Introduction to Attention

Danna Gurari
University of Colorado Boulder
Spring 2022

Review

- Last week:
  - Introduction to natural language processing
  - Text representation
  - Neural word embeddings
  - Programming tutorial

- Assignments (Canvas):
  - Lab assignment 3 due next week

- Questions?

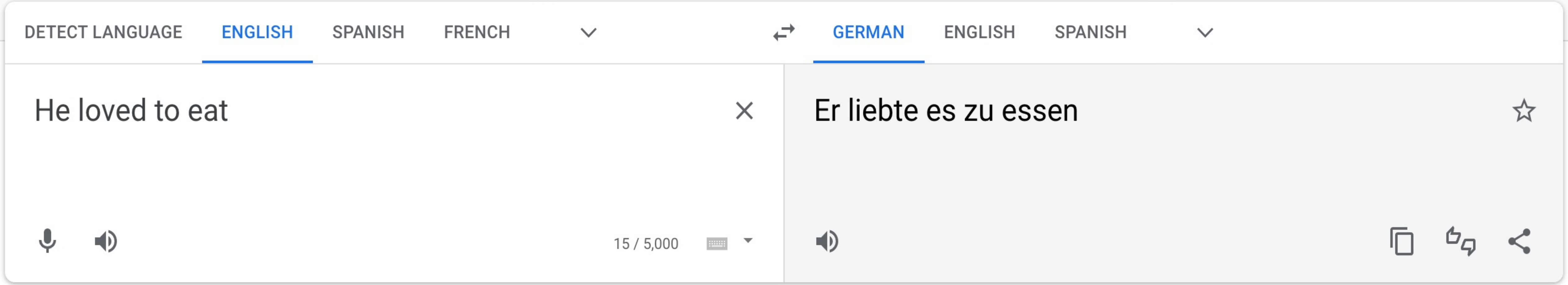
Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Task: Machine Translation



Pioneering Neural Network Approach

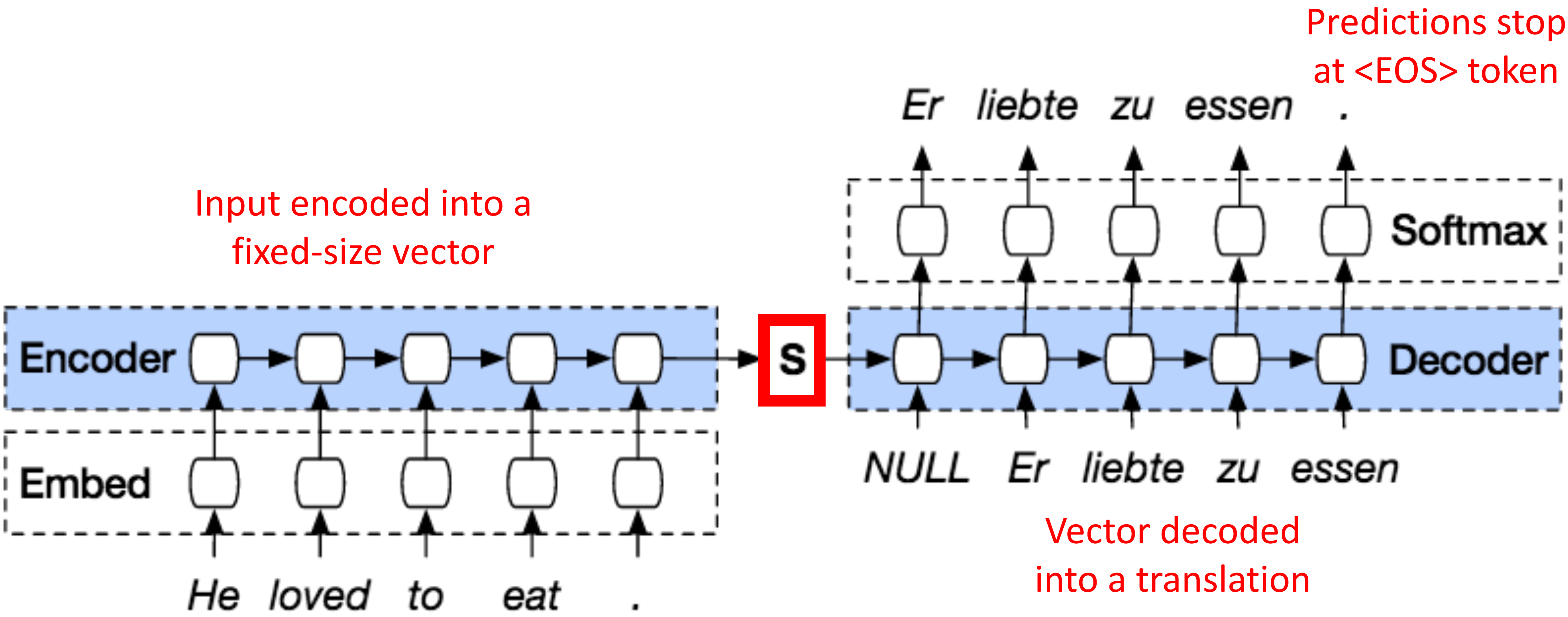


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

seq2seq: Sutskever et al. Sequence to Sequence Learning with Neural Networks. Neurips 2014.

Pioneering Neural Network Approach

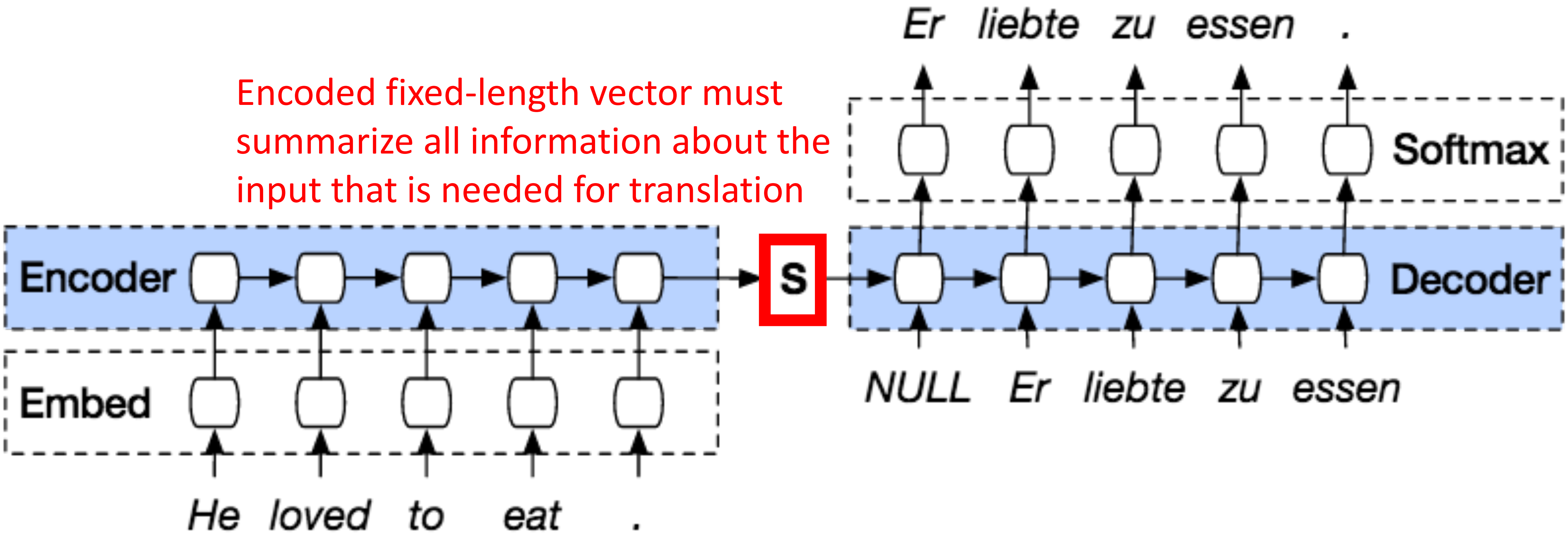
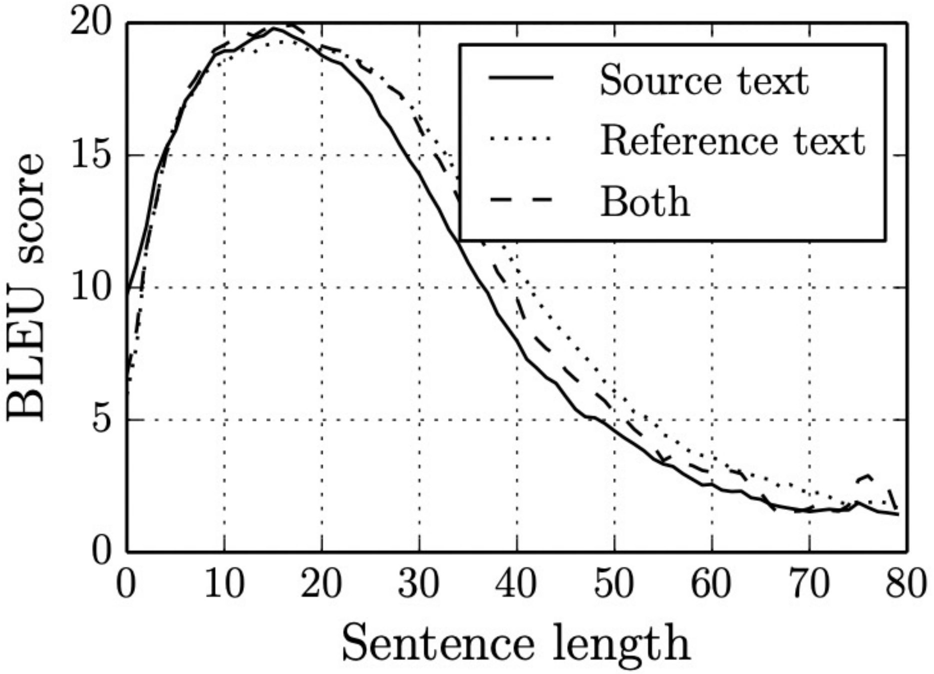
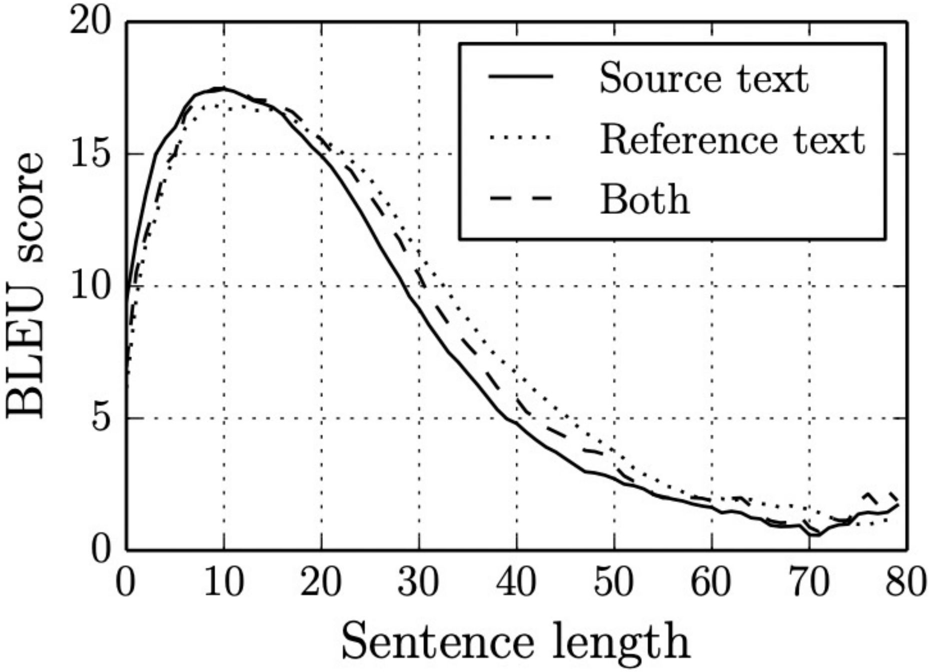


Image source: https://smerity.com/articles/2016/google_nmt_arch.html
Sutskever et al. Sequence to Sequence Learning with Neural Networks. Neurips 2014.

Analysis of Two Models

(larger scores are better)



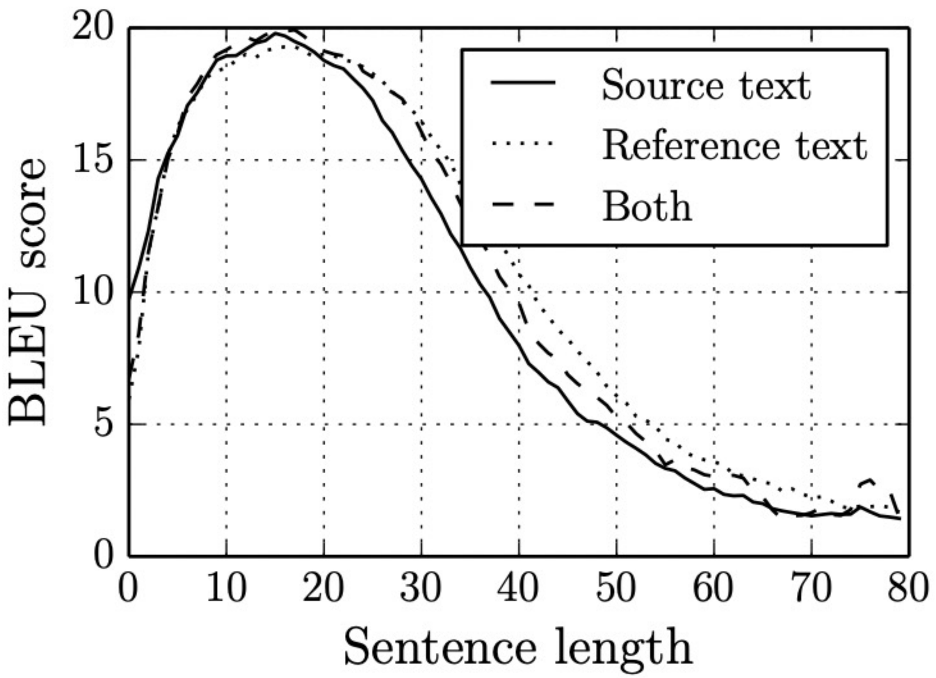


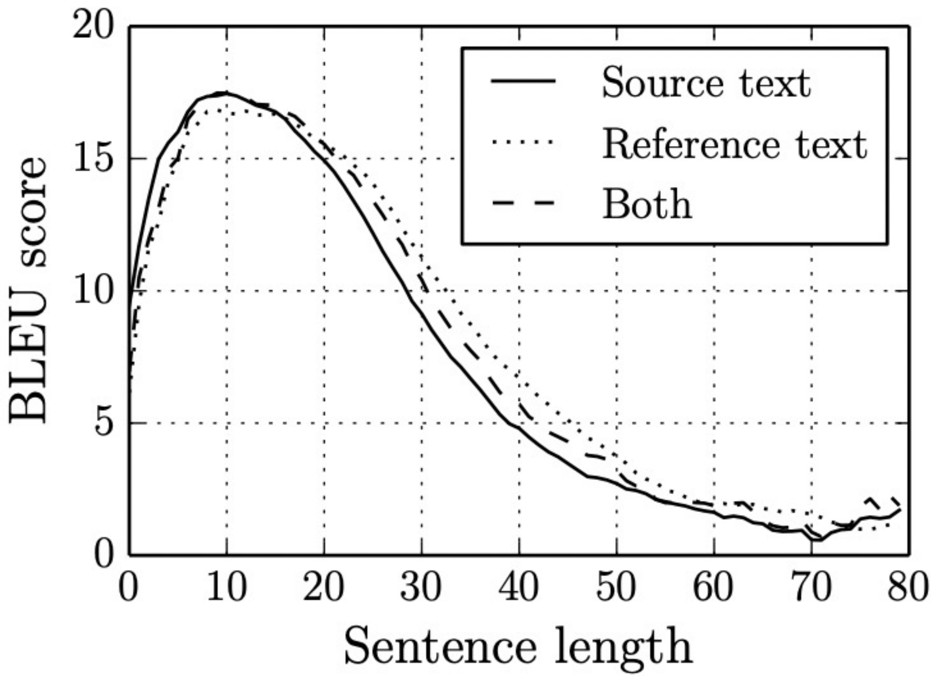
What performance trend is observed for inputs (source) and outputs (reference) as the number of words in each sentence grows?

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Analysis of Two Models

(larger scores are better)





Performance drops for longer sentences!

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Problem: Performance Drops As Sentence Length Grows

Hypothesis: fixed-length vector lacks sufficient capacity to capture all relevant information for long sentences

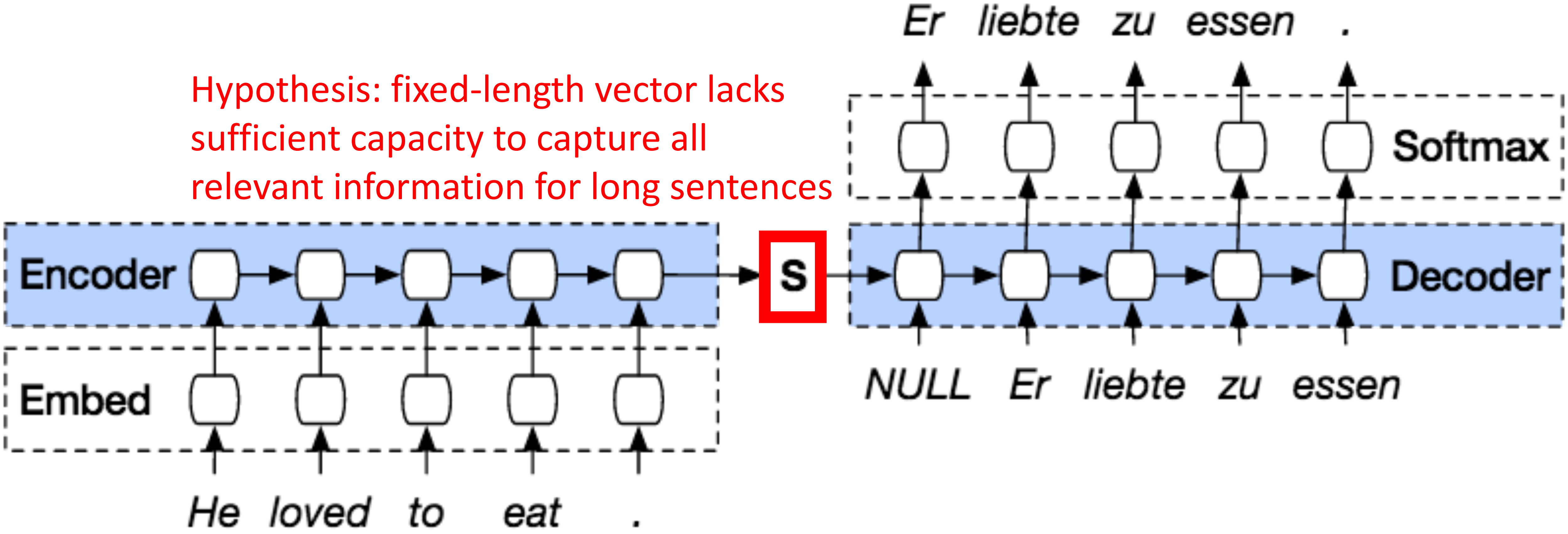


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Idea to Preserve Performance for Long Sentences: **Attention**

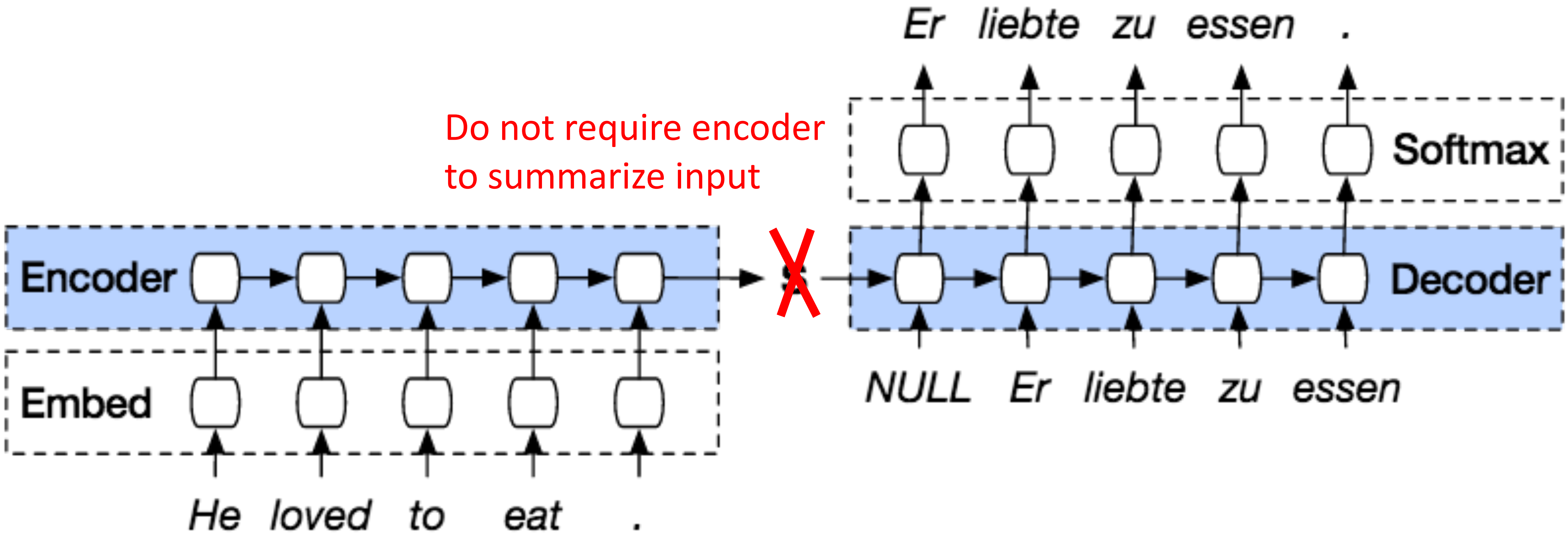


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Idea to Preserve Performance for Long Sentences: **Attention**

Instead, have the encoder pass **all** input's hidden states to the decoder to decide which to use for prediction at each time step

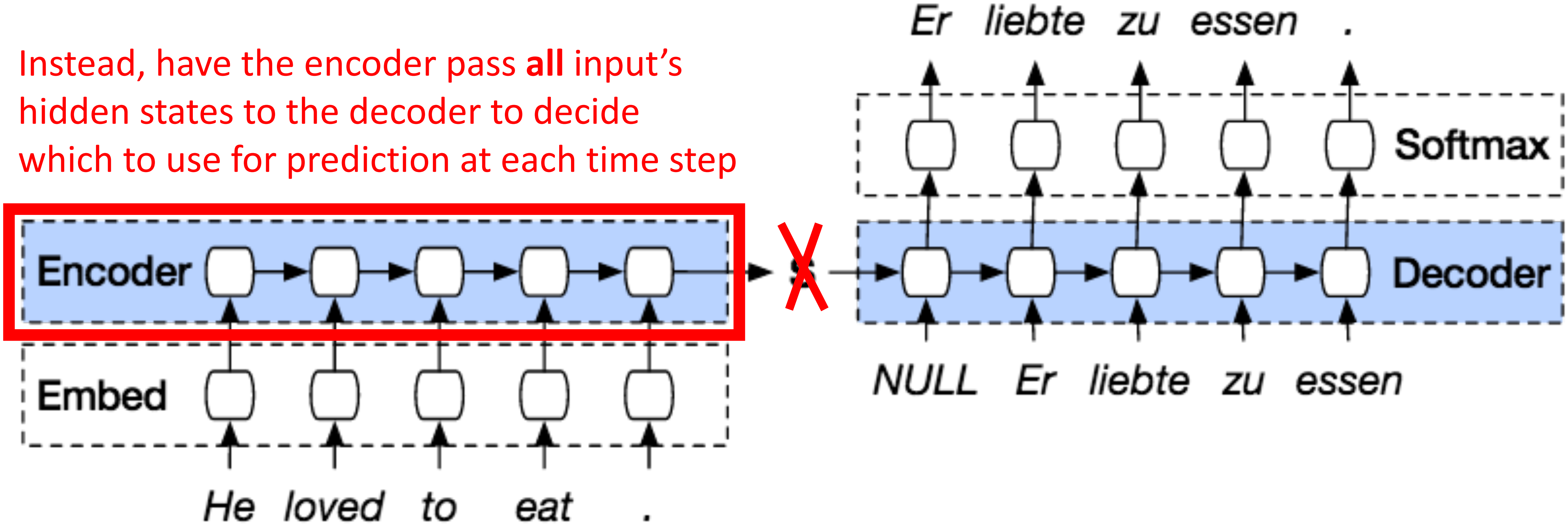
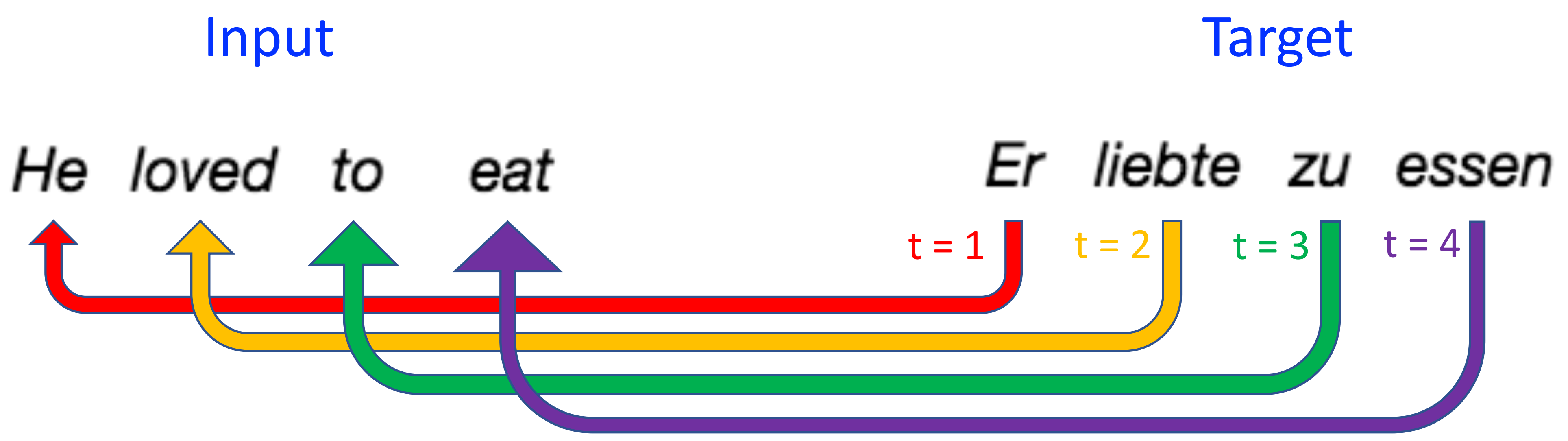


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “hard attention” focuses on one input

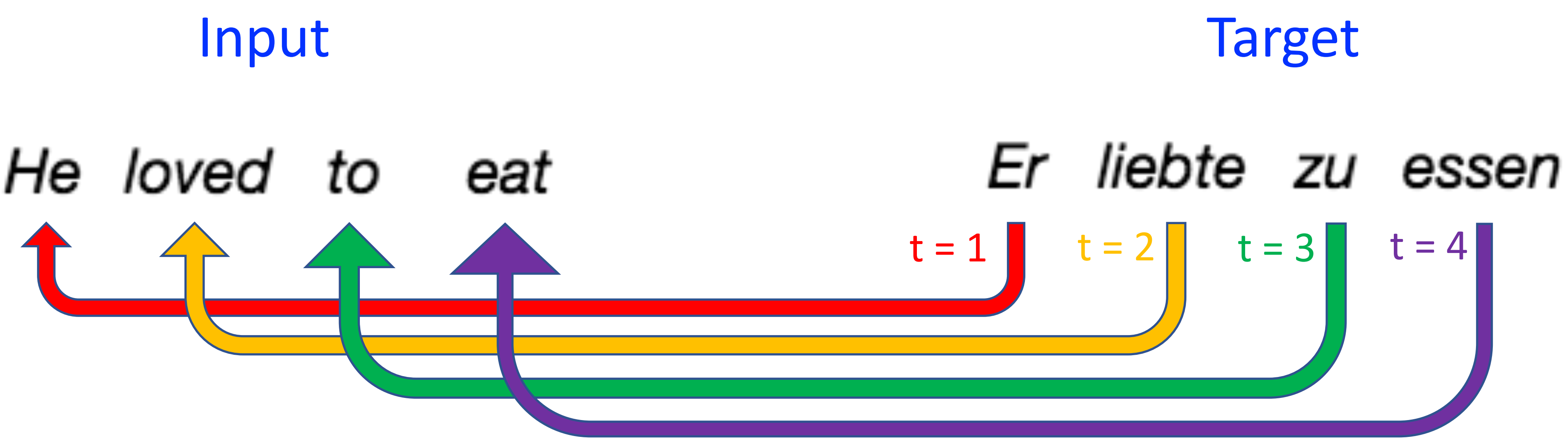


Note: while word order between the input and target align in this example, it can differ

https://deeplearning.cs.cmu.edu/F21/document/slides/lec18.attention.pdf

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “hard attention” focuses on one input



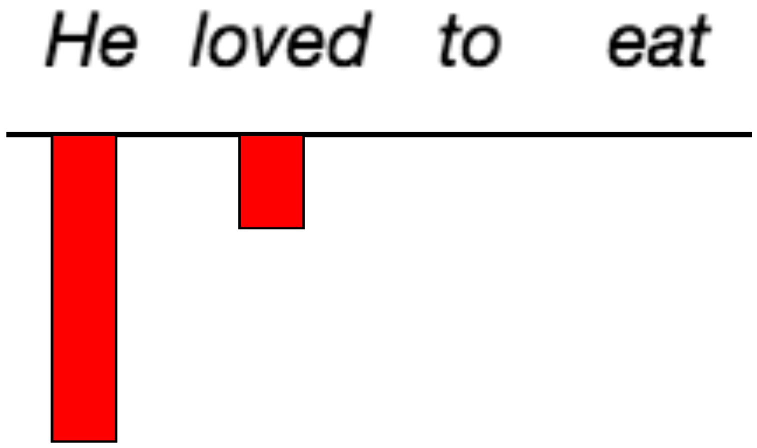
Limitations: a target word relies on information about one input word and “hard attention” is not differentiable

https://deeplearning.cs.cmu.edu/F21/document/slides/lec18.attention.pdf

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

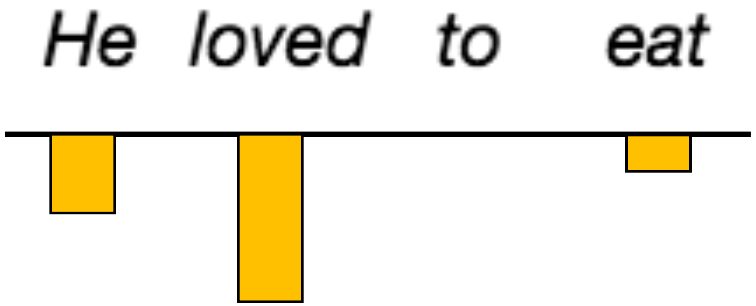
Er liebte zu essen

t = 1

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

Er liebte zu essen

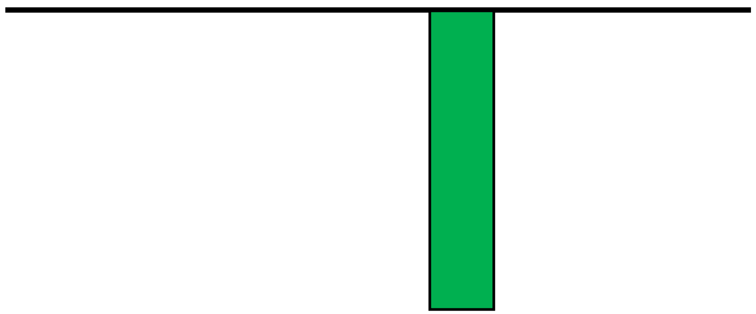
t = 1 t = 2

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input

He loved to eat



Target

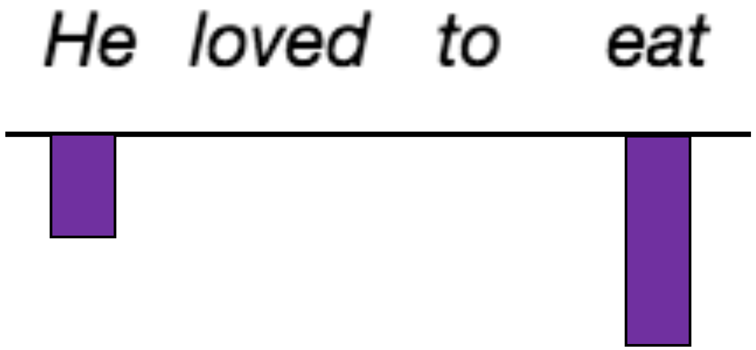
Er liebte zu essen

t = 1 t = 2 t = 3

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

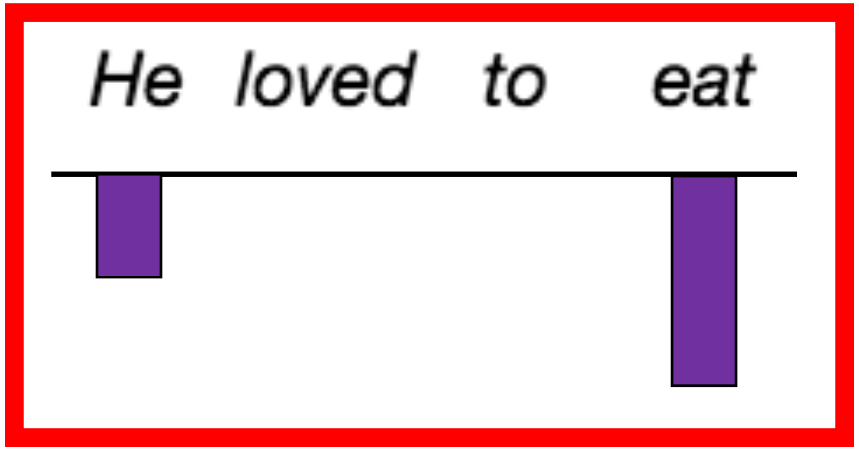
Er liebte zu essen

t = 1 t = 2 t = 3 t = 4

"Soft" Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., "soft attention" uses a weighted combination of the input

Input



Target

Er liebte zu essen

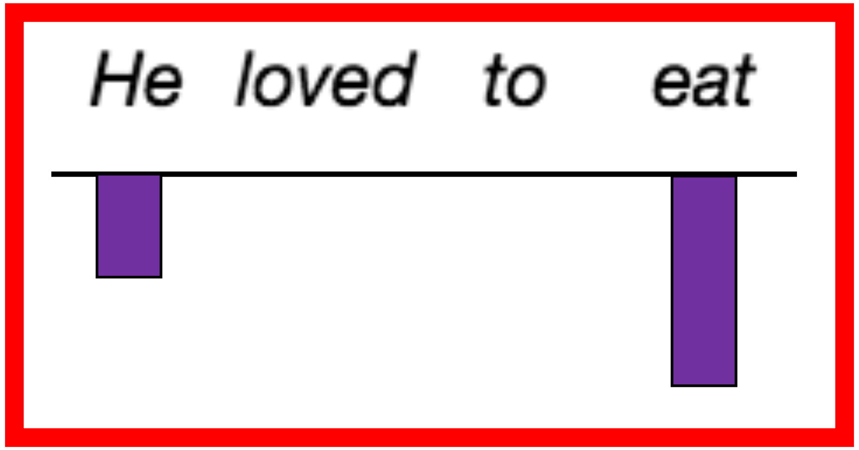
t = 1 t = 2 t = 3 t = 4

How should weights be chosen for each input?

“Soft” Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

Er liebte zu essen

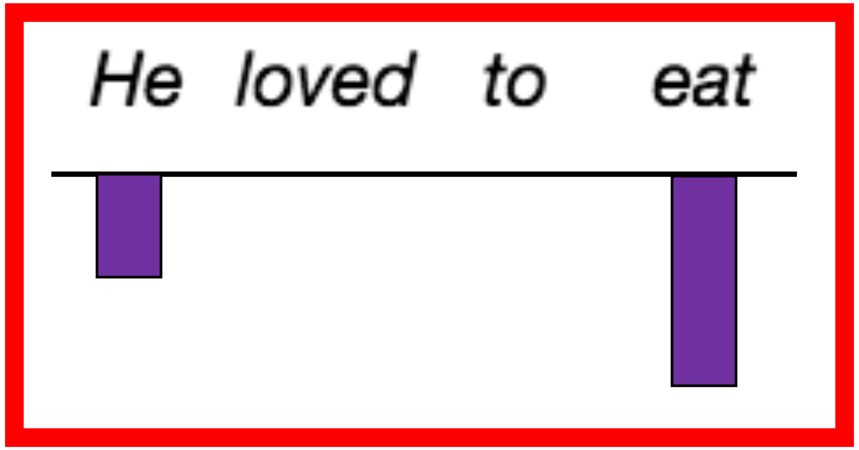
t = 1 t = 2 t = 3 t = 4

Could collect manual annotations and then incorporate into the loss function that predicted weights should match ground truth weights… but this approach is impractical

“Soft” Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Instead, have the model learn how to weight each input!

Target

Er liebte zu essen

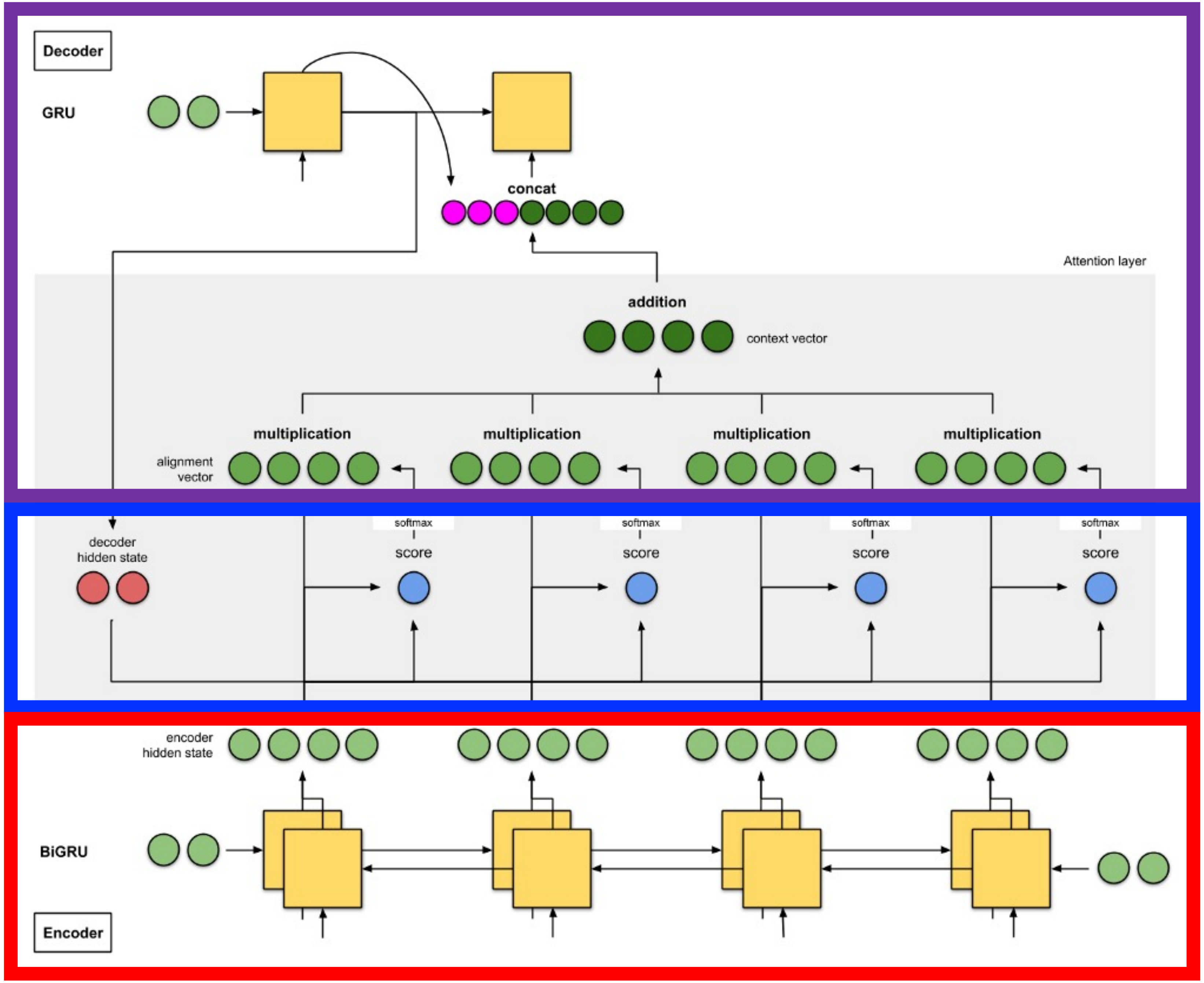
t = 1 t = 2 t = 3 t = 4

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction

1. Encoder produces hidden state for every input



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

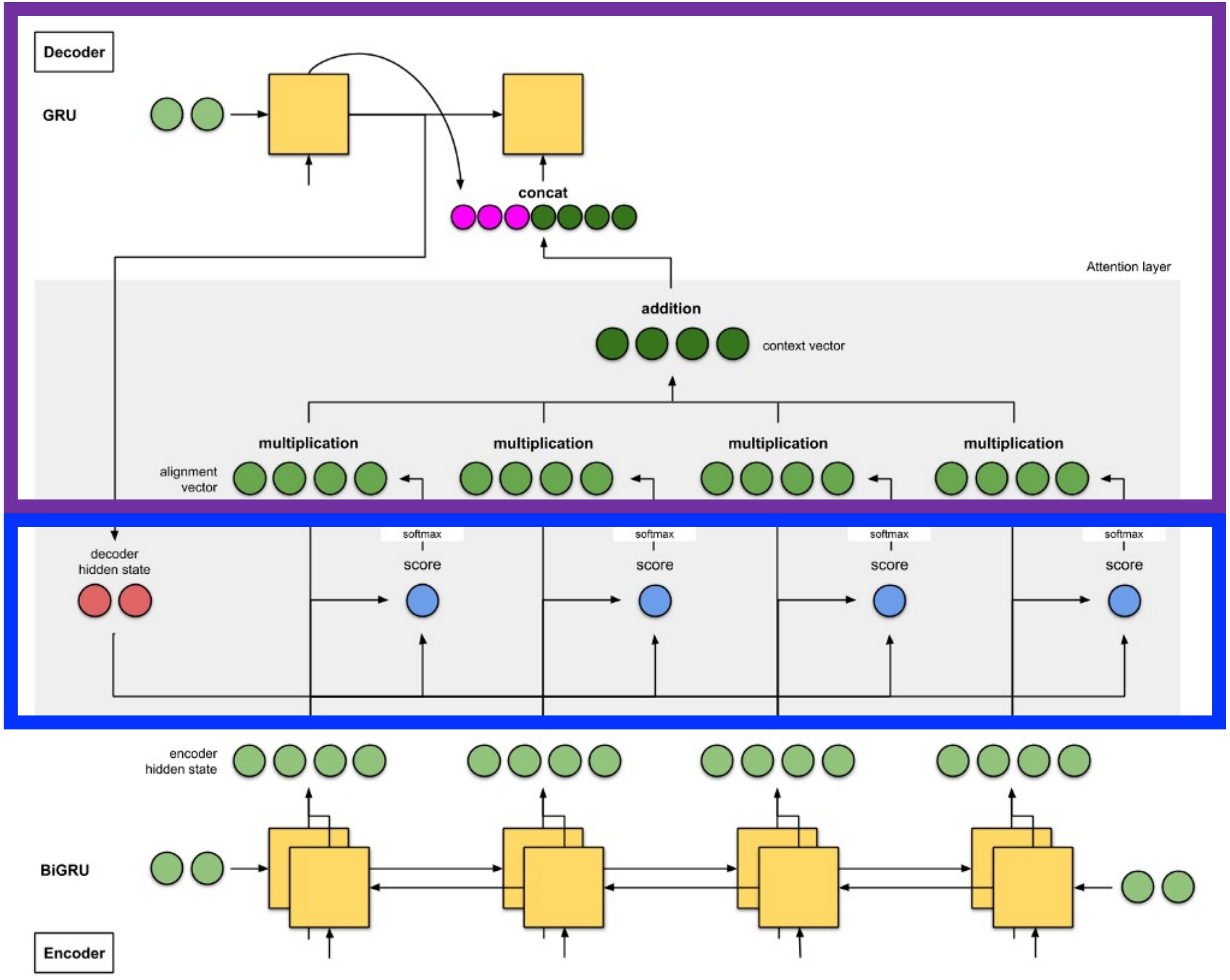
Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

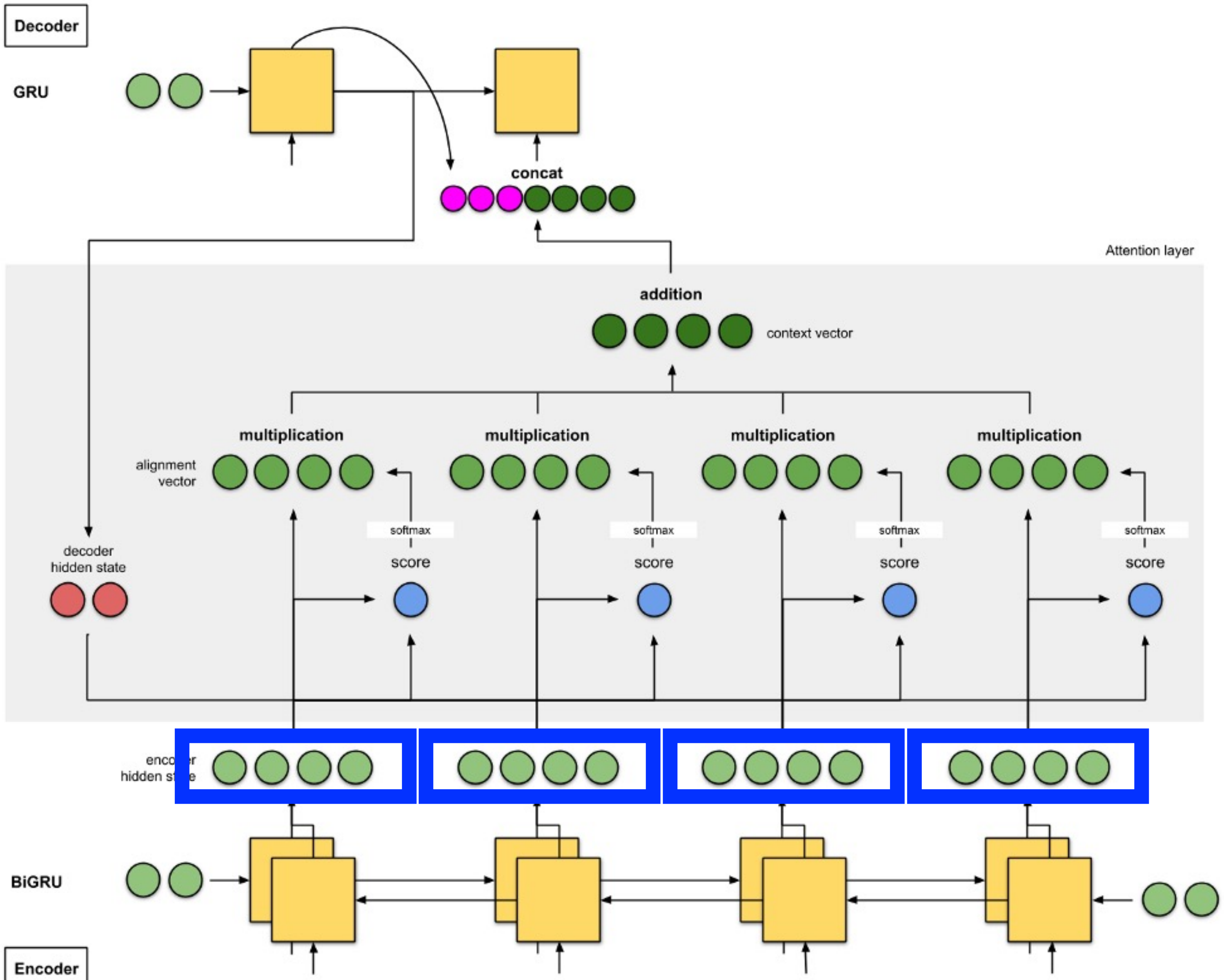
2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input’s Influence on the Prediction

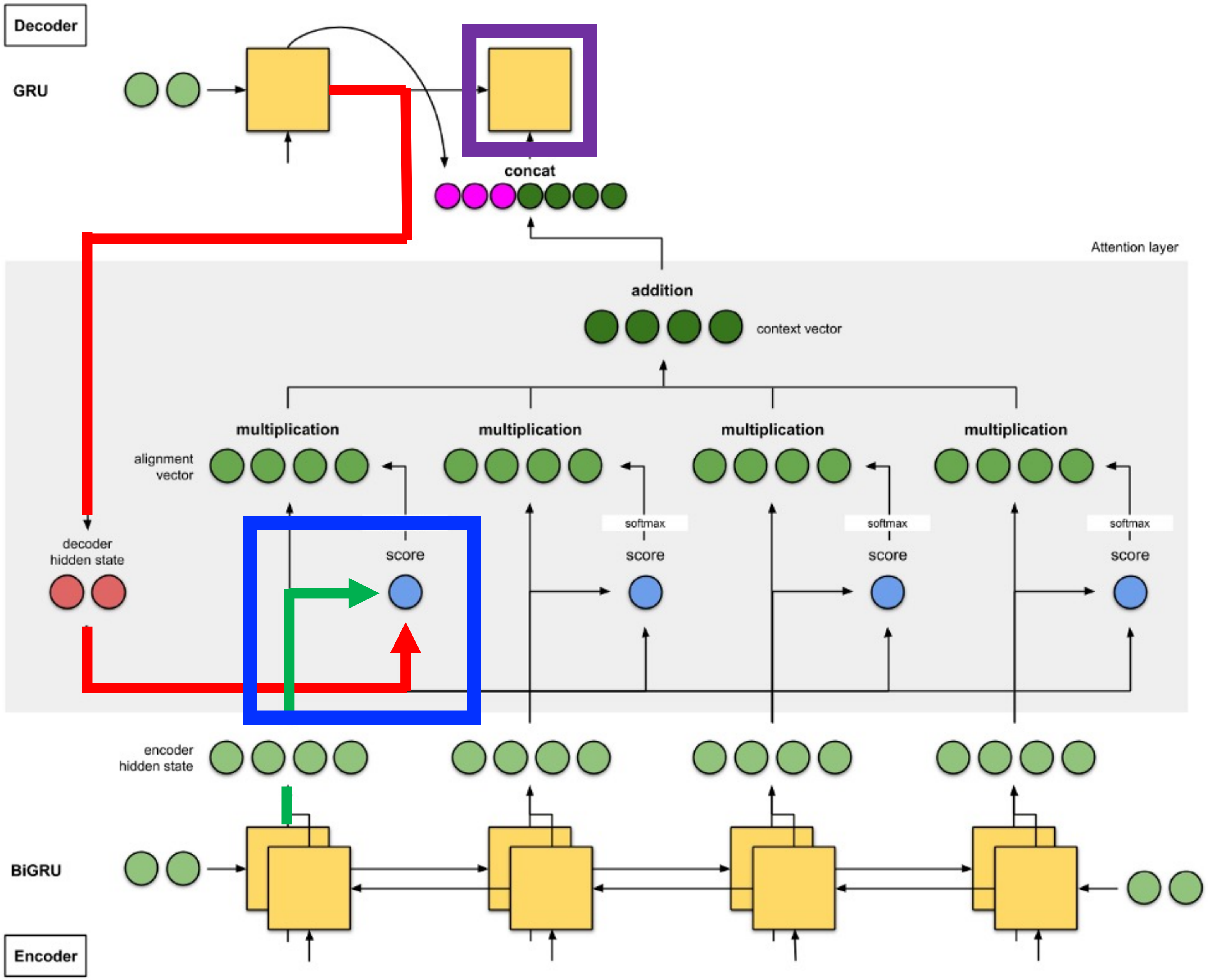
How many inputs are in this example?



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

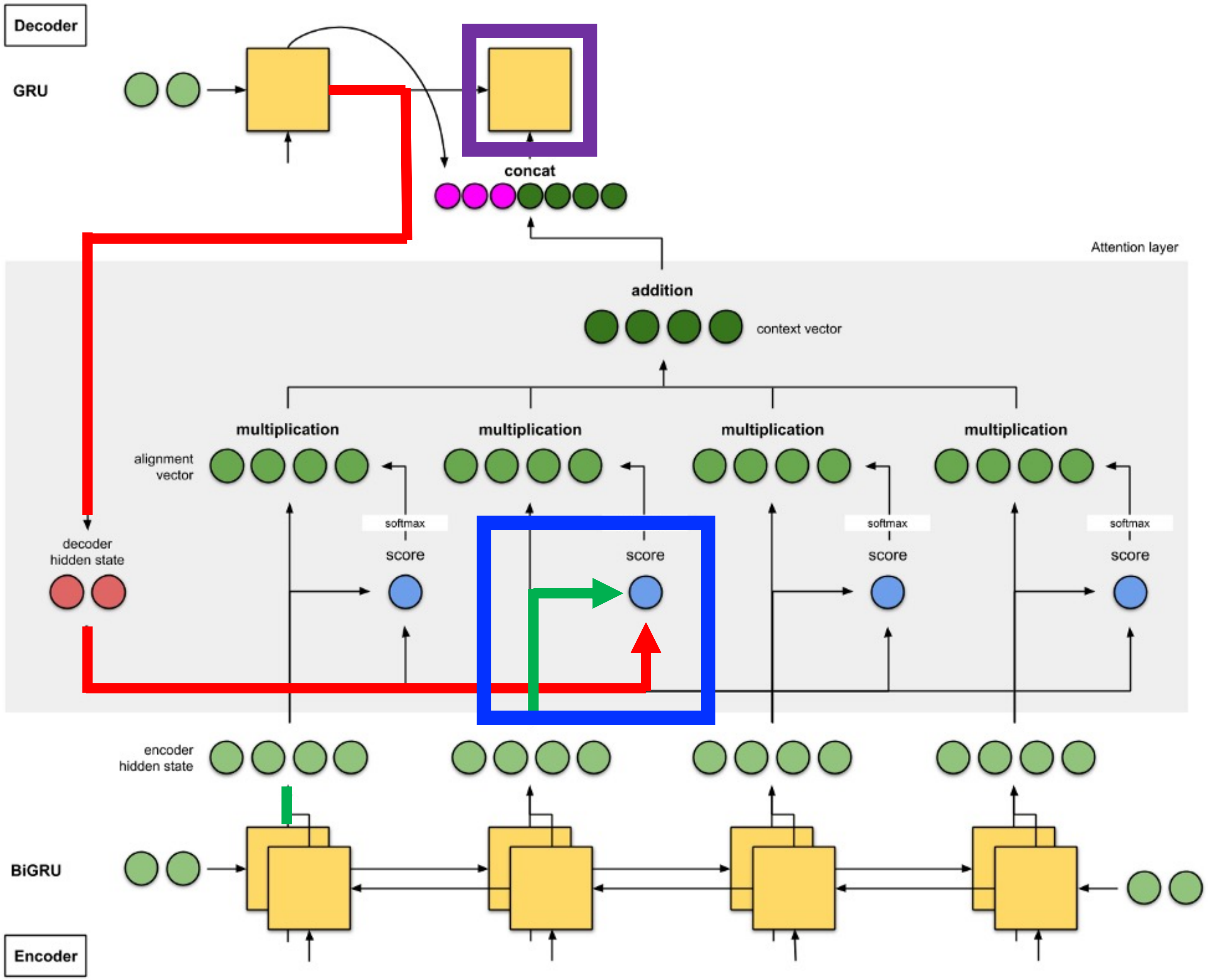
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

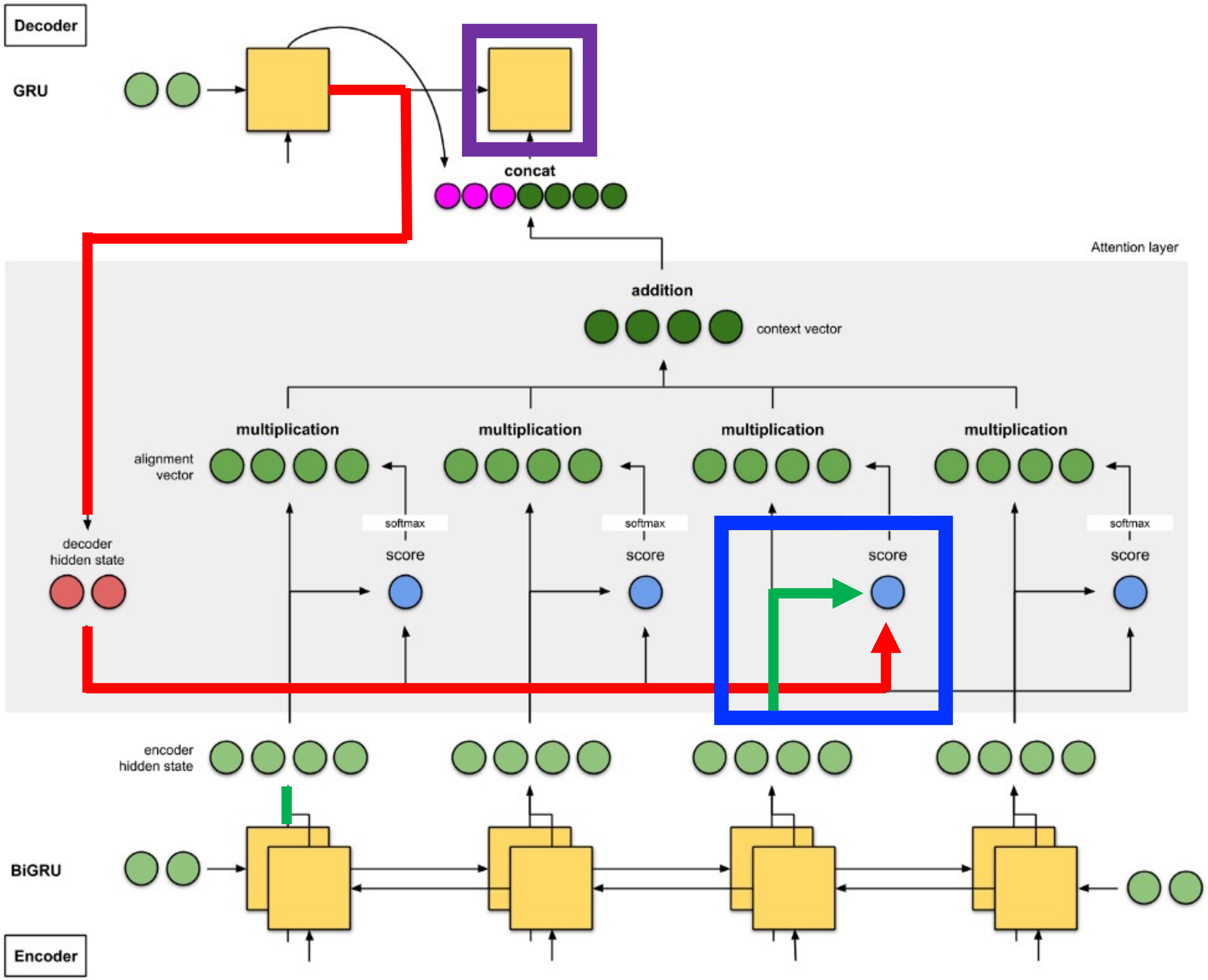
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

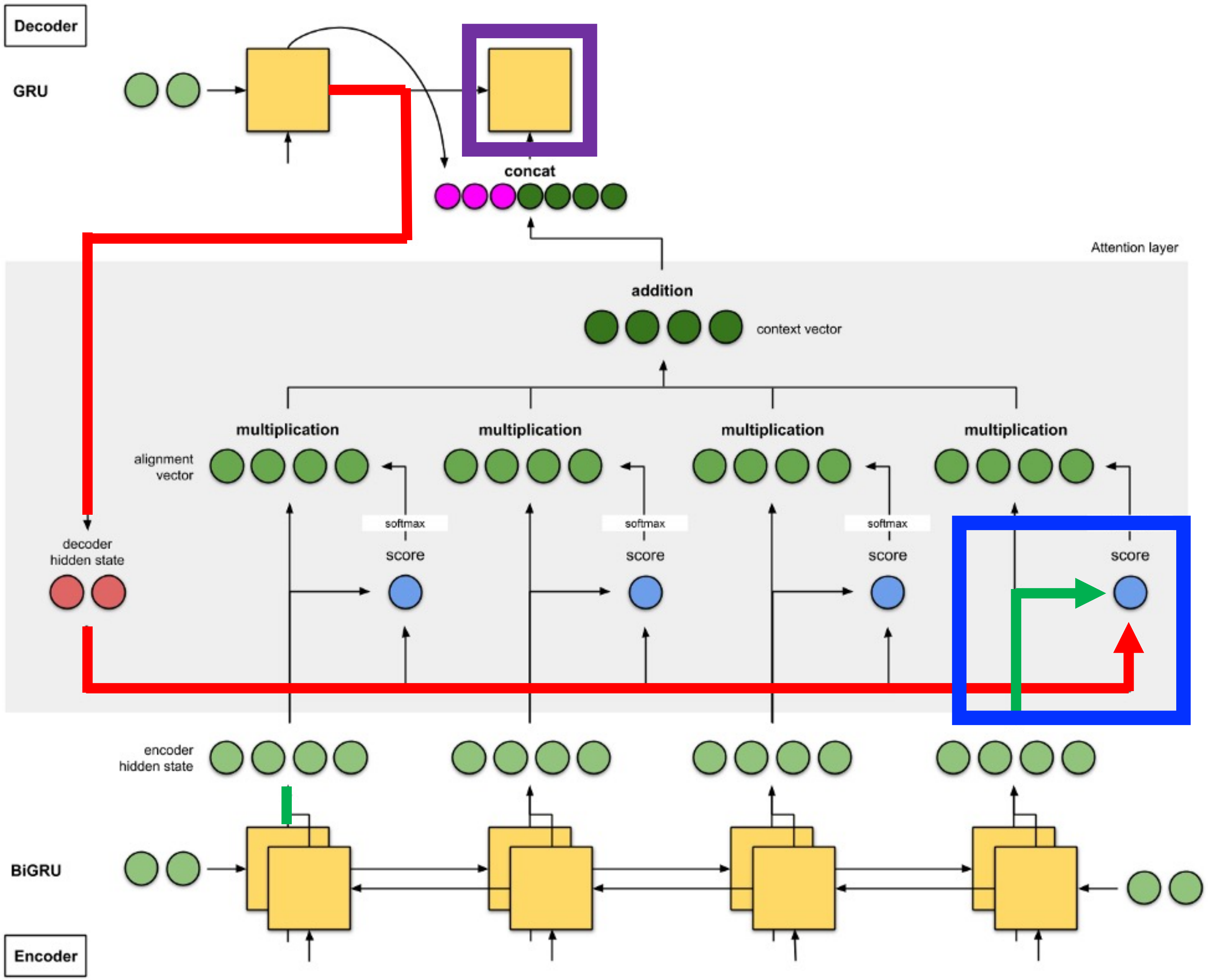
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

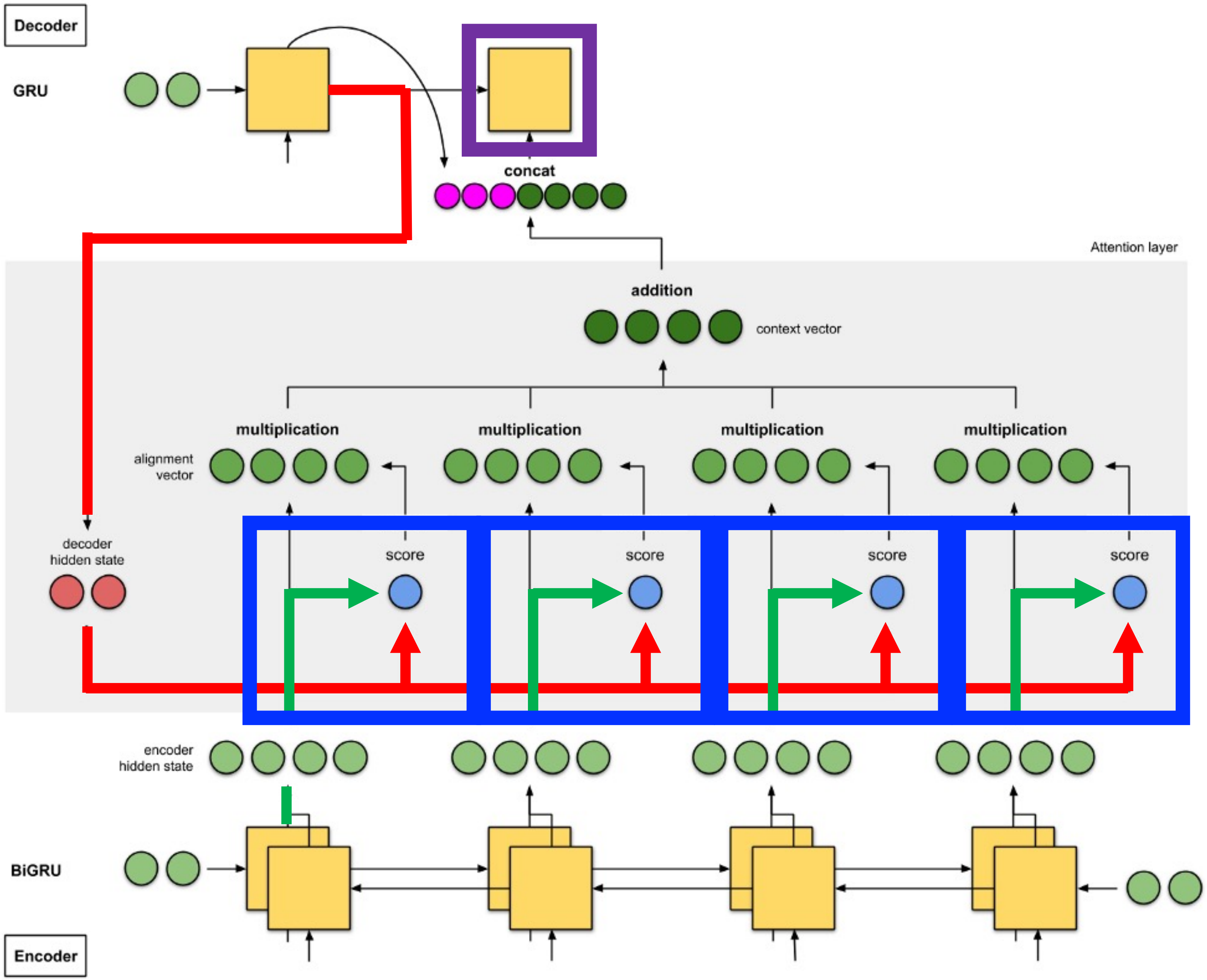
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

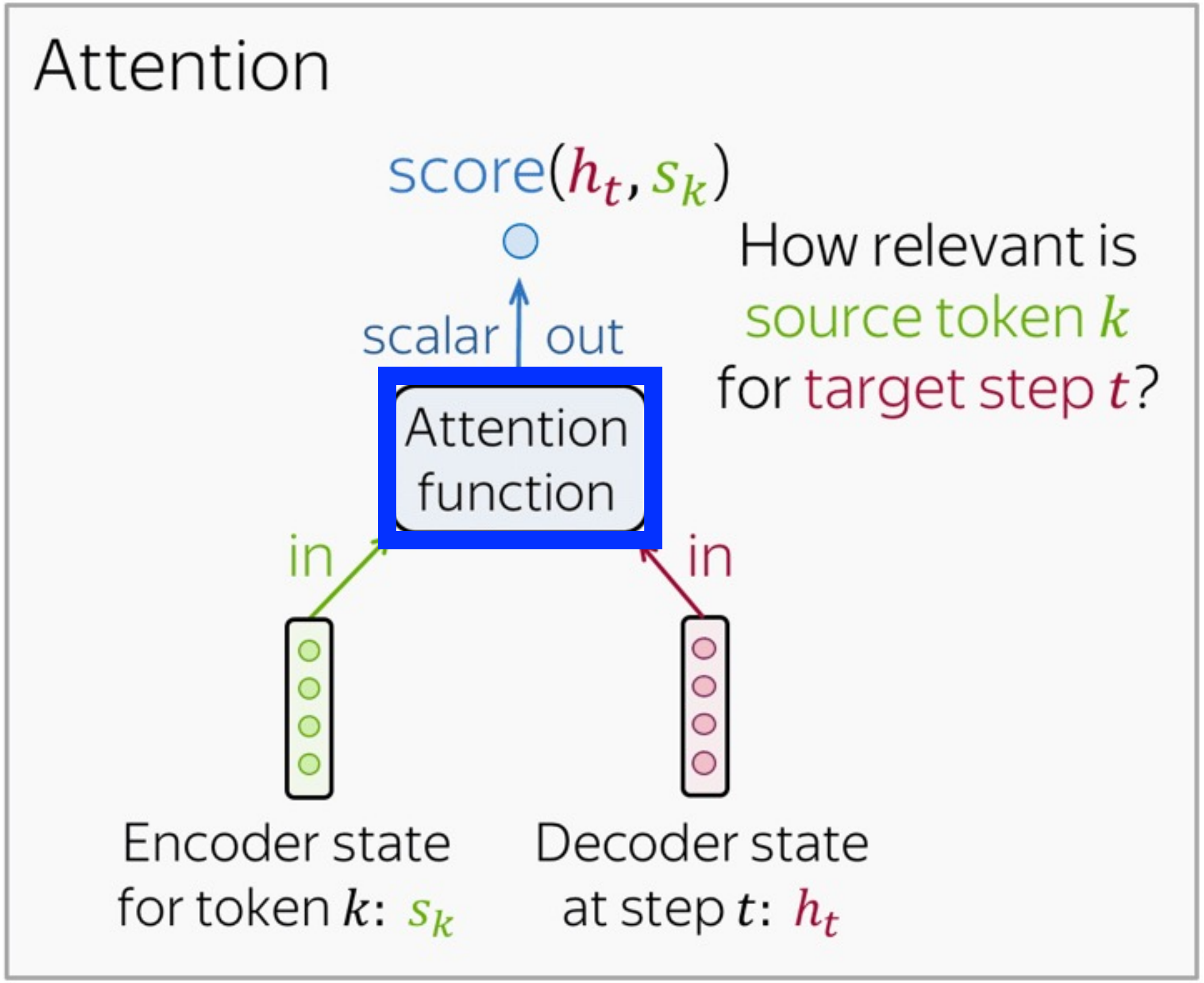
Measuring Each Input's Influence on the Prediction

How to measure the similarity between hidden states of the decoder and input?



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

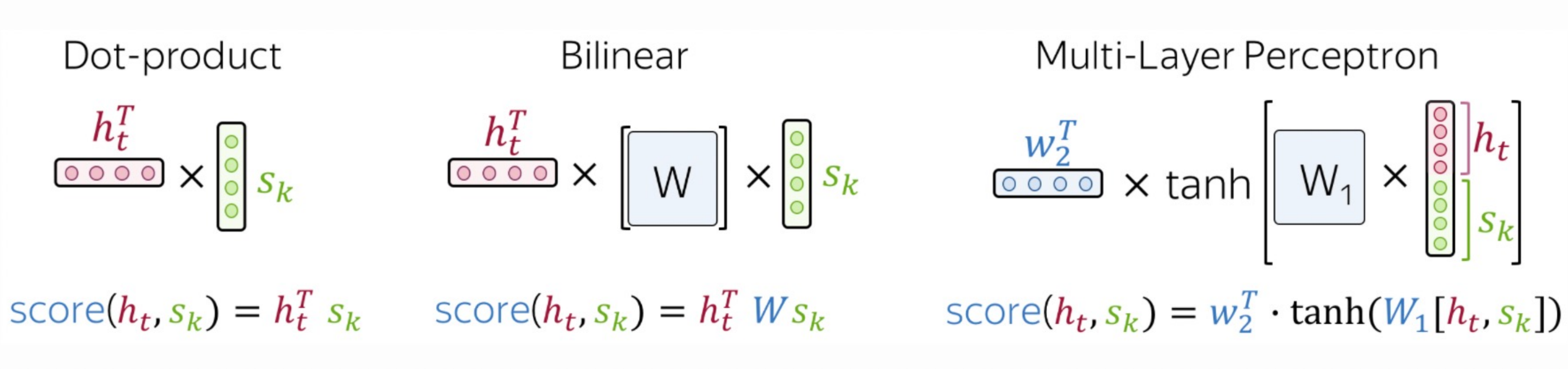
Similarity Measure for Hidden States of the Decoder and Encoder



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

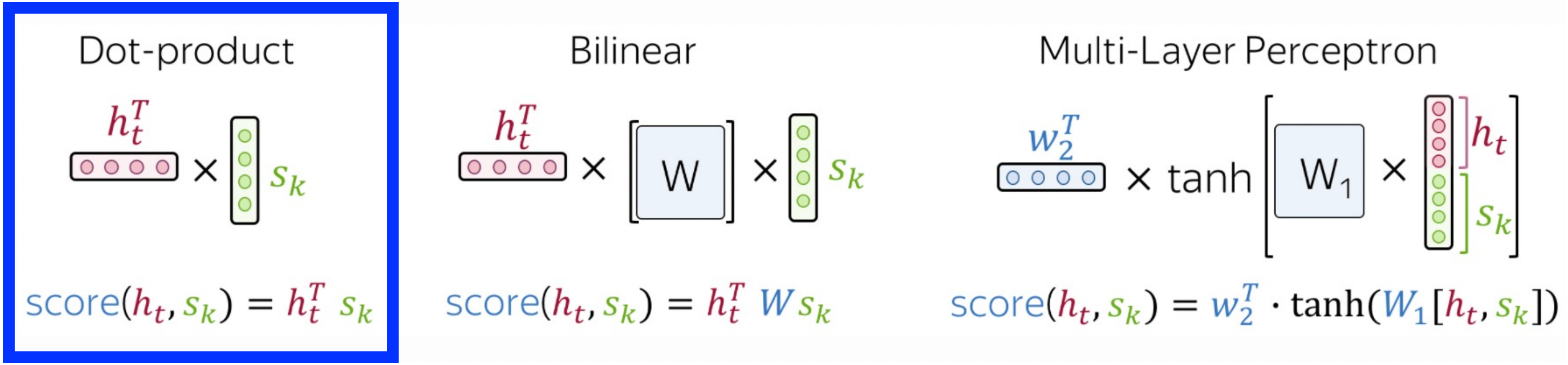
- Many options (function should be differentiable)



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)

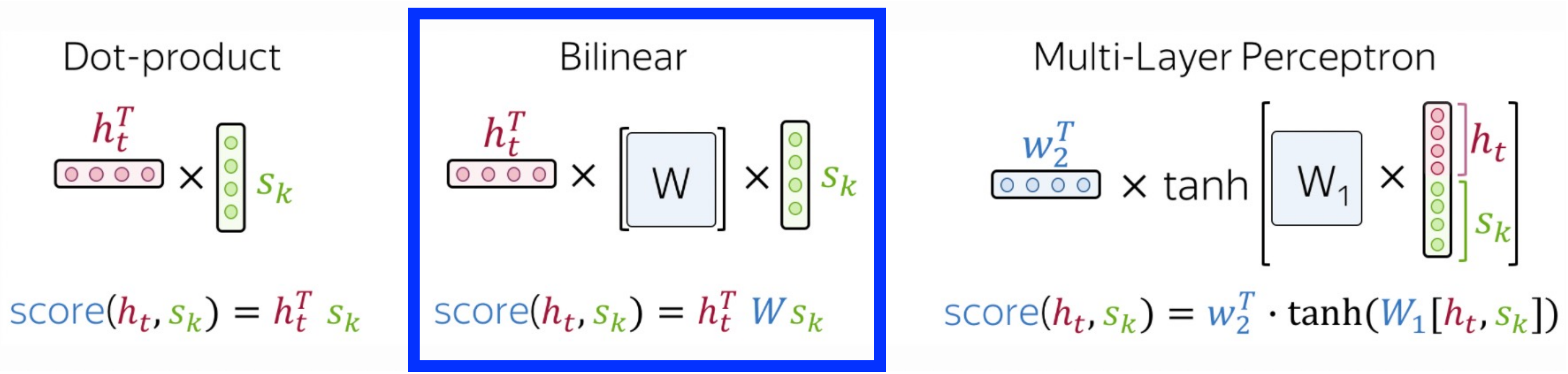


What model parameters must be learned when using dot-product?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)

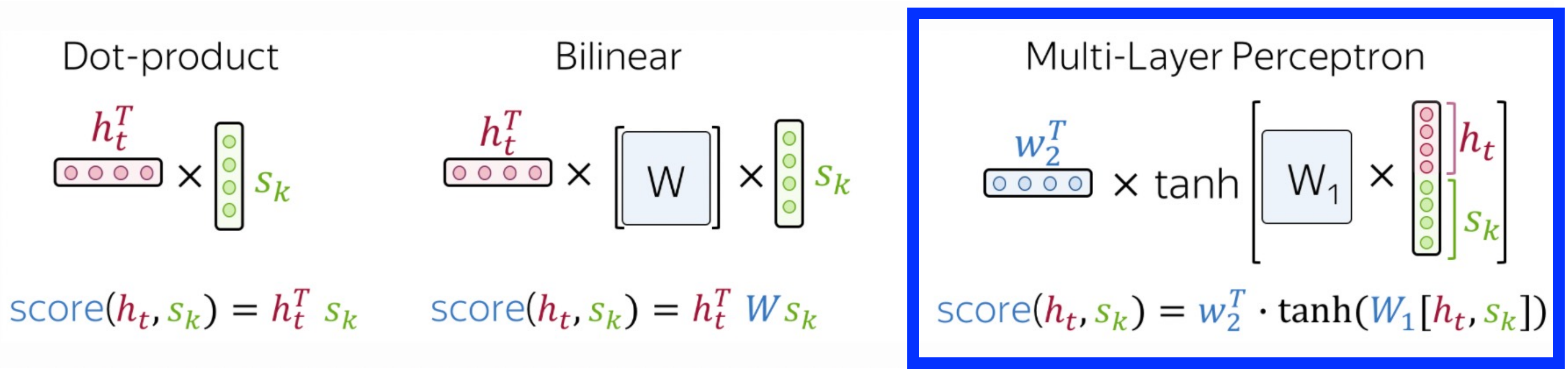


What model parameters must be learned when using bilinear?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)

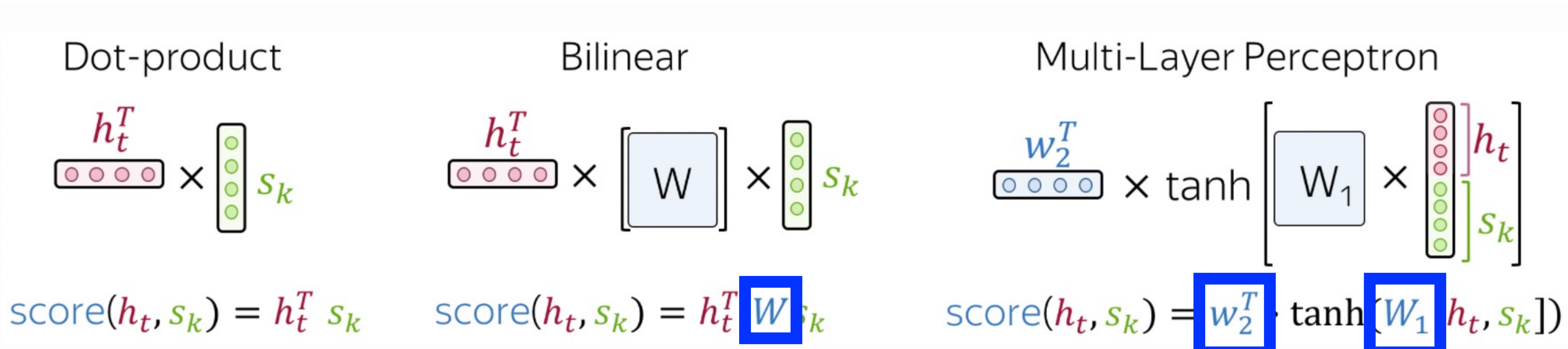


What model parameters must be learned when using multi-layer perceptron?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)



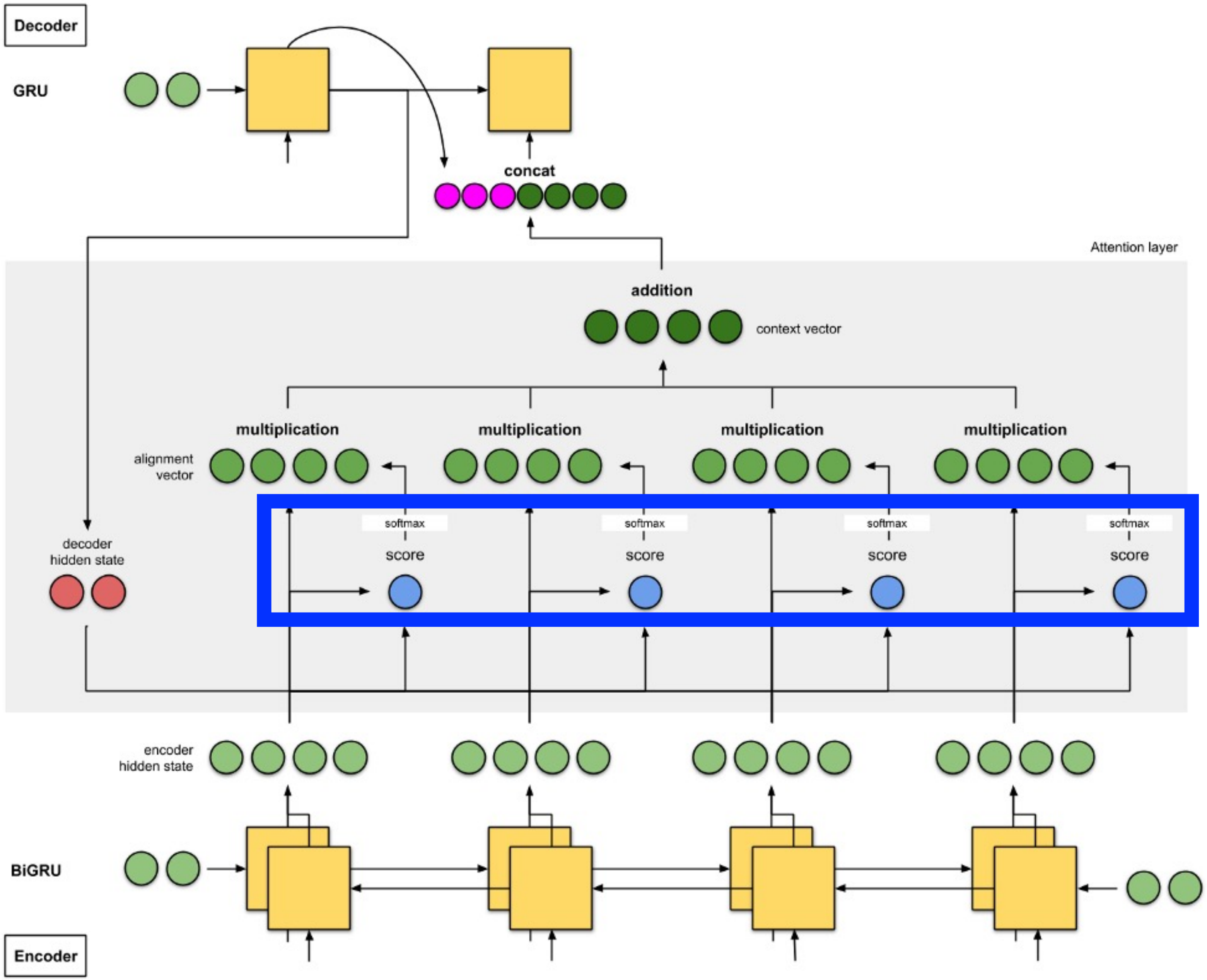
(no parameters)

Model parameters that must be learned

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Measuring Each Input's Influence on the Prediction

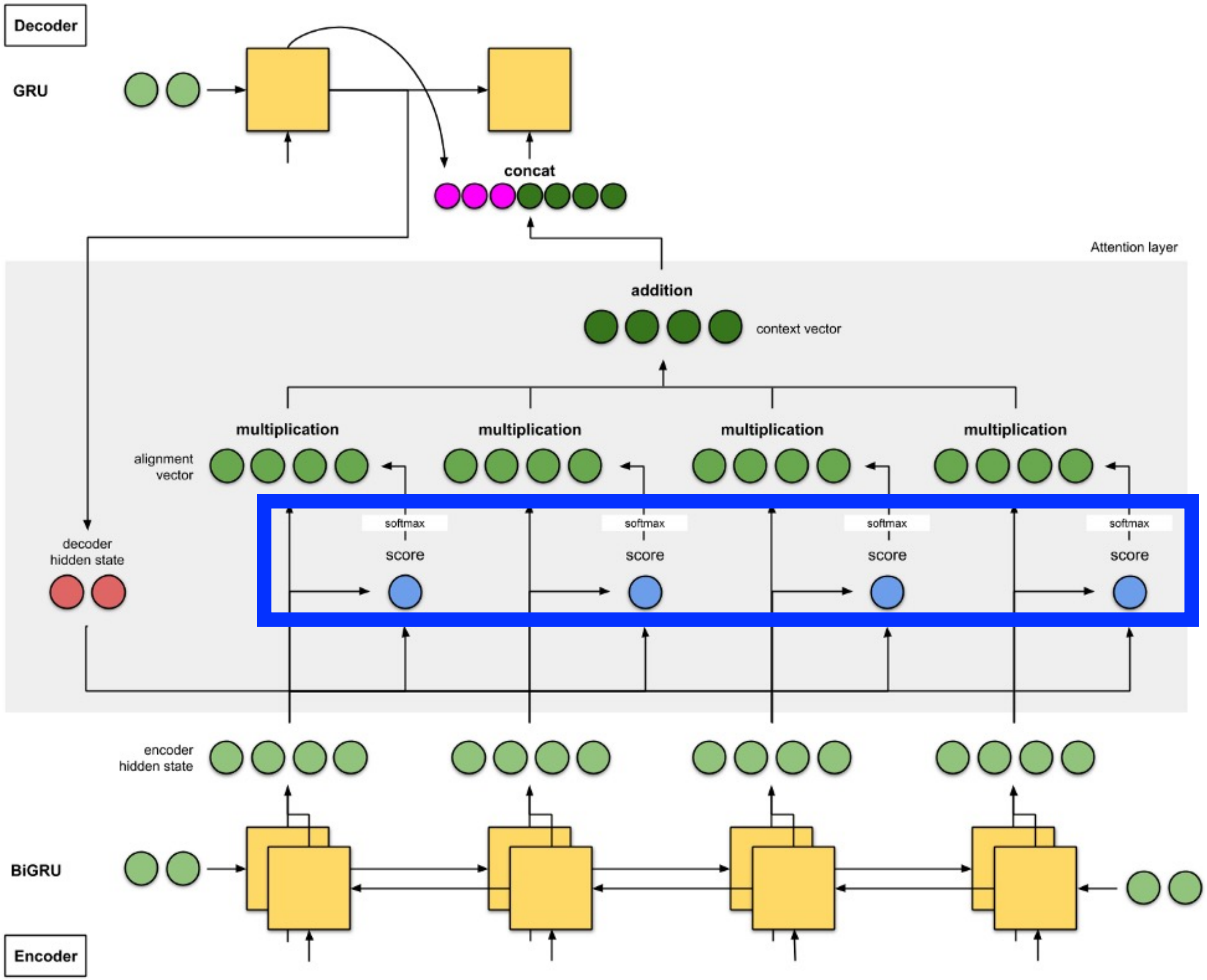
After computing the similarity scores for each input, then apply softmax so all inputs' weights sum to 1



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

We now have our attention weights!

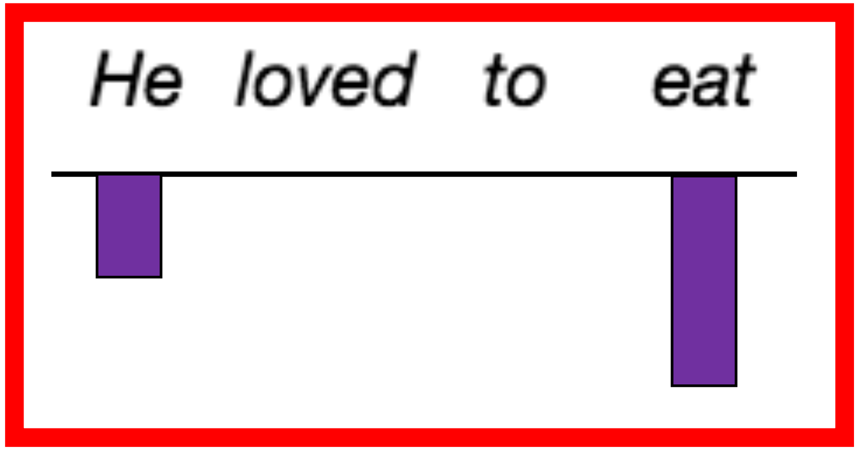


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

Intuitively:

Input



The model can weight each input at each time step!

Target

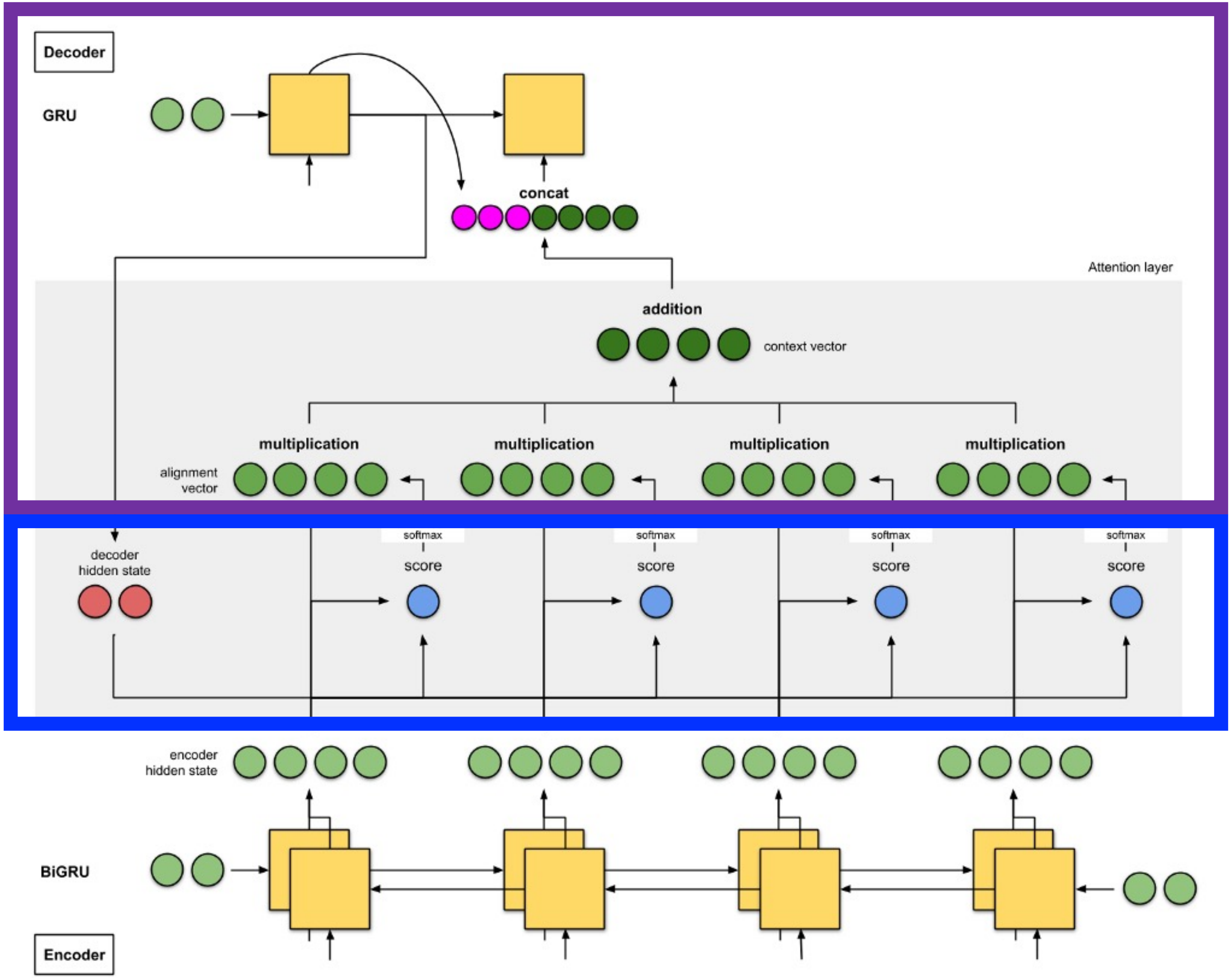
Er liebte zu essen

t = 4

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction



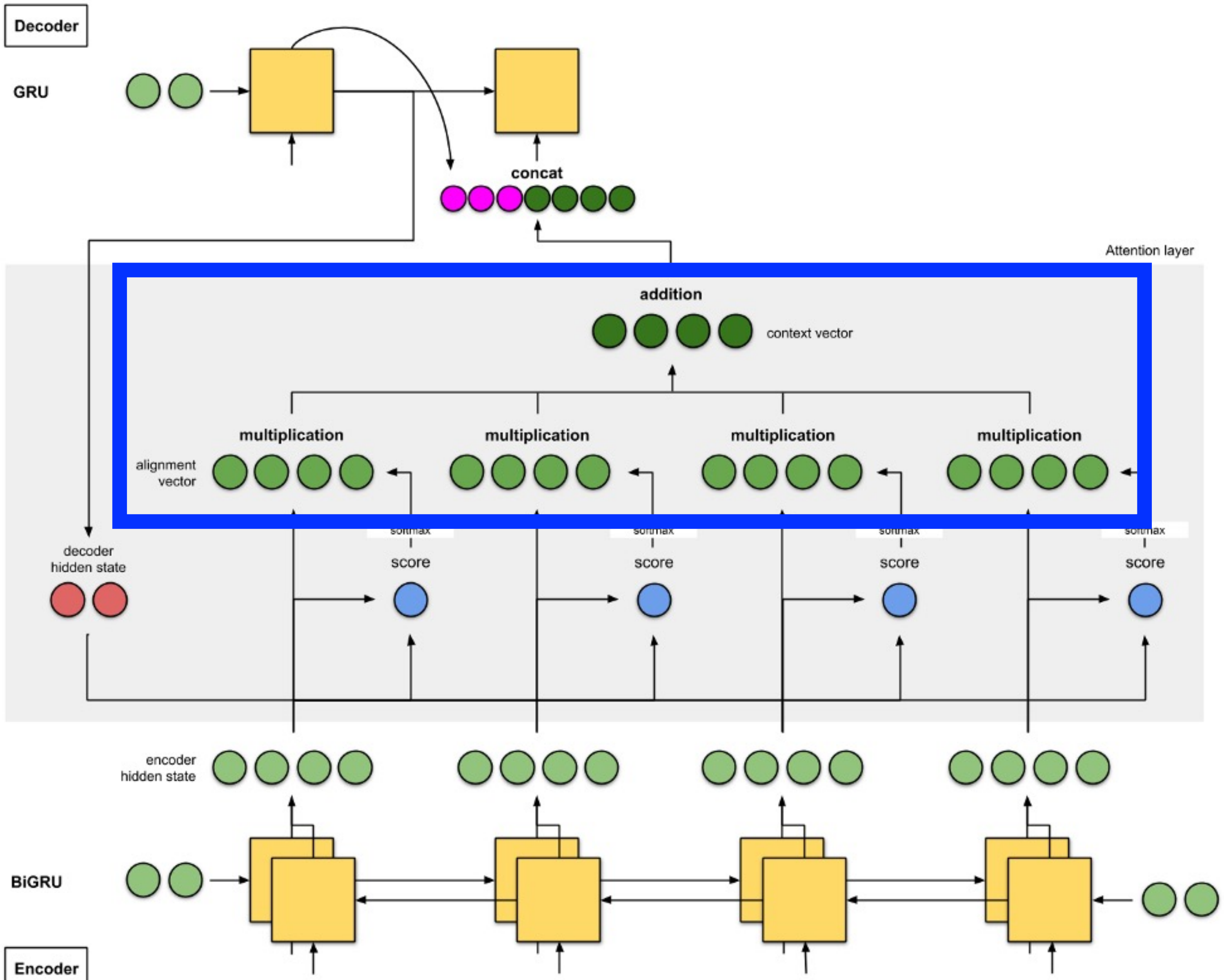
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Word Prediction

We compute at time step *t* for all *n* inputs a weighted sum*:*

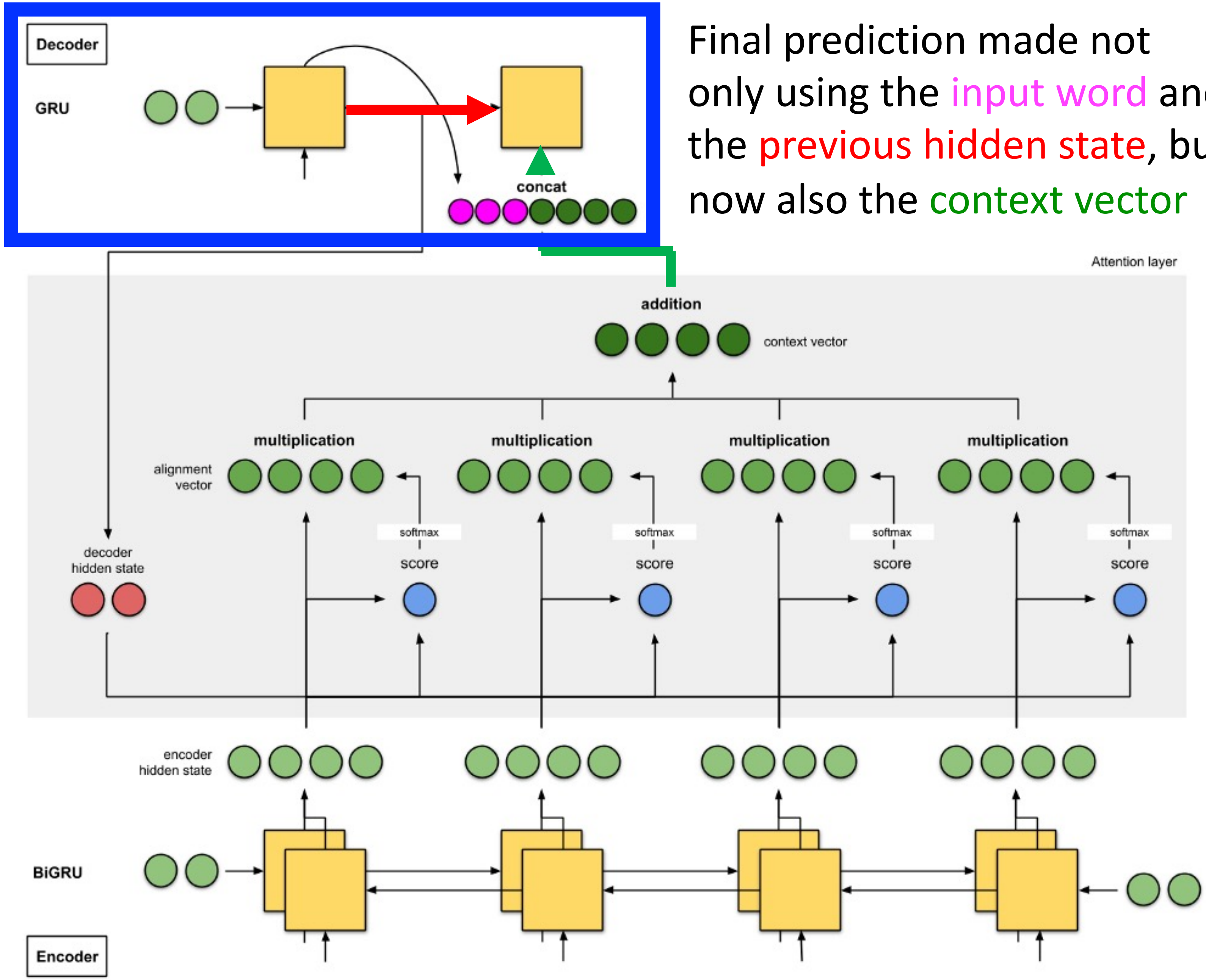
$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

The influence of inputs are **amplified** for large attention weights and repressed otherwise



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

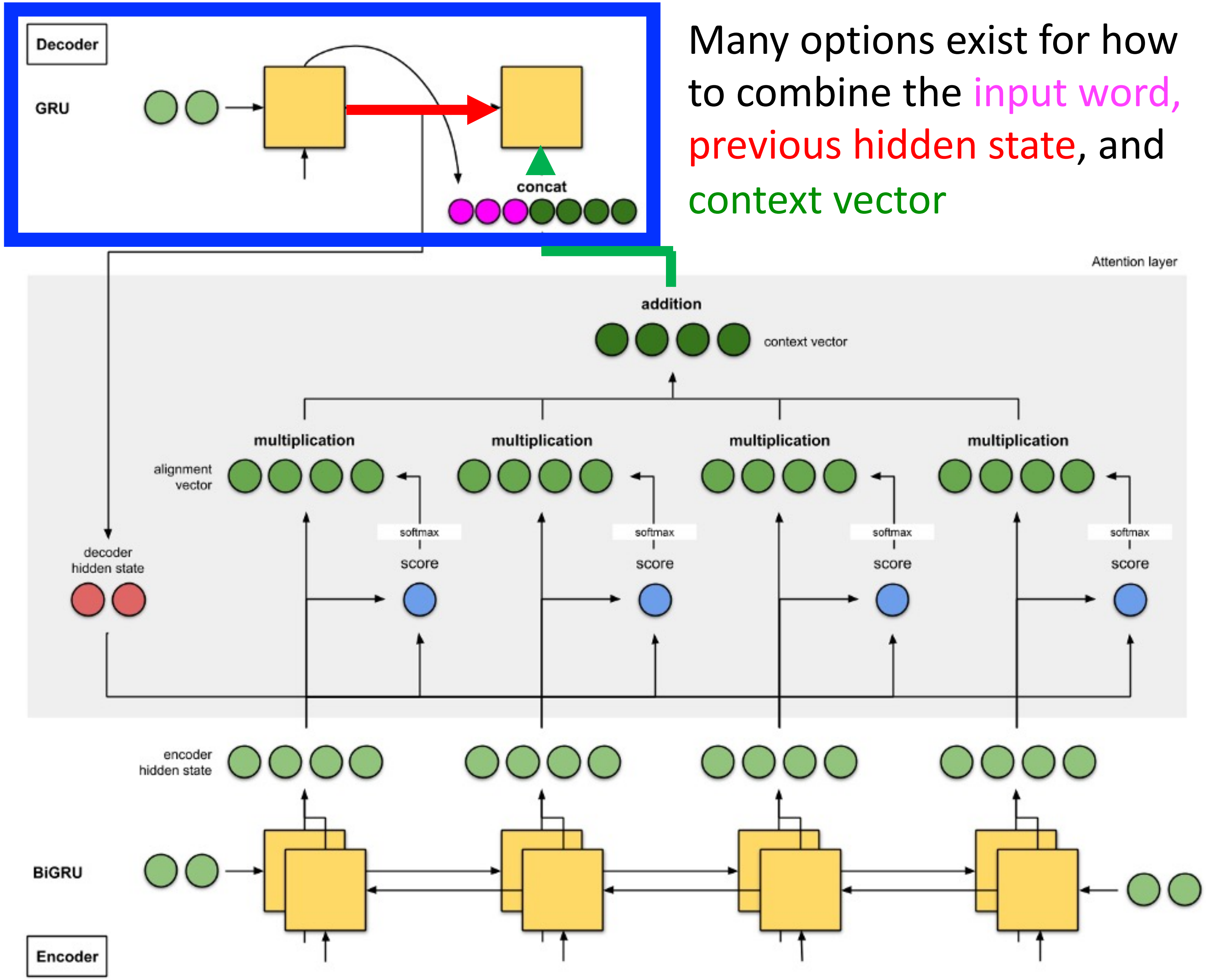
Word Prediction



Final prediction made not only using the input word and the previous hidden state, but now also the context vector

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Word Prediction



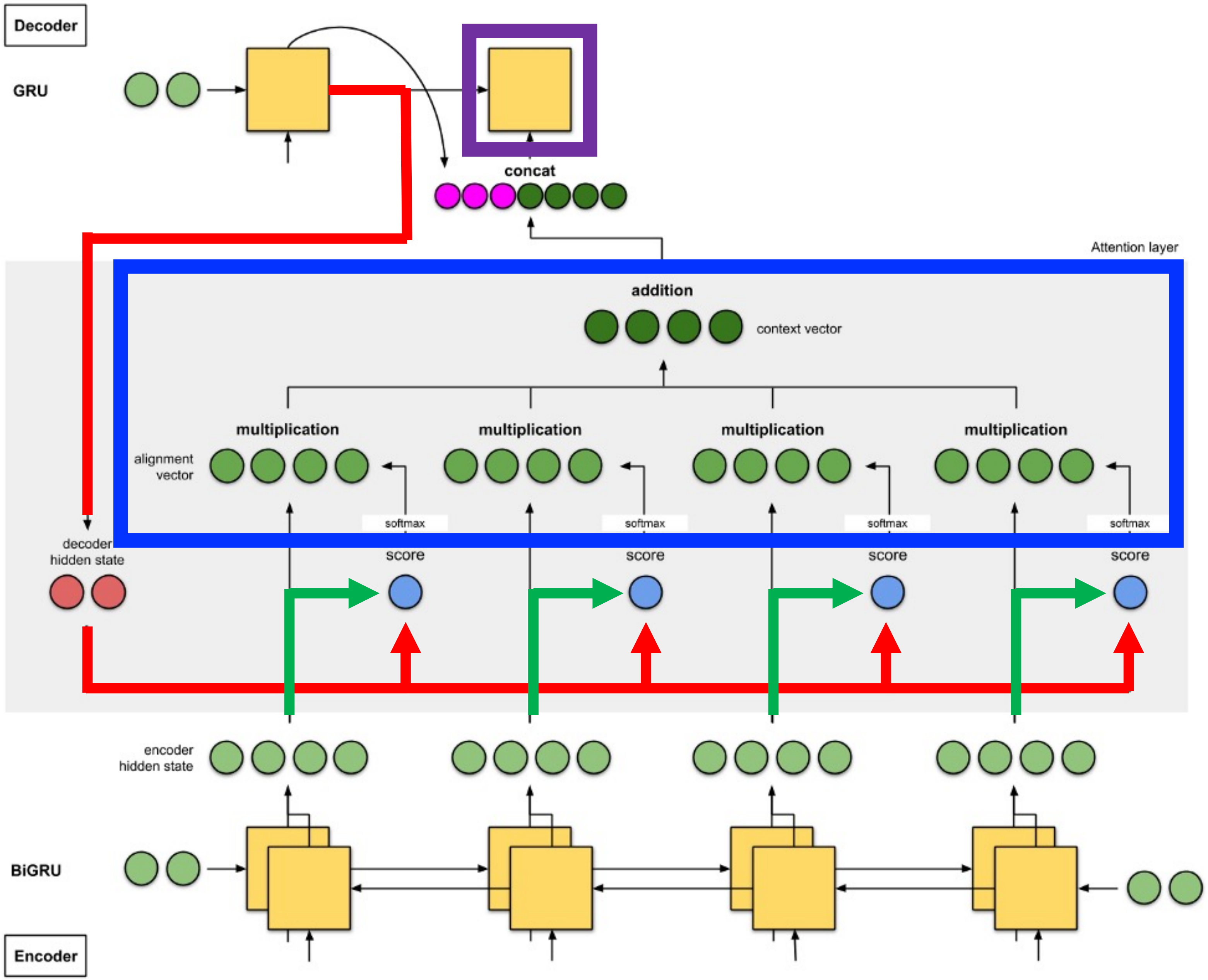
Many options exist for how to combine the input word, previous hidden state, and context vector

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Solution

What stays the same at each decoder time step?
- input's hidden state

What changes at each decoder time step?
- decoder's hidden state
- and so attention weights and context vector



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Summary: Attention (Computations at Each Decoder Step)

Decoder decides which inputs are needed for prediction at each time step with "soft attention", which results in a weighted combination of the input

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\mathrm{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\mathrm{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\mathrm{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

Attention input

$s_1, s_2, \ldots, s_m$ — all encoder states

h_t — one decoder state

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Summary: Attention (Computations at Each Decoder Step)

All parts are differentiable which means end-to-end training is possible

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\mathrm{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\mathrm{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\mathrm{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

Attention input

$s_1, s_2, \ldots, s_m$ — all encoder states

h_t — one decoder state

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Popular Choices for Encoding Input

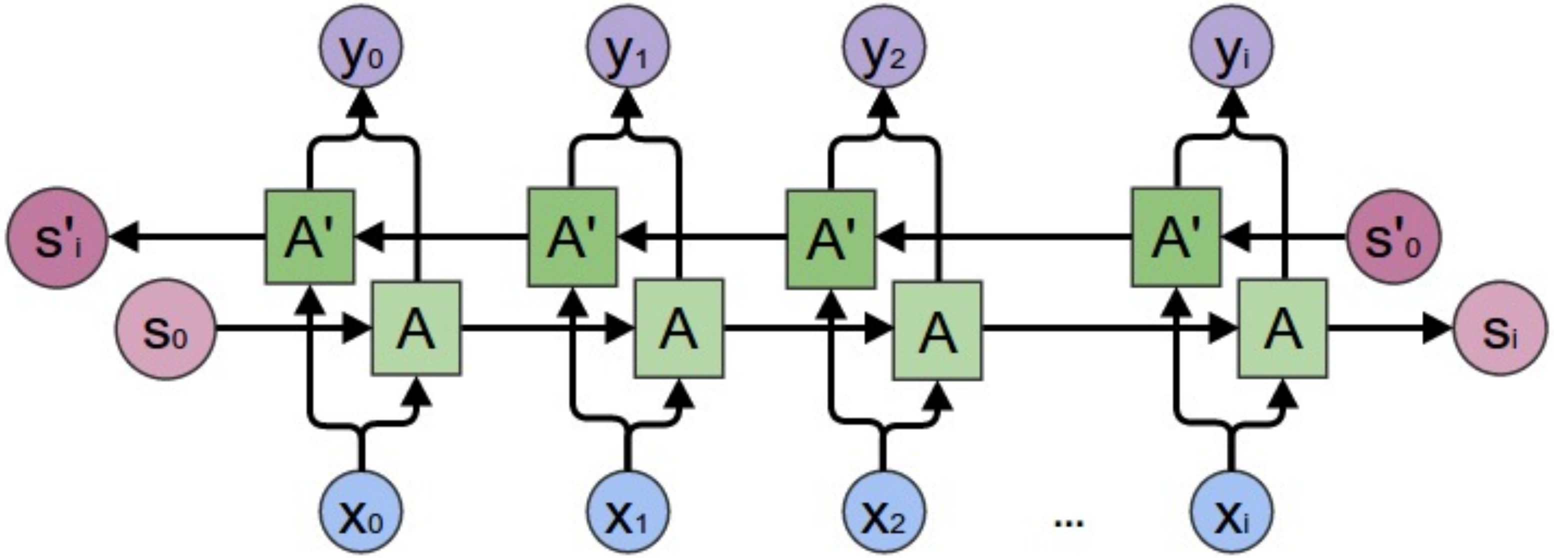
- Bi-directional RNN
- Stacked RNNs

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

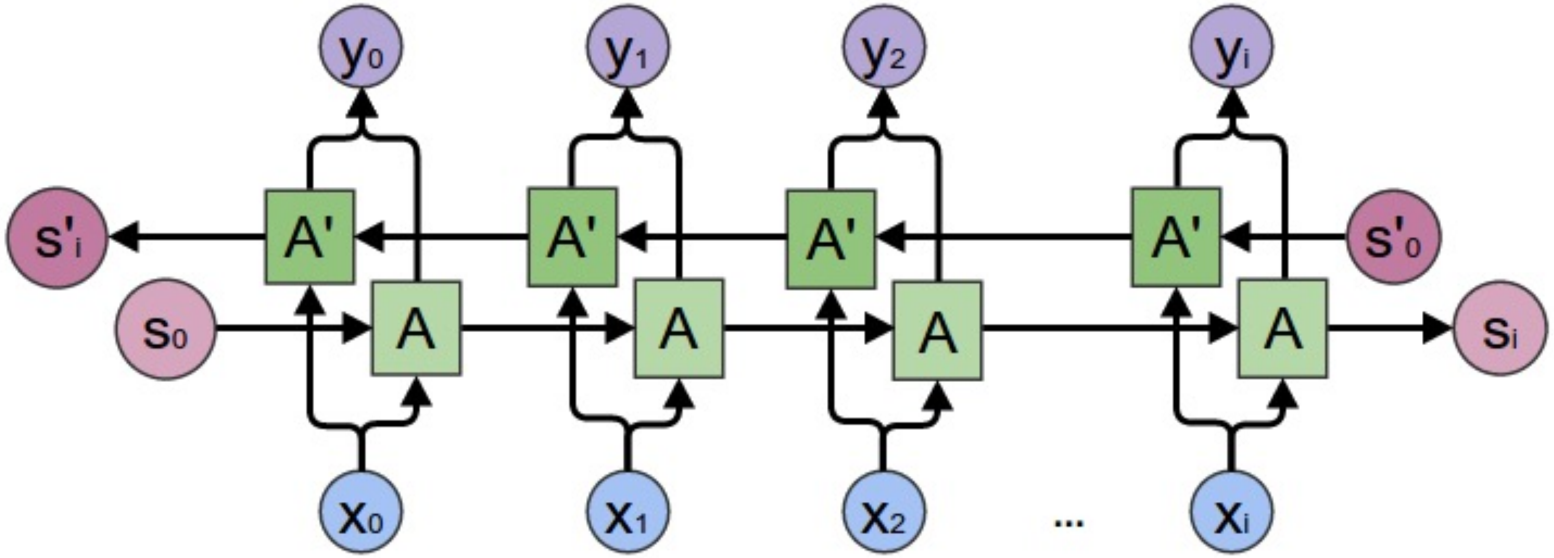


What are advantages of a bi-directional RNN compared to a single RNN?

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

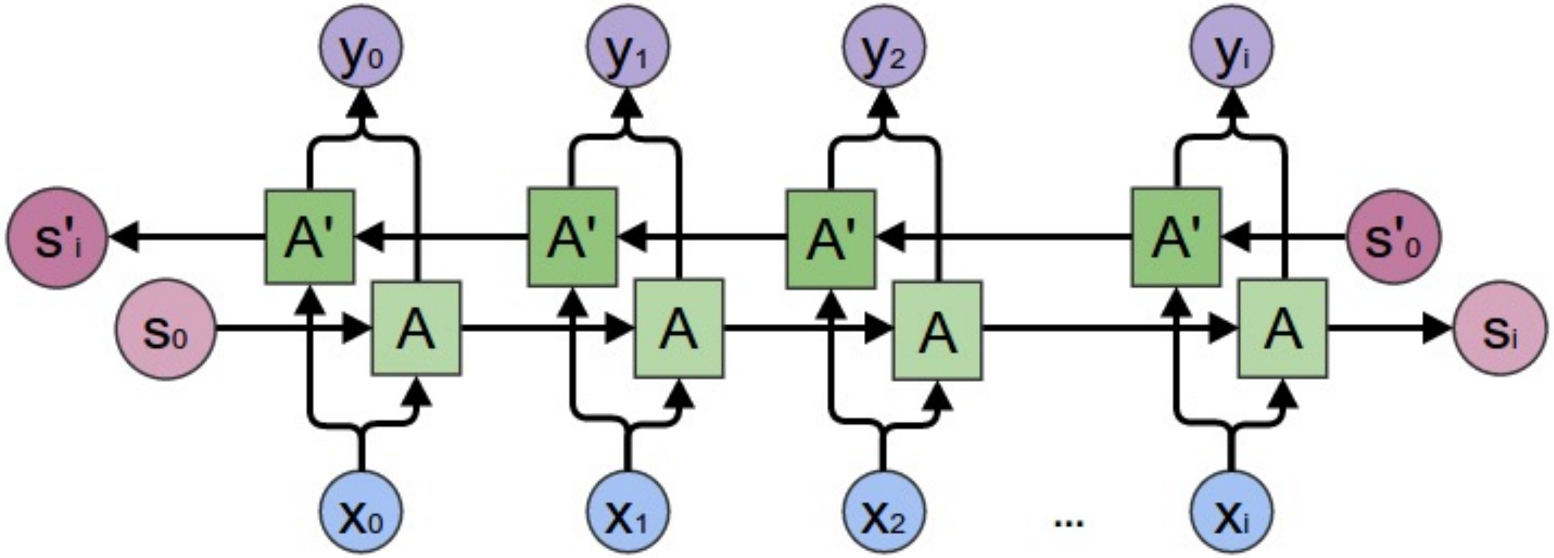


Can use information from the past and **future** to make predictions: e.g., can resolve for "Teddy is a …?” if Teddy refers to a “bear” or former US President Roosevelt

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

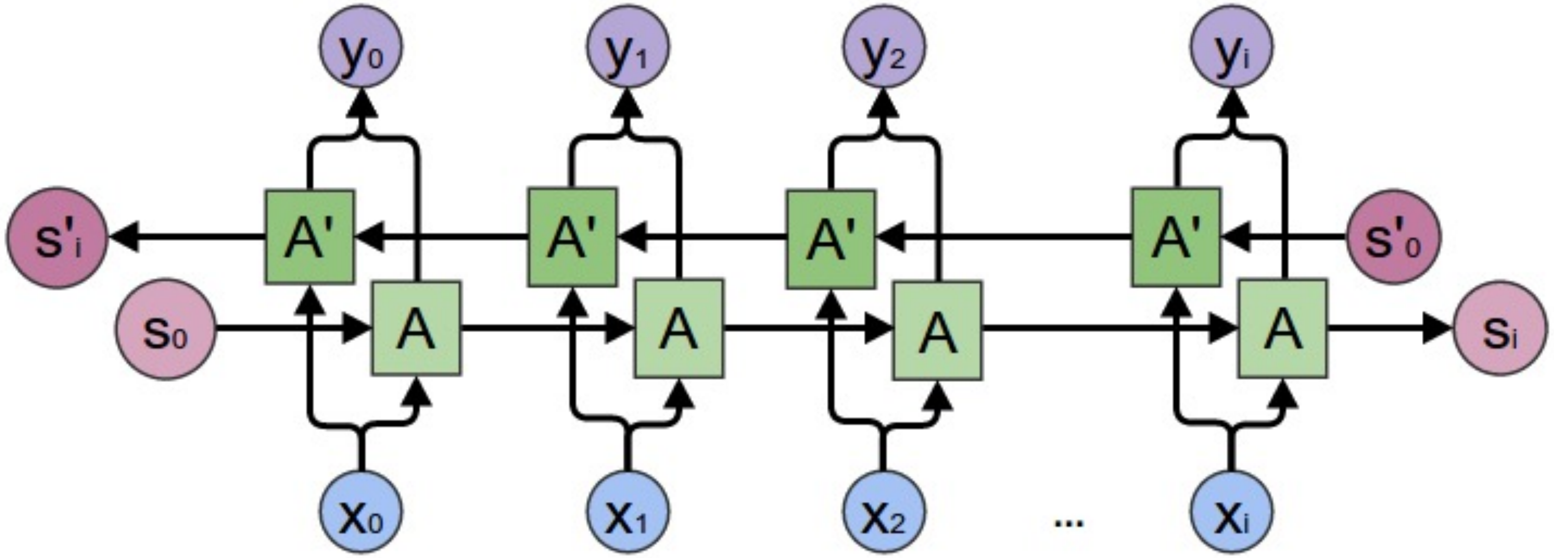


What are disadvantages of a bi-directional RNN compared to a single RNN?

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

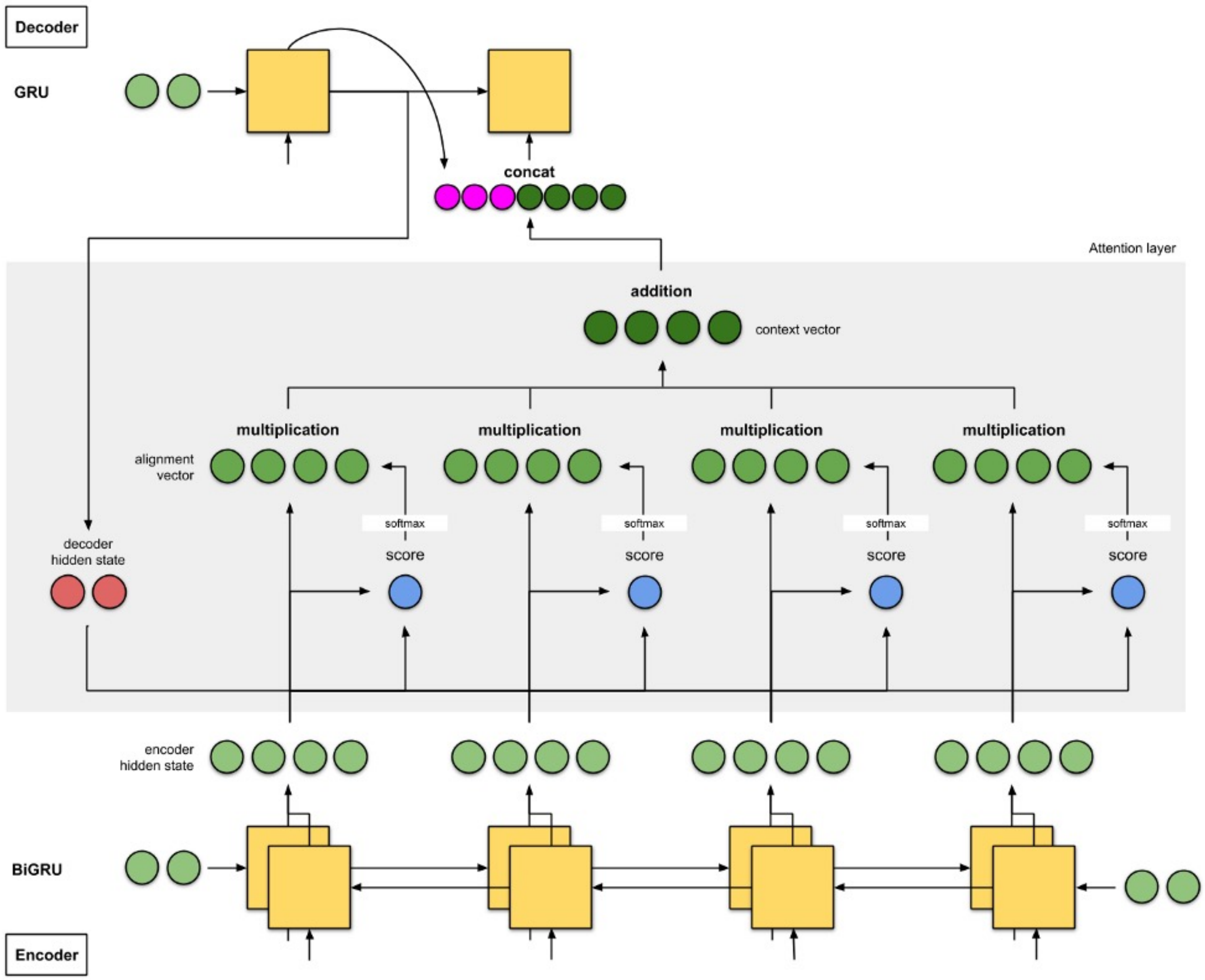
- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state



Entire sequence must be observed to make a prediction (e.g., unsuitable for text prediction)

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Bahdanau's Neural Machine Translation



Bi-directional RNN

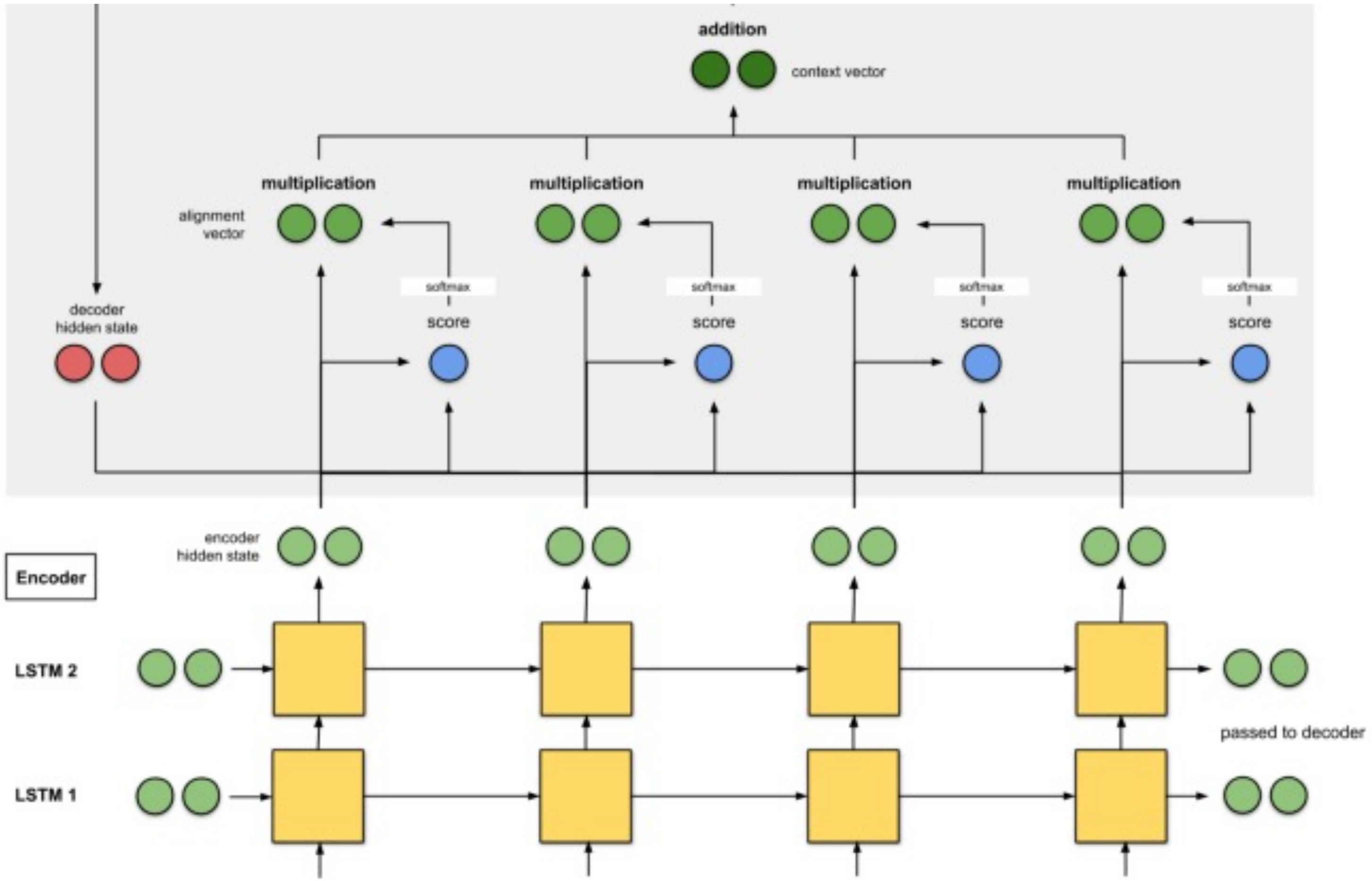
Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Luong’s Neural Machine Translation

2 layers



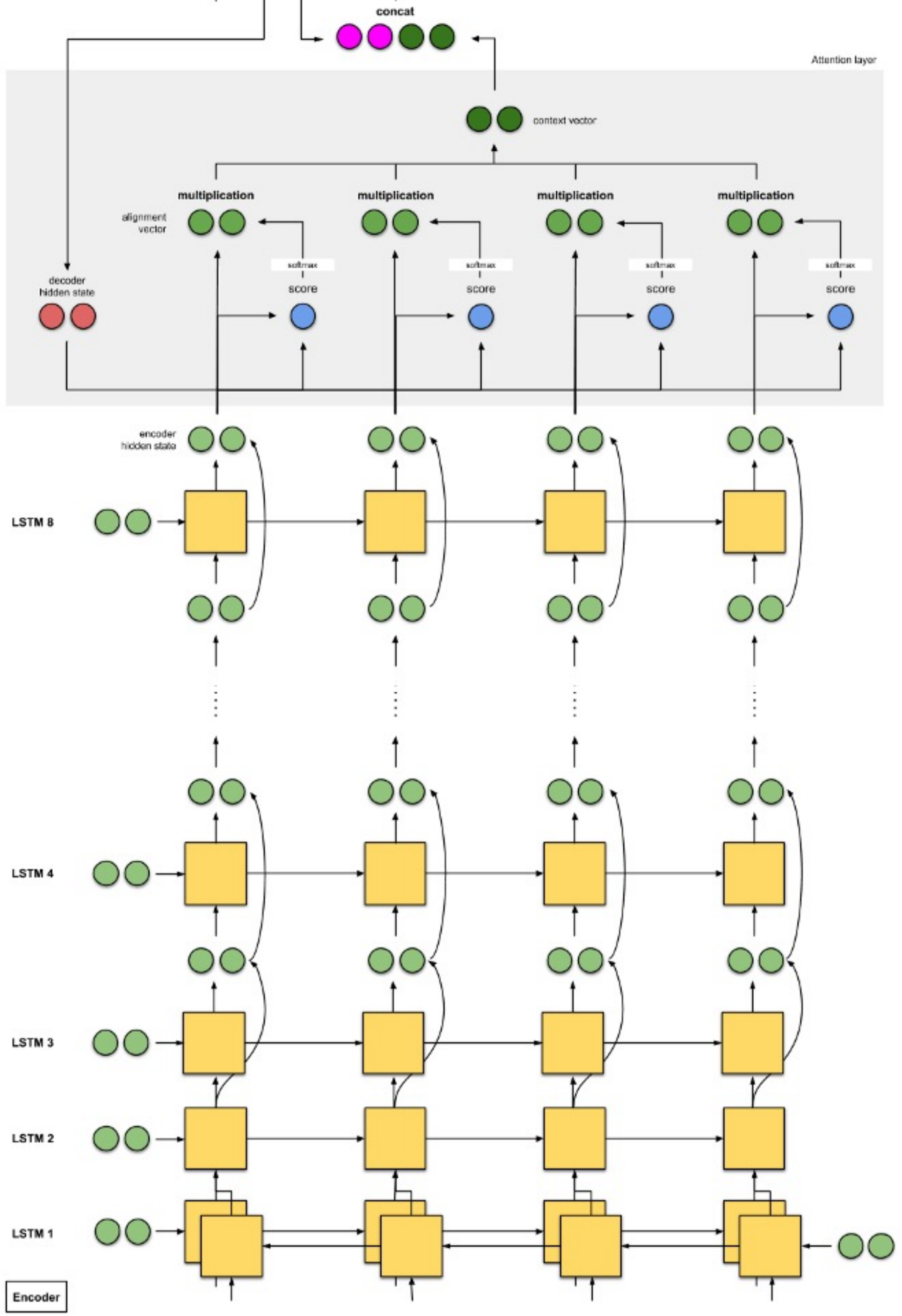
Luong et al. Effective Approaches to Attention-based Neural Machine Translation. EMNLP 2015
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#df28

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Google’s Neural Machine Translation

8 layers with 1rst layer bi-directional



Wu et al. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv 2016.
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#df28

Popular Choices for Encoding Input

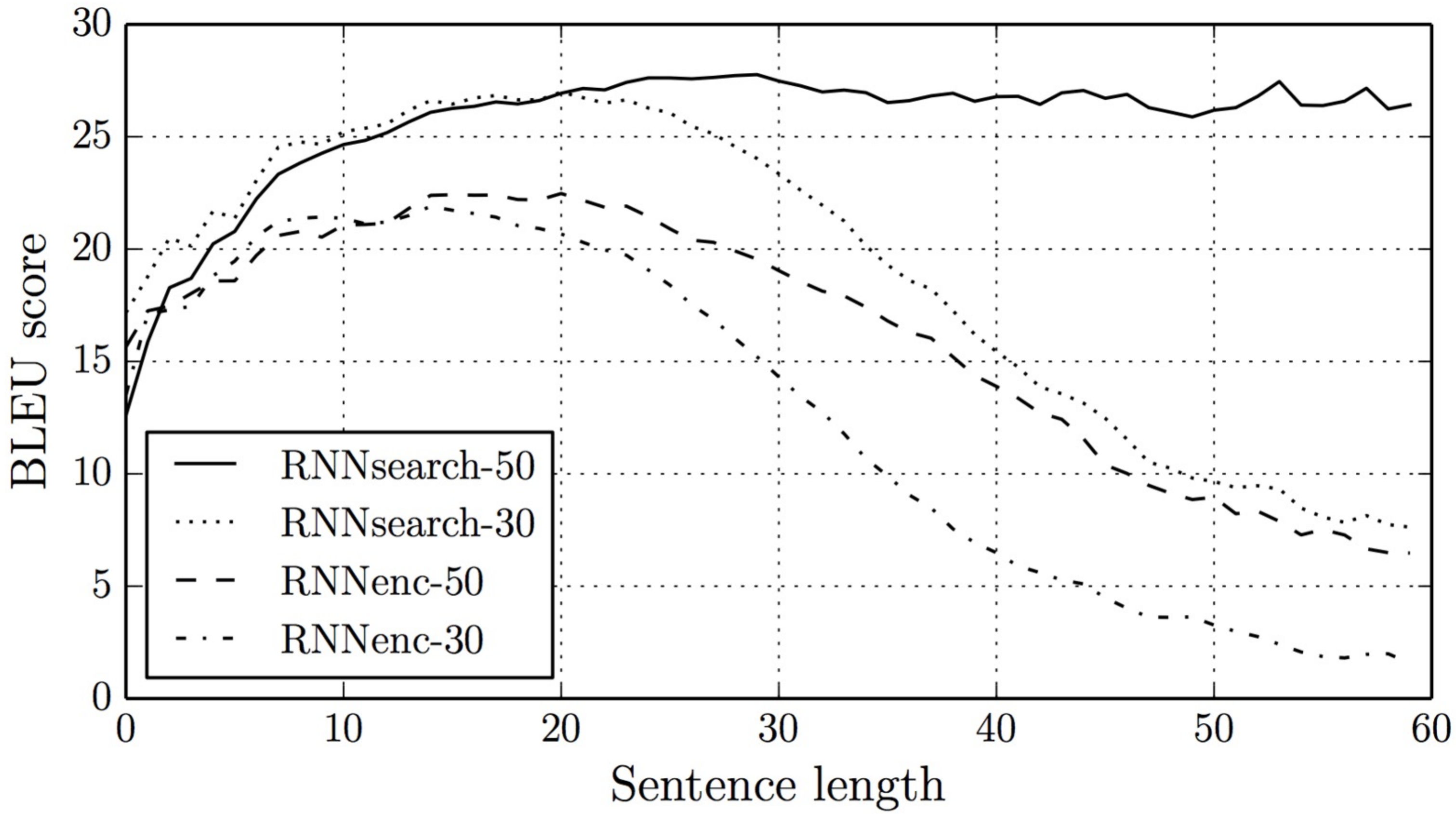
- Bi-directional RNN
- Stacked RNNs

Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Analysis of Attention Models

(larger scores are better)

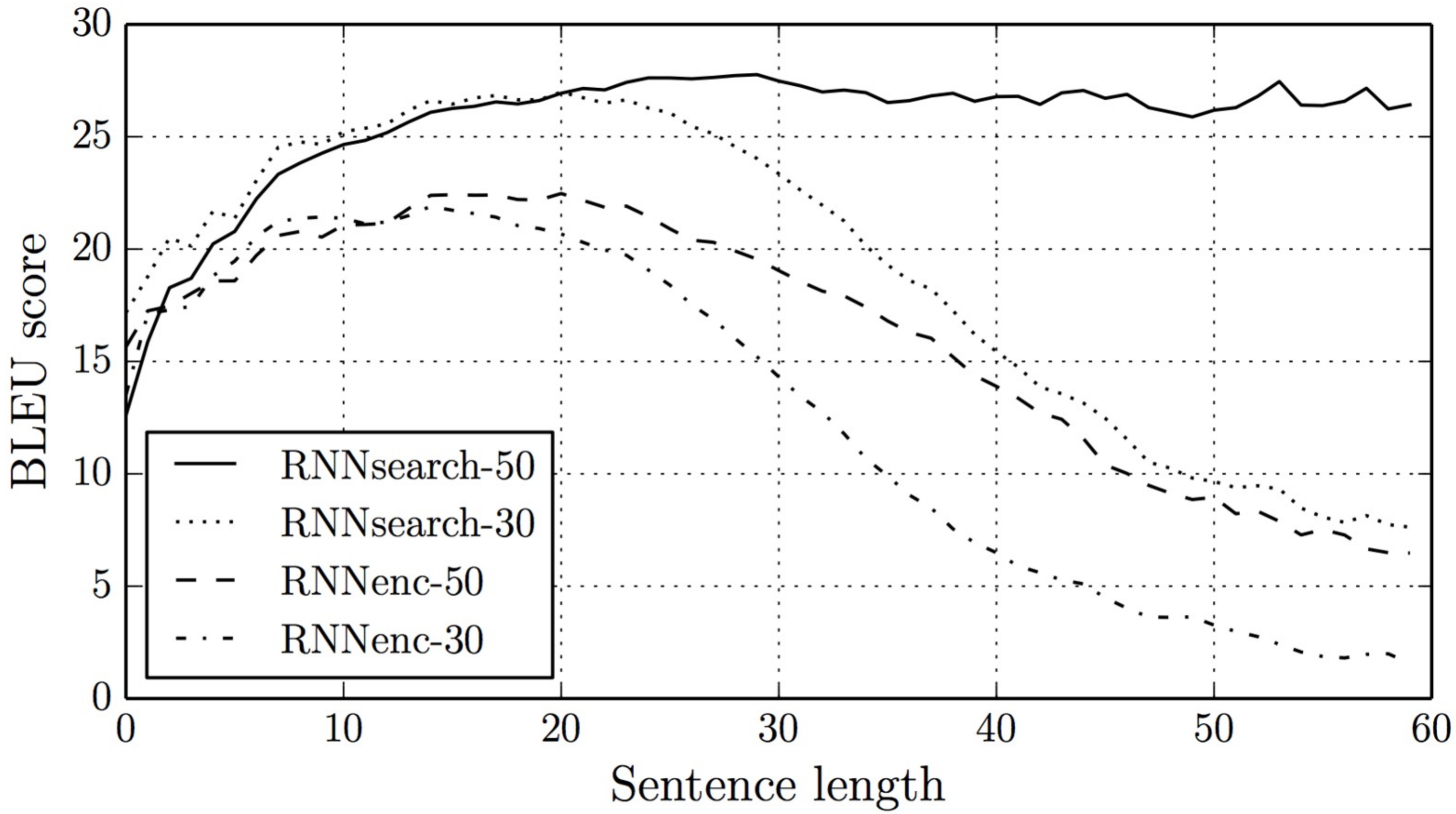


What performance trend is observed as the number of words in the input sentence grows?

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Analysis of Attention Models

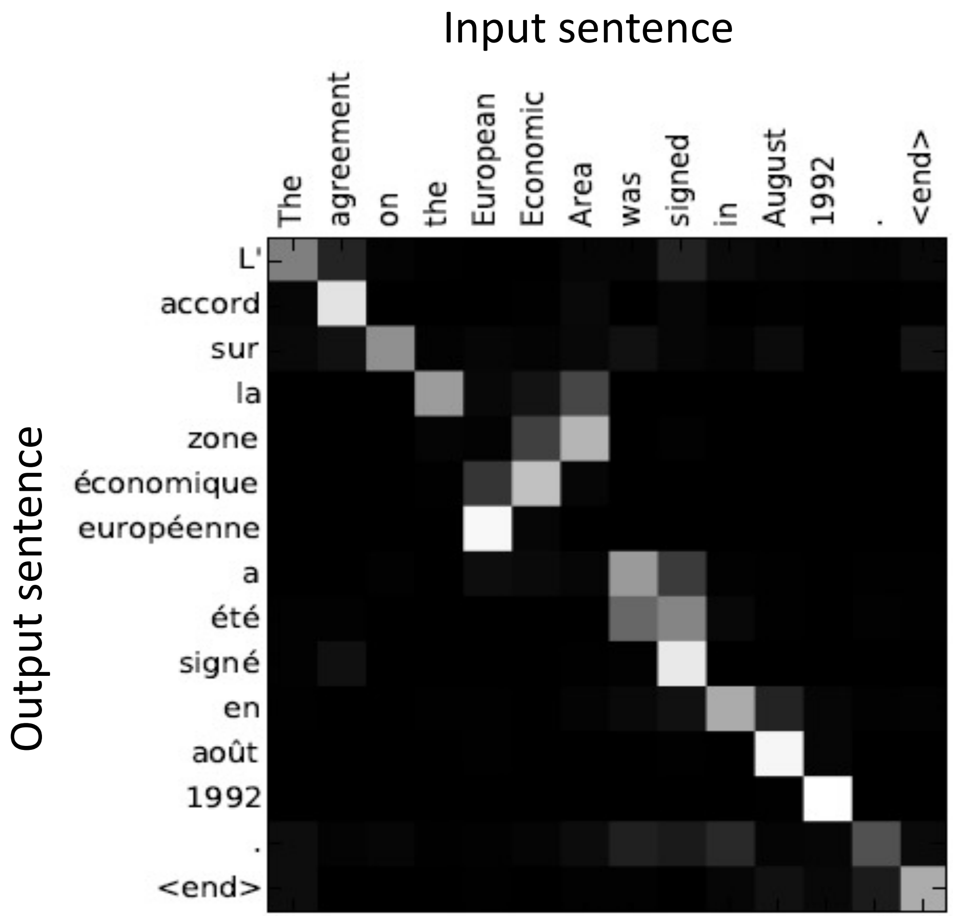
(larger scores are better)

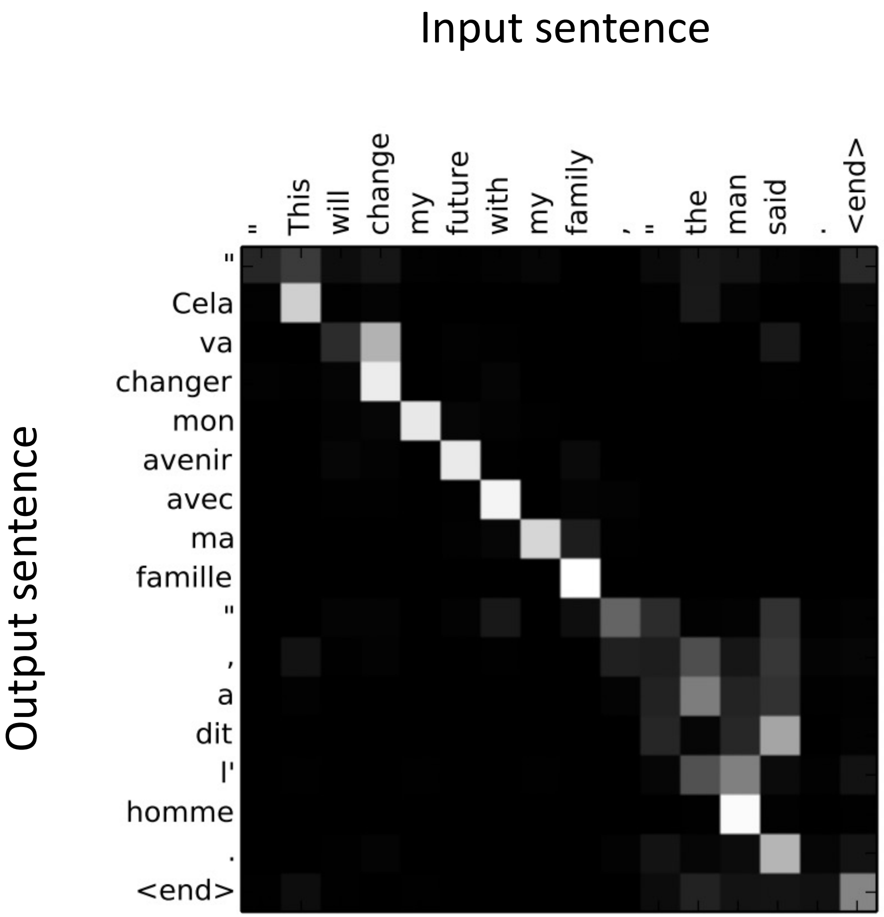


Performance no longer drops for longer sentences!

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

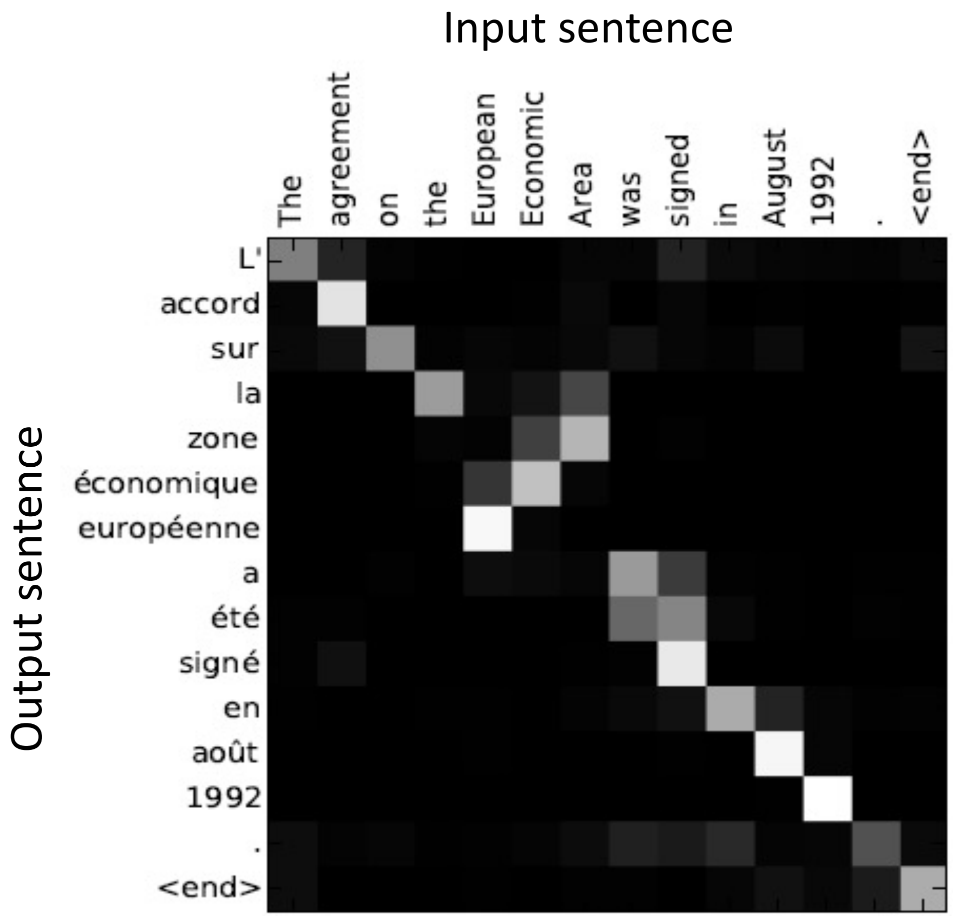


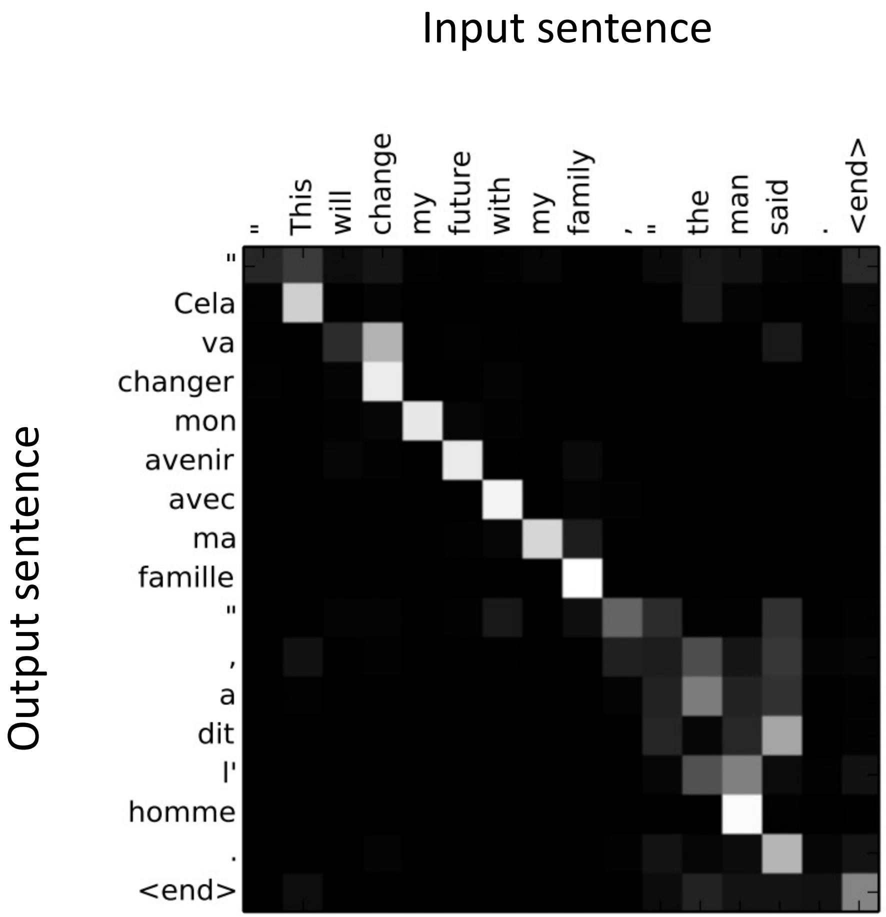


Values are 0 to 1, with whiter pixels indicating larger attention weights

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

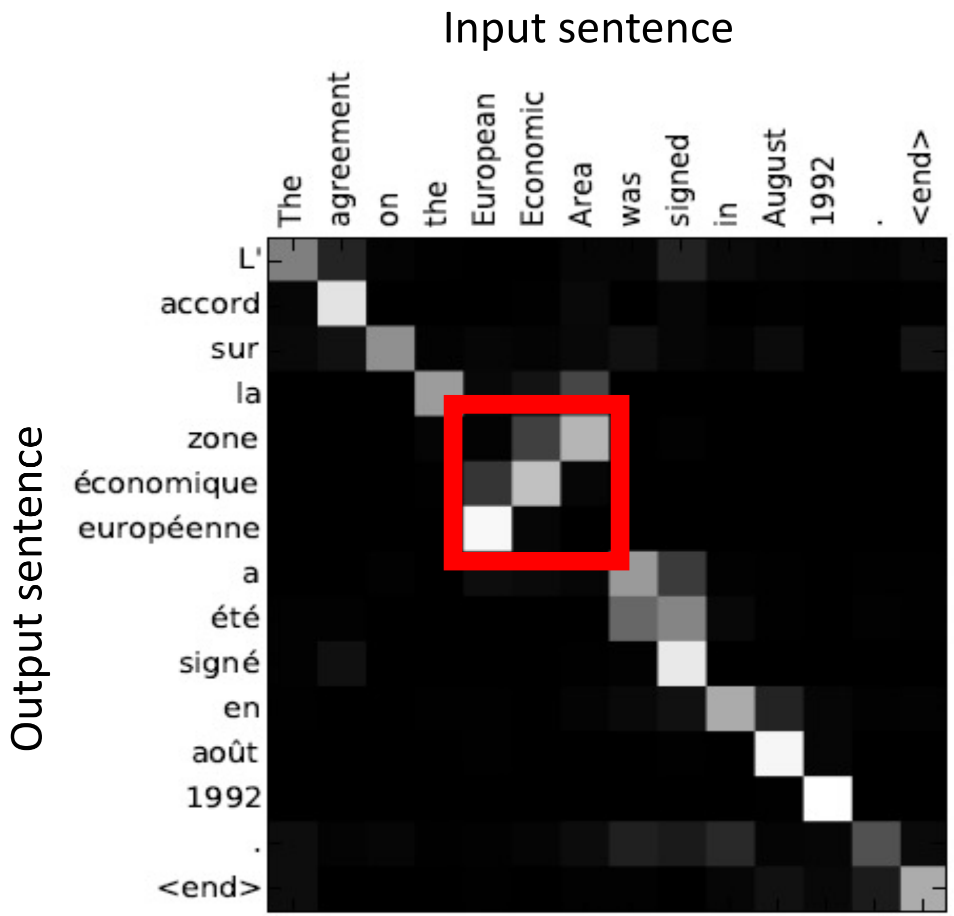


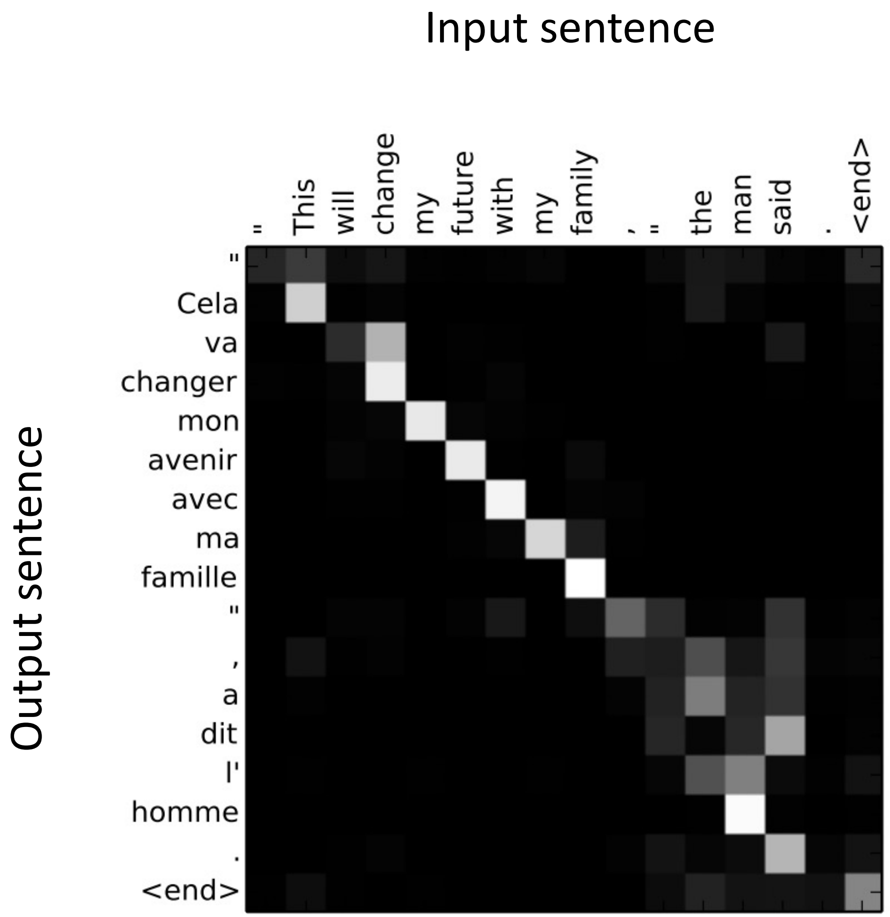


What insights can we glean from these examples?

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

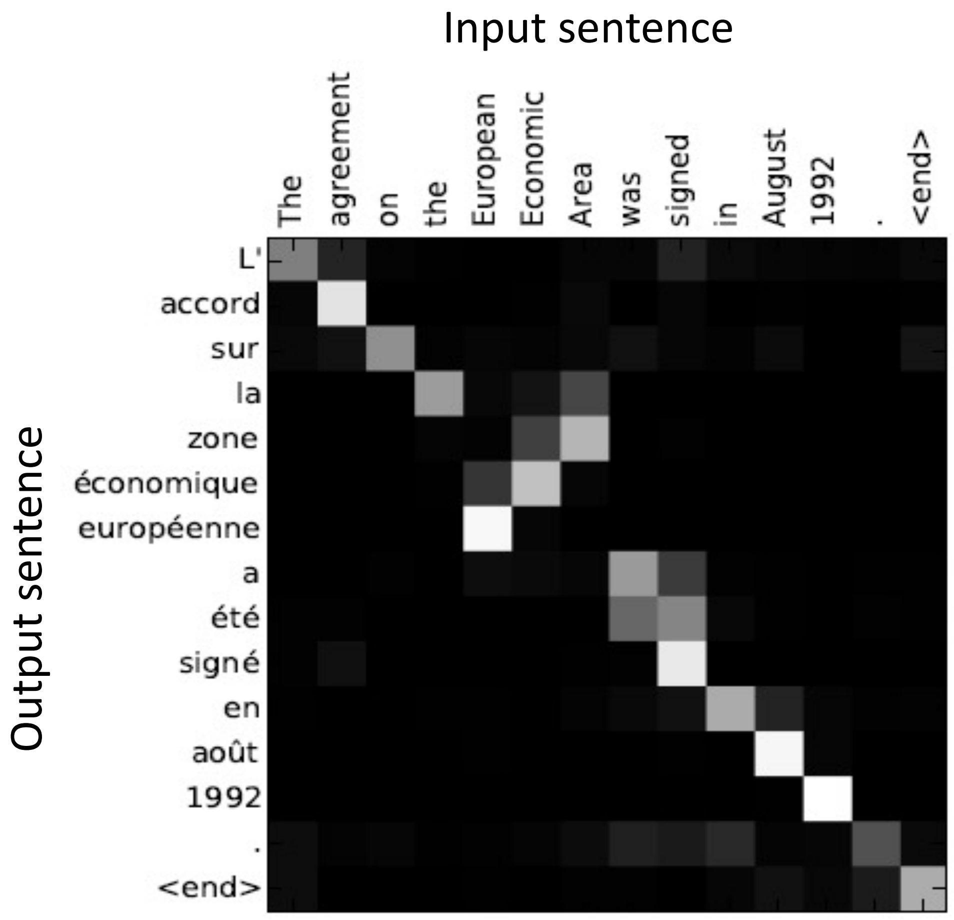


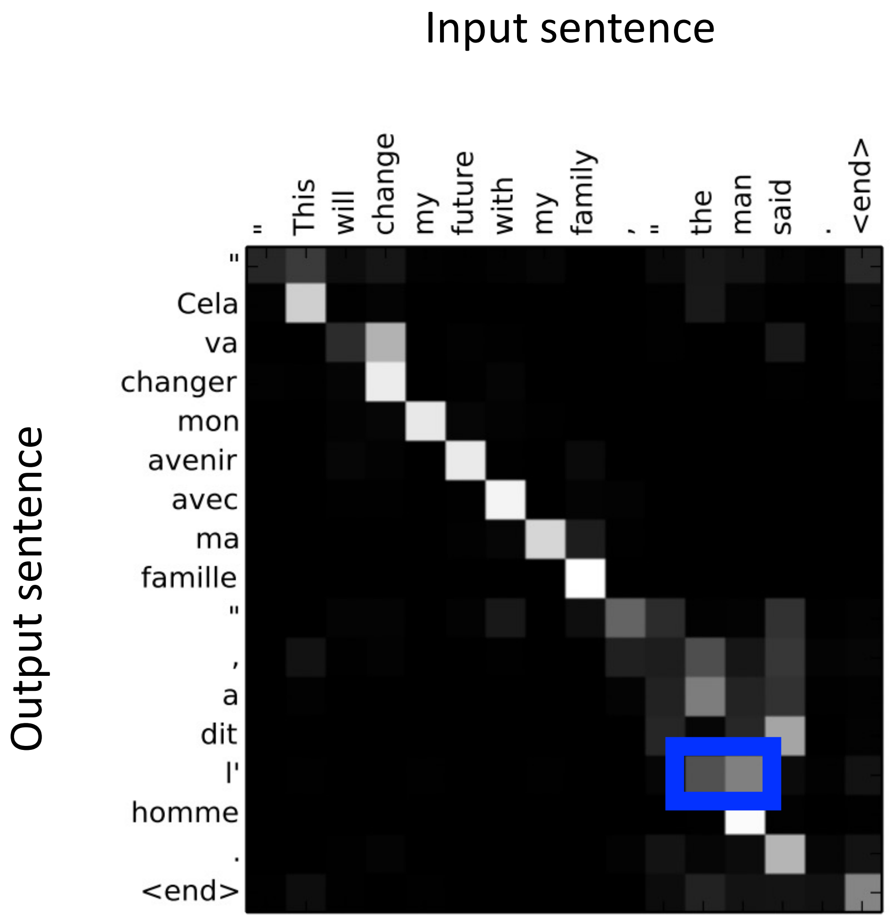


While a linear alignment between input and output sentences is common, there are exceptions (e.g., order of adjectives and nouns can differ)

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

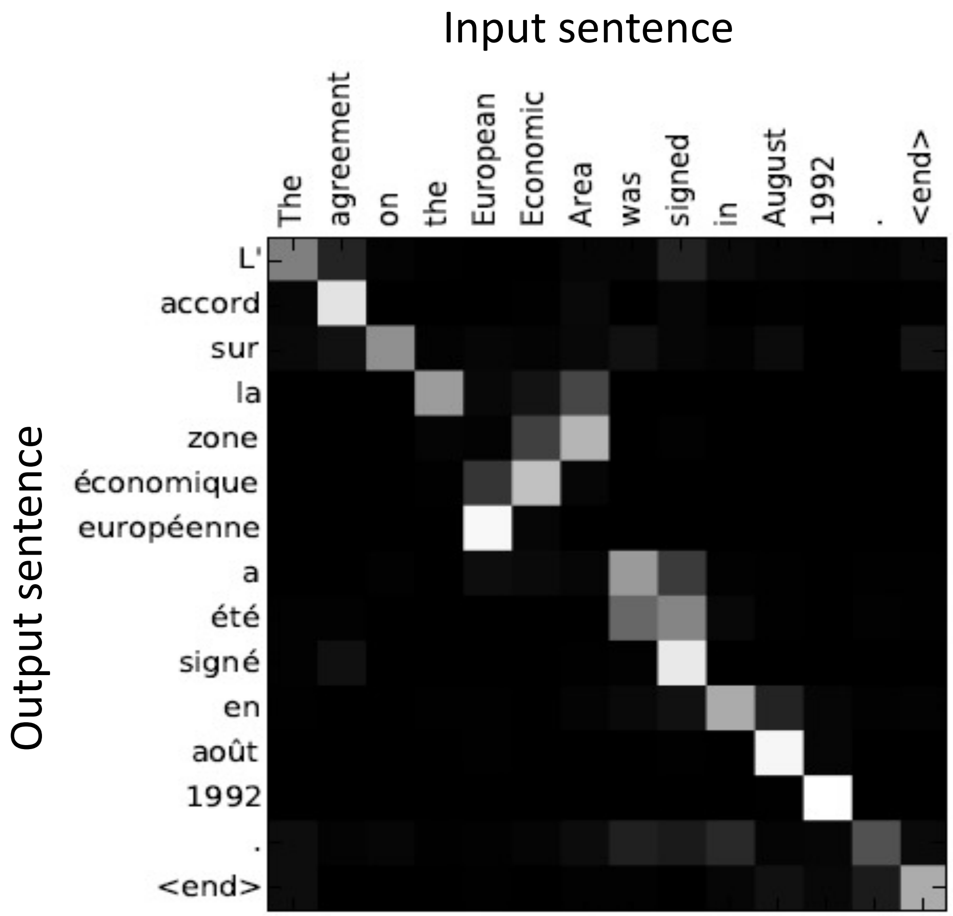


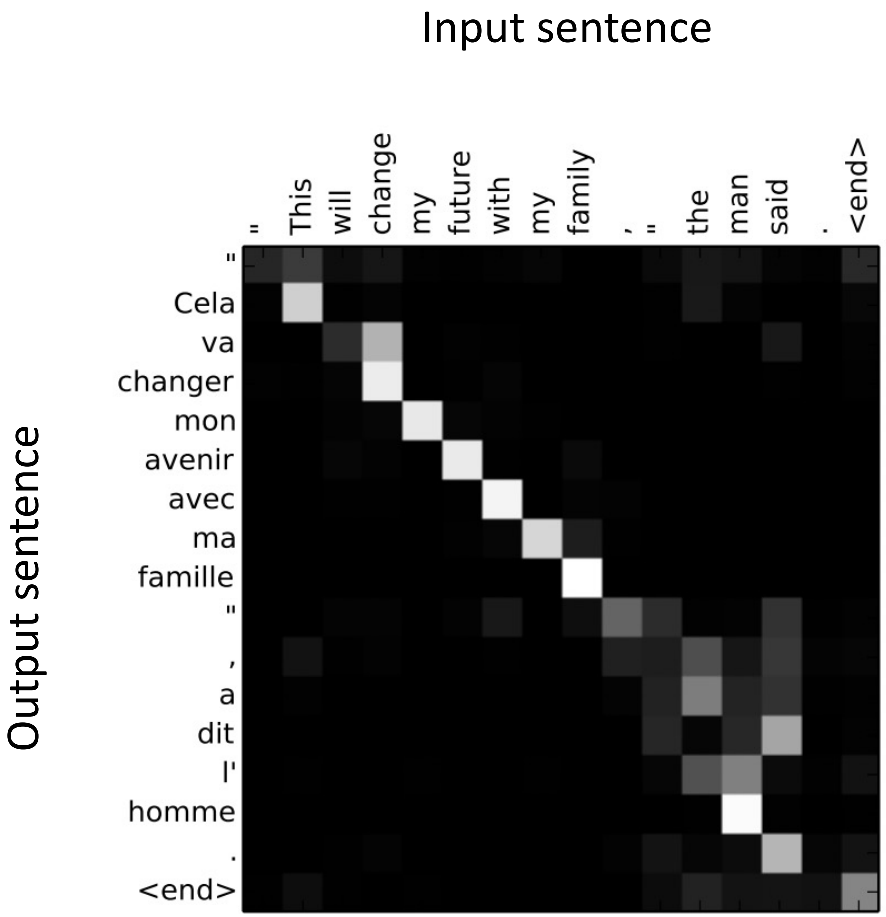


Output words are often informed by more than one input word;
e.g., “man” indicates translation of “the” to l’ instead of le, la, or les

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention





It naturally handles different input and output lengths
(e.g., 1 extra output word for both examples)

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Today’s Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial

Today's Topics

- Motivation: machine neural translation for long sentences
- Decoder: attention
- Encoder
- Performance evaluation
- Programming tutorial


The End