Machine Translation

Session 10: Sequence-to-sequence with attention

Sharid Loáiciga — June 24th, 2020

Slides credits: Alessandro Raganato
(who credits Abigail See & Rico Sennrich)

On the previous episode of NMT course...

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network (end-to-end)*

On the previous episode of NMT course...

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network (end-to-end)*

- Sequence-to-sequence is the architecture for NMT

- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

On the previous episode of NMT course...

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network (end-to-end)*
- Sequence-to-sequence is the architecture for NMT
- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*
- We saw how to generate (or “decode”) the target sentence by beam search

On the previous episode of NMT course...

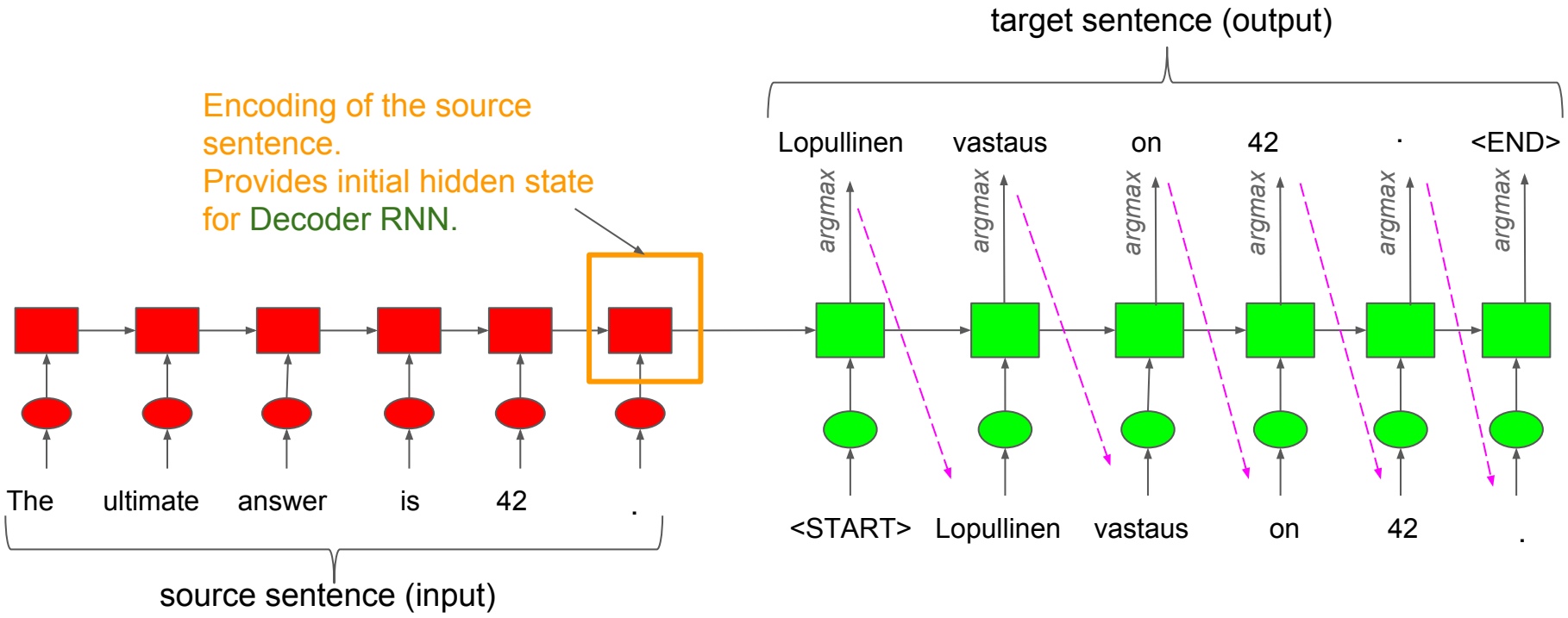
- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network (end-to-end)*
- Sequence-to-sequence is the architecture for NMT
- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*
- We saw how to generate (or “decode”) the target sentence by beam search
- Byte Pair Encoding (BPE) as a solution for Out-Of-Vocabulary (OOV) words

NMT course continues

- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

NMT: Issues

- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

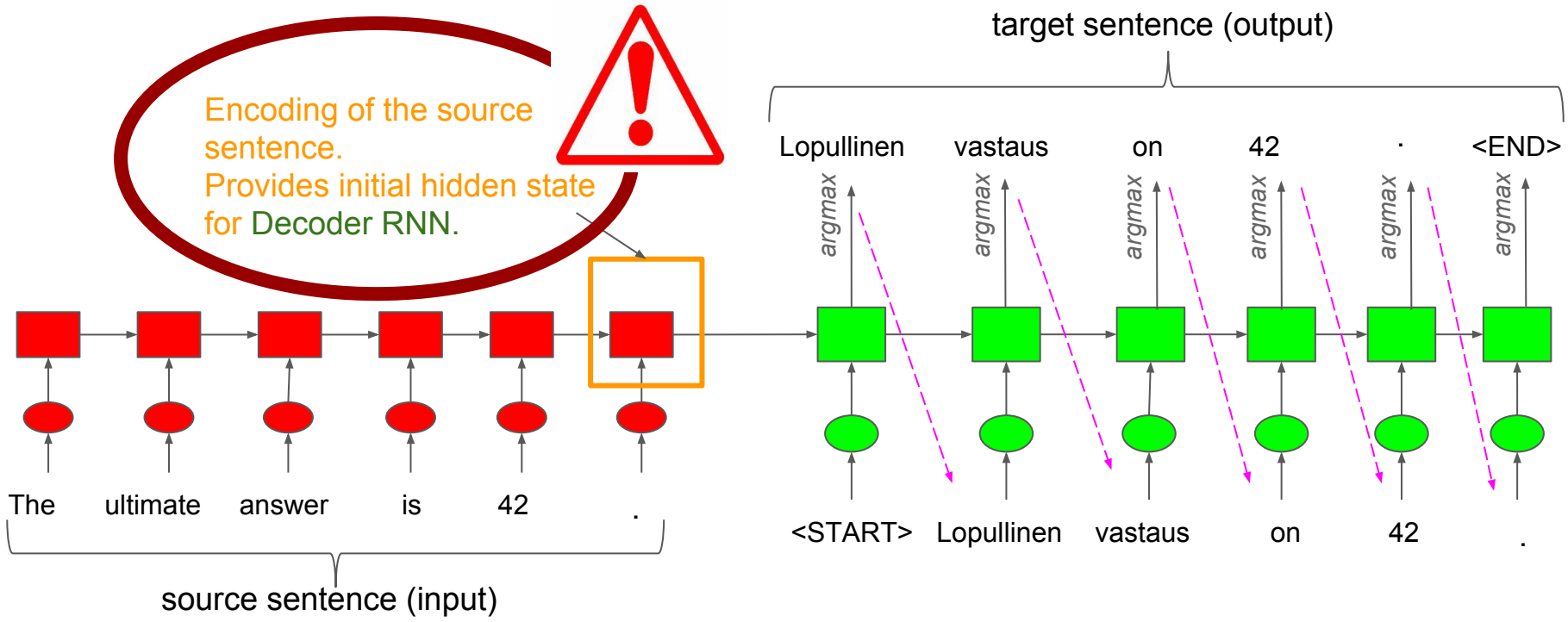


The **encoder** creates a representation of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned* on encoding.

NMT: Issues

- The sequence-to-sequence model is an example of a Conditional Language Model:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*



The **encoder** creates a representation of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned* on encoding.

Summary vector as information bottleneck

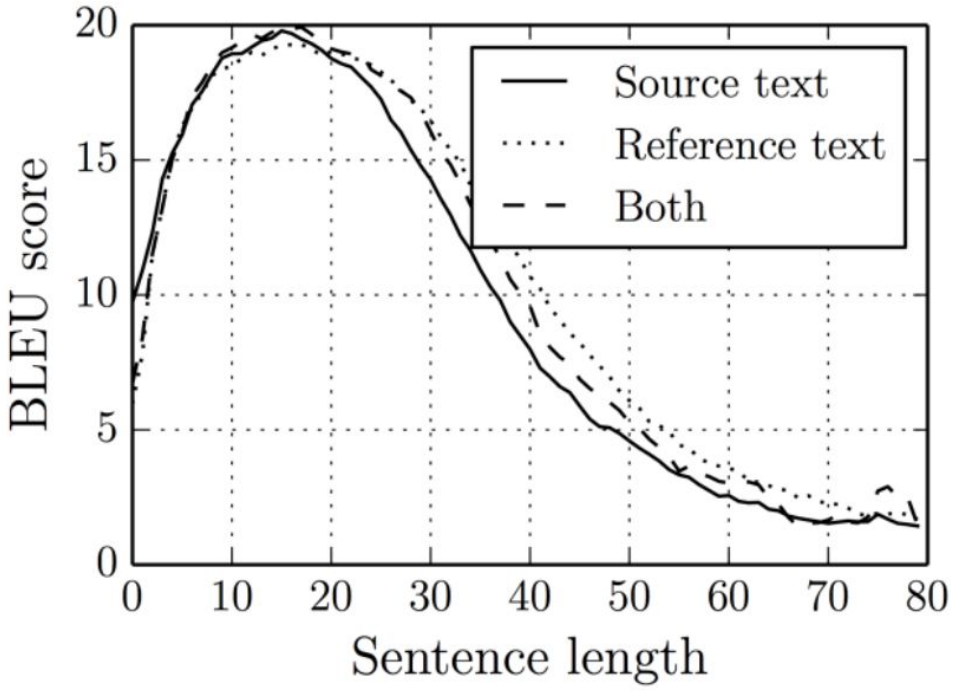
- Last encoder hidden-state “summarises” source sentence.
- This needs to capture all information about the source sentence.

- **Problem**: *Information bottleneck!*

Summary vector as information bottleneck

- Last encoder hidden-state "summarises" source sentence.
- This needs to capture all information about the source sentence.

- **Problem**: *Information bottleneck!*
- Fixed sized representation degrades as sentence length increases



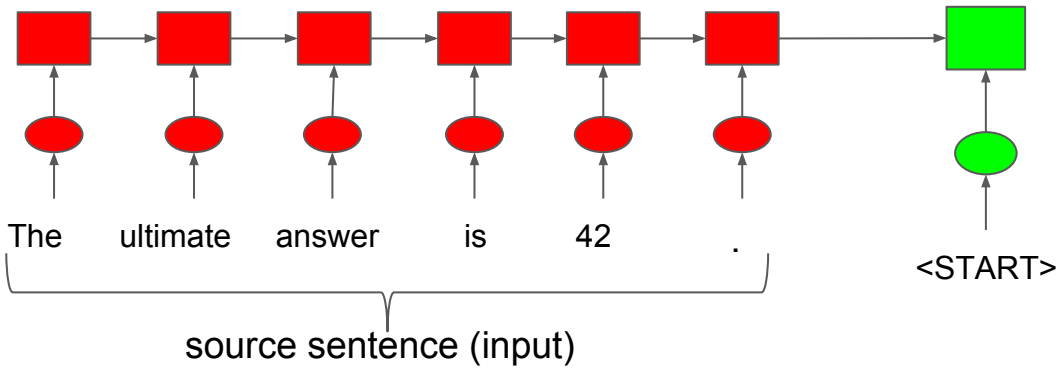
[Cho et al., 2014]

Attention

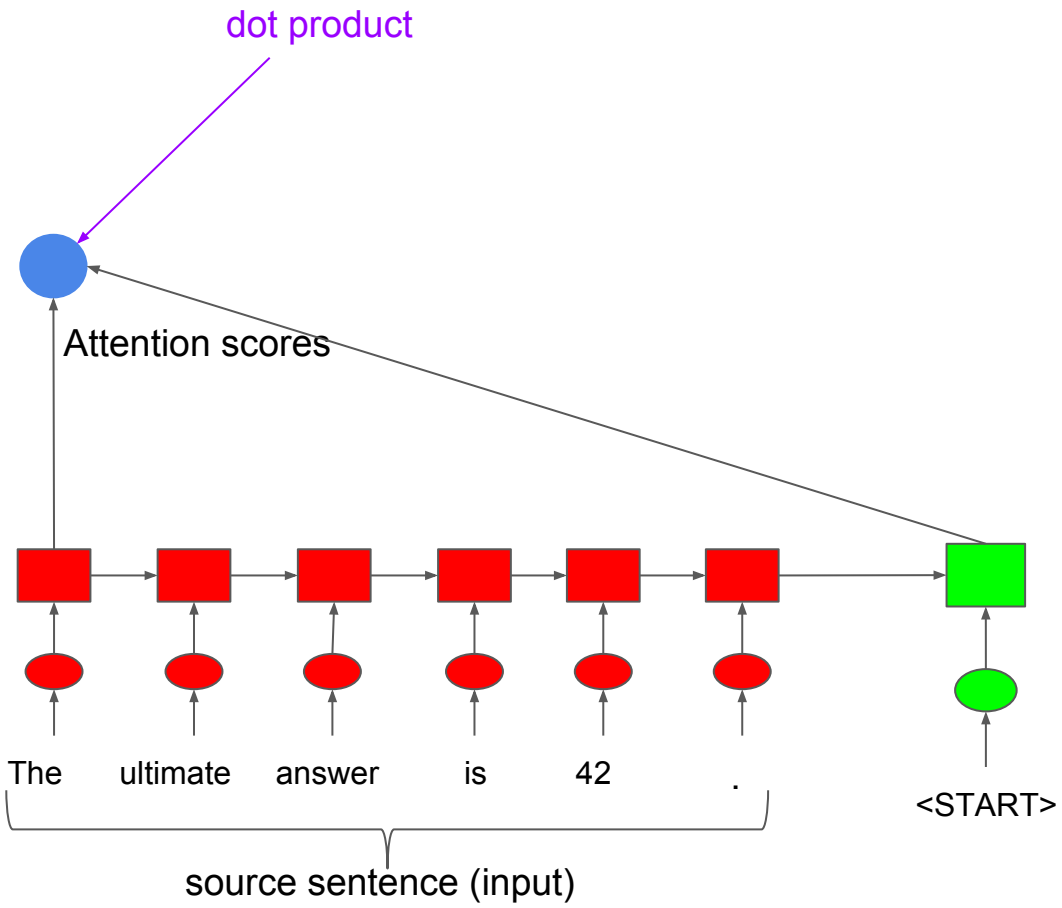
- **Attention** provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence

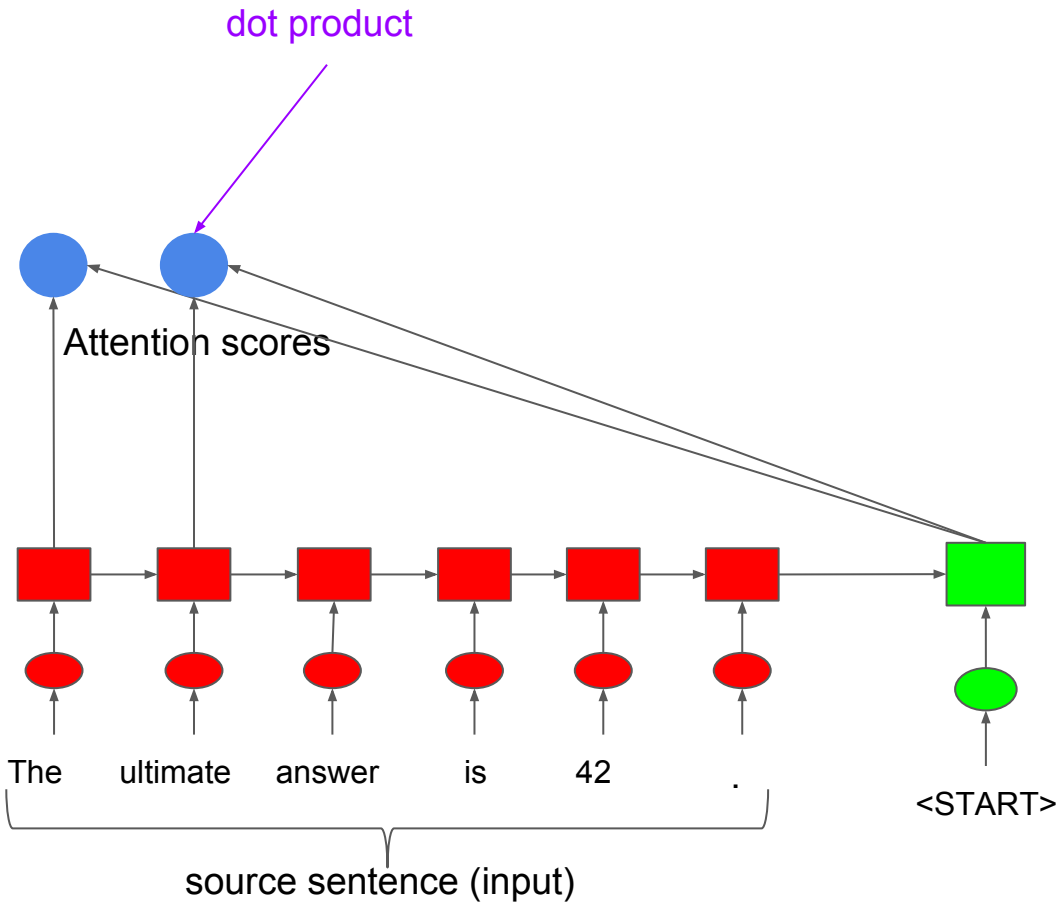
Sequence-to-sequence with attention



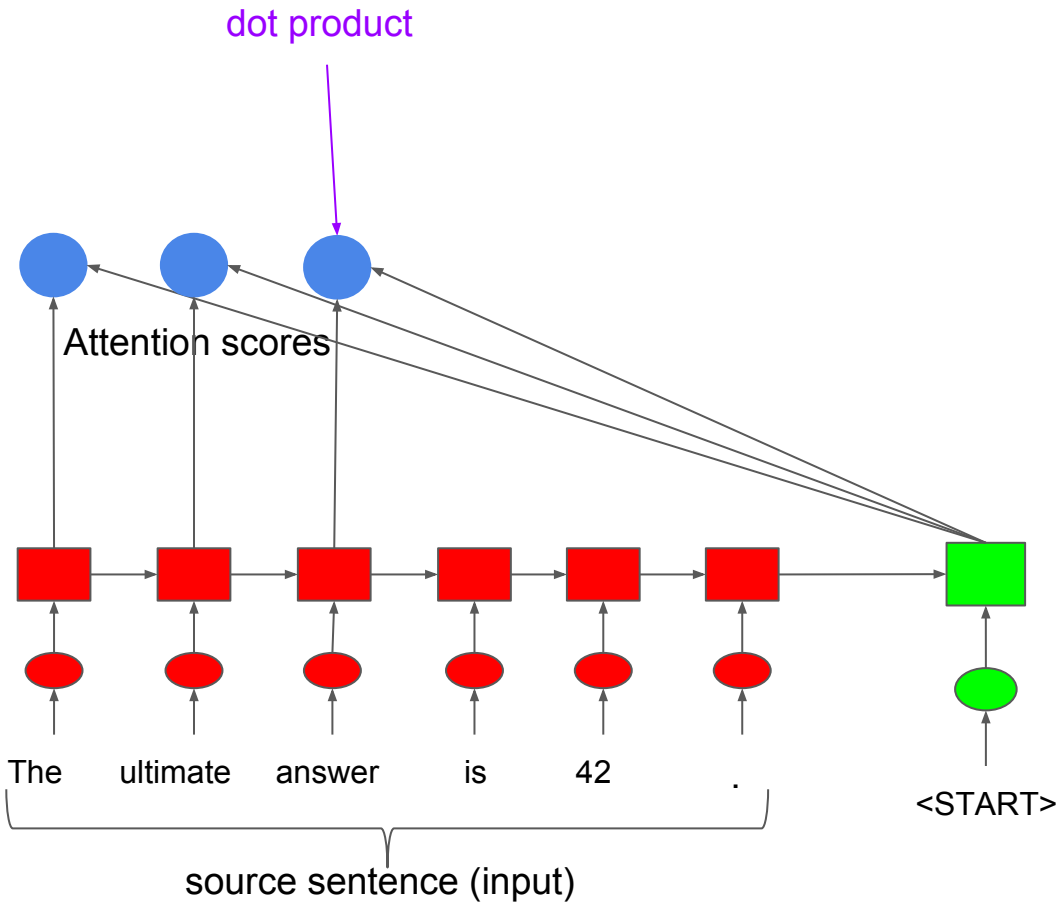
Sequence-to-sequence with attention



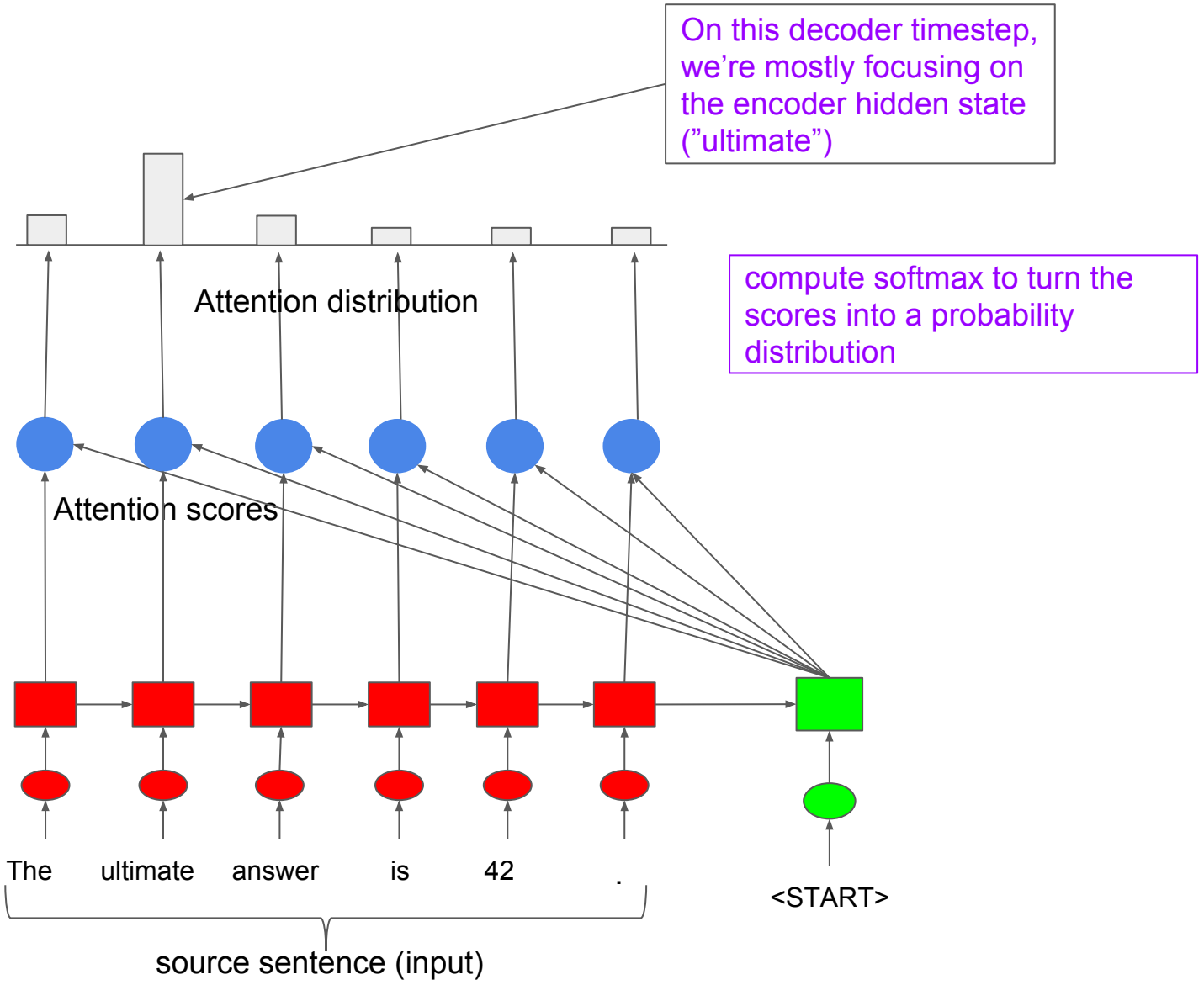
Sequence-to-sequence with attention



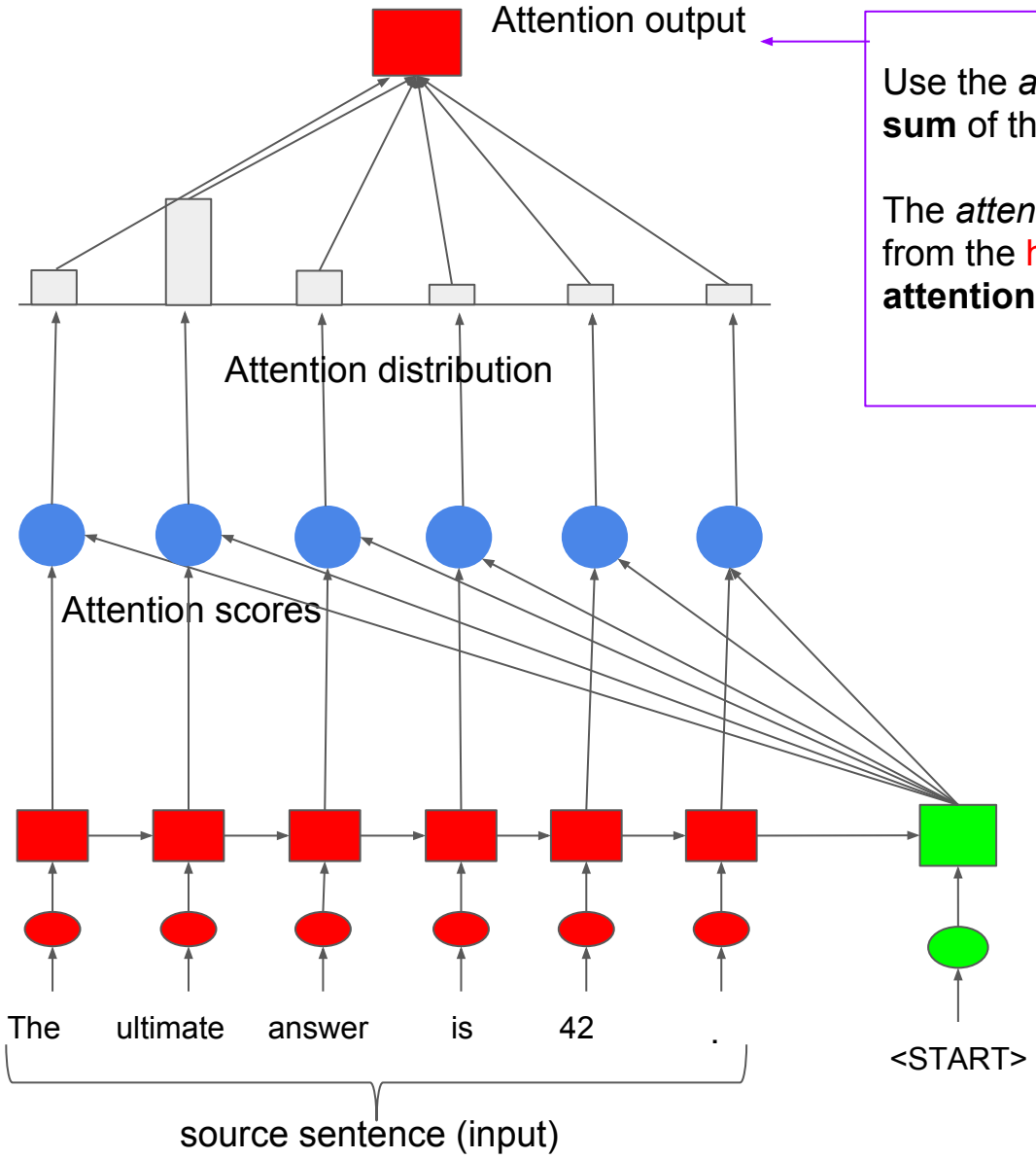
Sequence-to-sequence with attention



Sequence-to-sequence with attention



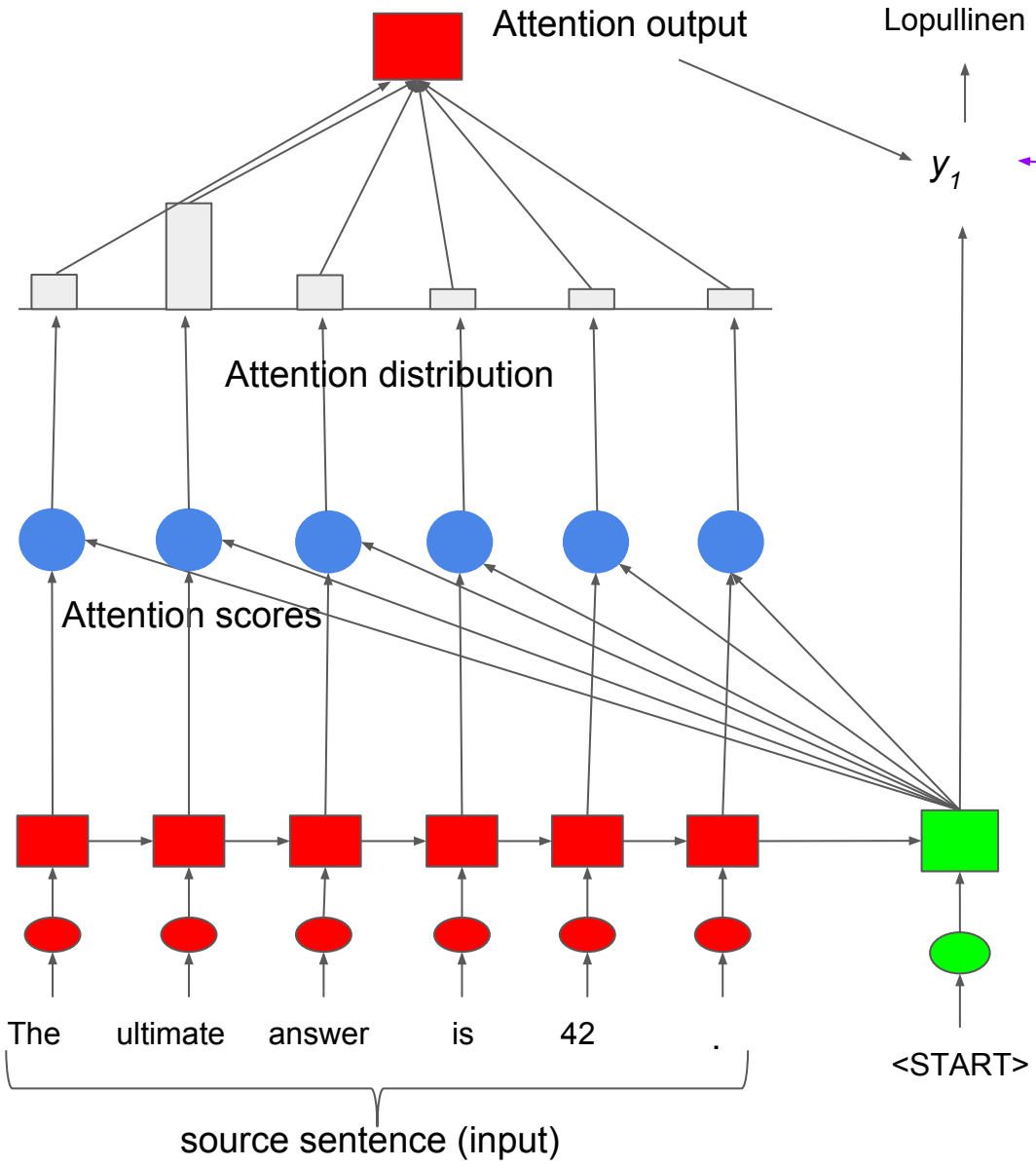
Sequence-to-sequence with attention



Use the *attention distribution* to take a **weighted sum** of the encoder hidden states.

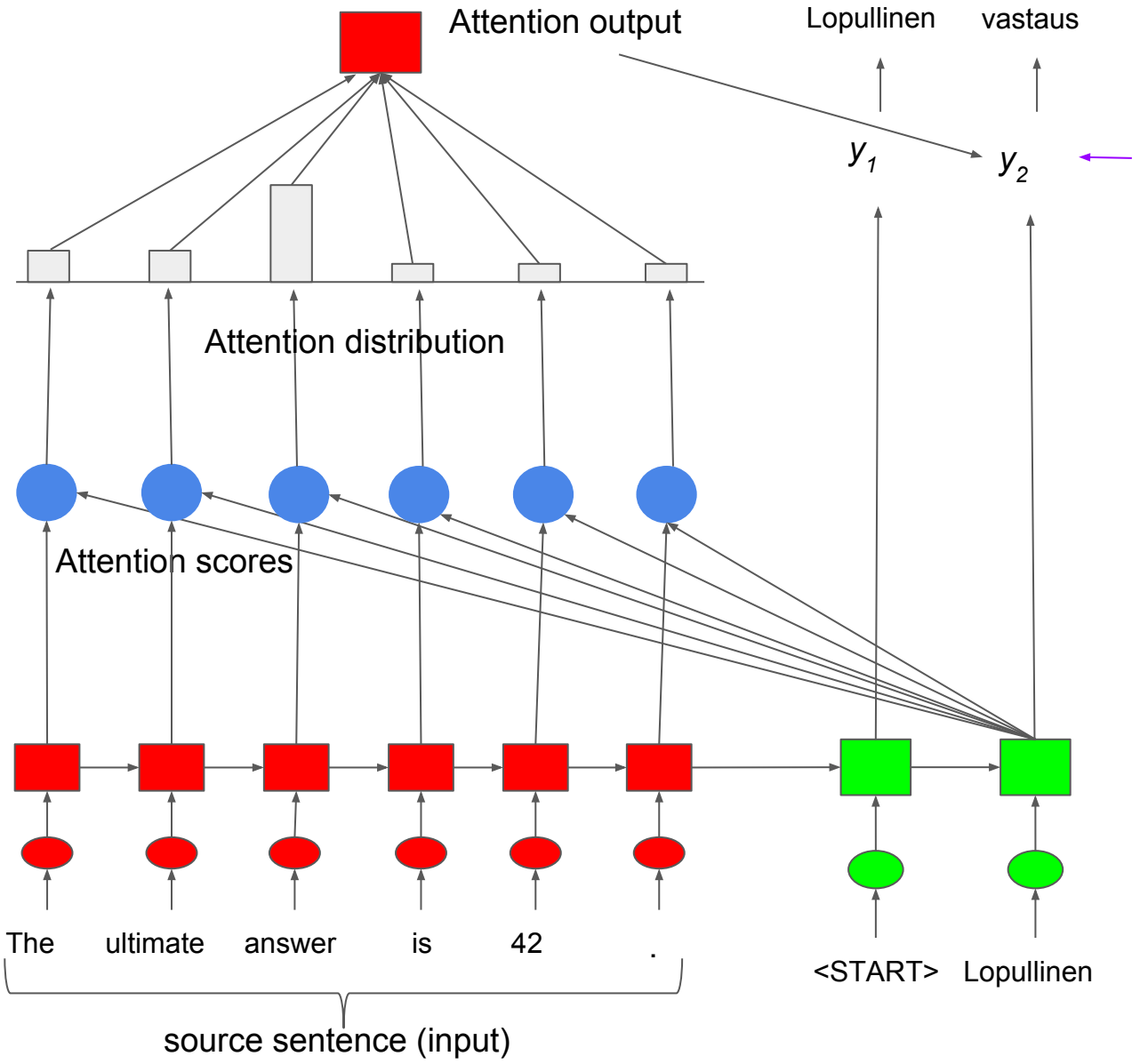
The *attention output* mostly contains information from the hidden states that received **high attention.**

Sequence-to-sequence with attention



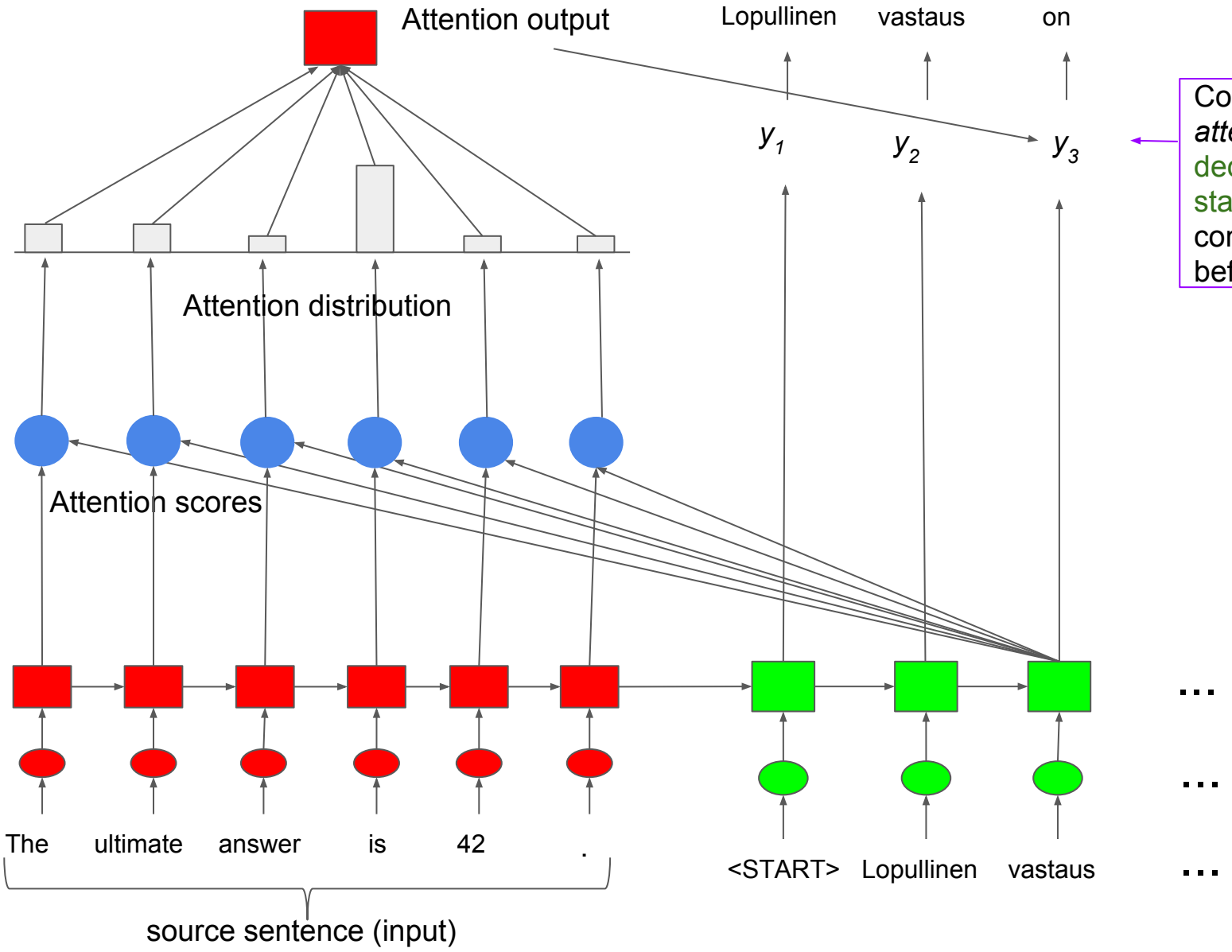
Concatenate *attention output* with decoder hidden state, then use to compute y_1 as before

Sequence-to-sequence with attention



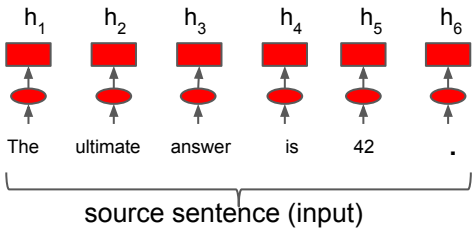
Concatenate *attention output* with decoder hidden state, then use to compute y_2 as before

Sequence-to-sequence with attention



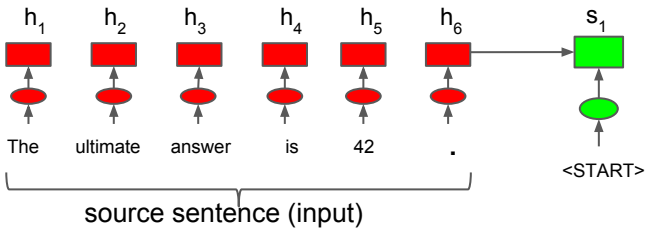
Attention: in equations

Attention: in equations



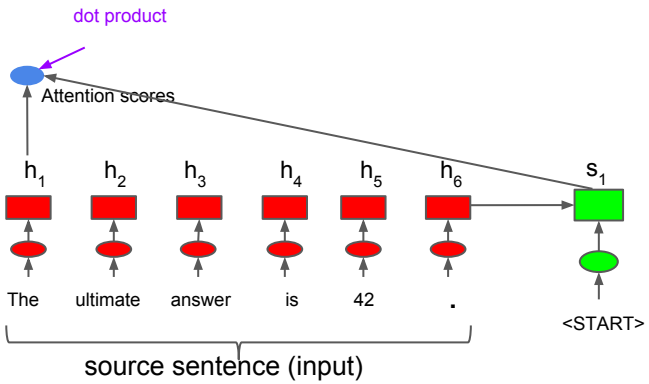
- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$

Attention: in equations



- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$

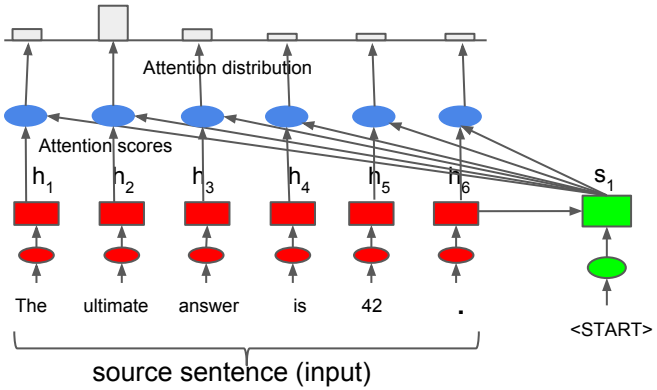
Attention: in equations



- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the *attention scores* $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

Attention: in equations



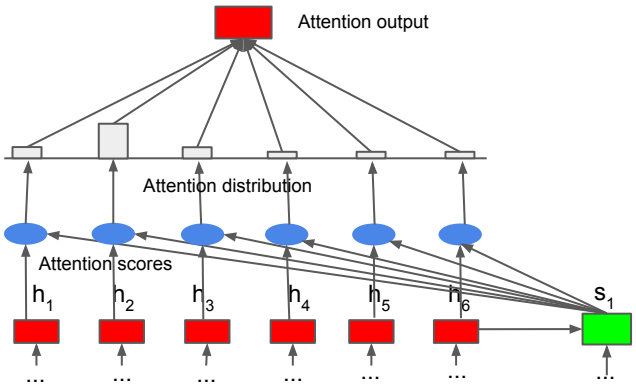
- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the *attention scores* $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the *attention distribution* α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

Attention: in equations



- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the *attention scores* $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

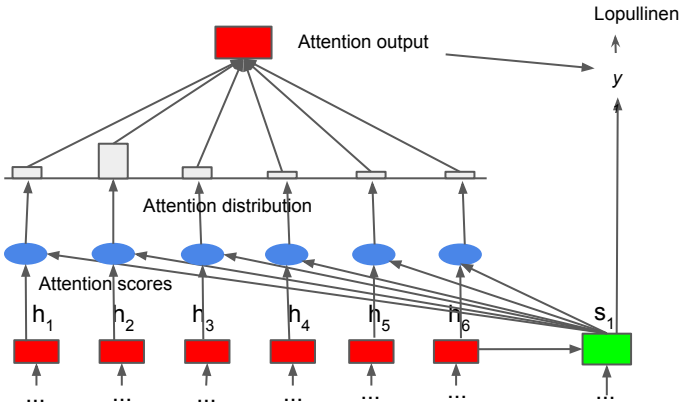
- We take softmax to get the *attention distribution* α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a *weighted sum* of the encoder hidden states to get the *attention output* $\boldsymbol{a}_t$

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

Attention: in equations



- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the *attention scores* $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the *attention distribution* α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a *weighted sum* of the encoder hidden states to get the *attention output* $\boldsymbol{a}_t$

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the *attention output* $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great

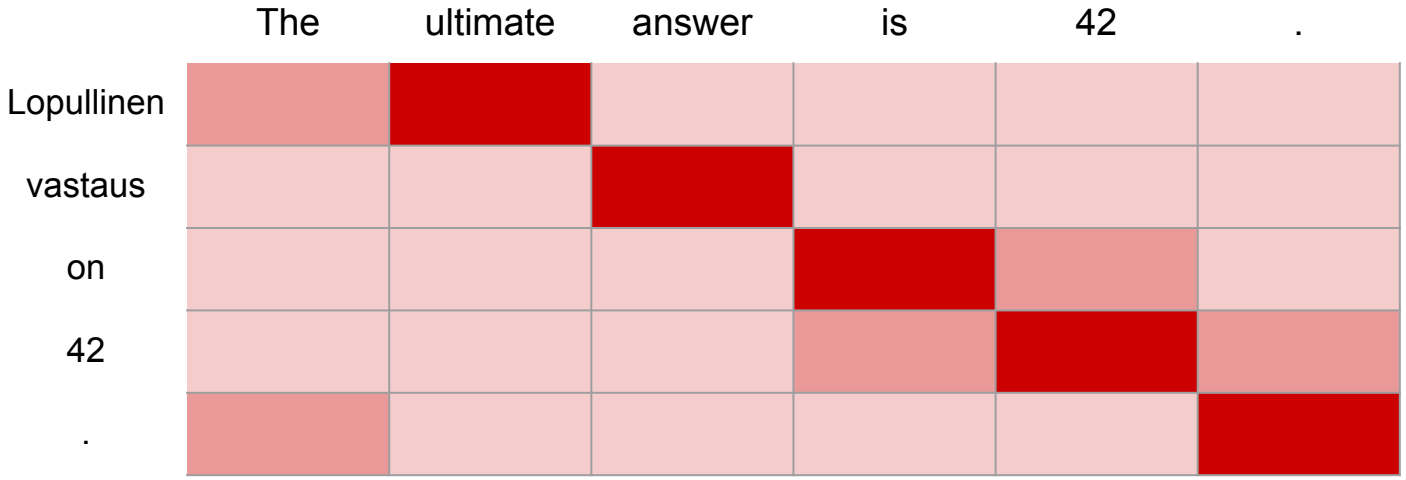
- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source

Attention is great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck

Attention is great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention provides some interpretability:
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

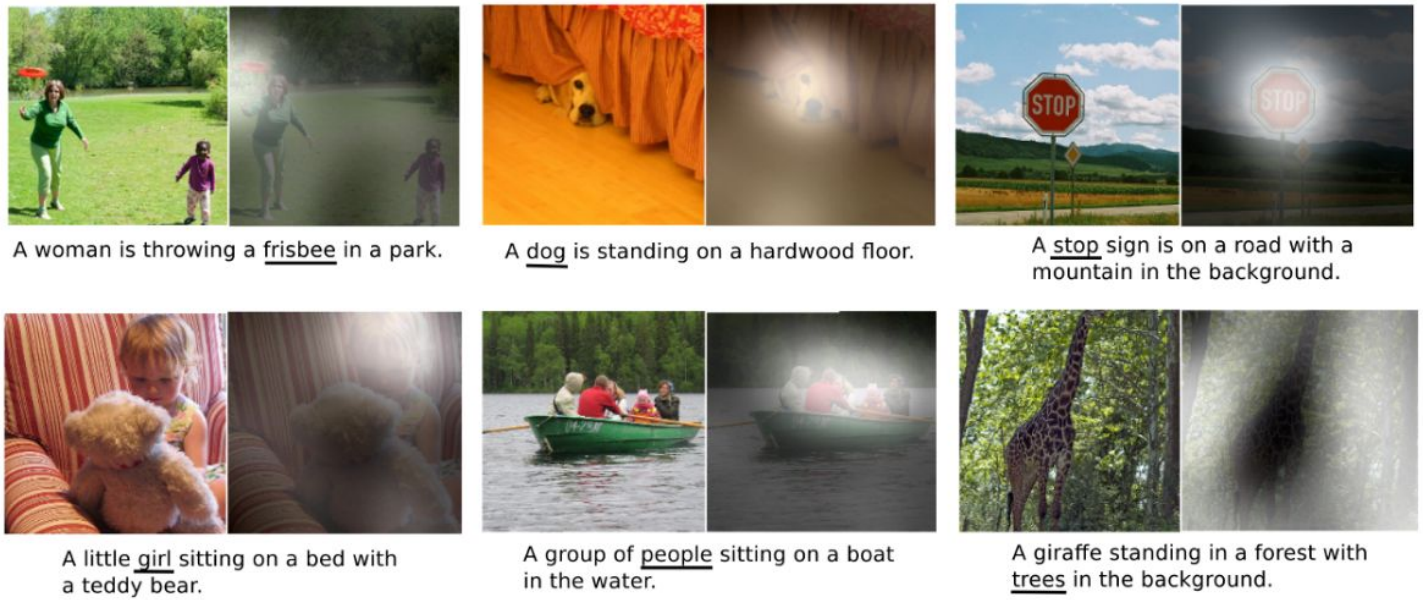


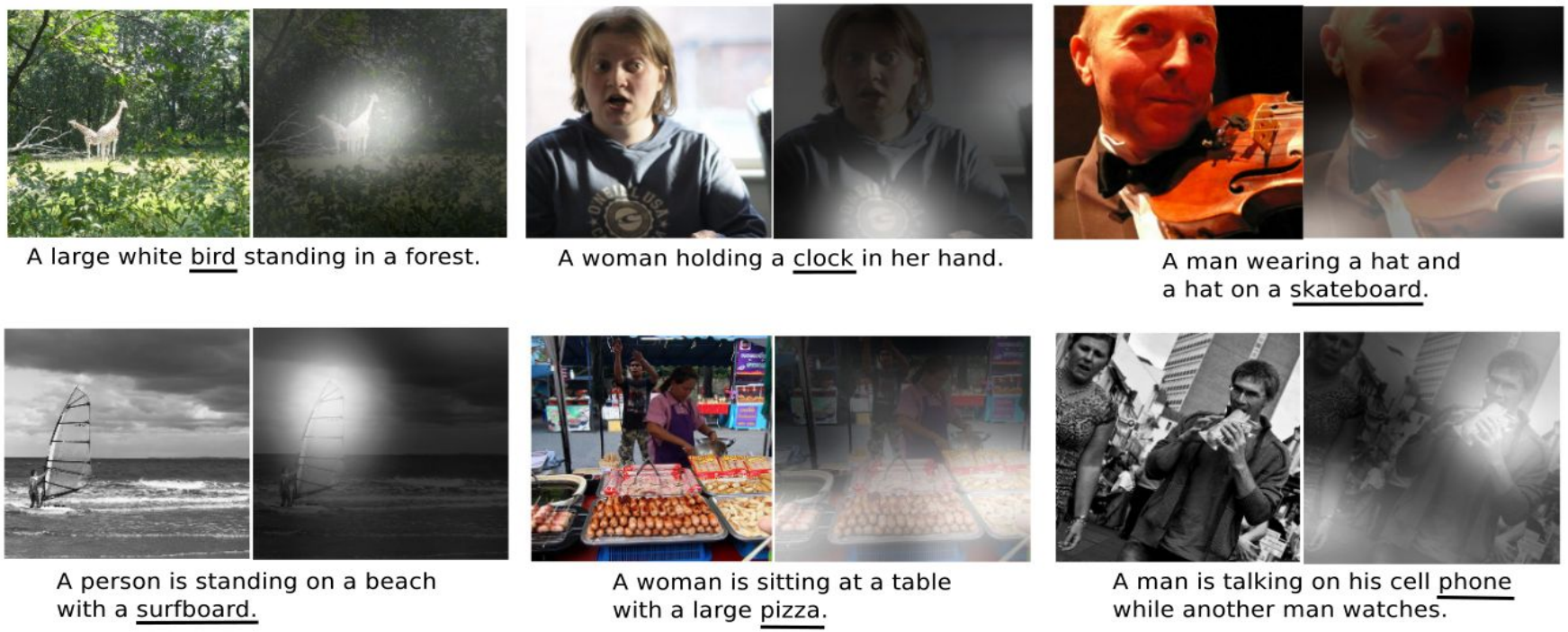
Fig. 5. Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word) [22]

[Xu et al., 2015]

Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



[Xu et al., 2015]

Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- More general definition of attention:

  Given a set of vector values, and a vector query, ***attention*** is a technique to compute a weighted sum of the values, dependent on the query.

Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- More general definition of attention:

  Given a set of vector values, and a vector query, ***attention*** is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends* to the values.
- For example, in the seq2seq + attention model, each decoder hidden state (**query**) *attends* to all the encoder hidden states (**values**).

Attention is a *general* Deep Learning technique

- More general definition of attention:

  Given a set of vector values, and a vector query, ***attention*** is a technique to compute a weighted sum of the values, dependent on the query.

- **Intuition**:
  - The weighted sum is a selective summary of the information contained in the values, where the query determines which values to focus on.
  - Attention is a way to obtain a fixed-size representation of an arbitrary set of representations (the values), dependent on some other representation (the query).

There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$
  2. Taking softmax to get *attention distribution* α:

$$\alpha = \text{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

  thus obtaining the *attention output* $\boldsymbol{a}$ (sometimes called the context vector)

There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$

     There are multiple ways to do this

  2. Taking softmax to get *attention distribution* α:

$$\alpha = \text{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

  thus obtaining the *attention output* $\boldsymbol{a}$ (sometimes called the context vector)

Attention variants

- There are several ways you can compute the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$ from the values $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

Attention variants

- There are several ways you can compute the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$ from the values $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier

Attention variants

- There are several ways you can compute the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$ from the values $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier

- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix

Attention variants

- There are several ways you can compute the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$ from the values $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier

- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix

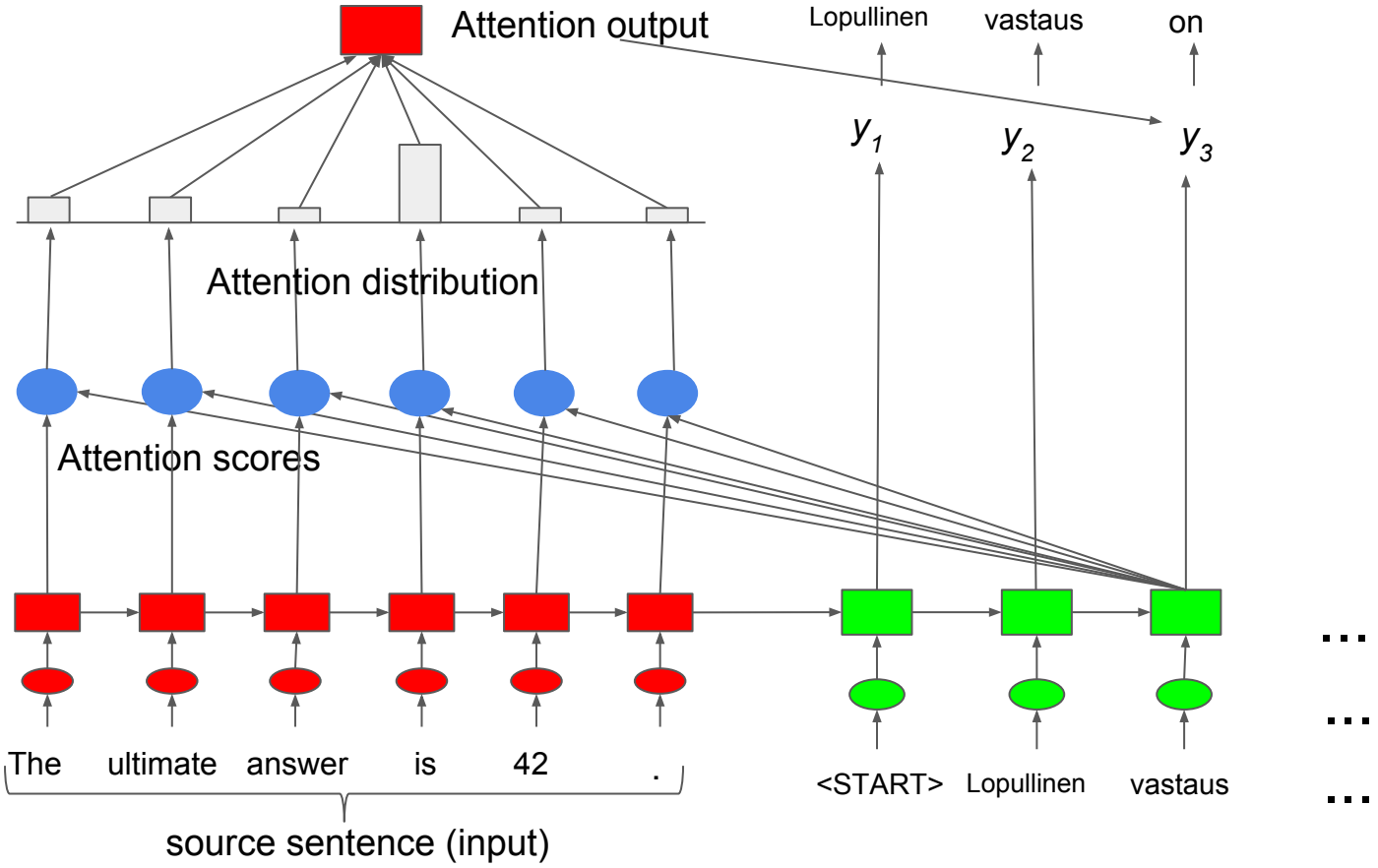
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector
  - d_3 (the attention dimensionality) is a hyperparameter

More information:

"Deep Learning for NLP Best Practices", Ruder, 2017. http://ruder.io/deep-learning-nlp-best-practices/index.html#attention

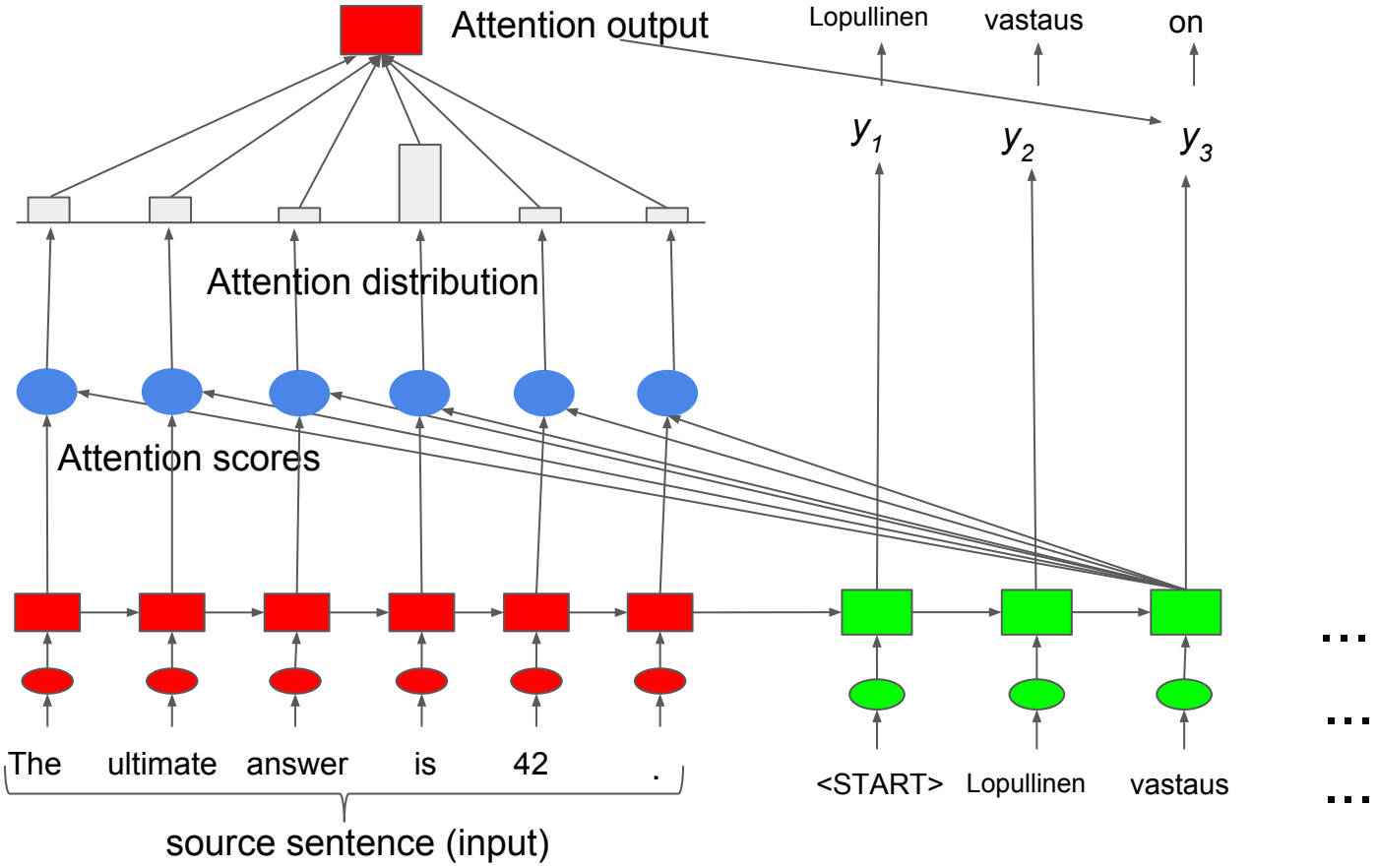
"Massive Exploration of Neural Machine Translation Architectures", Britz et al, 2017, https://arxiv.org/pdf/1703.03906.pdf

Sequence-to-sequence with attention



- Are we forgetting something behind?

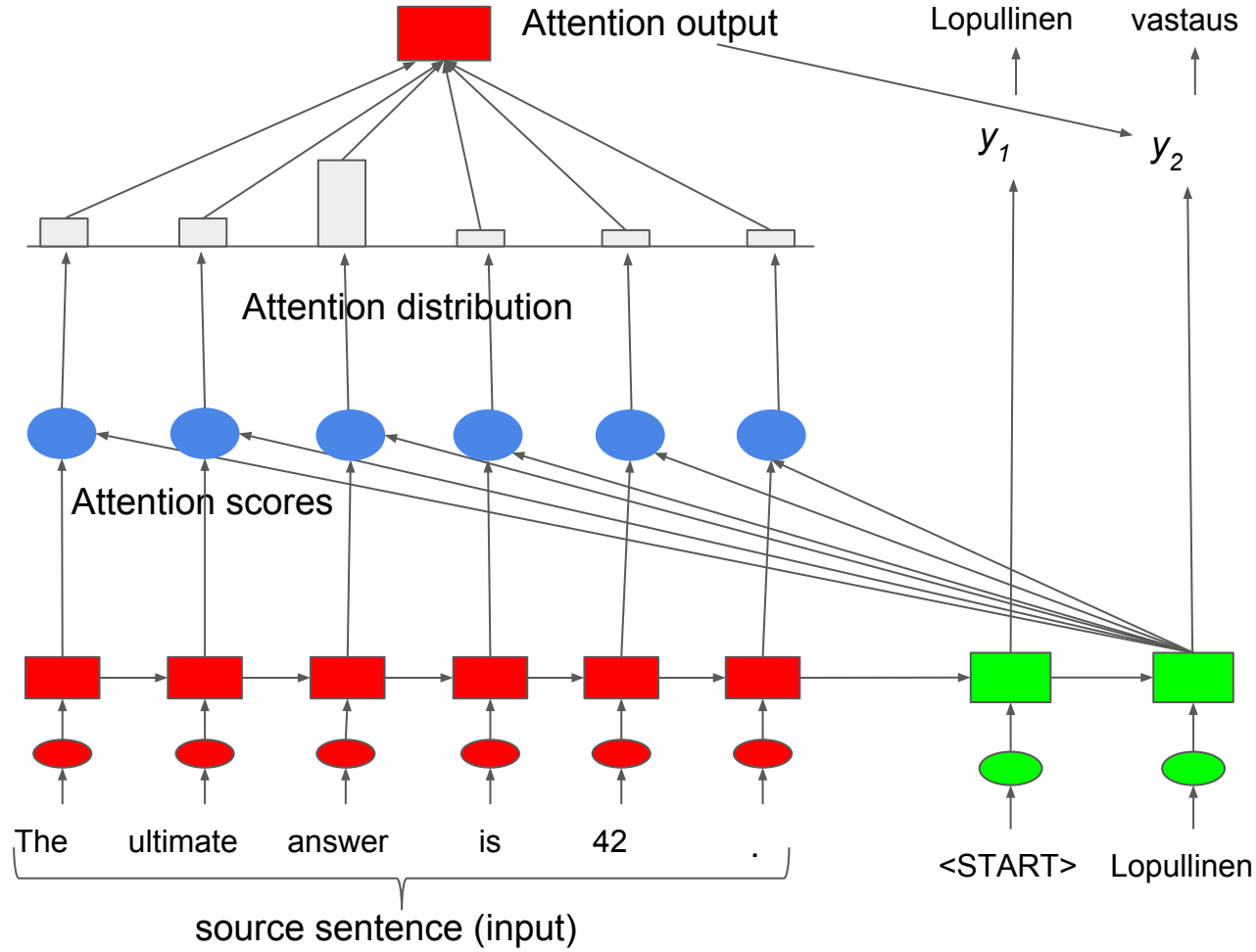
Sequence-to-sequence with attention



- <u>Are we forgetting something behind?</u>
  Attention decisions are made independently (which is *suboptimal*)

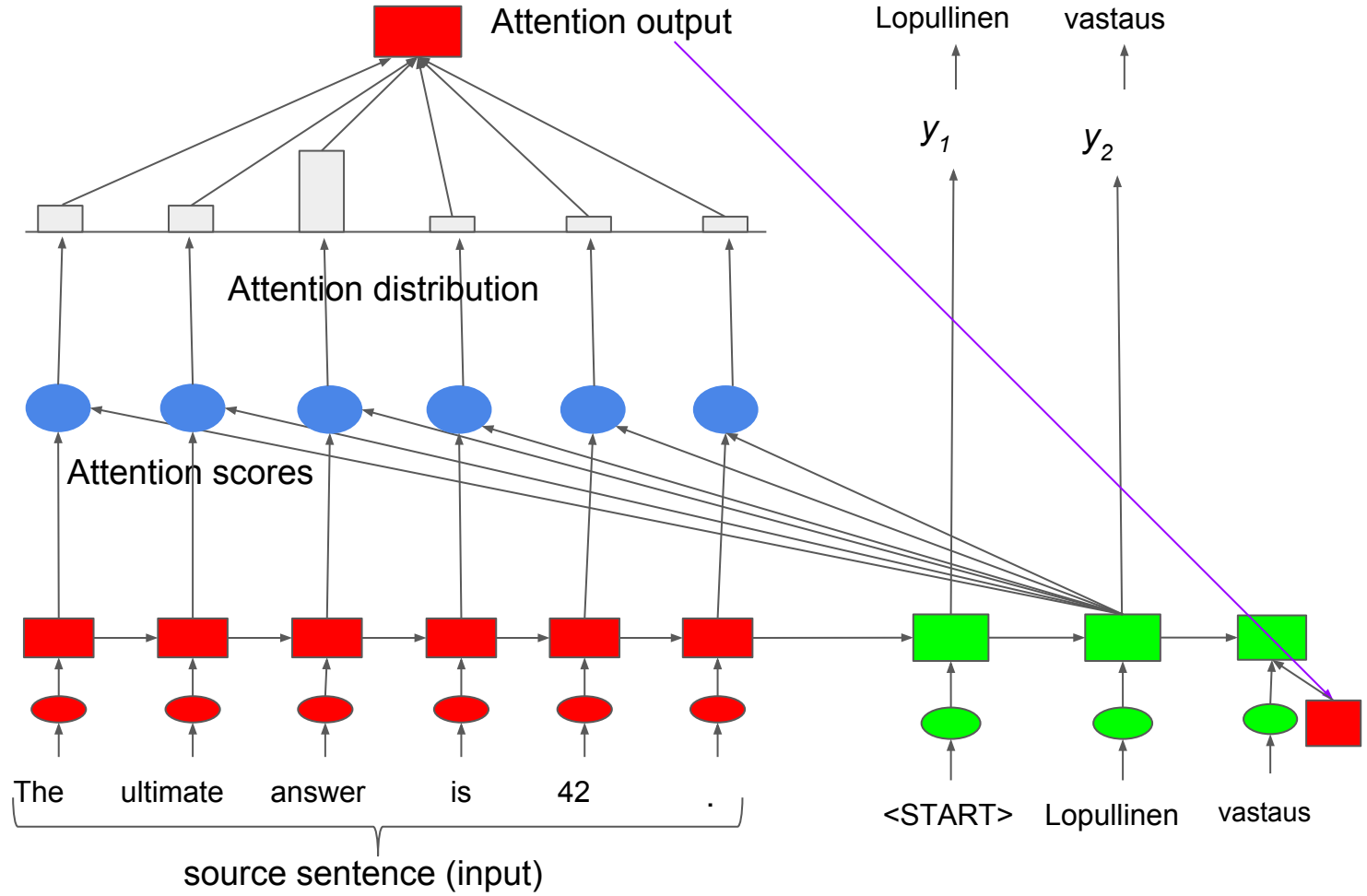
Input feeding

- Attention decisions should be made jointly taking into account past attention information.



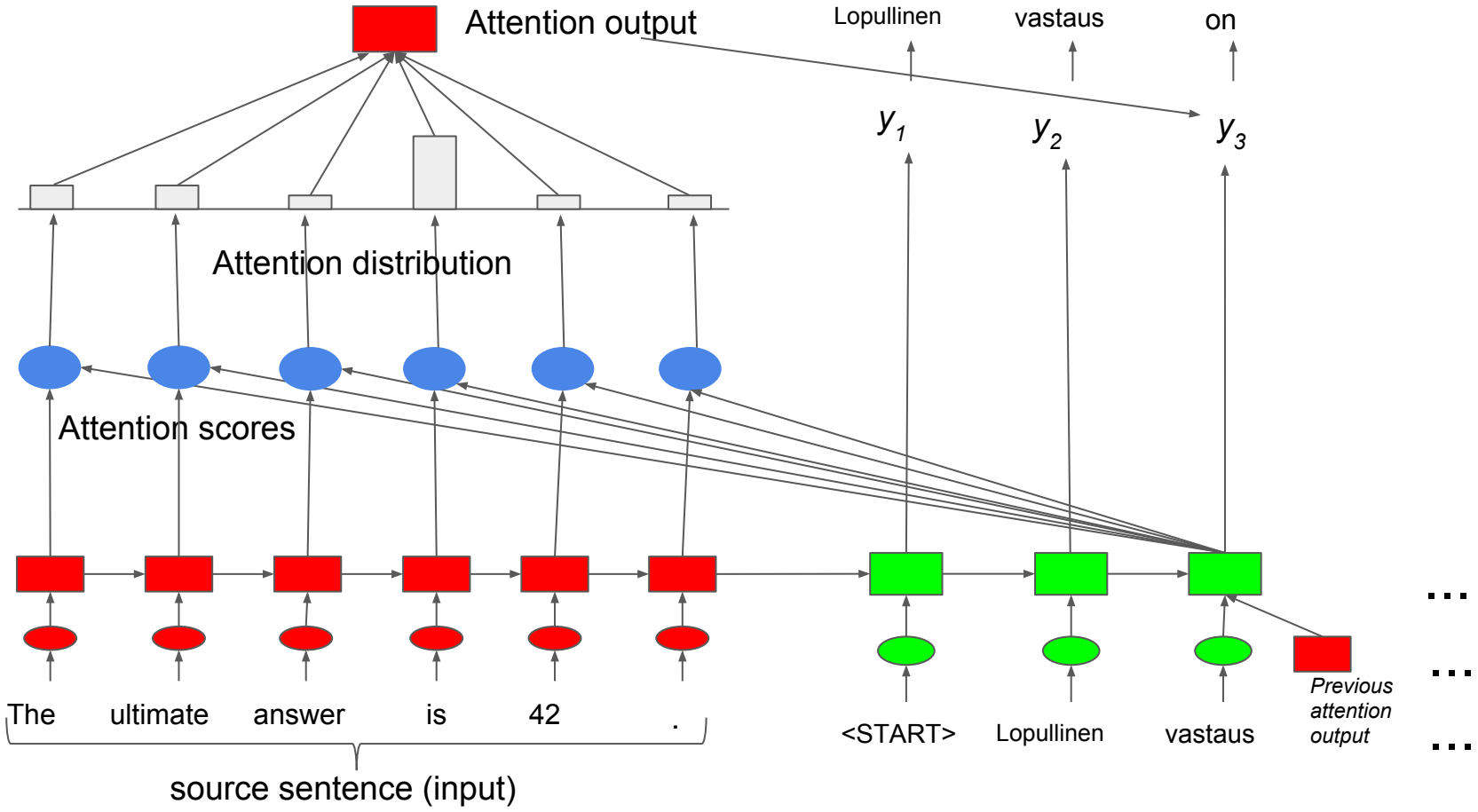
Input feeding

- Attention decisions should be made jointly taking into account past attention information.



Input feeding

- Attention decisions should be made jointly taking into account past attention information:
  - we hope to make the model fully aware of previous attention choices
  - usually the attentional vector is concatenated with the input at the next time step



NMT research continues

NMT is the **flagship task** for Natural Language Processing (NLP) Deep Learning

- NMT research has pioneered many of the recent innovations of NLP Deep Learning

- In **2019**: NMT research continues to thrive
  - Researchers have found many, many improvements to the seq2seq NMT system
  - what we have seen today is considered the new "vanilla" seq2seq NMT architecture:
    - seq2seq + attention + input feeding

Readings

- Attention models:
  Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. (2014)
  Neural machine translation by jointly learning to align and translate.
  https://arxiv.org/pdf/1409.0473.pdf

  Luong, T., Pham, H., and Manning, C. D. (2015).
  Effective Approaches to Attention-based Neural Machine Translation.
  https://aclweb.org/anthology/D15-1166

  Hamidreza Ghader and Christof Monz (2017)
  What does Attention in Neural Machine Translation Pay Attention to?
  https://arxiv.org/pdf/1710.03348.pdf

- BackTranslations:
  Rico Sennrich, Barry Haddow and Alexandra Birch (2016)
  Improving Neural Machine Translation Models with Monolingual Data
  https://arxiv.org/pdf/1511.06709.pdf

Readings

Xu, Kelvin, et al. (2015)
Show, attend and tell: Neural image caption generation with visual attention.
http://www.jmlr.org/proceedings/papers/v37/xuc15.pdf

Cho, Kyunghyun, et al. (2014)
On the properties of neural machine translation: Encoder-decoder approaches.
https://arxiv.org/pdf/1409.1259.pdf

Raganato, Alessandro, et al. (2018)
The University of Helsinki submissions to the WMT18 news task.
https://www.aclweb.org/anthology/W18-6425

Lample, Guillaume, et al. (2018)
Phrase-based & neural unsupervised machine translation.
https://arxiv.org/pdf/1804.07755.pdf

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre. (2018)
Unsupervised statistical machine translation.
https://arxiv.org/pdf/1809.01272.pdf