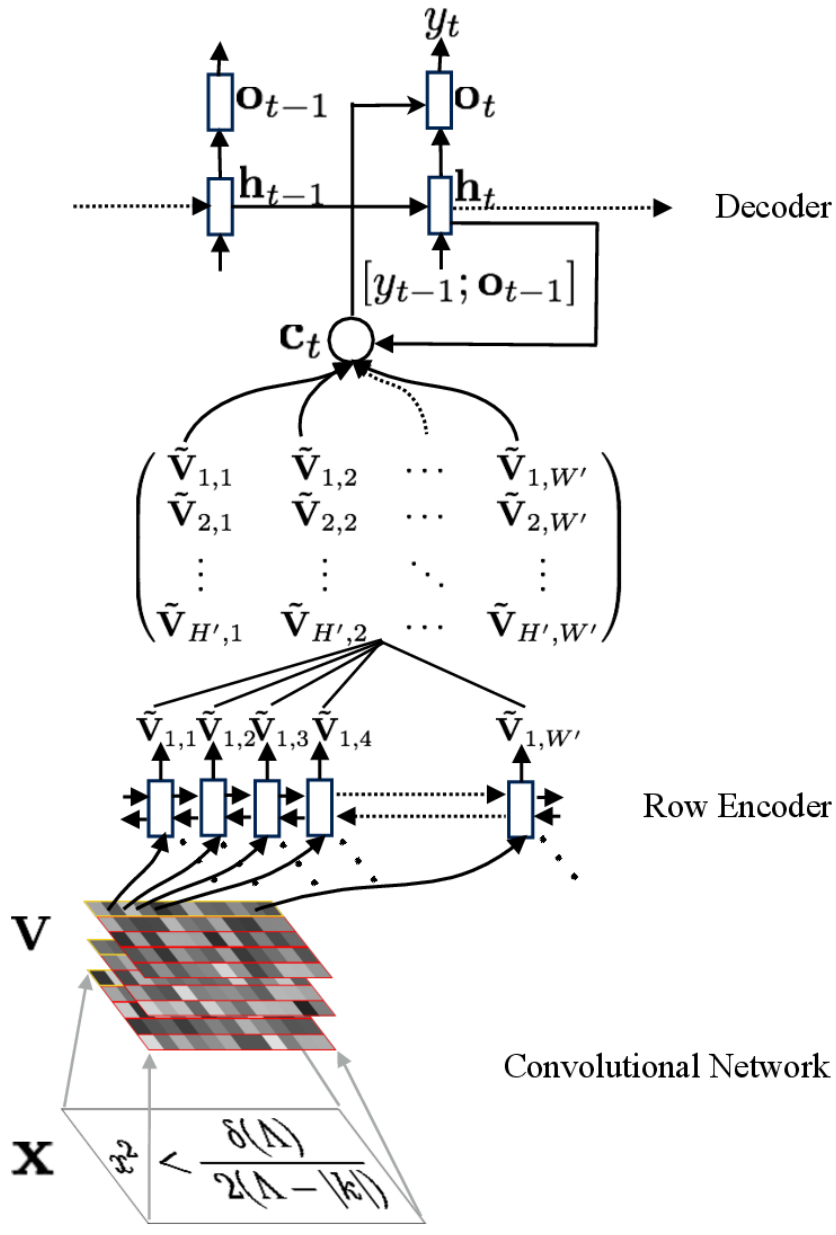


Applications From HarvardNLP: Summarization

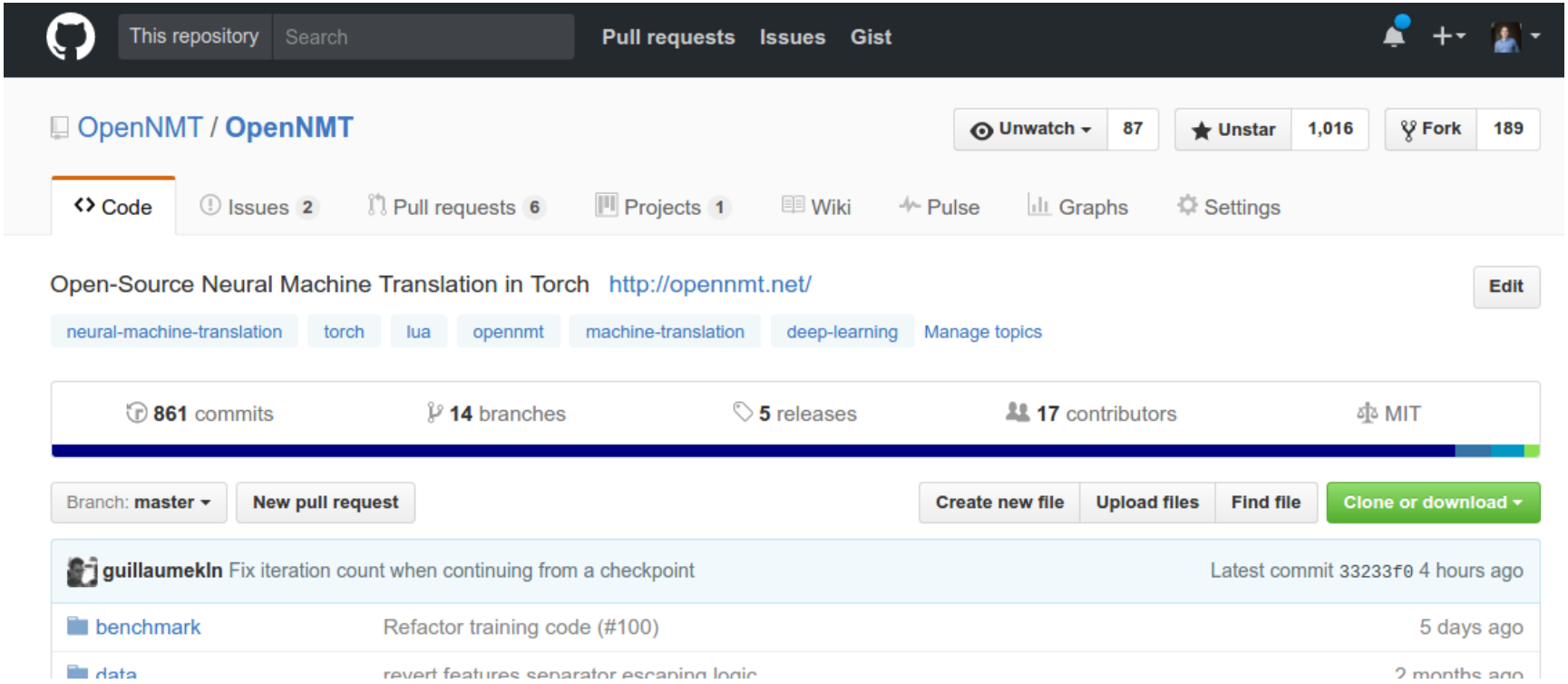


Applications From HarvardNLP: Image-to-Latex




y_t
$\mathbf{o}_{t-1}$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$
$\mathbf{h}_t$
Decoder
$[y_{t-1}; \mathbf{o}_{t-1}]$
$\mathbf{c}_t$
$\tilde{\mathbf{V}}_{1,1}$ $\tilde{\mathbf{V}}_{1,2}$ $\cdots$ $\tilde{\mathbf{V}}_{1,W'}$
$\tilde{\mathbf{V}}_{2,1}$ $\tilde{\mathbf{V}}_{2,2}$ $\cdots$ $\tilde{\mathbf{V}}_{2,W'}$
$\tilde{\mathbf{V}}_{H',1}$ $\tilde{\mathbf{V}}_{H',2}$ $\cdots$ $\tilde{\mathbf{V}}_{H',W'}$
$\tilde{\mathbf{V}}_{1,1}$ $\tilde{\mathbf{V}}_{1,2}$ $\tilde{\mathbf{V}}_{1,3}$ $\tilde{\mathbf{V}}_{1,4}$ $\tilde{\mathbf{V}}_{1,W'}$
Row Encoder
$\mathbf{V}$
Convolutional Network
$\mathbf{X}$

Applications From HarvardNLP: OpenNMT



References I