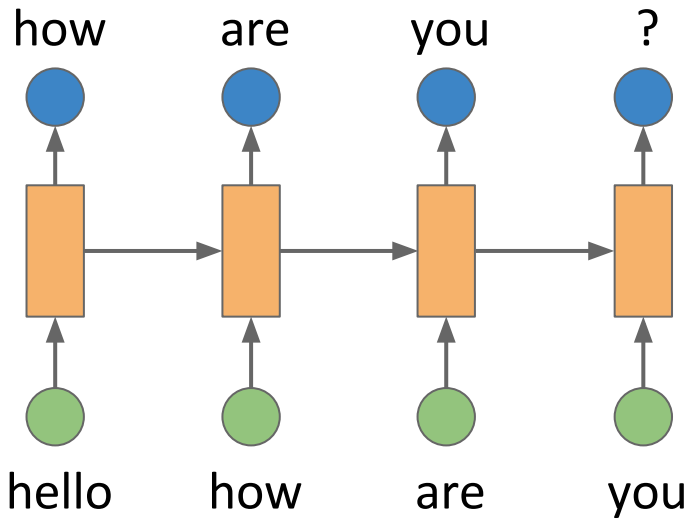
Sequence to Sequence Models

Lecture # 7

Hassan Sajjad and Fahim Dalvi
Qatar Computing Research Institute, HBKU

Recap

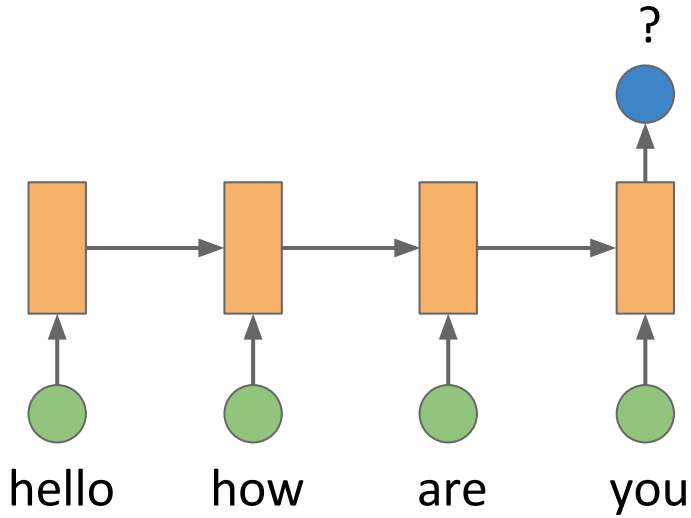
- RNN predicts a word based on the context
- A context vector at time t can be seen as a summary of previous words



$$P(\text{current word}|\text{previous words})$$

Recap

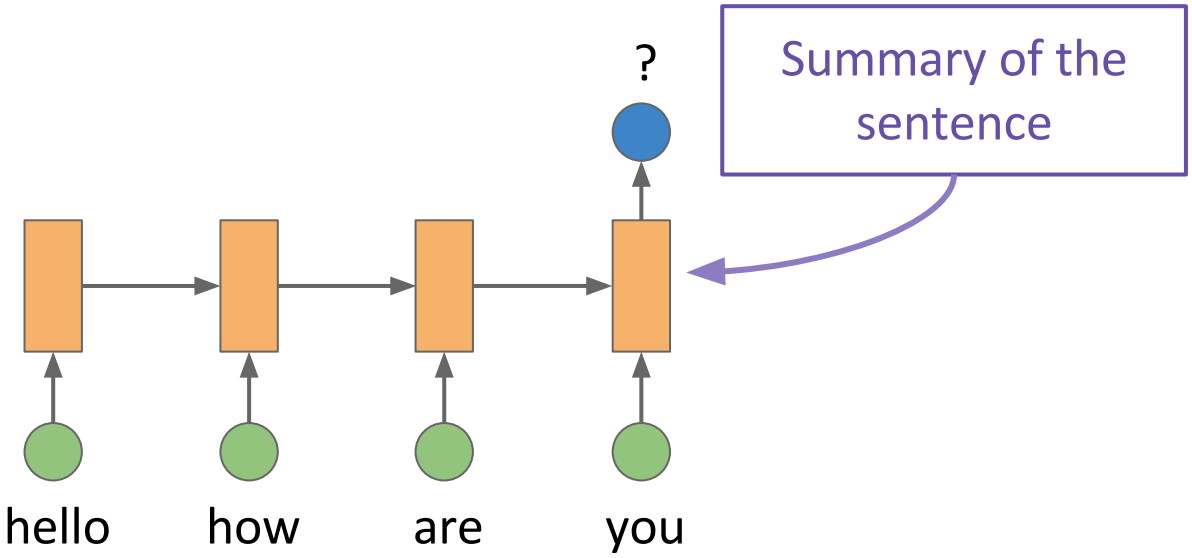
- RNN predicts a word based on the context
- A context vector at time t can be seen as a summary of previous words



$$P(?|\text{previous words})$$

Recap

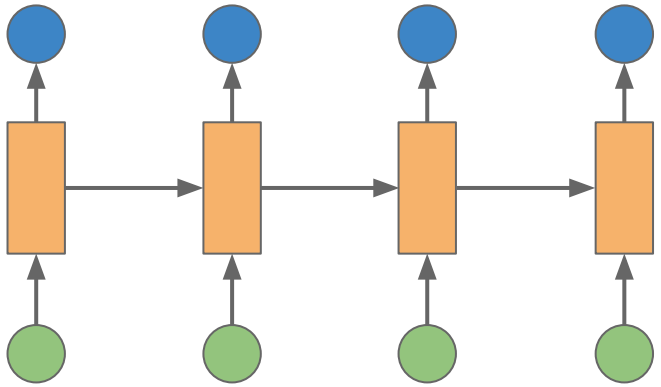
- RNN predicts a word based on the context
- A context vector at time t can be seen as a summary of previous words



Bilingual Recurrent Neural Networks

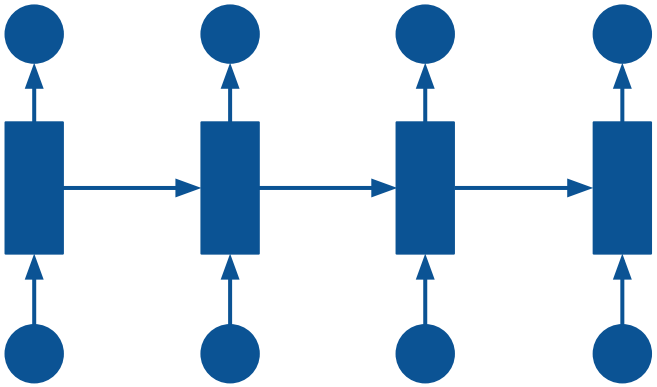
Sequence to Sequence Model

- Bilingual RNN
  - So far, we have seen RNN's with only one language involved:

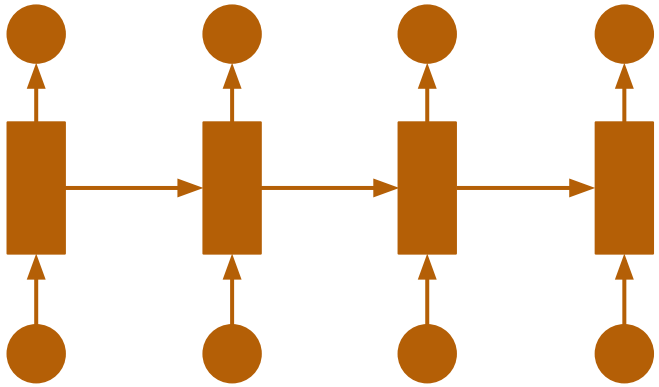


Sequence to Sequence Model

- Bilingual RNN
  - Consider two language models:



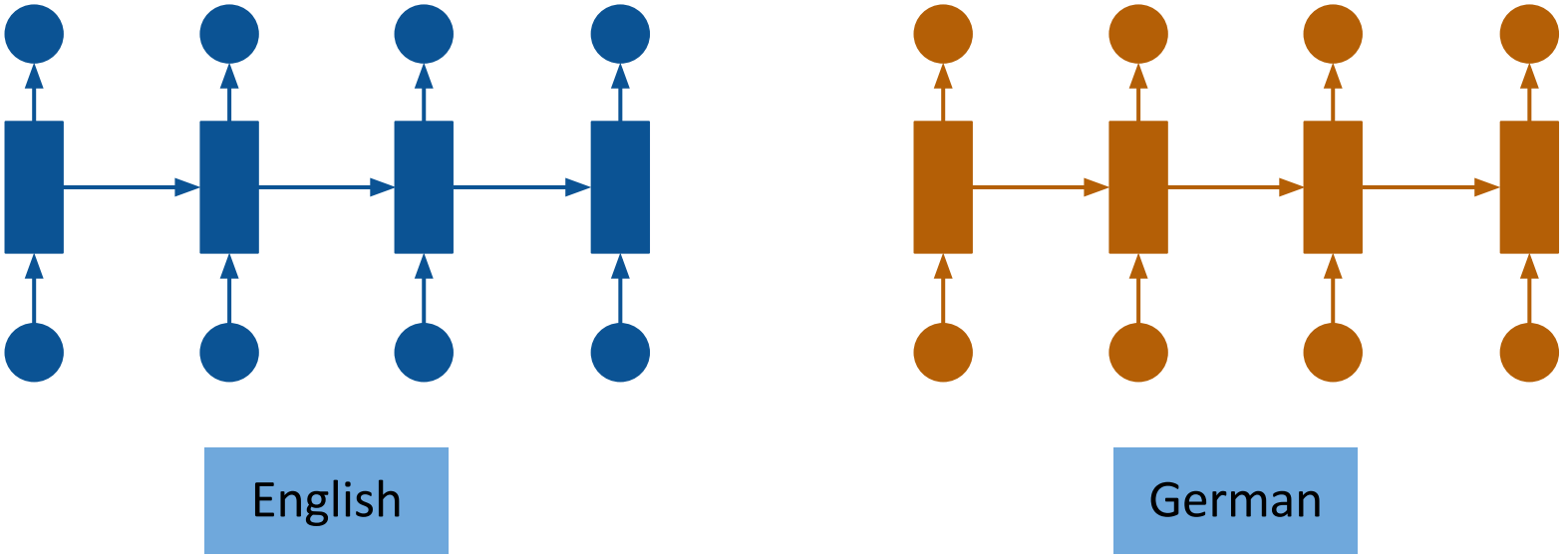
English



German

Sequence to Sequence Model

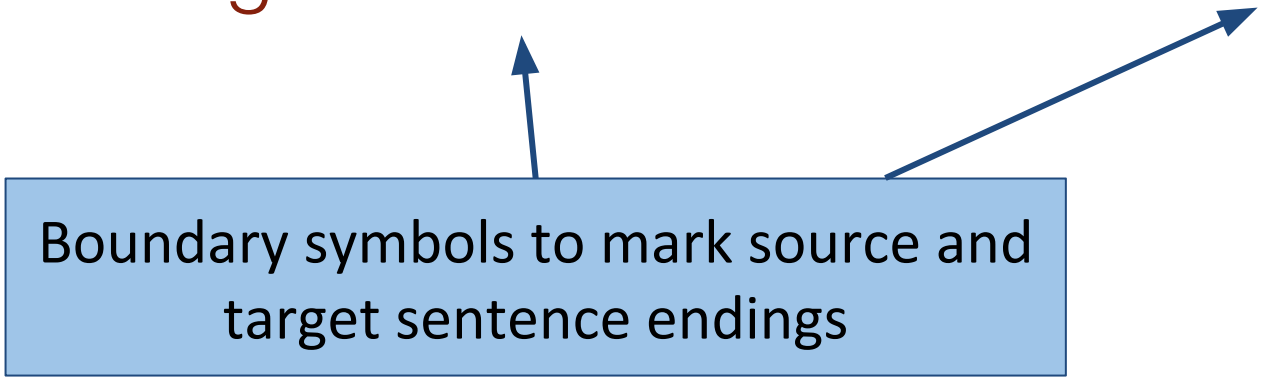
- Bilingual RNN
  - Can we combine English (source) and German (target) sentences to predict a German word?



Sequence to Sequence Model

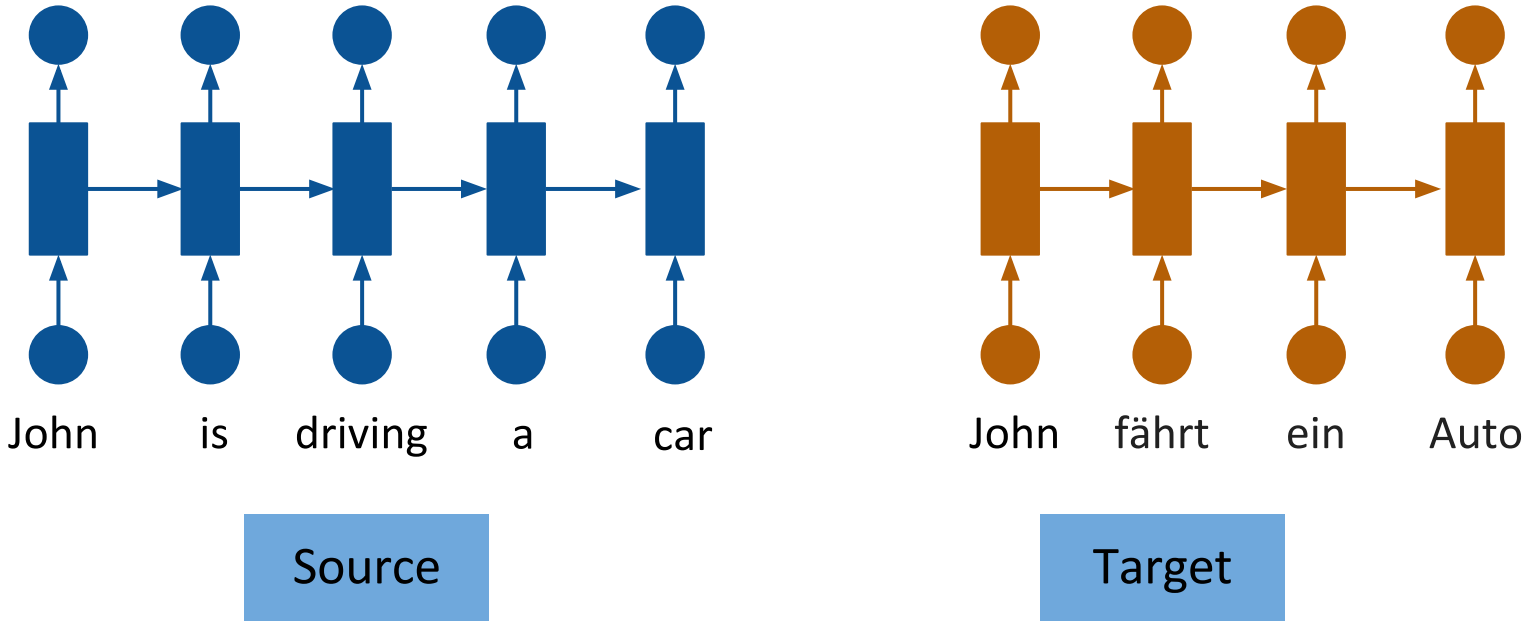
- Bilingual RNN
  - Imagine English and German sentences as a single sequence of strings

John is driving a car . John fährt ein Auto .



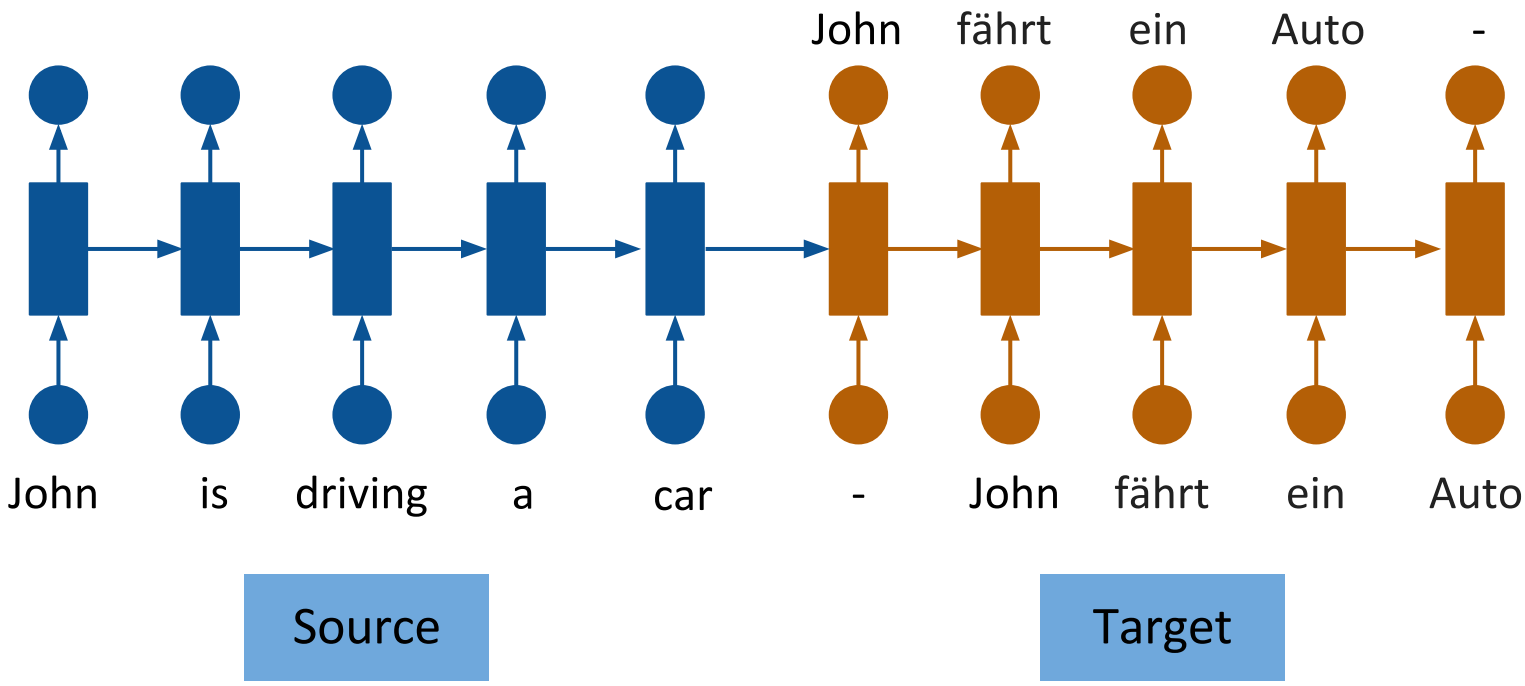
Sequence to Sequence Model

- Consider this combined form as a single language



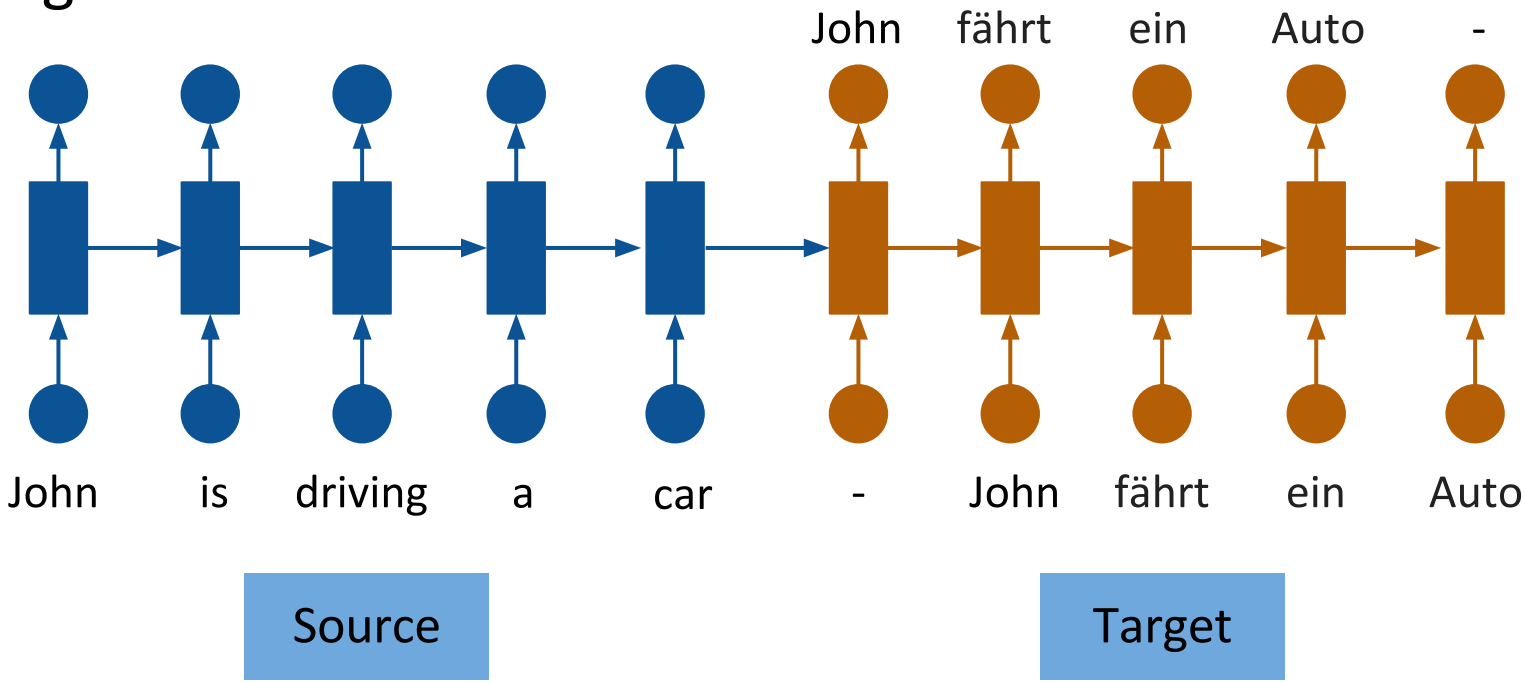
Sequence to Sequence Model

- Consider this combined form as a single language

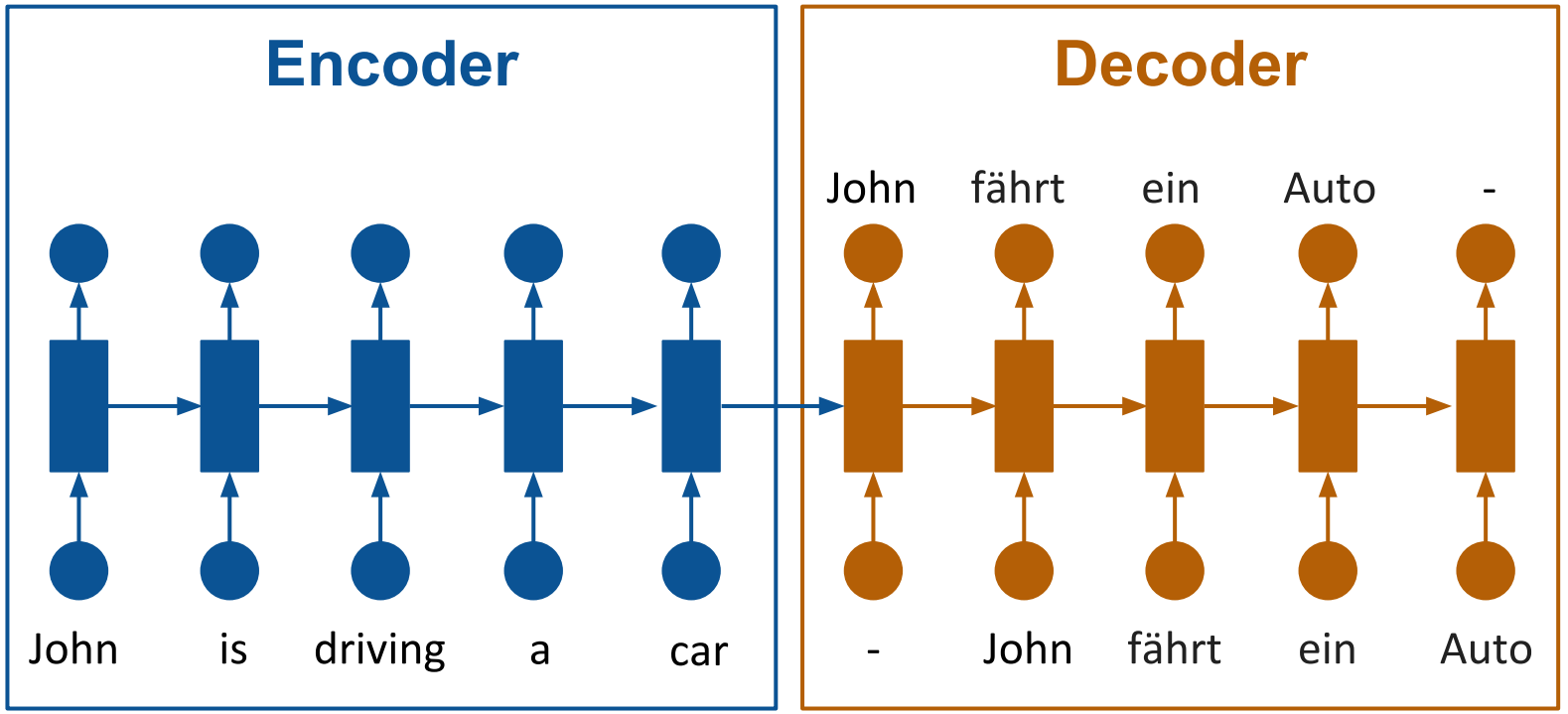


Sequence to Sequence Model

- Essentially, the first RNN is summarizing the English sentence into a vector, and the second RNN uses this to generate German!

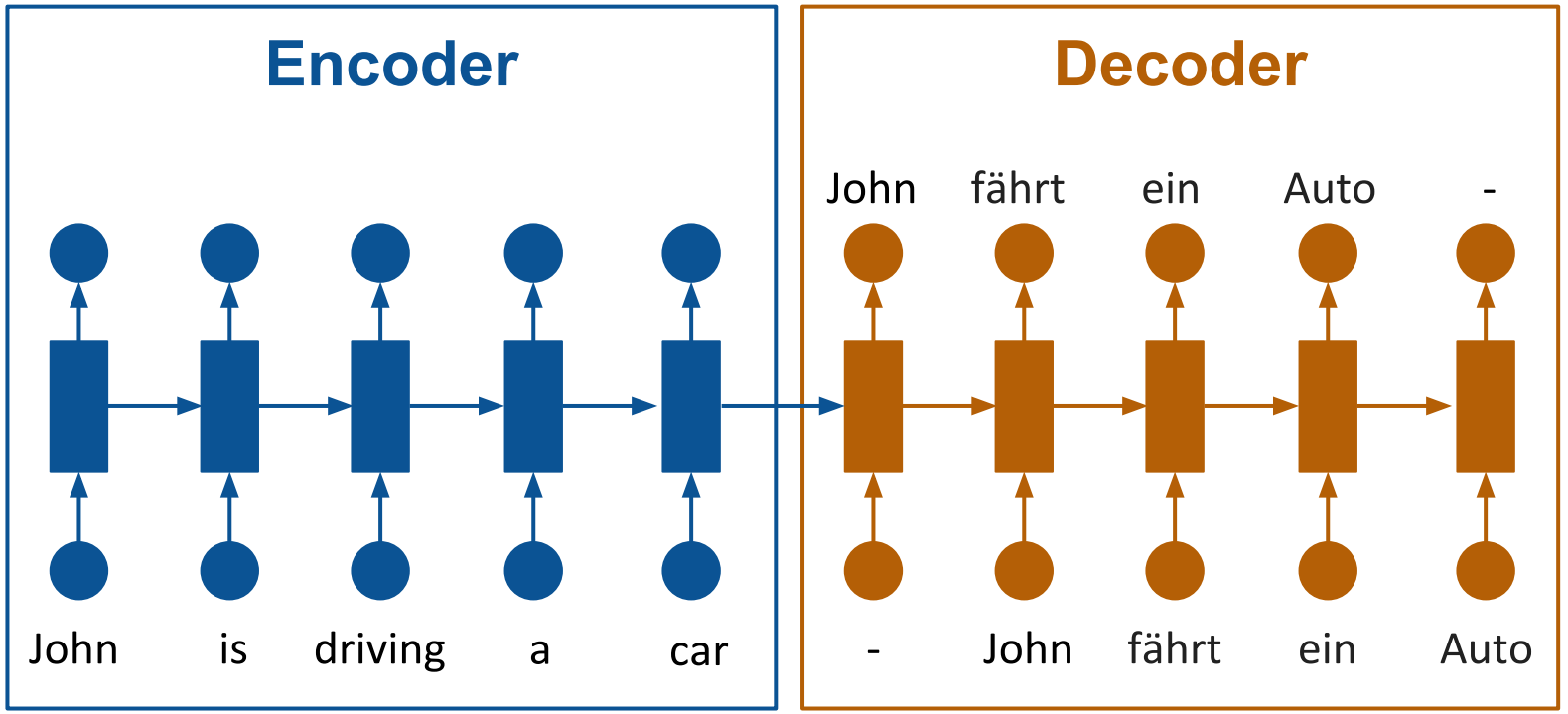


Sequence to Sequence Model



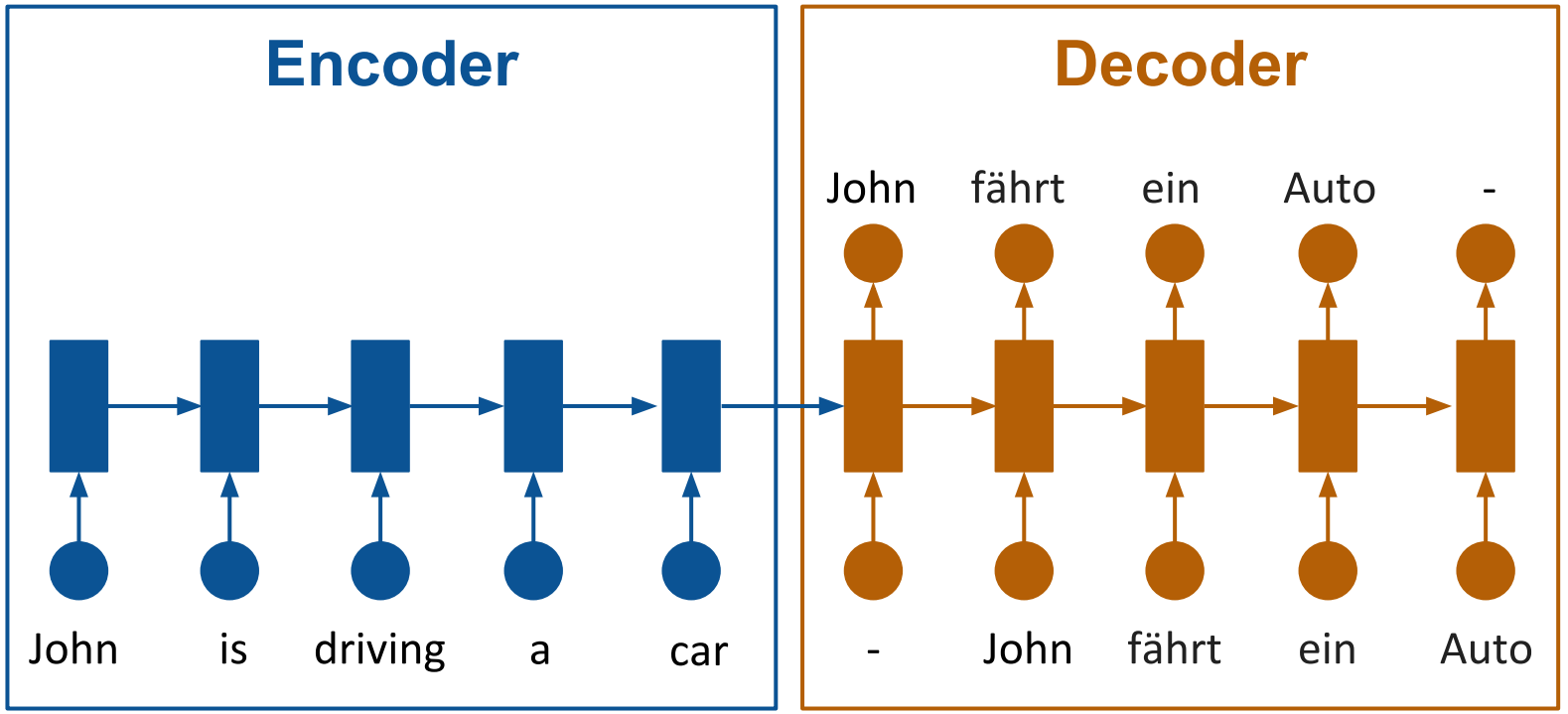
This is the Encoder-Decoder architecture!

Sequence to Sequence Model



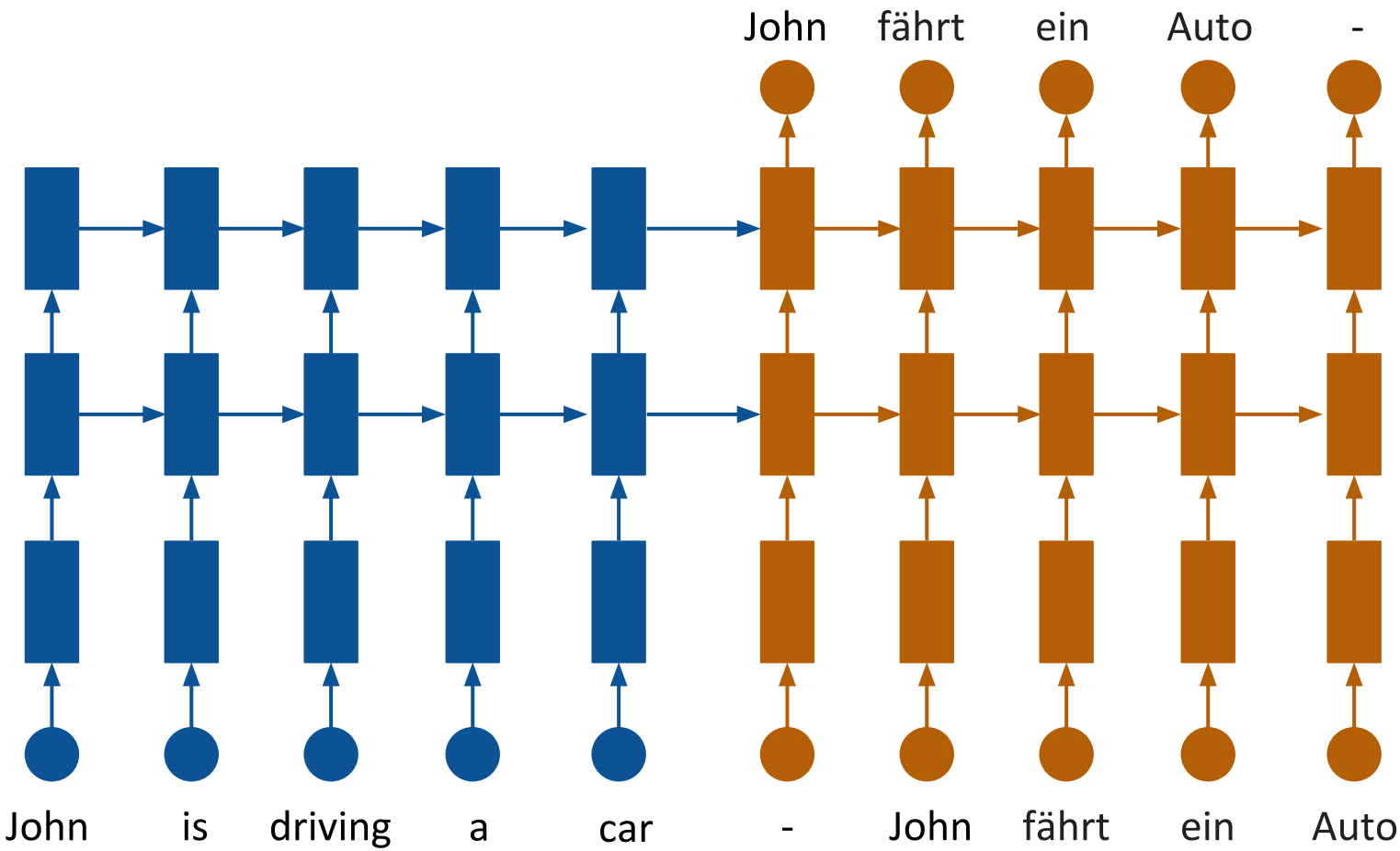
The model learns to read a source sentence and predict a target sentence

Sequence to Sequence Model

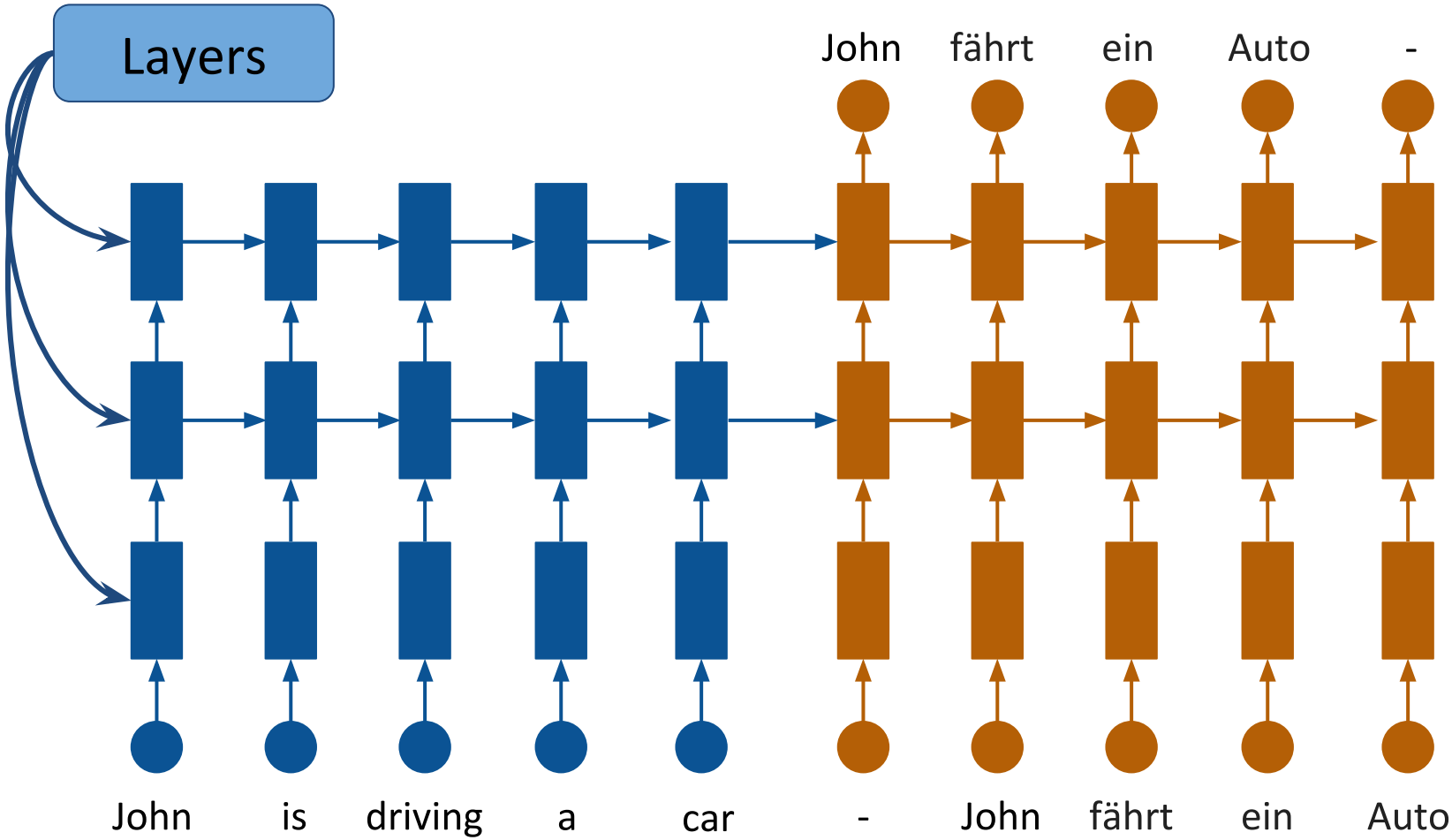


We generally do not produce any outputs in the Encoder, since we are interested in the translation only

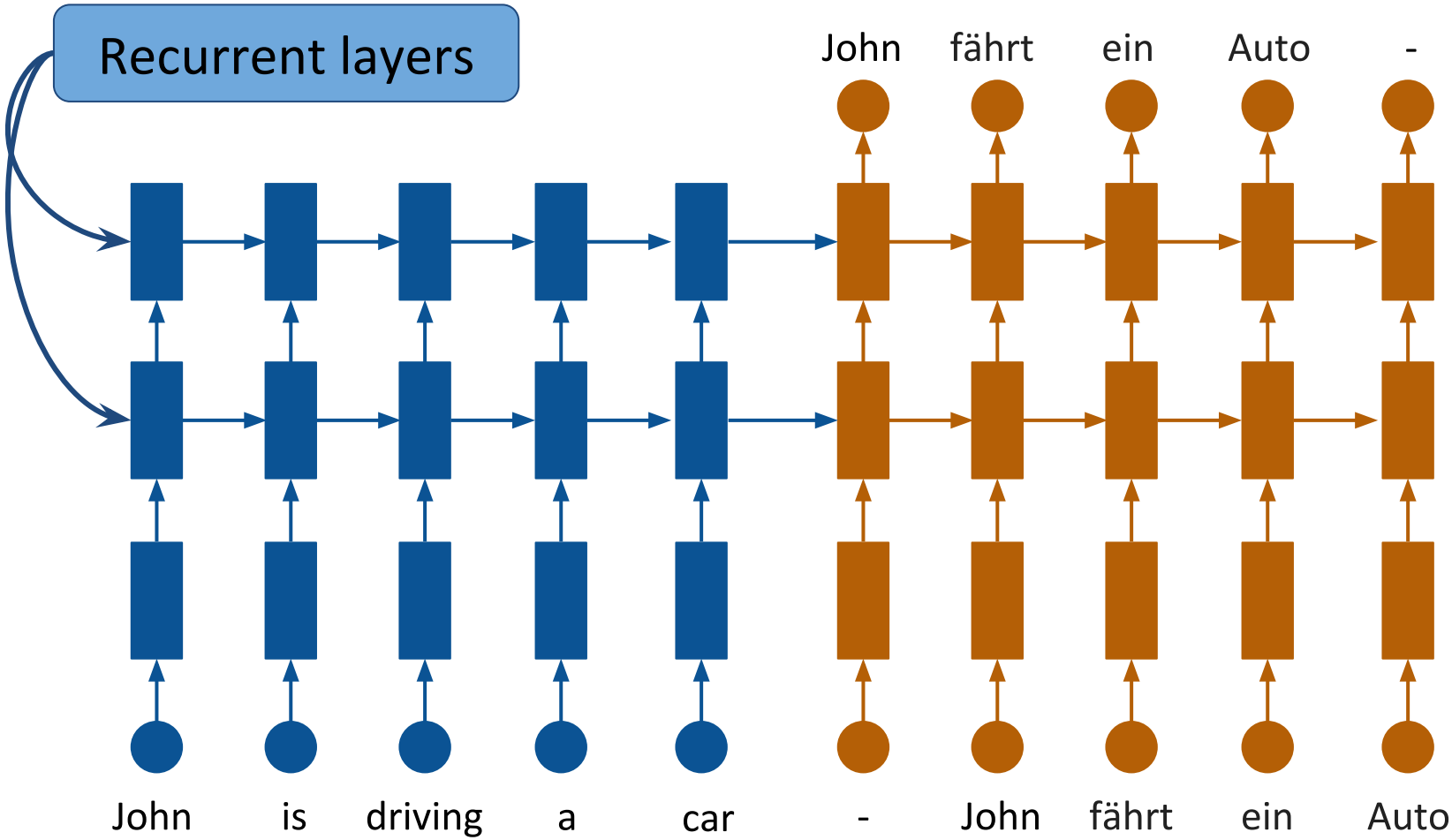
Sequence to Sequence Model



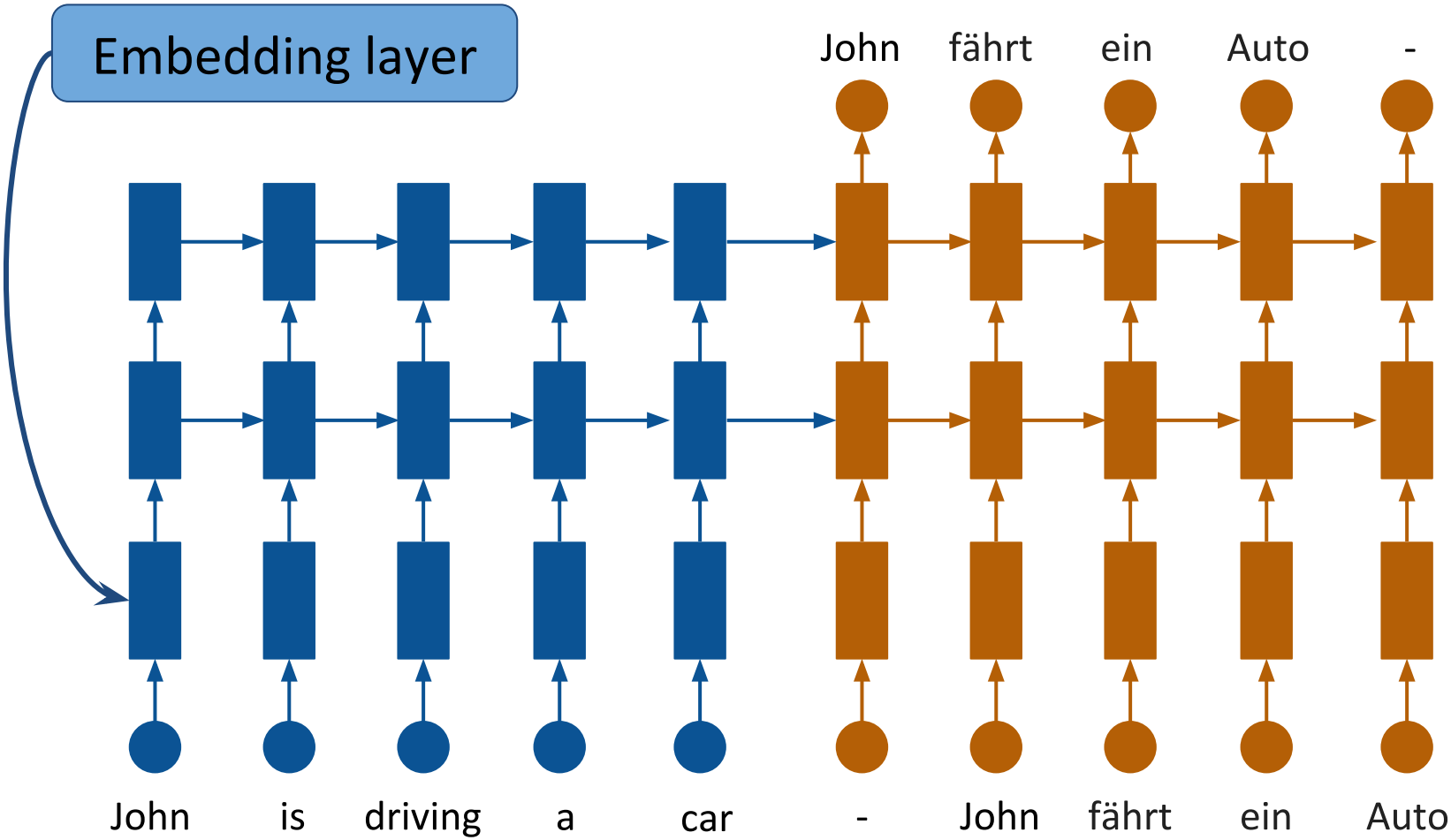
Sequence to Sequence Model



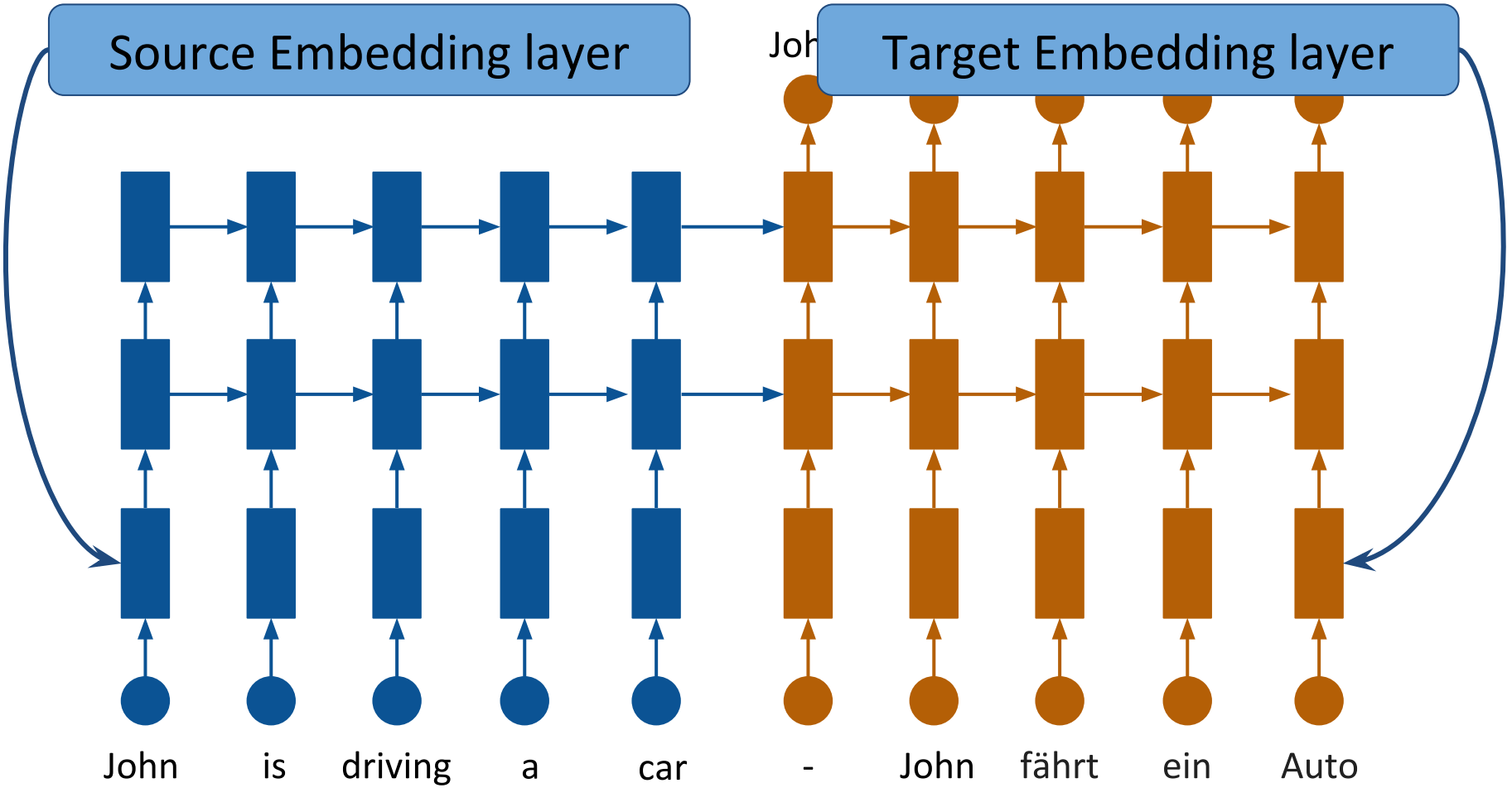
Sequence to Sequence Model



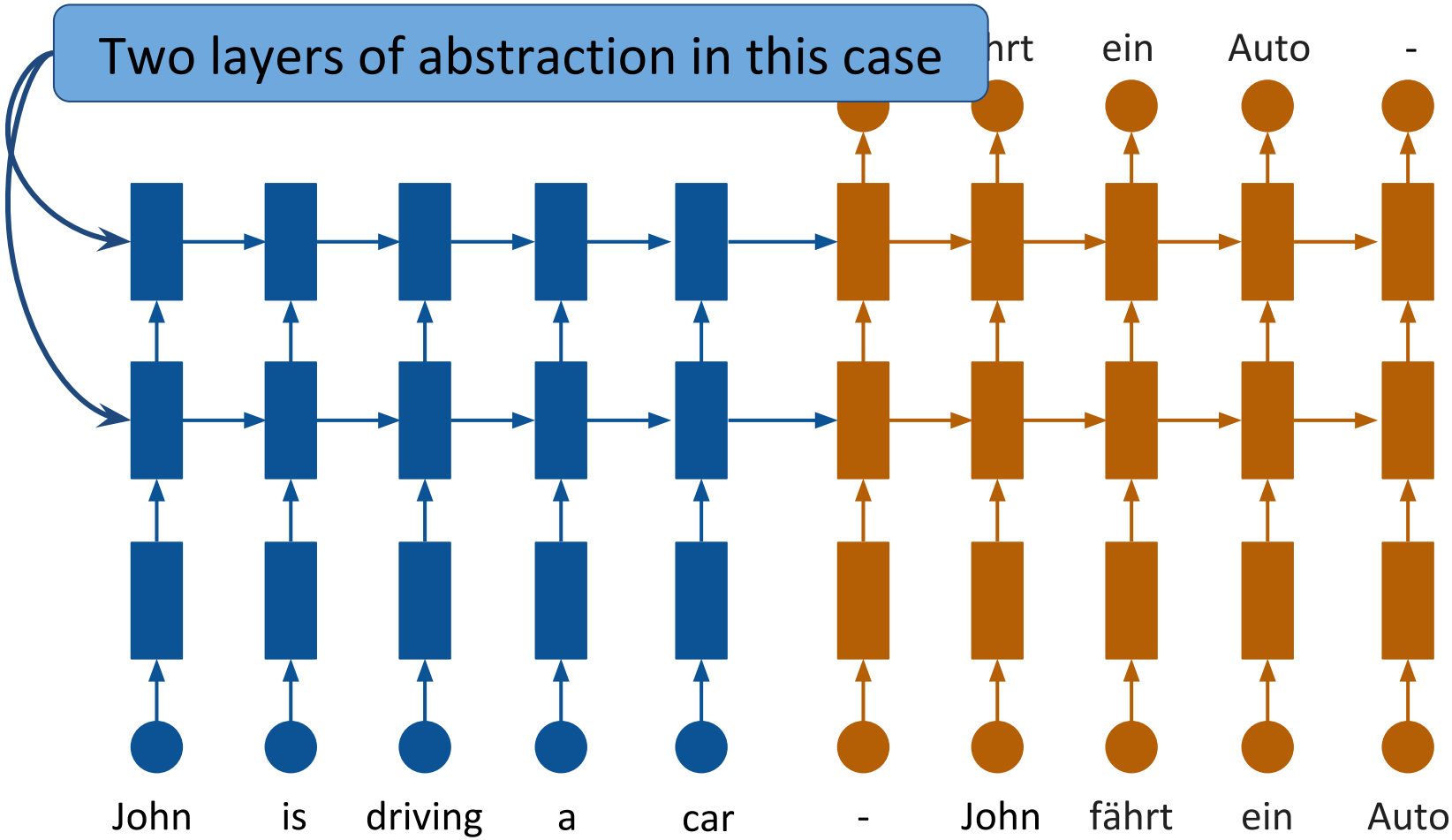
Sequence to Sequence Model



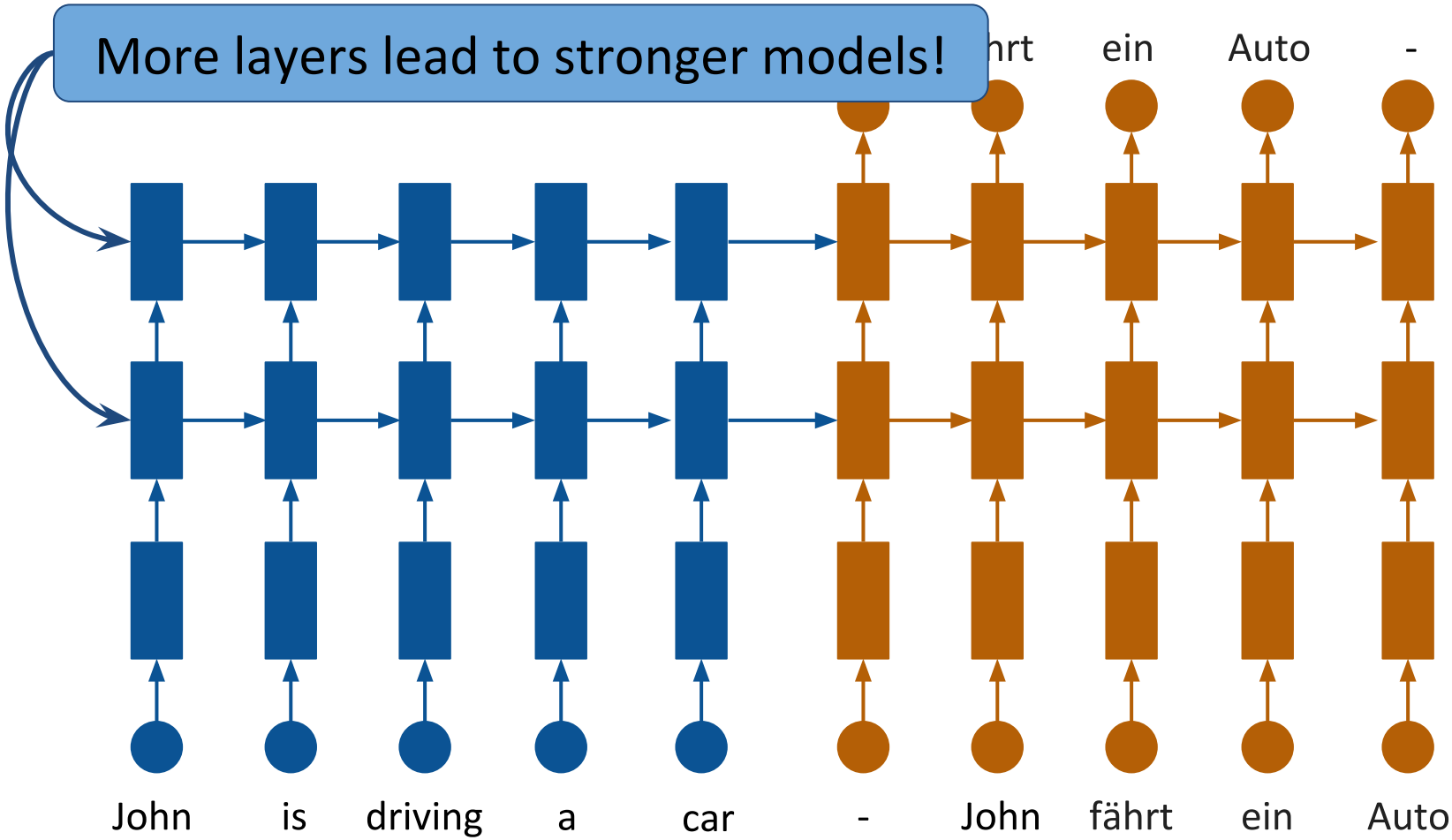
Sequence to Sequence Model



Sequence to Sequence Model



Sequence to Sequence Model



Sequence to Sequence Model

Intuitively, the **seq2seq** model learns to:

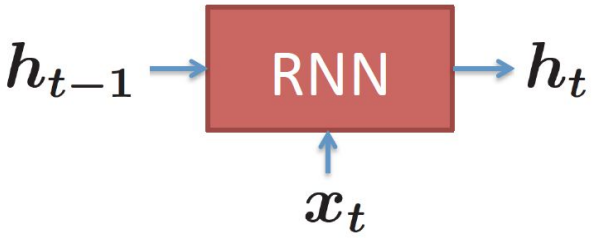
- Read a source sentence
- Predict a target word

The network learns to see a *source boundary symbol* to start predicting target words till a *target boundary marker* is predicted

Sequence to Sequence Model

Other than the embedding layer, each unit of encoder and decoder layer is recurrent

- Vanilla

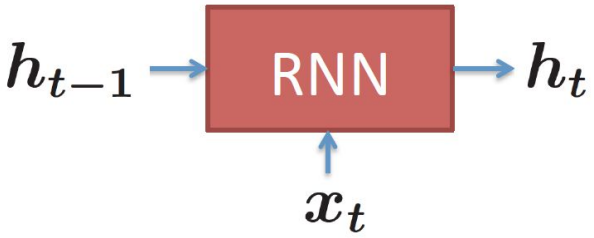


Vanishing gradient problem

Sequence to Sequence Model

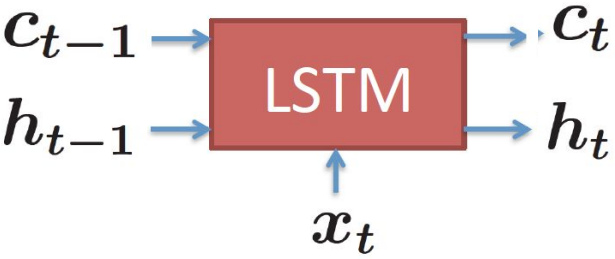
Other than the embedding layer, each unit of encoder and decoder layer is recurrent

- Vanilla



Vanishing gradient problem

- LSTM



Adapted from Thang Luong - Neural Machine Translation

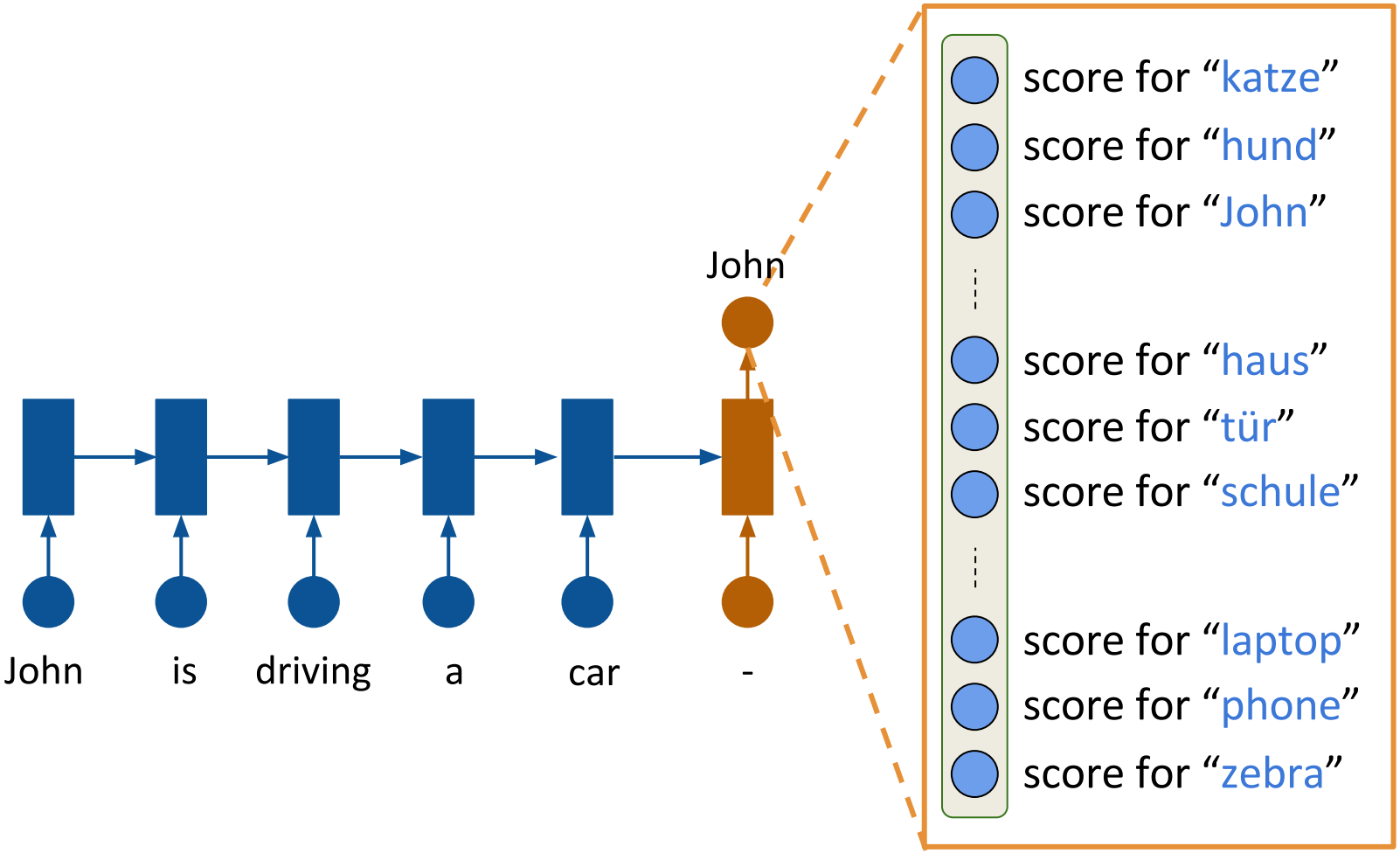
Reminder: Output Layer

- Similar to softmax linear classifier (lecture 5), the output scores are converted to a probability distribution using softmax

$$p_i = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

Here, total classes is equal to **vocabulary size of the target language**

Output Layer



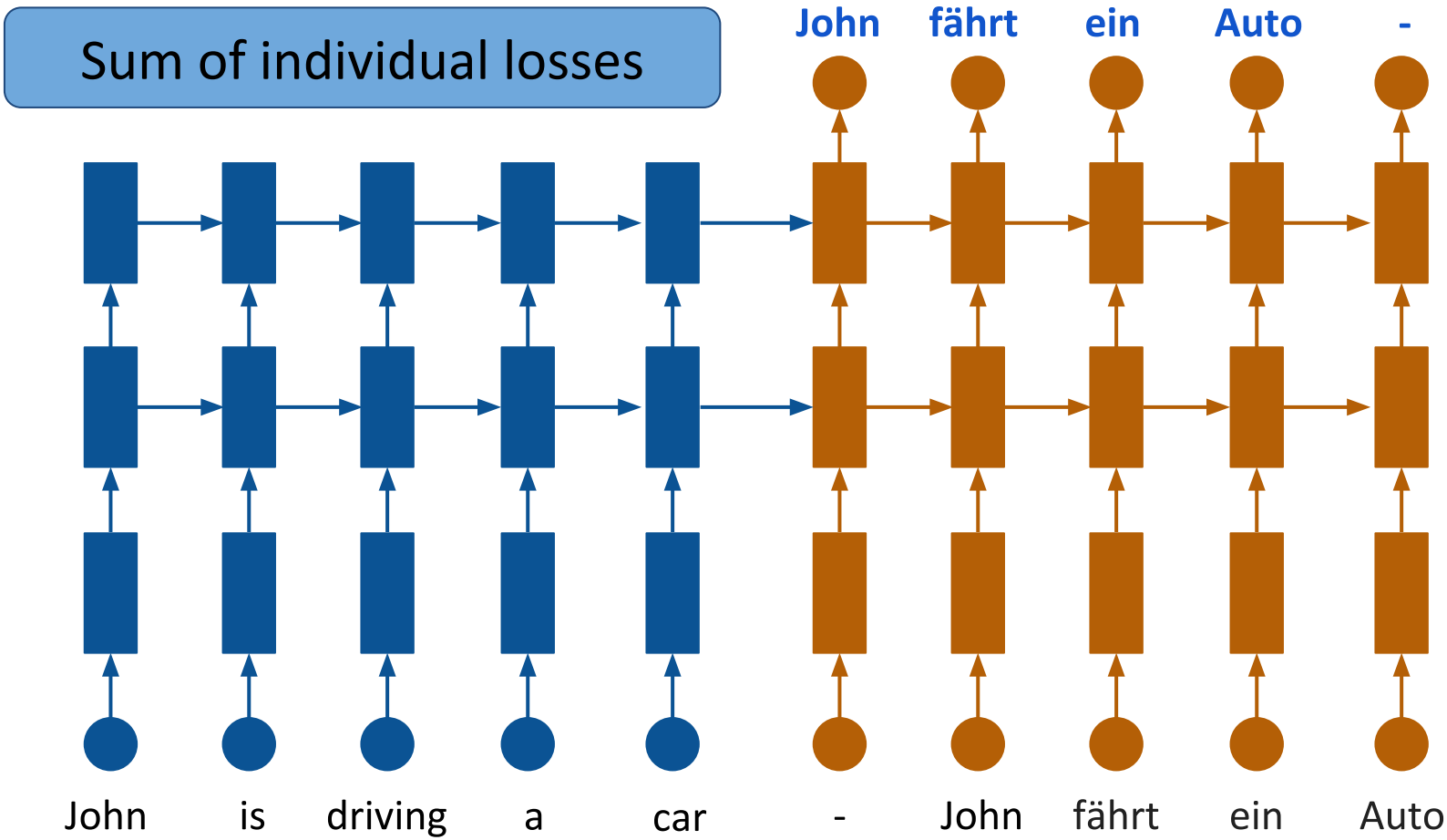
Training Loss

- Individual loss at instance i is (*lecture 5*):

$$L_i = -\log(softmax(f)_c)$$

- Total loss of the network is equal to **sum of individual losses** for each predicted word

Sequence to Sequence Model



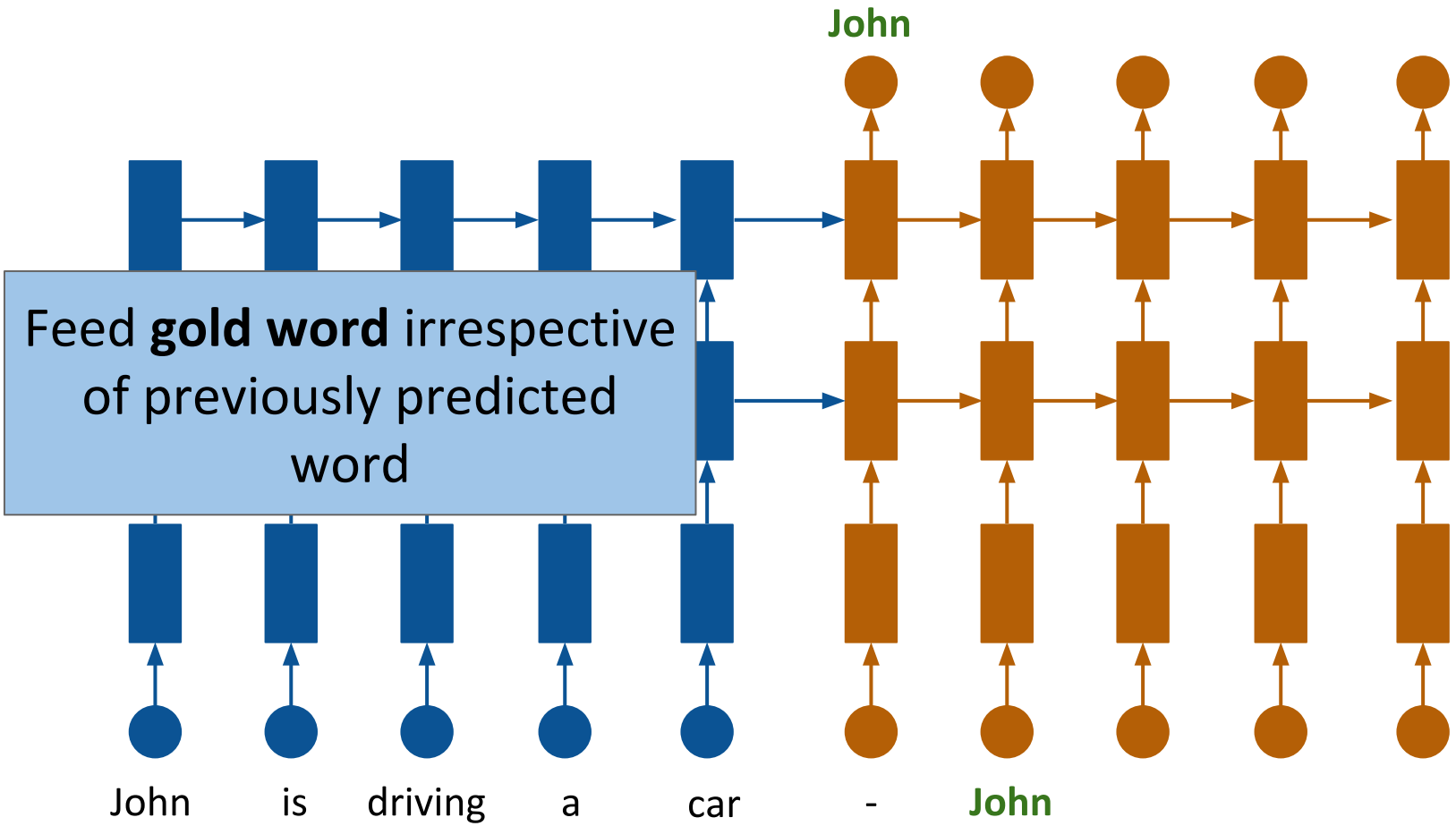
Training a Neural MT Model

- Read a source sentence till end-of-sentence boundary symbol is read
- Start predicting a target word at each time step until the target end-of-sentence symbol is produced
- **Loss:** how bad are we in predicting target words
- Backpropagate the accumulated loss to all parameters using **Backpropagation through time**
- Repeat till convergence
- Usually use minibatch gradient descent with batch sizes of 64-128

Training vs. Testing

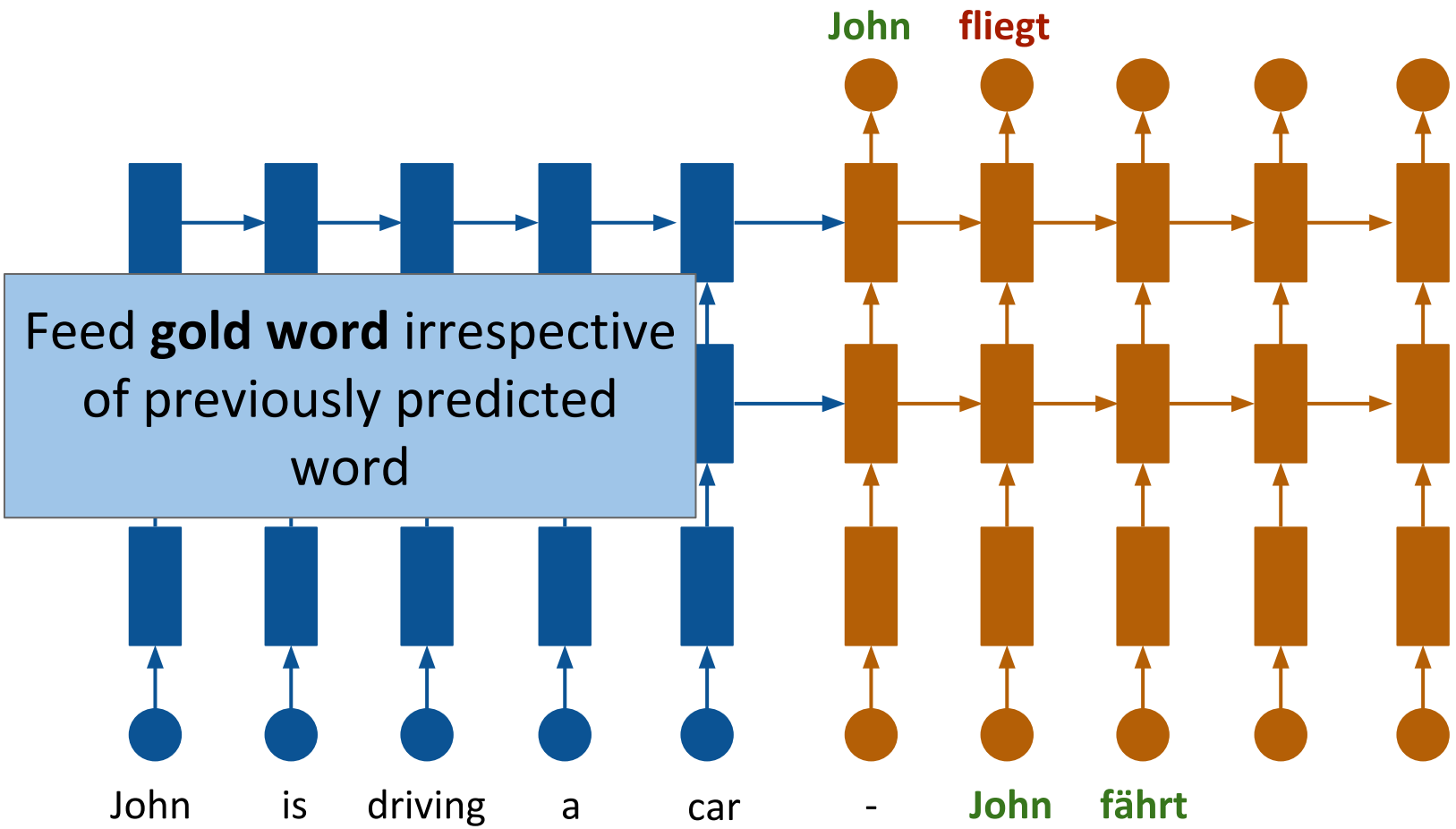
- During training, both source and target sentences are available
  - for every next target word to predict, previous correct word is known
- At test time, only source sentences are available
  - feed most probable word at step `t-1` as input to step `t`
  - Decode using beam search

NMT: Training



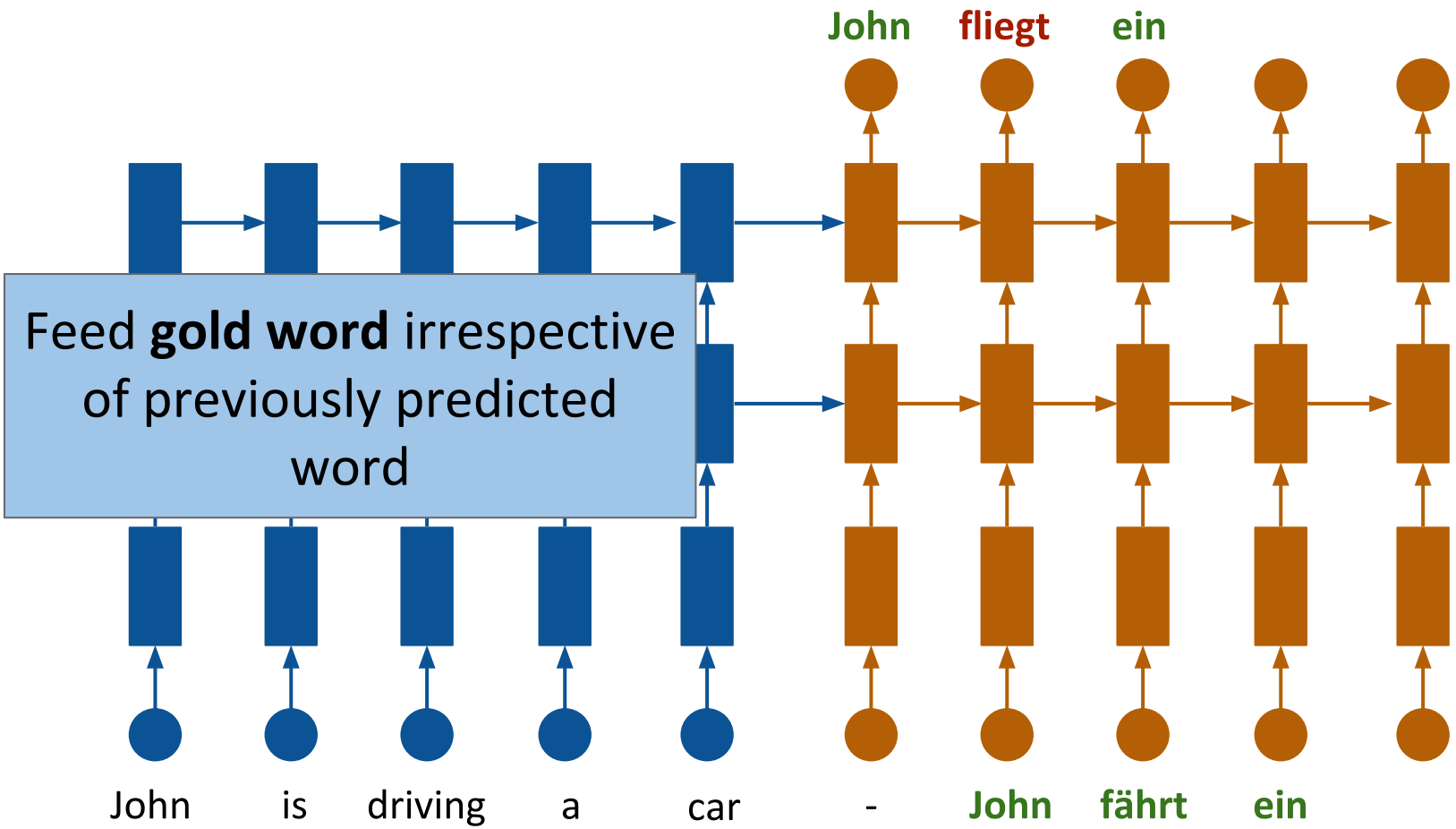
Adapted from Thang Luong - Neural Machine Translation

NMT: Training



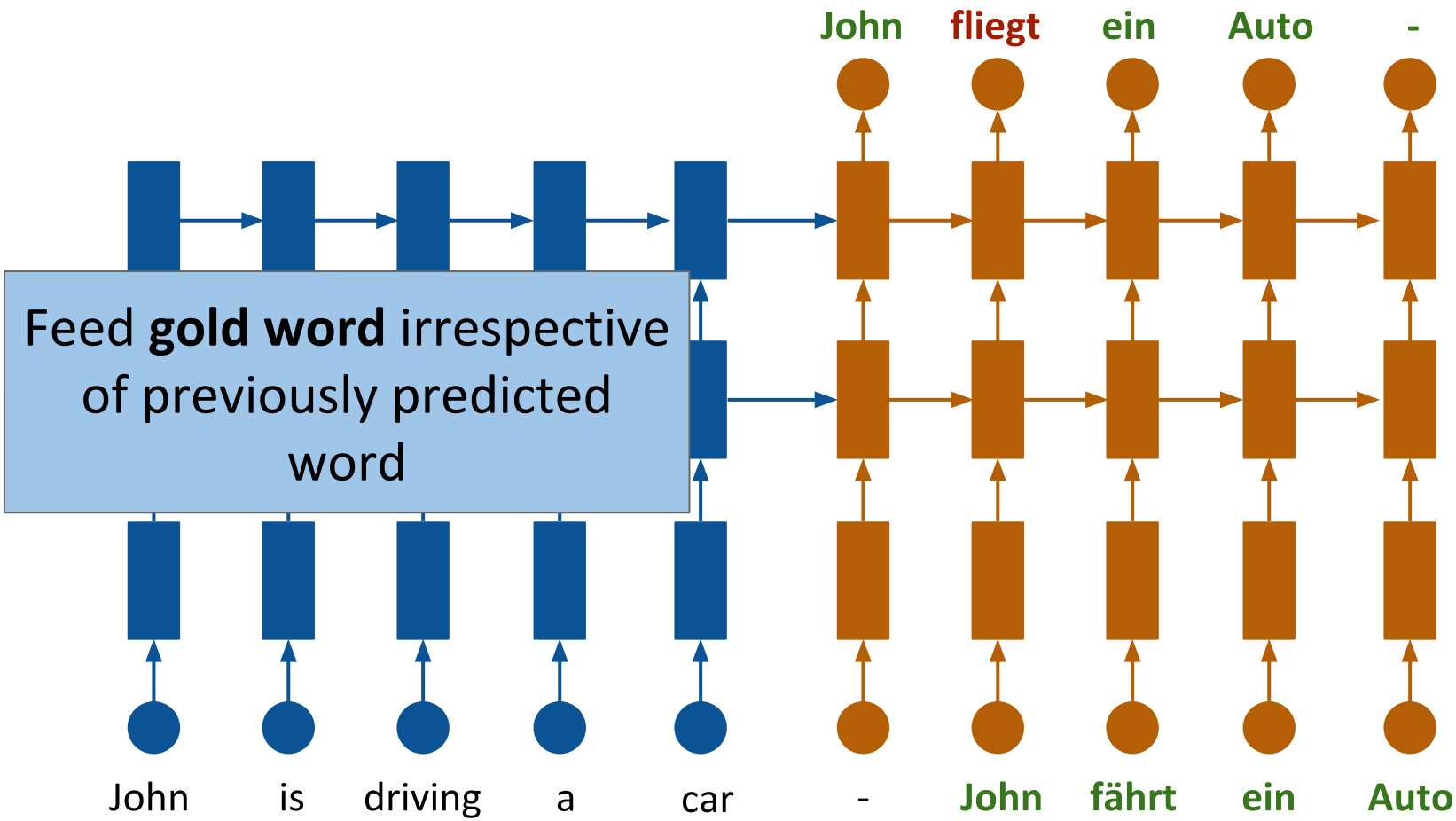
Adapted from Thang Luong - Neural Machine Translation

NMT: Training



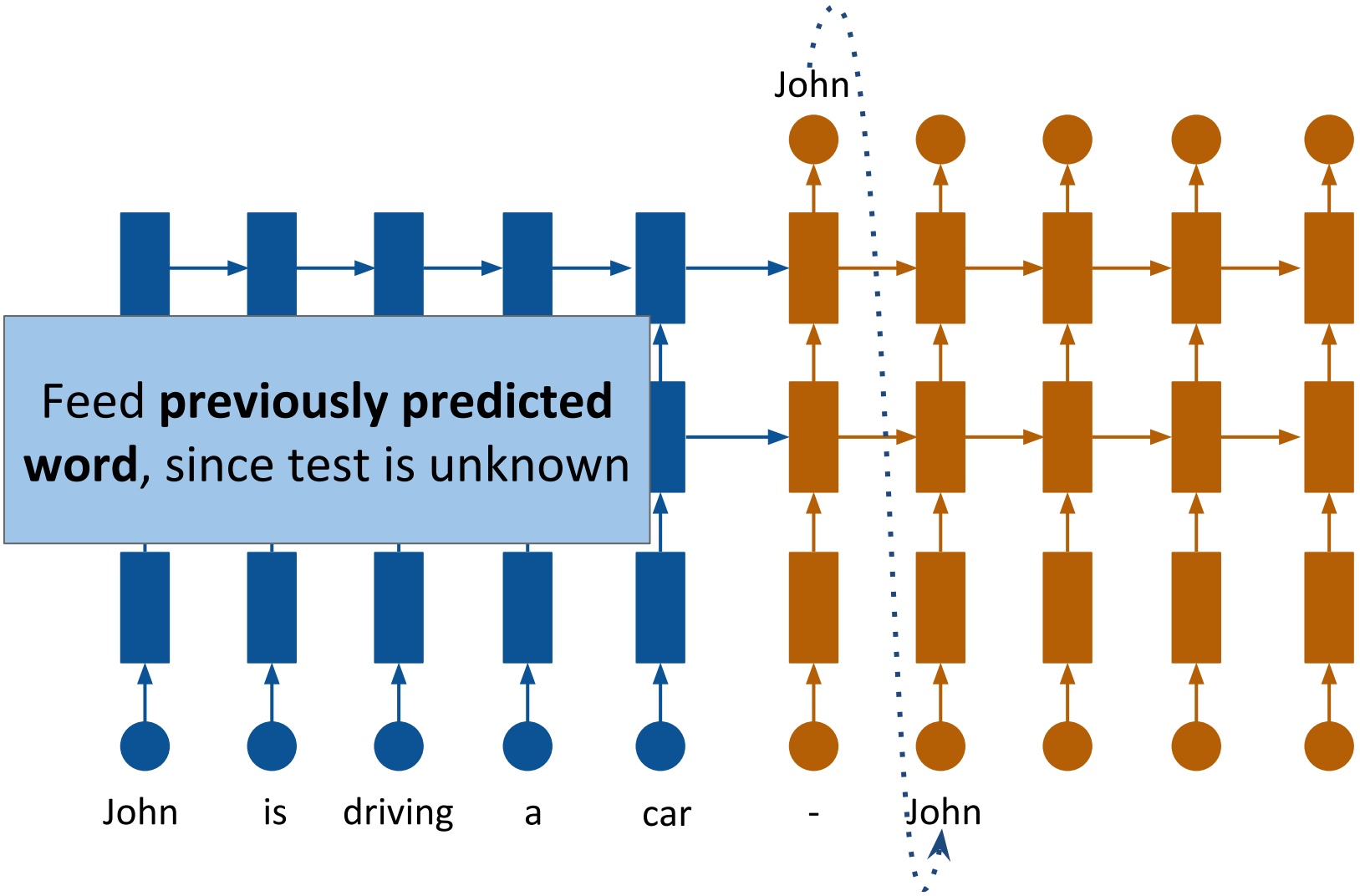
Adapted from Thang Luong - Neural Machine Translation

NMT: Training



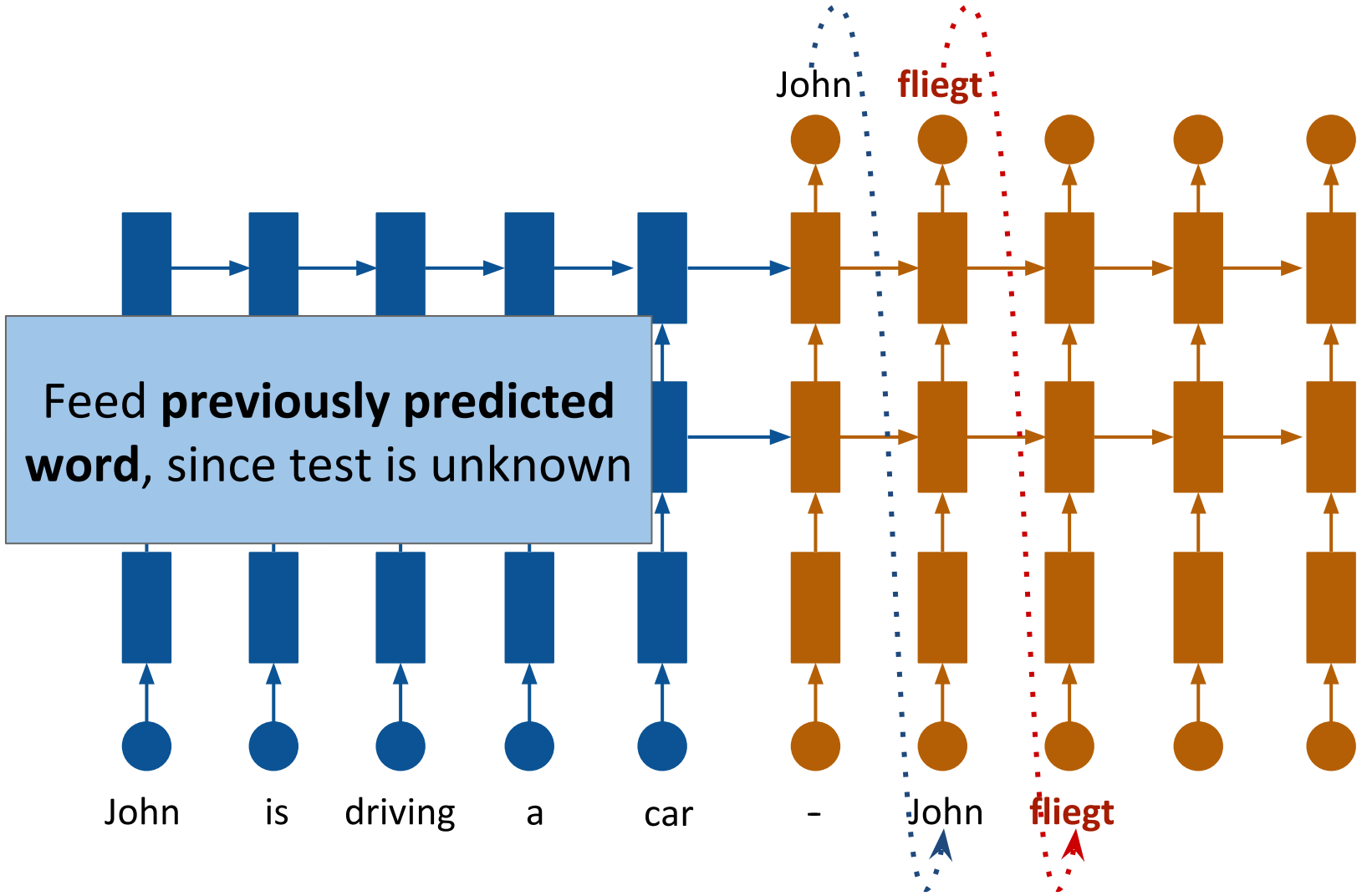
Adapted from Thang Luong - Neural Machine Translation

NMT: Testing



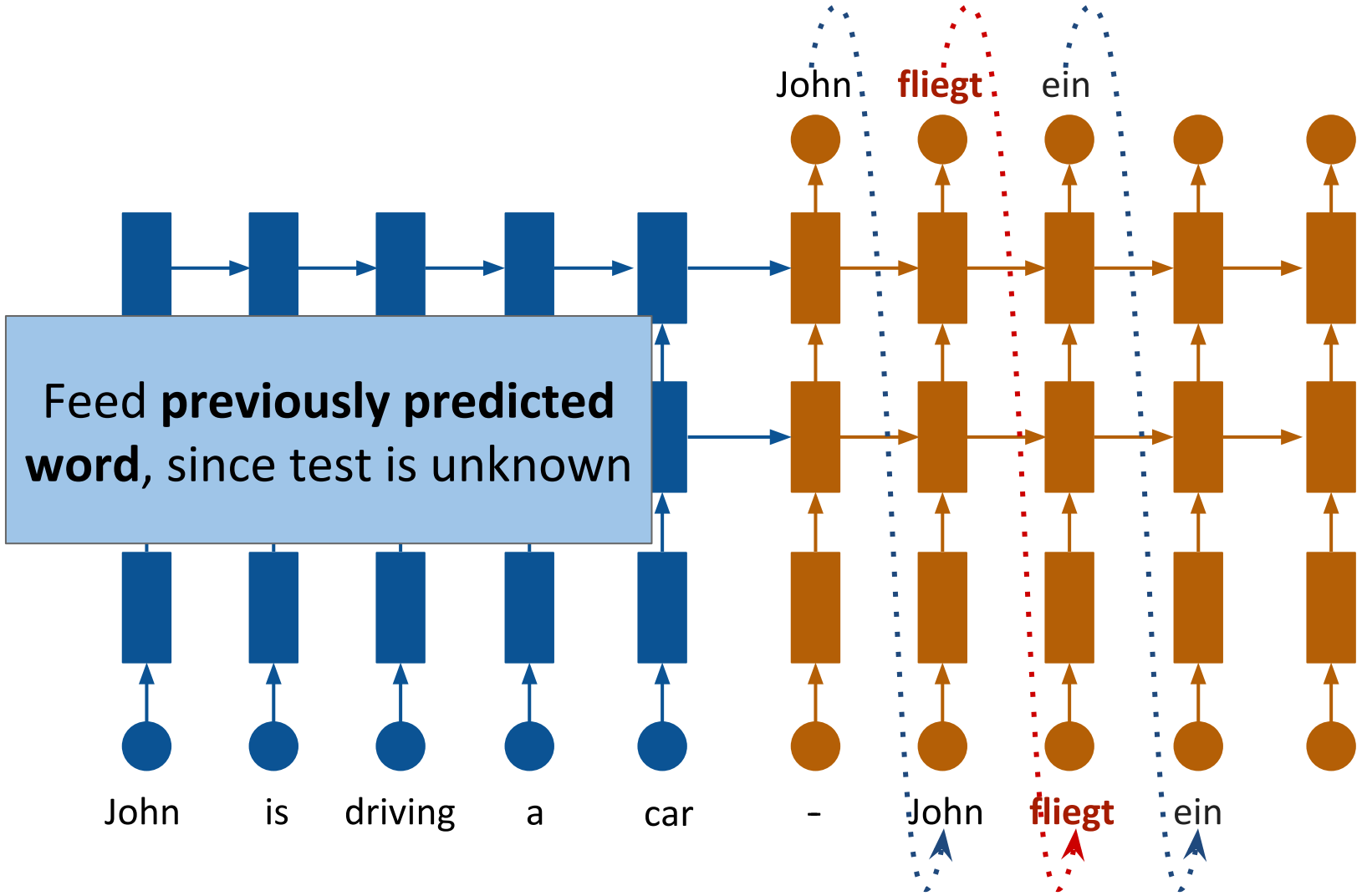
Adapted from Thang Luong - Neural Machine Translation

NMT: Testing



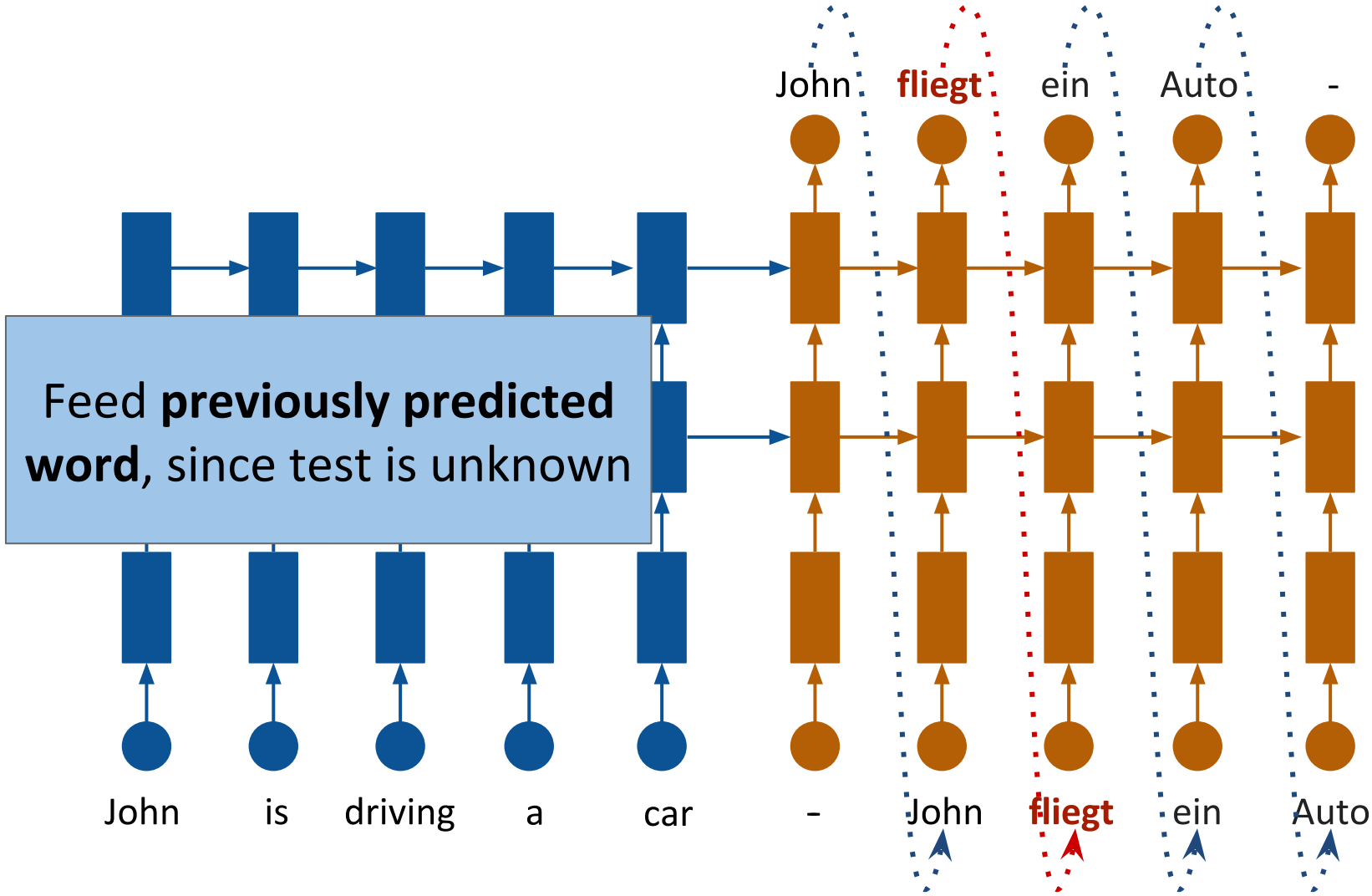
Adapted from Thang Luong - Neural Machine Translation

NMT: Testing



Adapted from Thang Luong - Neural Machine Translation

NMT: Testing



Adapted from Thang Luong - Neural Machine Translation

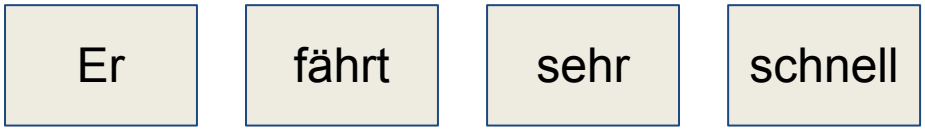
Testing - Greedy Search

- Choose the most probable word at each time step t
- As we have seen before, it's efficient but not very accurate

Testing - Beam Search

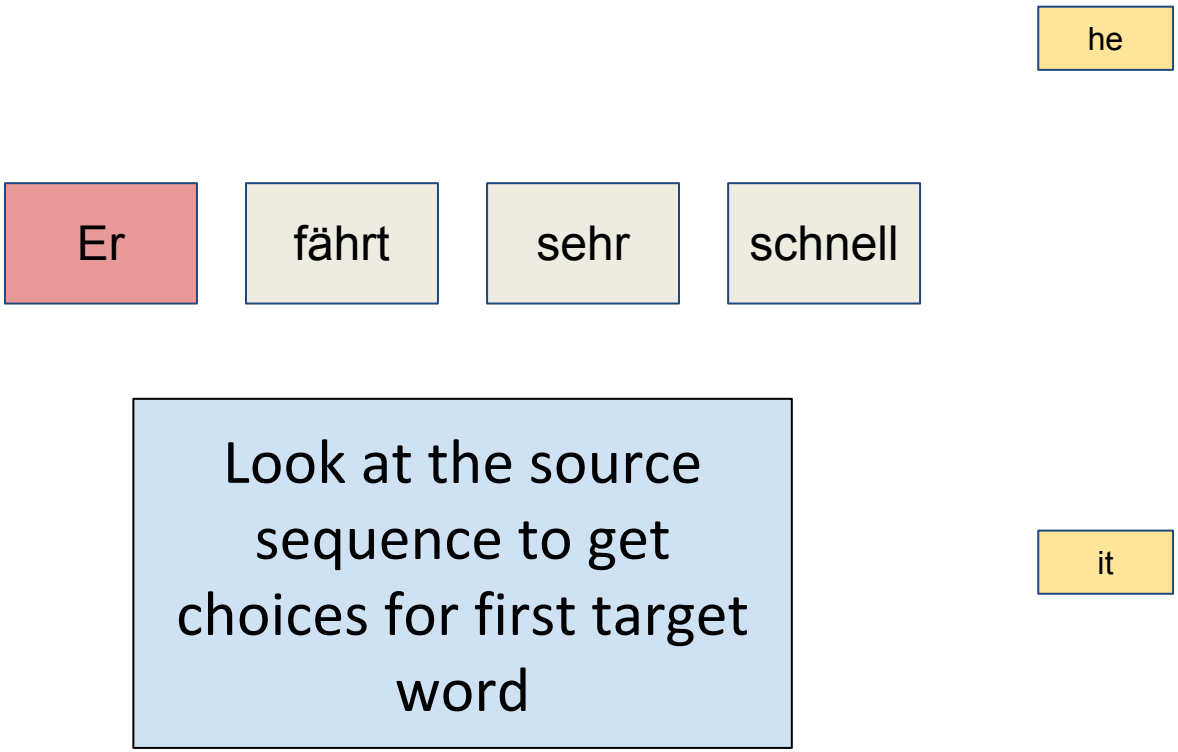
- Maintain k hypothesis at any given time
- Computationally expensive, but results in better quality

Testing - Beam Search

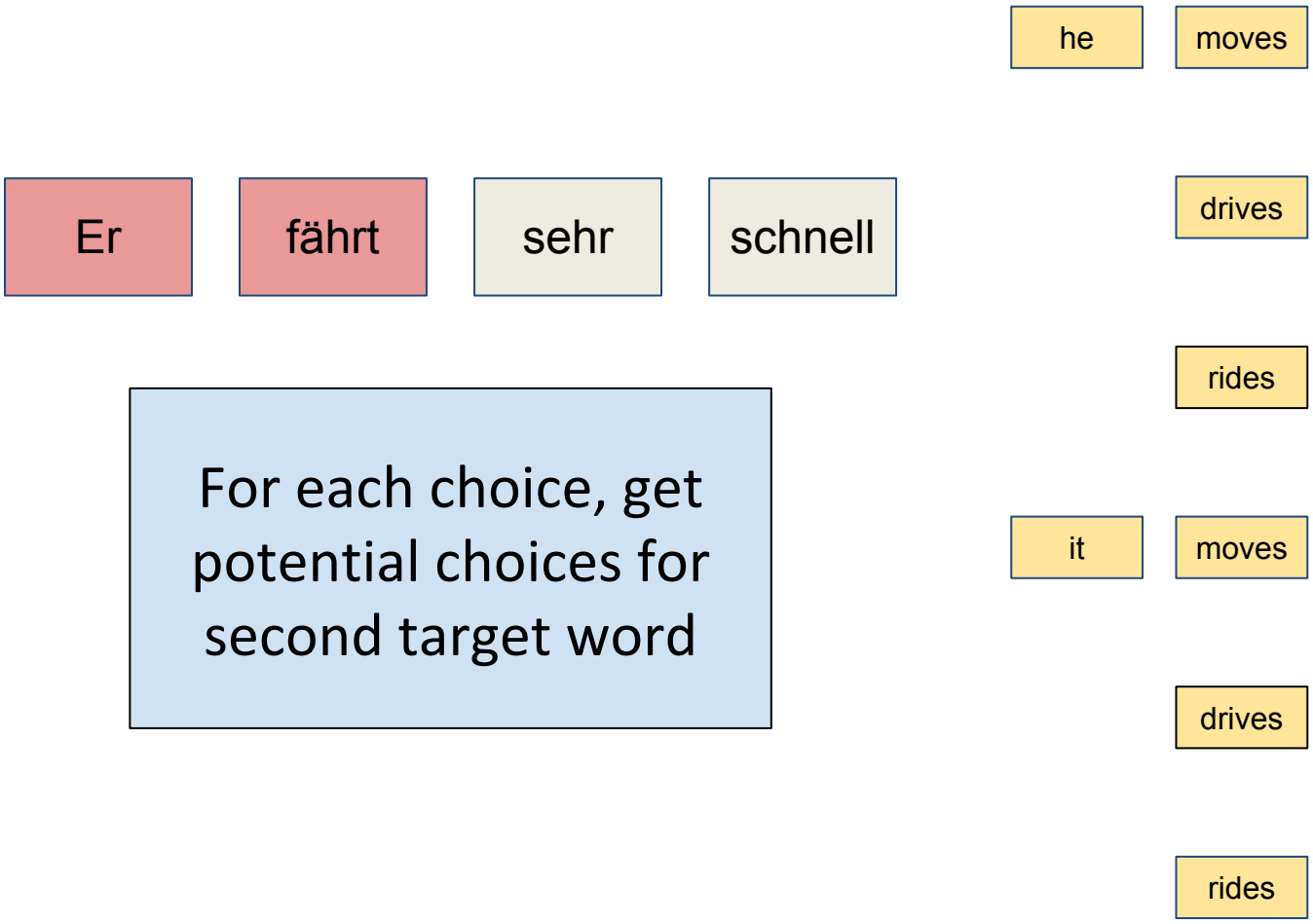


Let us consider an example with k=4, i.e. we will keep at most 4 “best” hypothesis

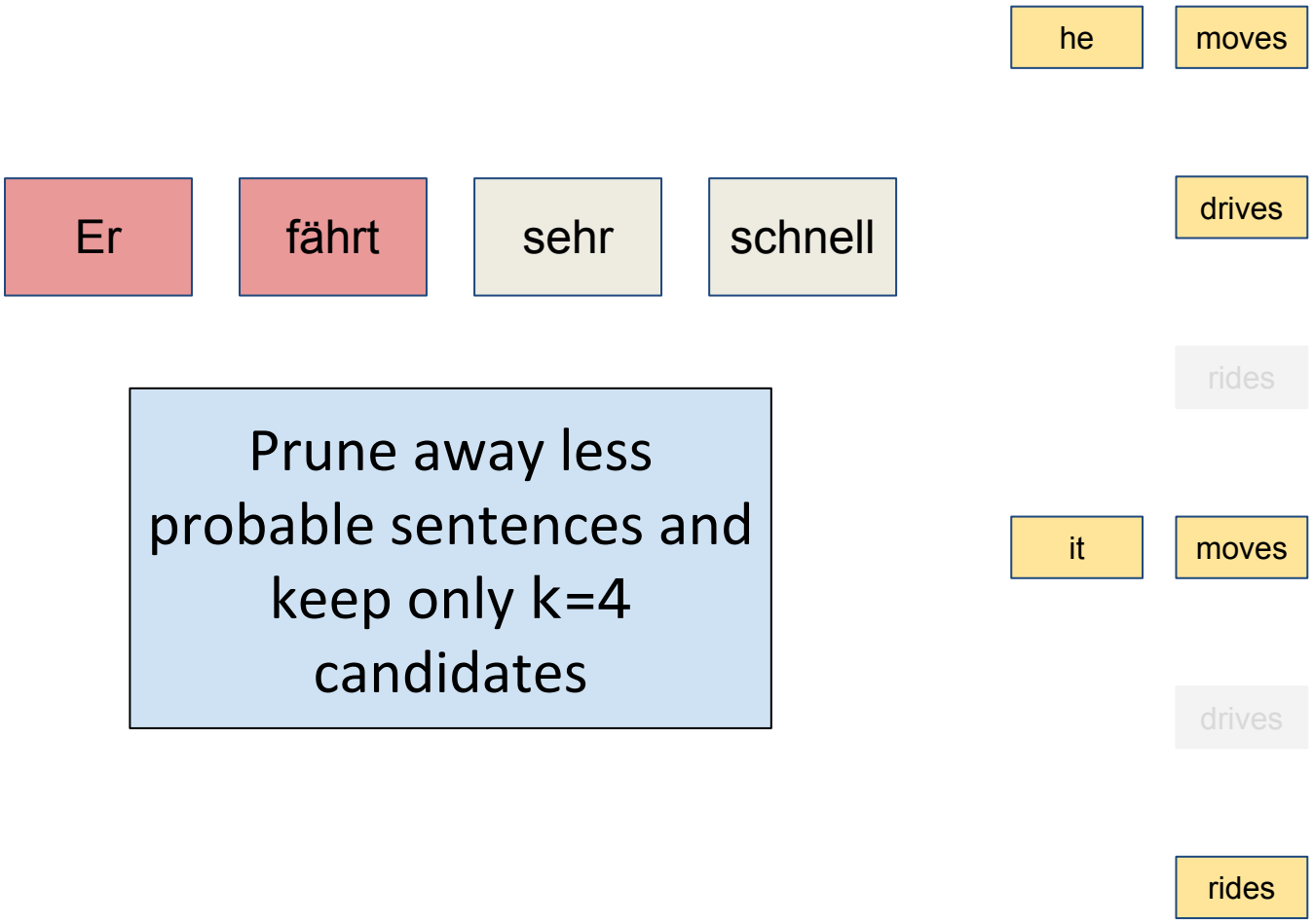
Testing - Beam Search



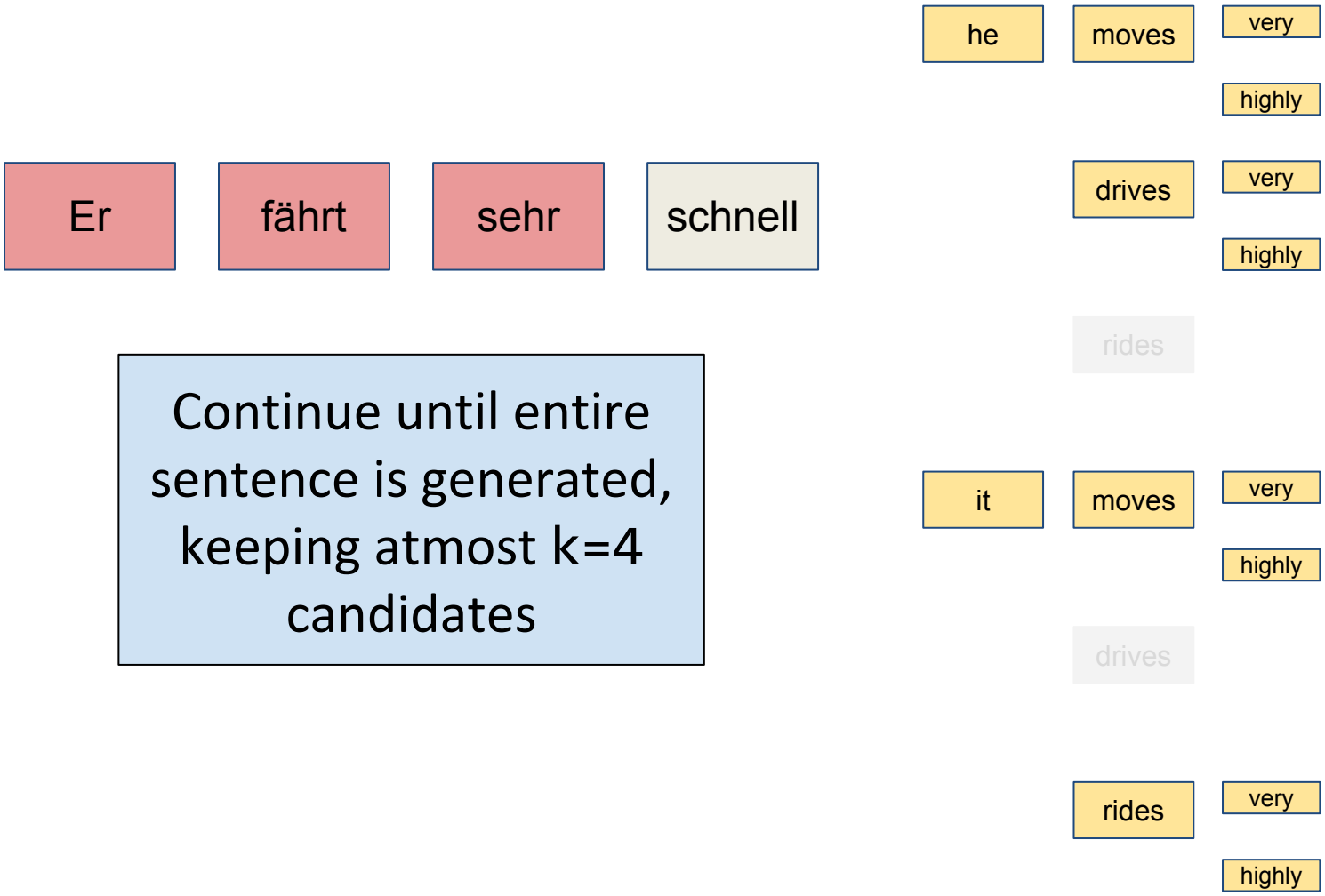
Testing - Beam Search



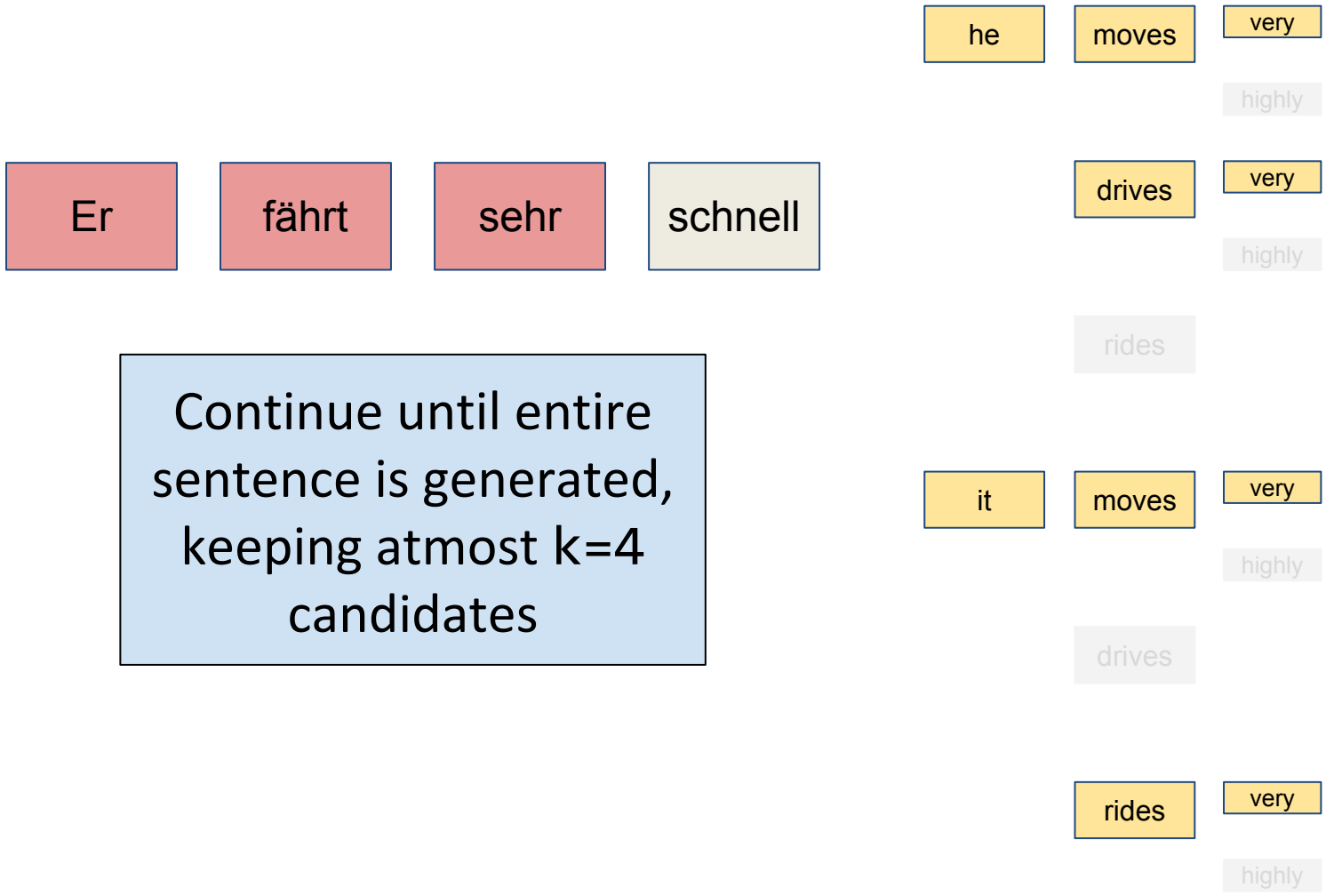
Testing - Beam Search



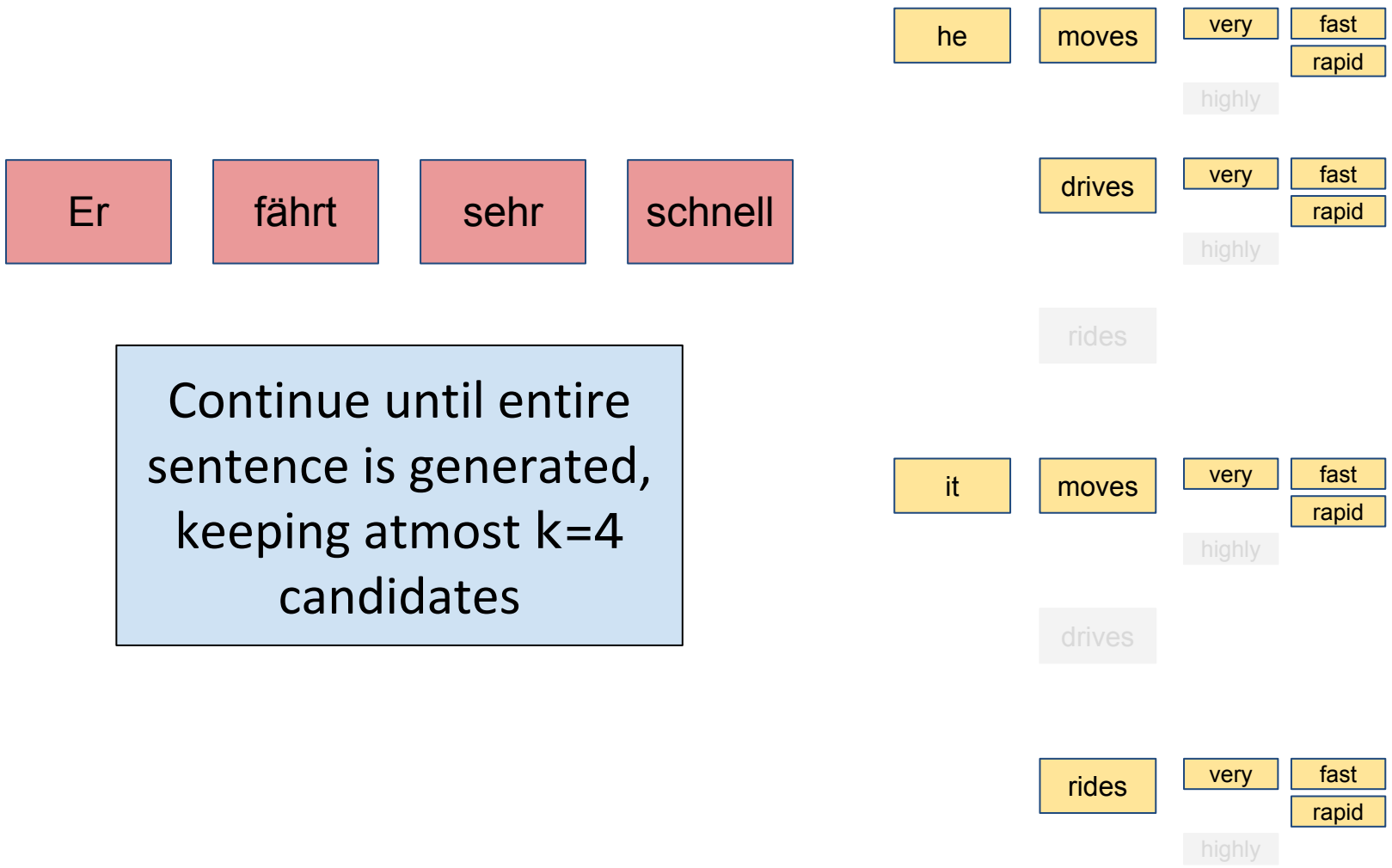
Testing - Beam Search



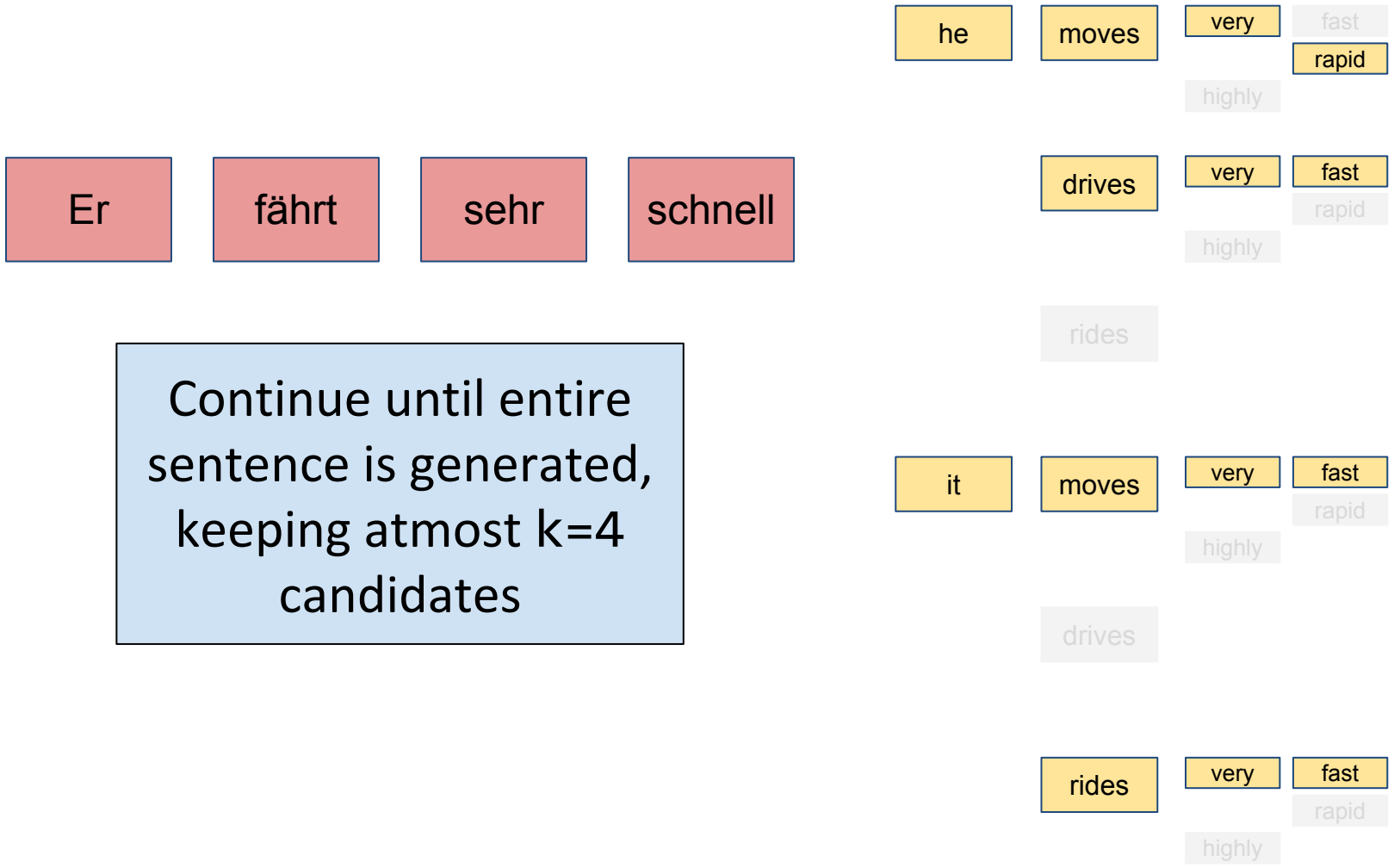
Testing - Beam Search



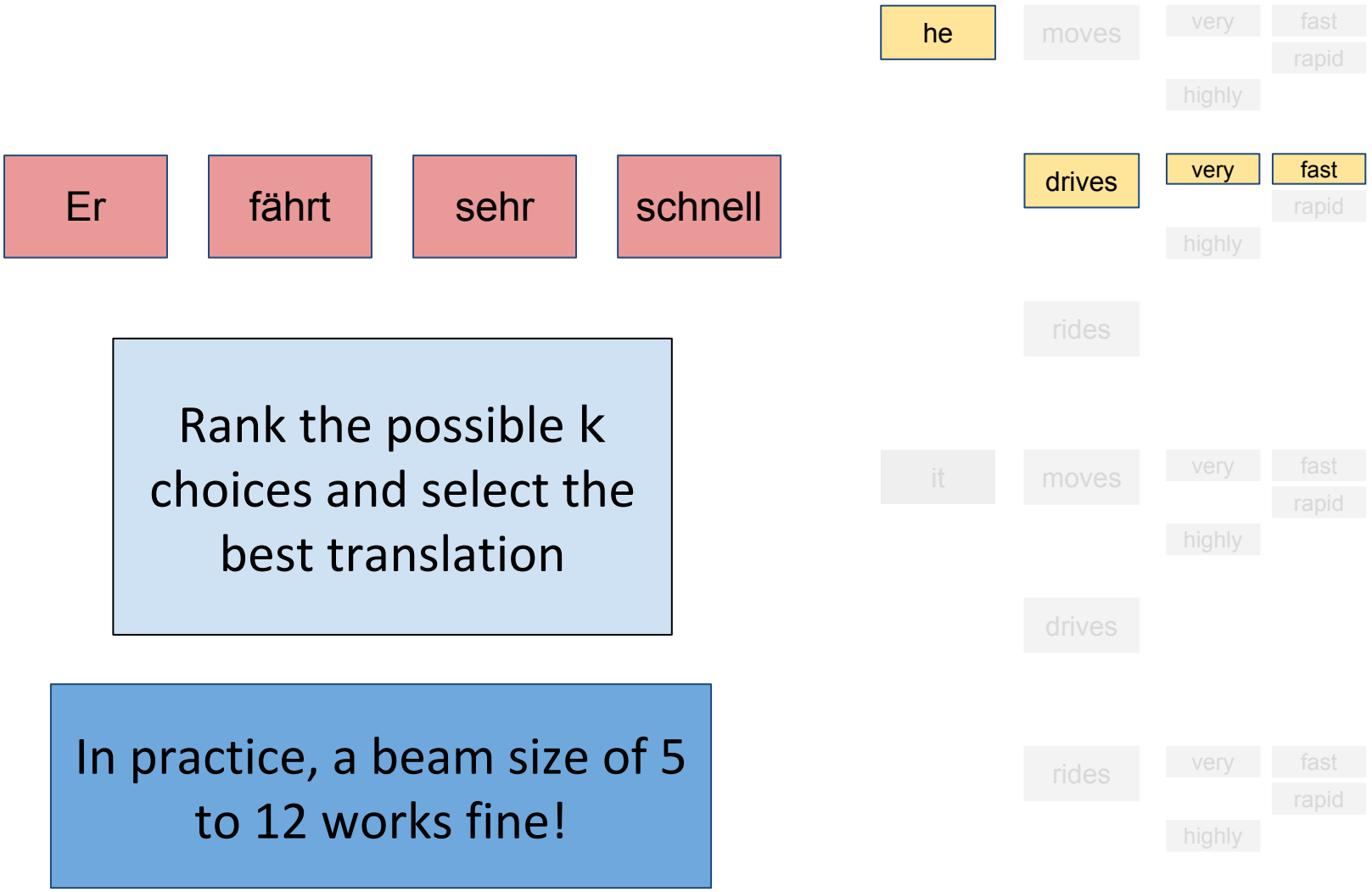
Testing - Beam Search



Testing - Beam Search



Testing - Beam Search



Rank the possible k choices and select the best translation

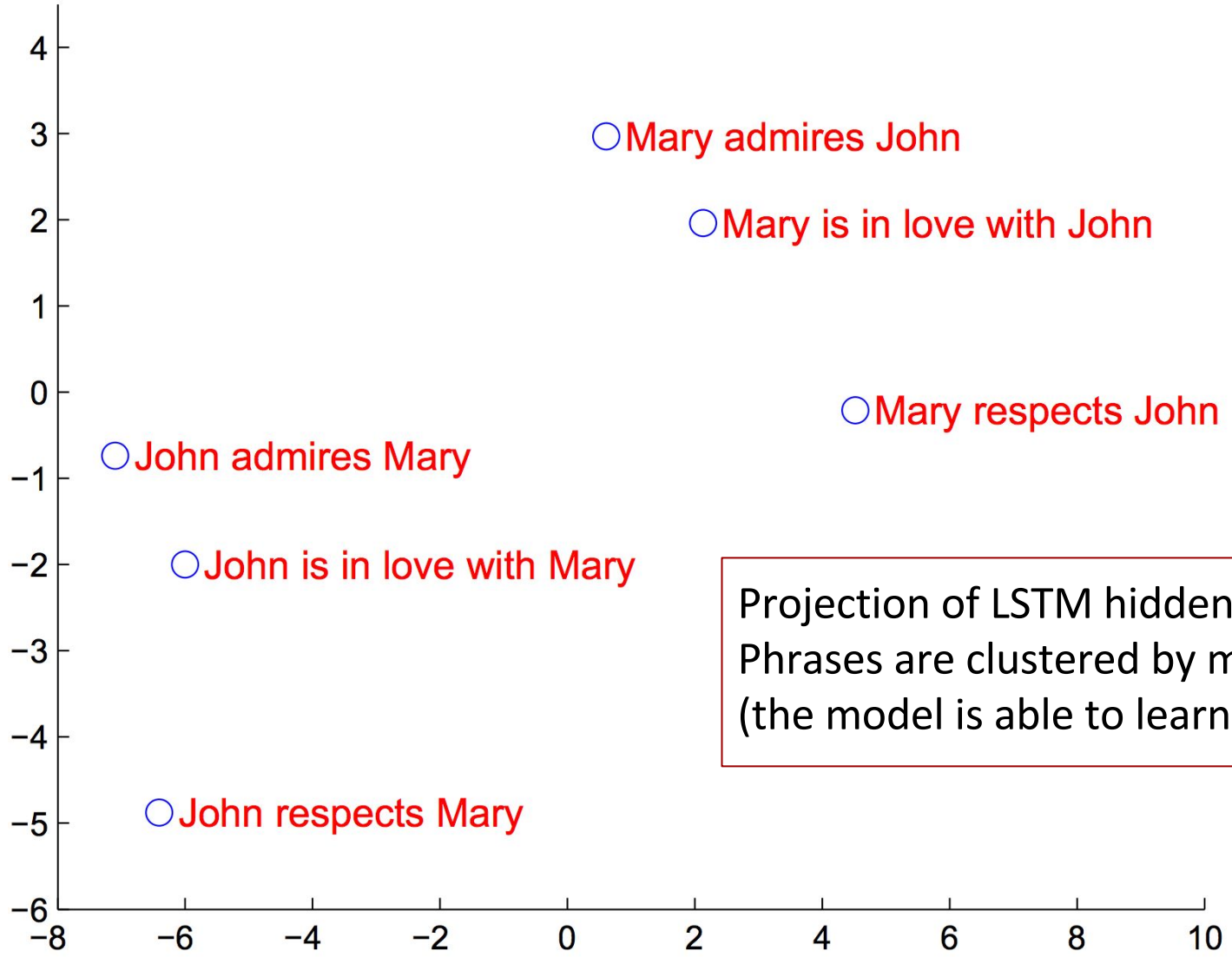
In practice, a beam size of 5 to 12 works fine!

Neural Machine Translation

- Sequence to sequence model
- **End-to-end training:** one loss to optimize all parameters (translation, alignment, LMs)
- **Better use of context:** use source sentences and previously predicted target words
- **Distributed word representations**: semantic similarities

Neural Machine Translation

Sequence to Sequence Learning with Neural Networks
https://arxiv.org/pdf/1409.3215.pdf

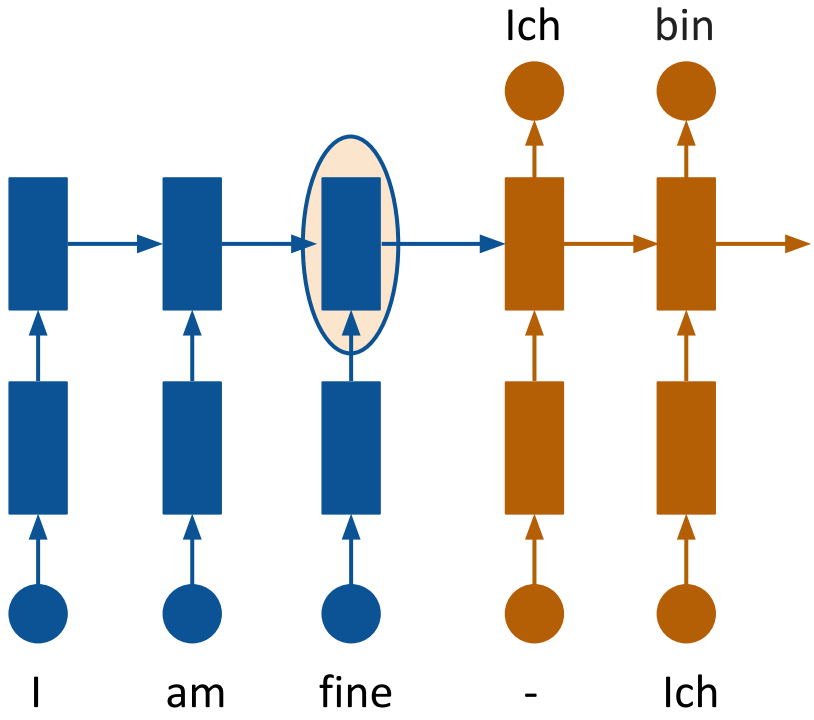


Projection of LSTM hidden states.
Phrases are clustered by meaning
(the model is able to learn roles)

Attention Mechanism

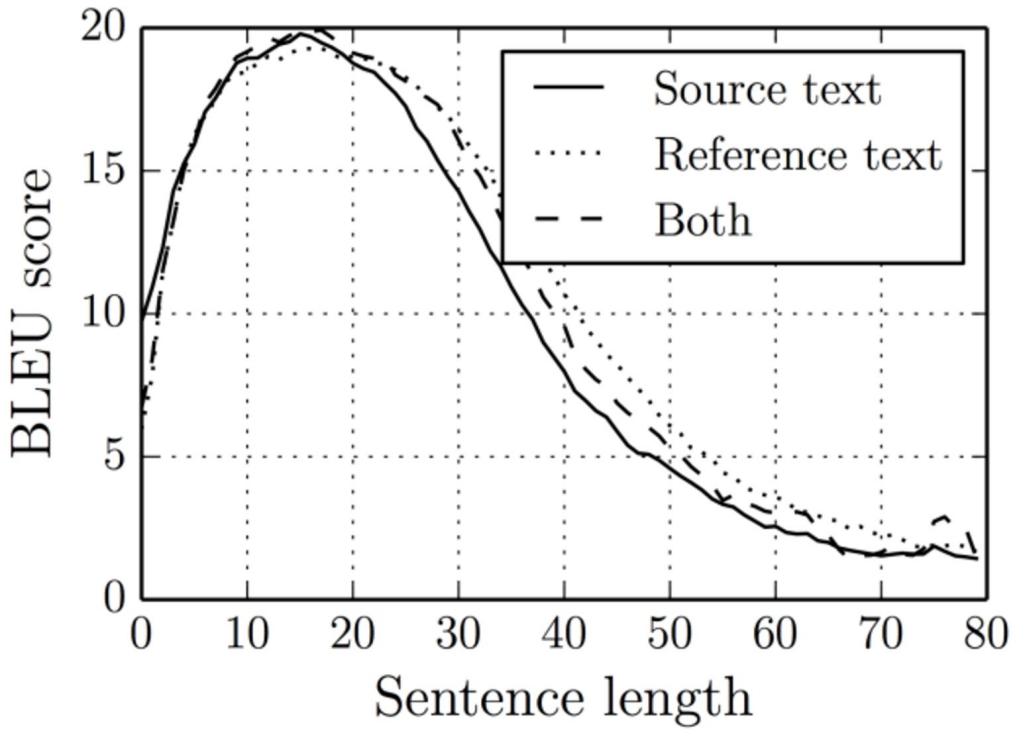
- Sequence to sequence model performs poorly when translating long sentences

- **Issue**: a sentence is represented as a fixed vector
- Relationship between source and target words is very abstract
- All words are not equally important to predict a target word



Attention Mechanism

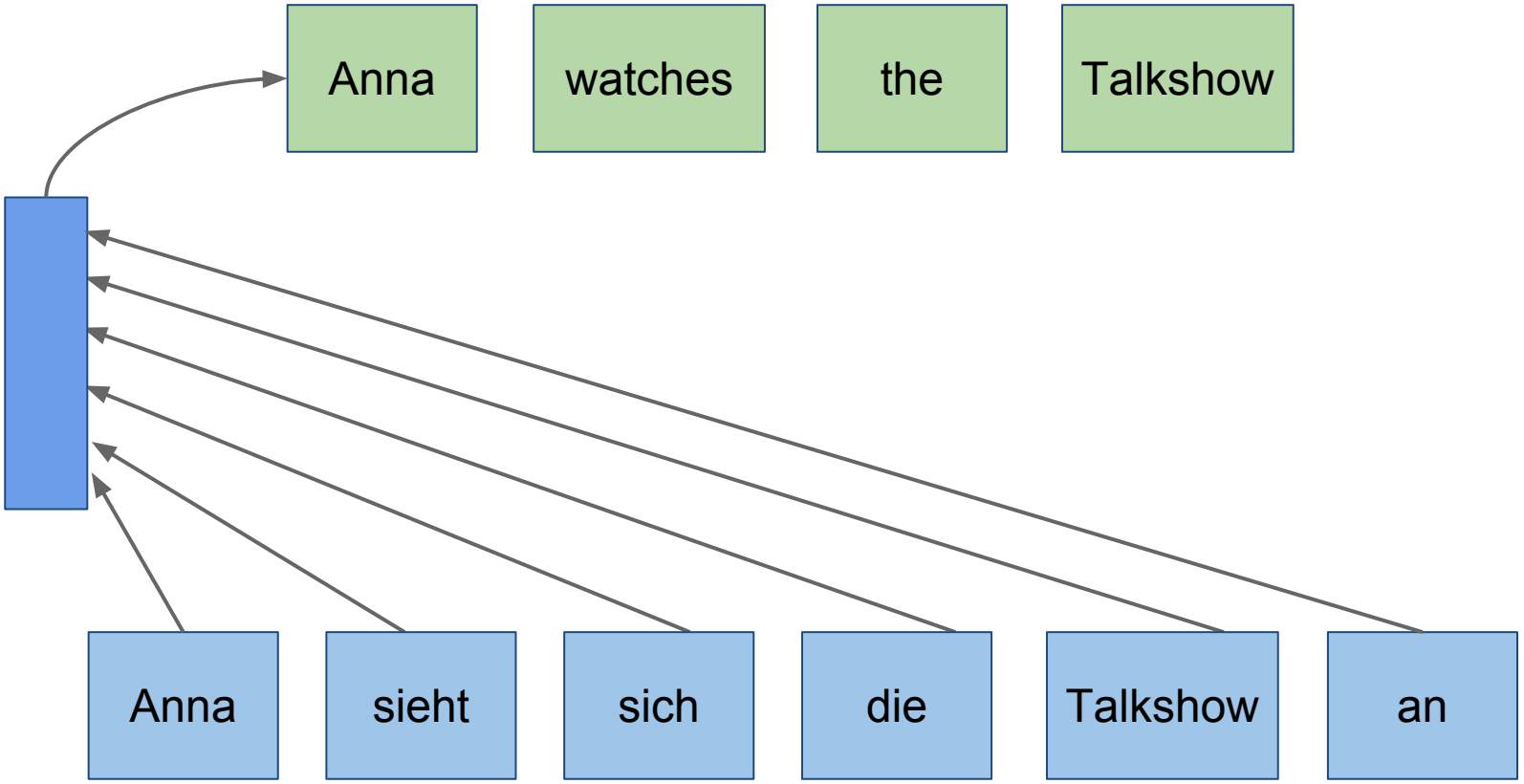
- Sequence to sequence model performs poorly when translating long sentences



https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

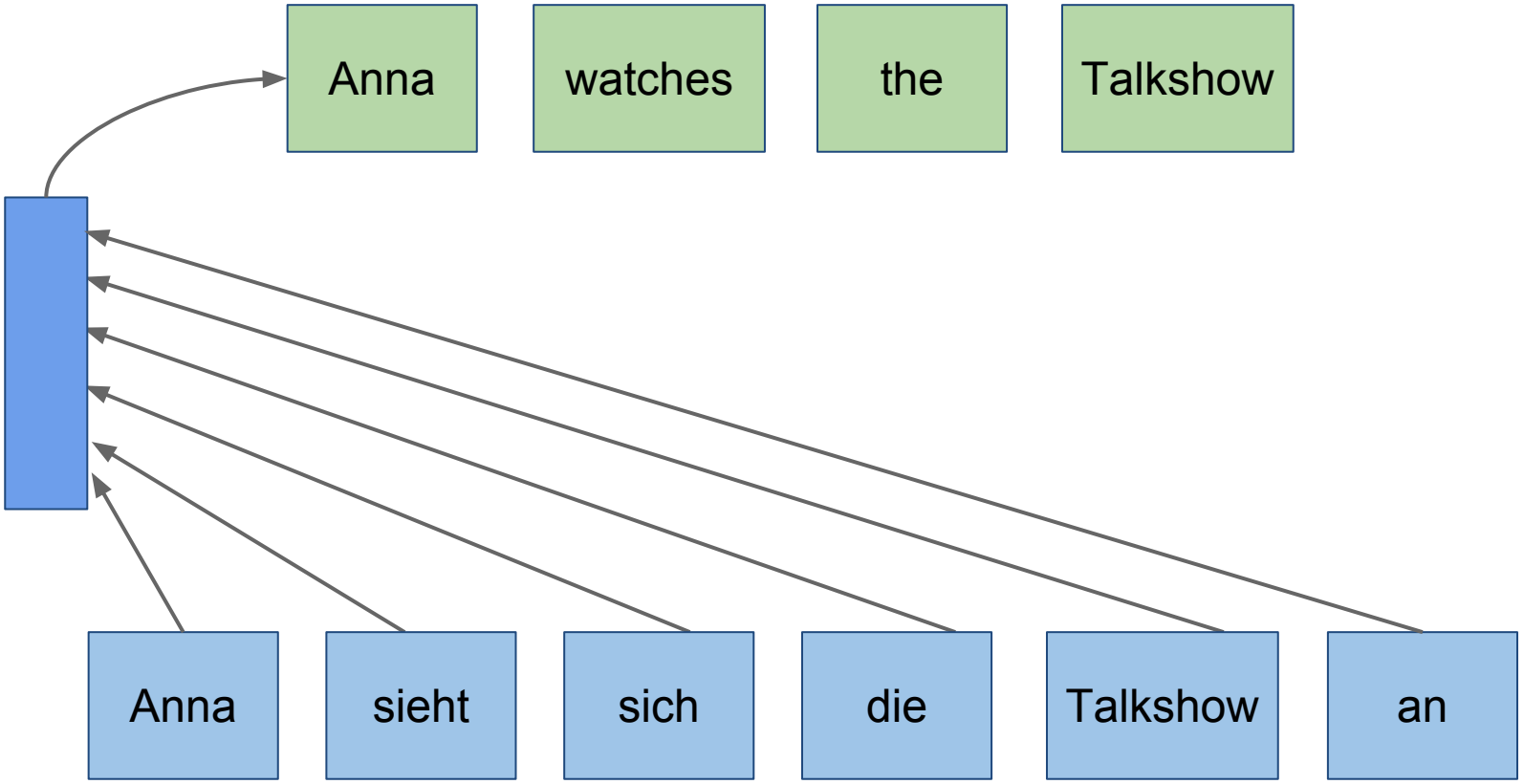
Attention Mechanism

One vector represents all source words



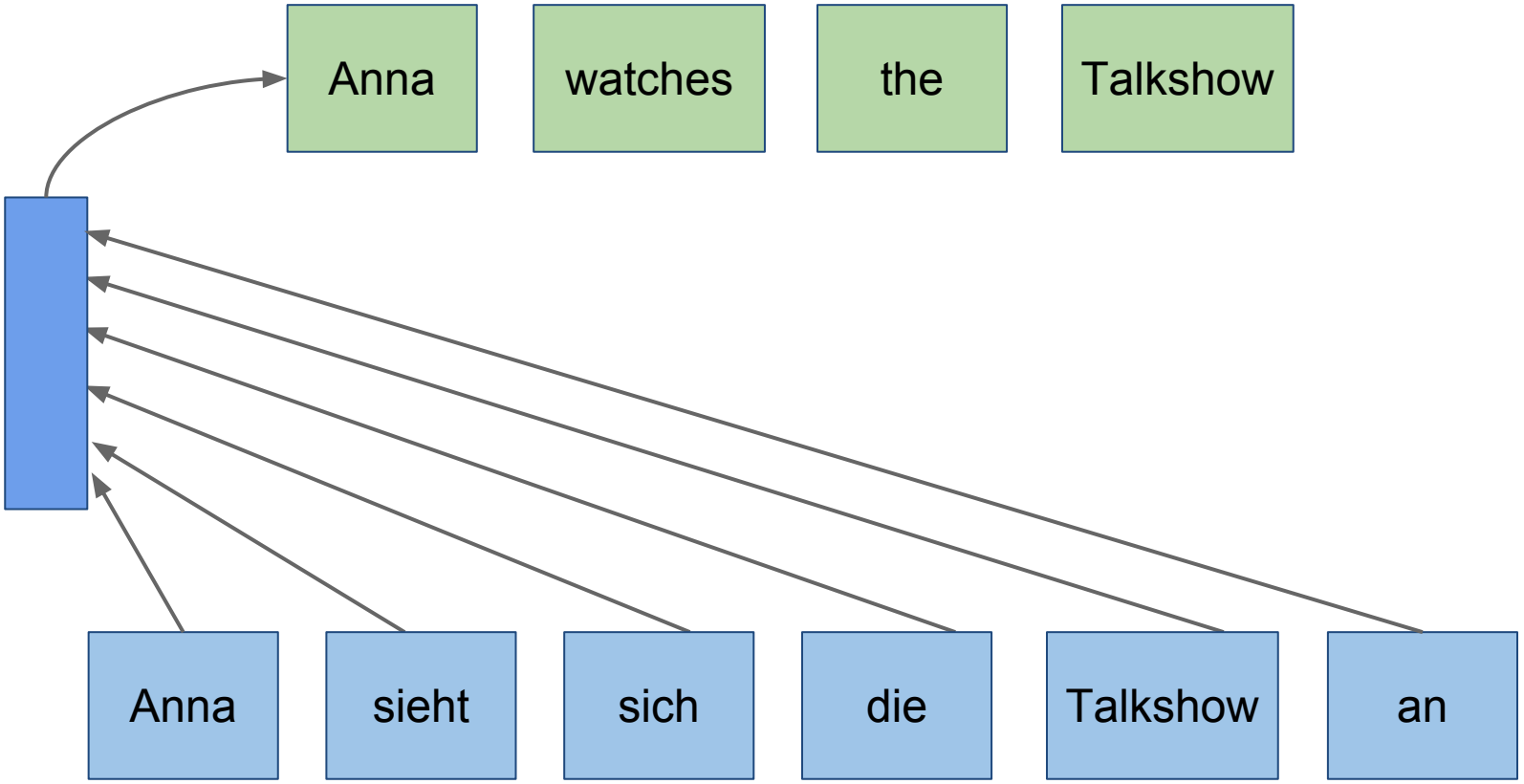
Attention Mechanism

Source and target words inherently have relationships



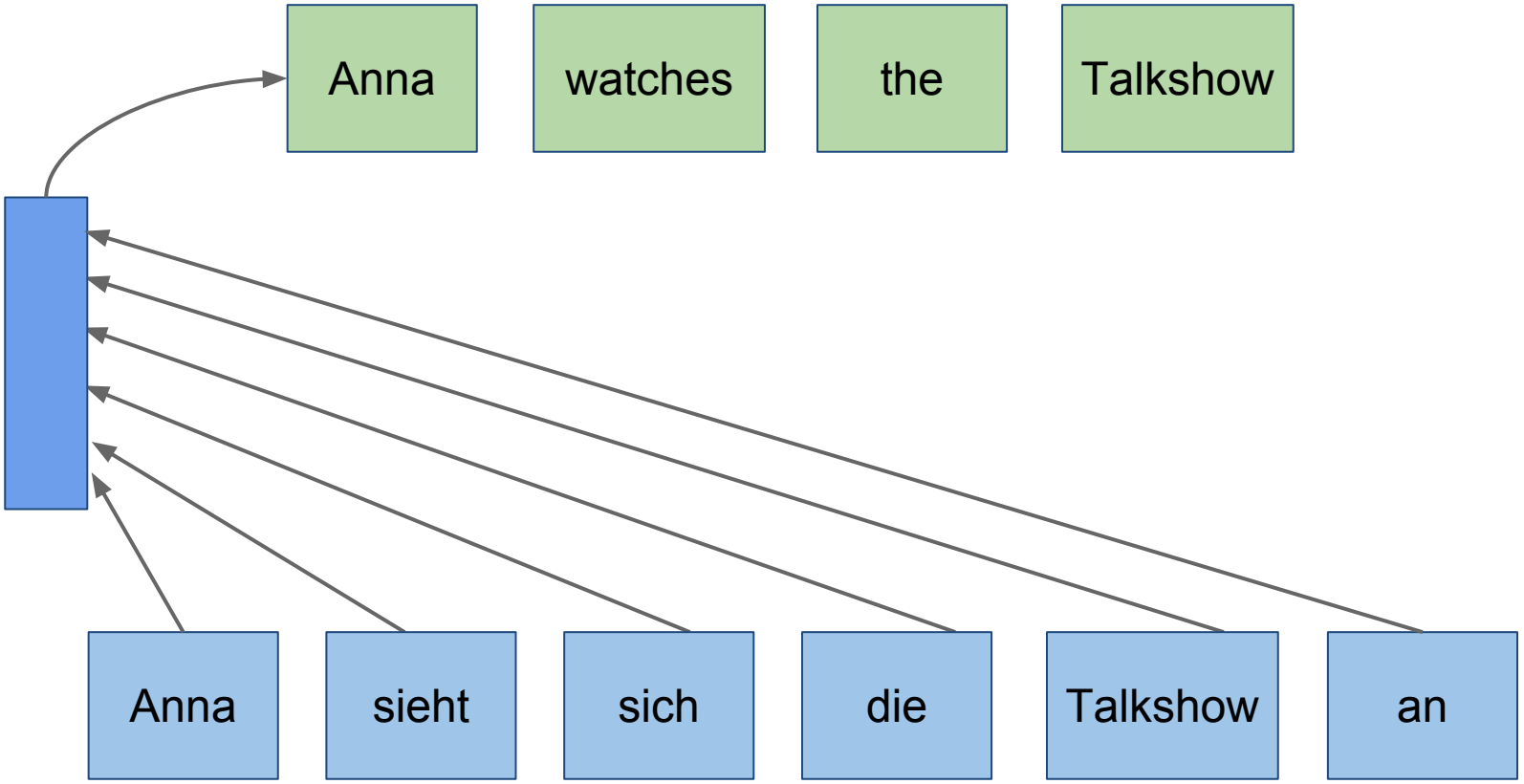
Attention Mechanism

Anna,Anna are more relevant to each other than Anna,Talkshow



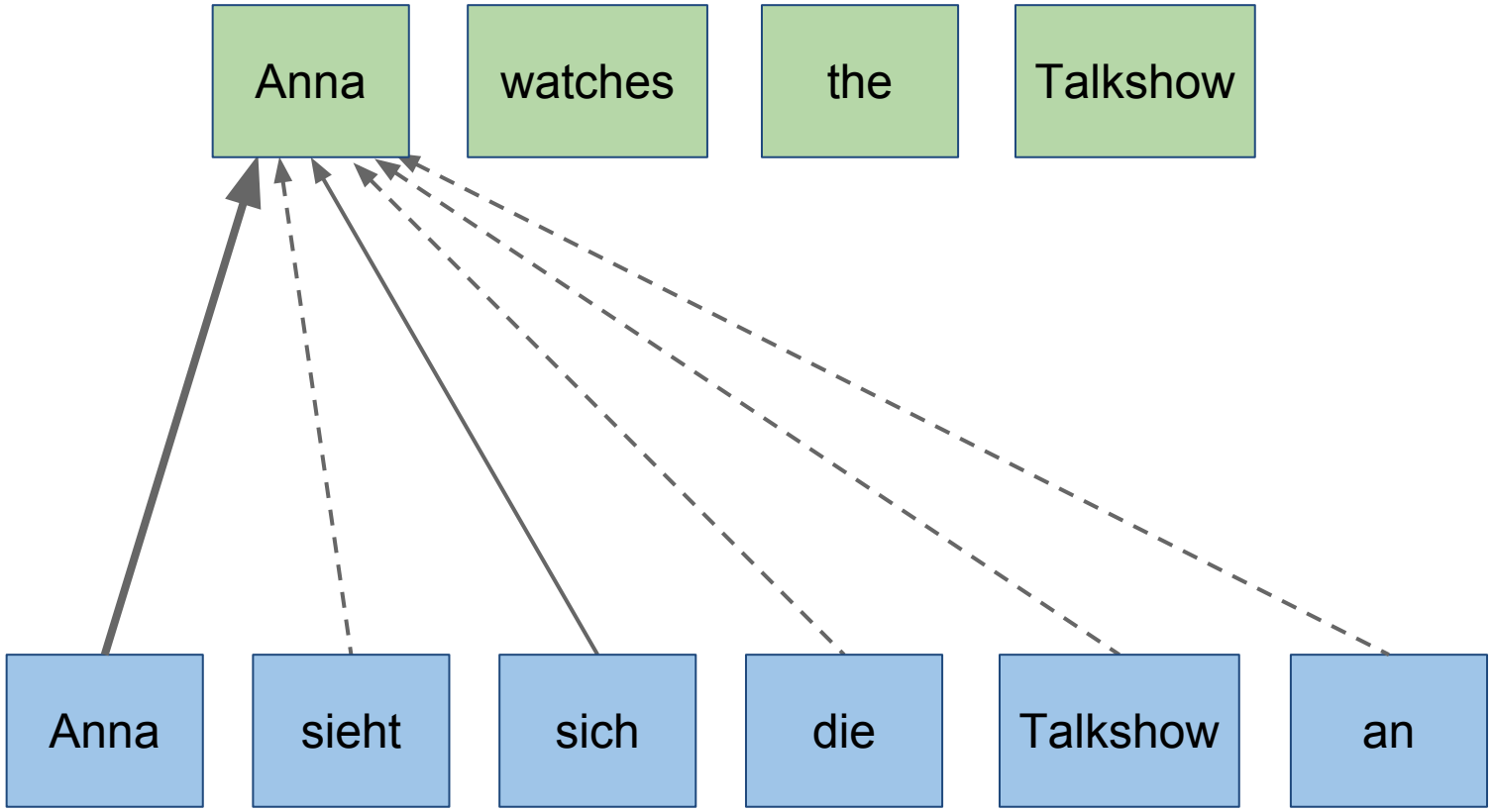
Attention Mechanism

Solution: For each target word, concentrate only on specific source words



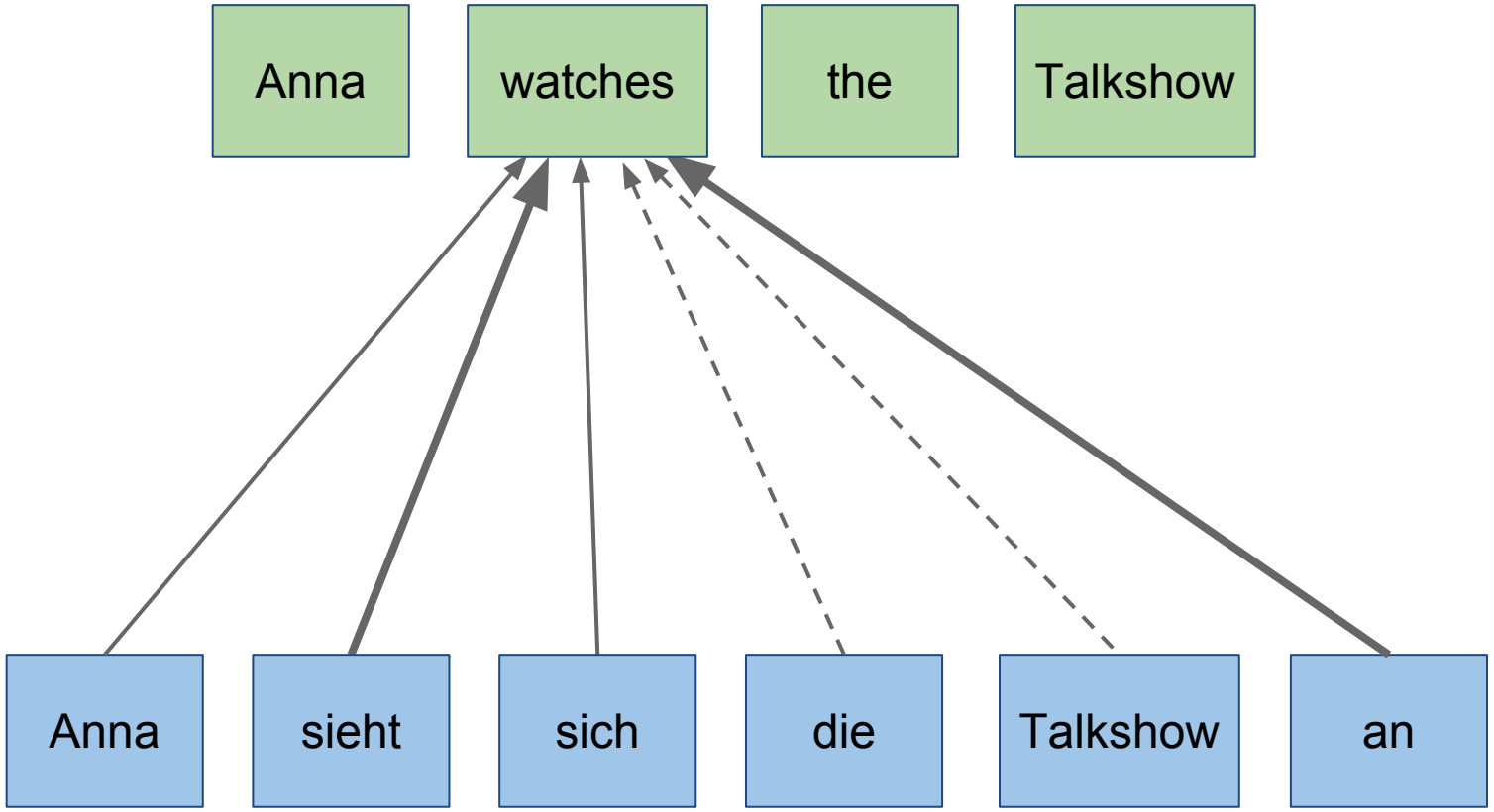
Attention Mechanism

Solution: For each target word, concentrate only on specific source words



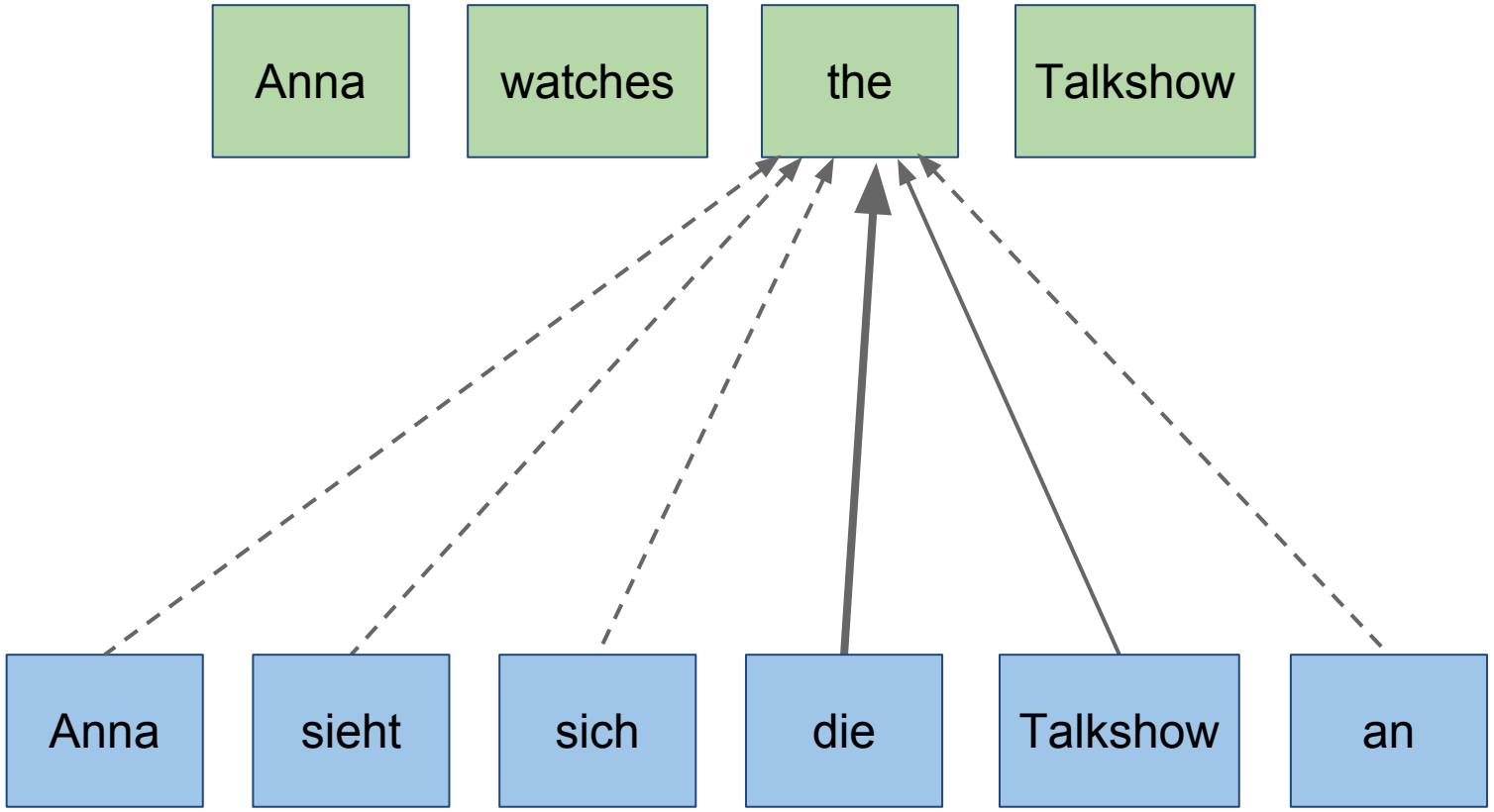
Attention Mechanism

Solution: For each target word, concentrate only on specific source words



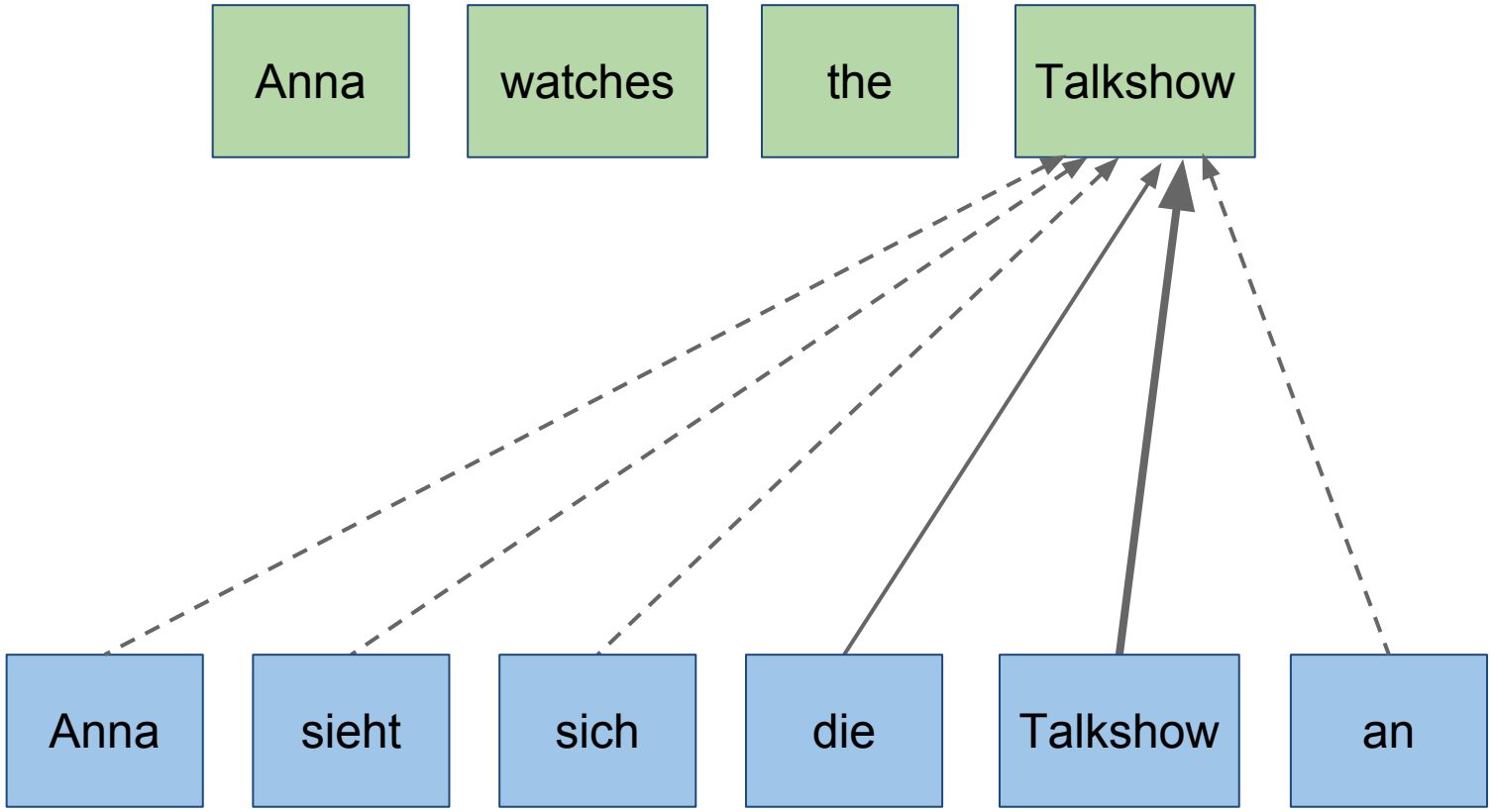
Attention Mechanism

Solution: For each target word, concentrate only on specific source words



Attention Mechanism

Solution: For each target word, concentrate only on specific source words



Attention Mechanism

- **Issue**: a sentence is represented as a fixed vector
- Relationship between source and target words is very abstract
- All words are not equally important to predict a target word
- Long distance dependencies are not captured well
  - I thought LSTM can do it?

Attention Mechanism

- **Issue**: a sentence is represented as a fixed vector
- Relationship between source and target words is very abstract
- All words are not equally important to predict a target word
- Long distance dependencies are not captured well
  - I thought LSTM can do it? Practically, it's still working with a fixed sized vector, so the problem surfaces again with longer distance dependencies

Attention Mechanism

A few hacks

- Feed the sentence backwards
- Repeat the input sentence a couple of times to improve memorization (ref)

Attention Mechanism

A few hacks

- Feed the sentence backwards
- Repeat the input sentence a couple of times to improve memorization (ref)

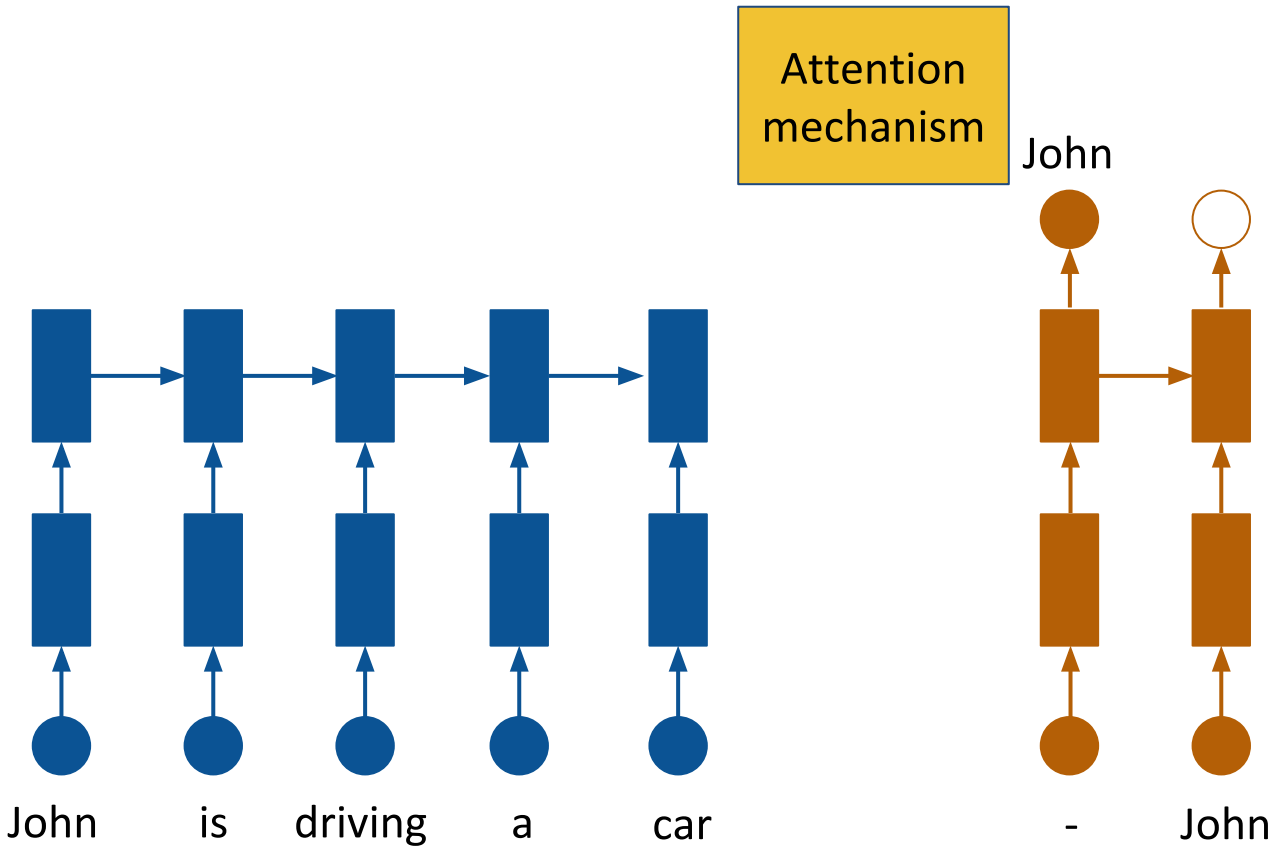
A modeling solution

- Let the model learn dependency between source and target words

Attention Mechanism

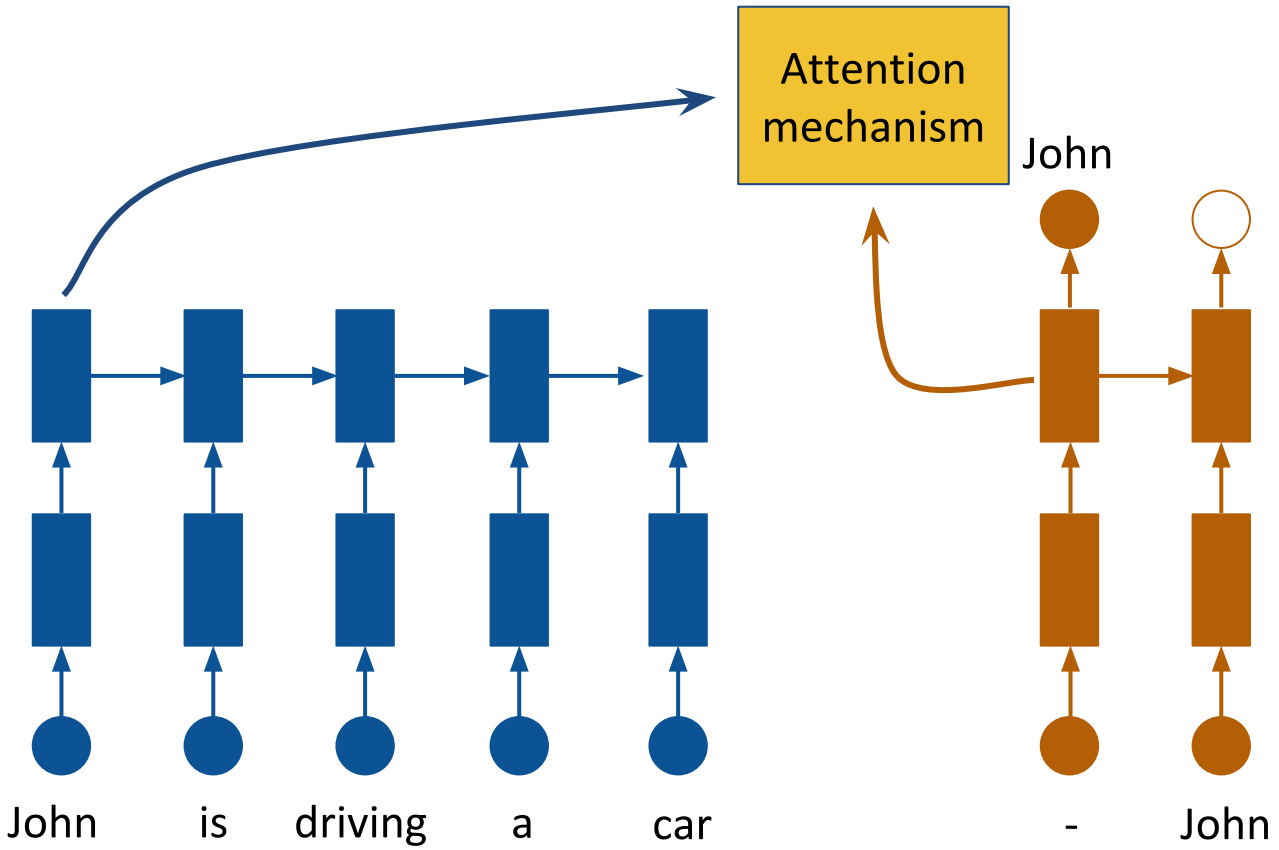
- Instead of using only the last state while predicting a target word **consider all input states**

Attention Mechanism



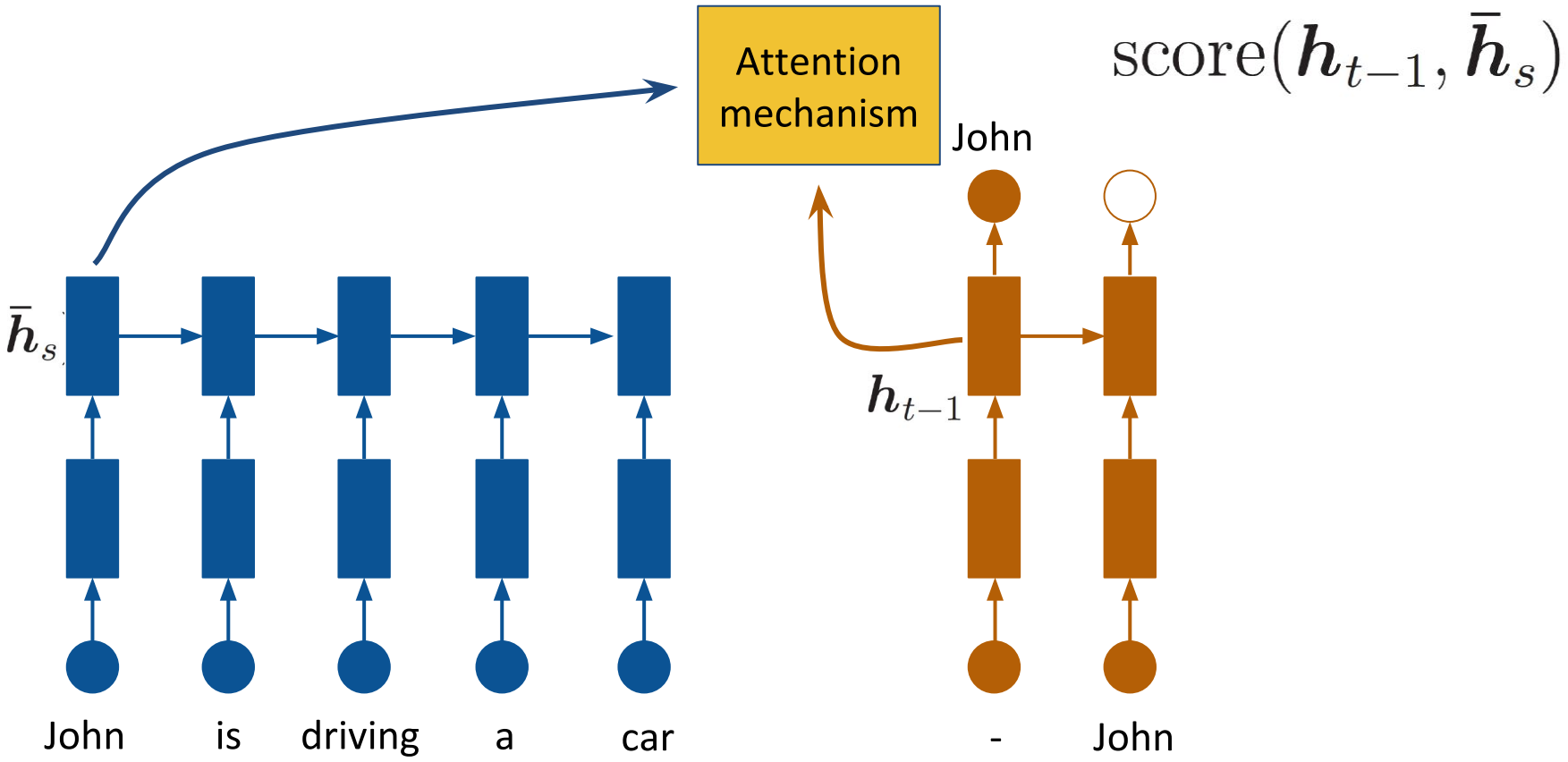
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



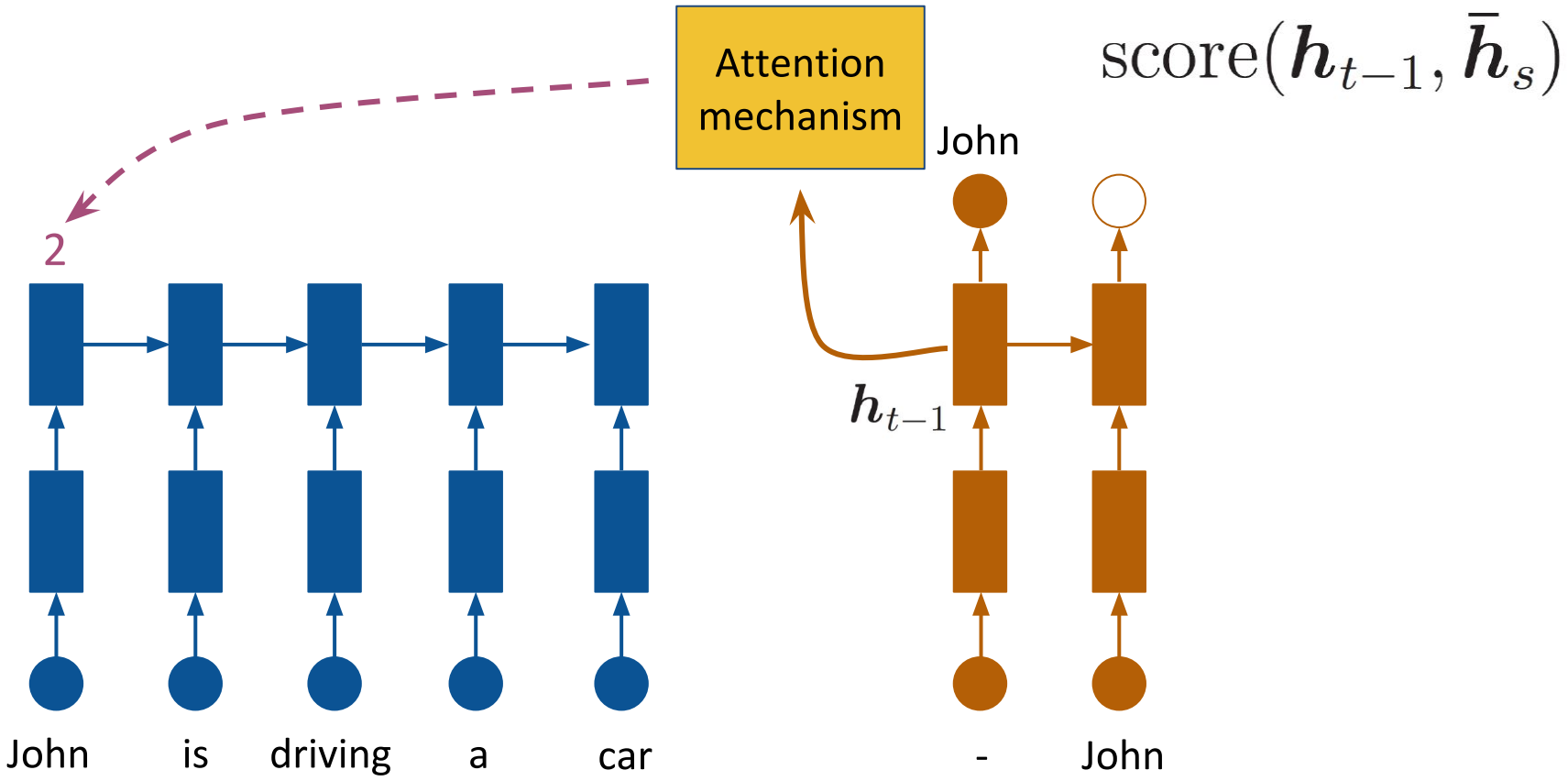
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



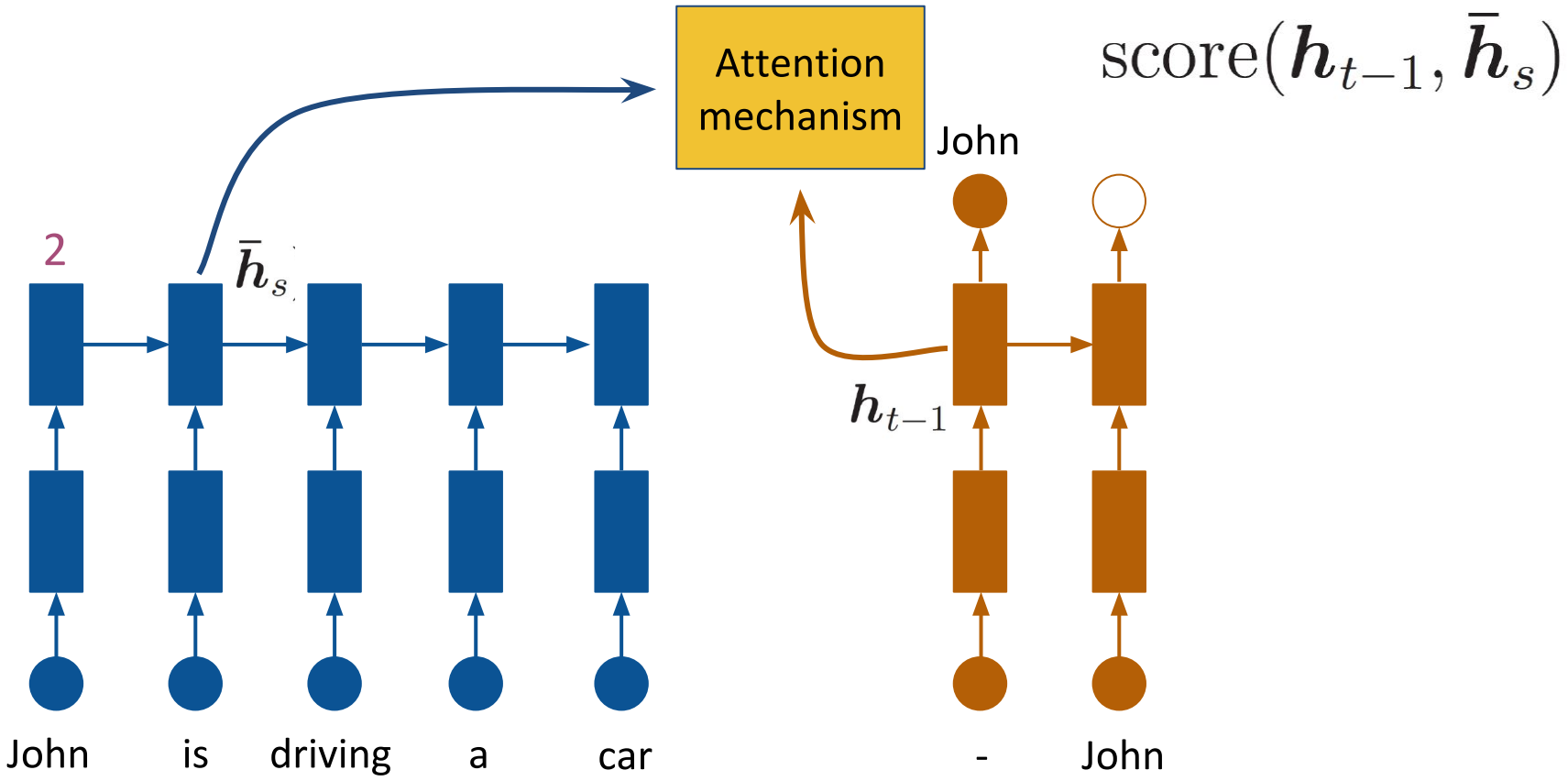
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



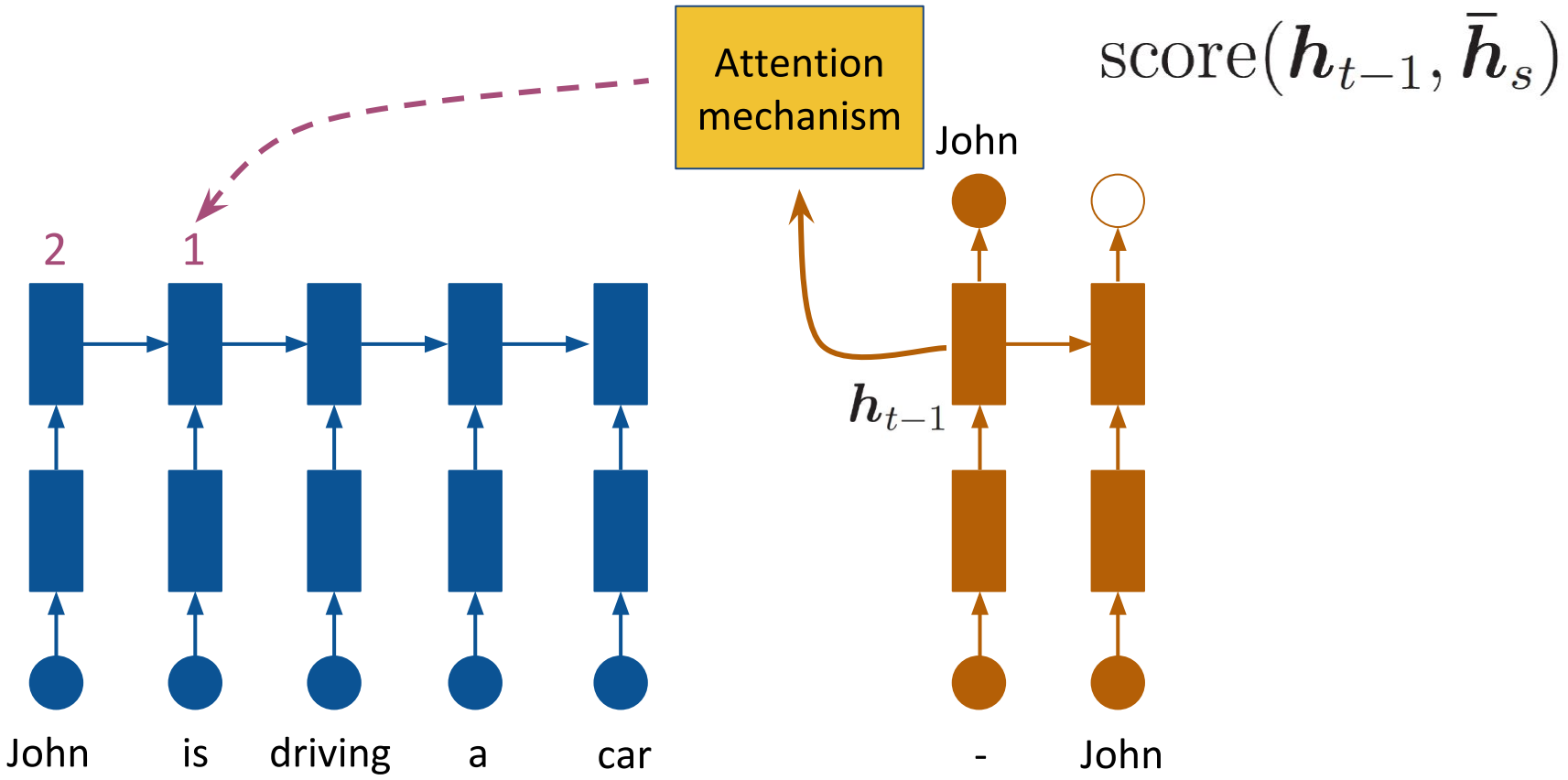
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



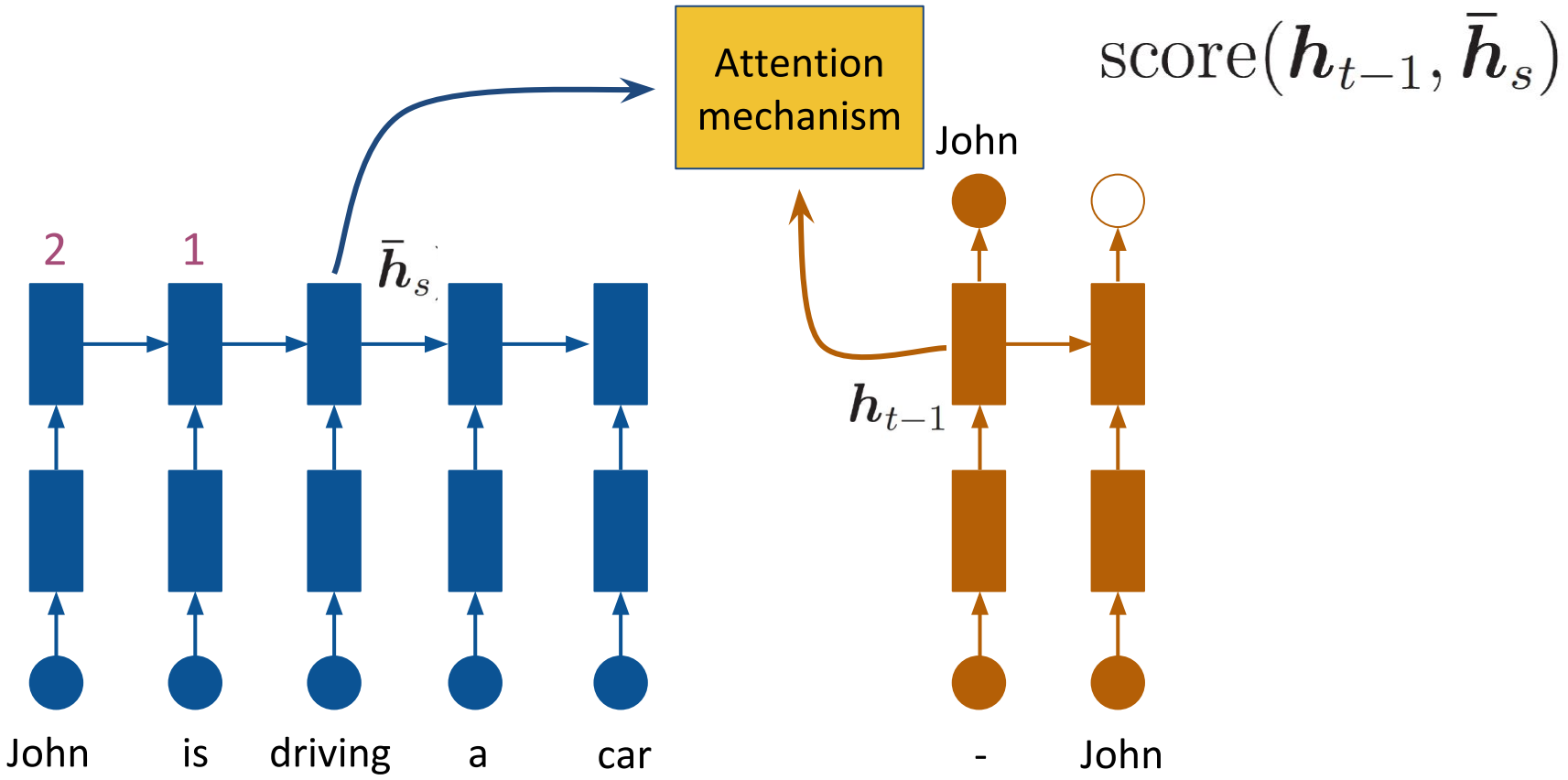
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



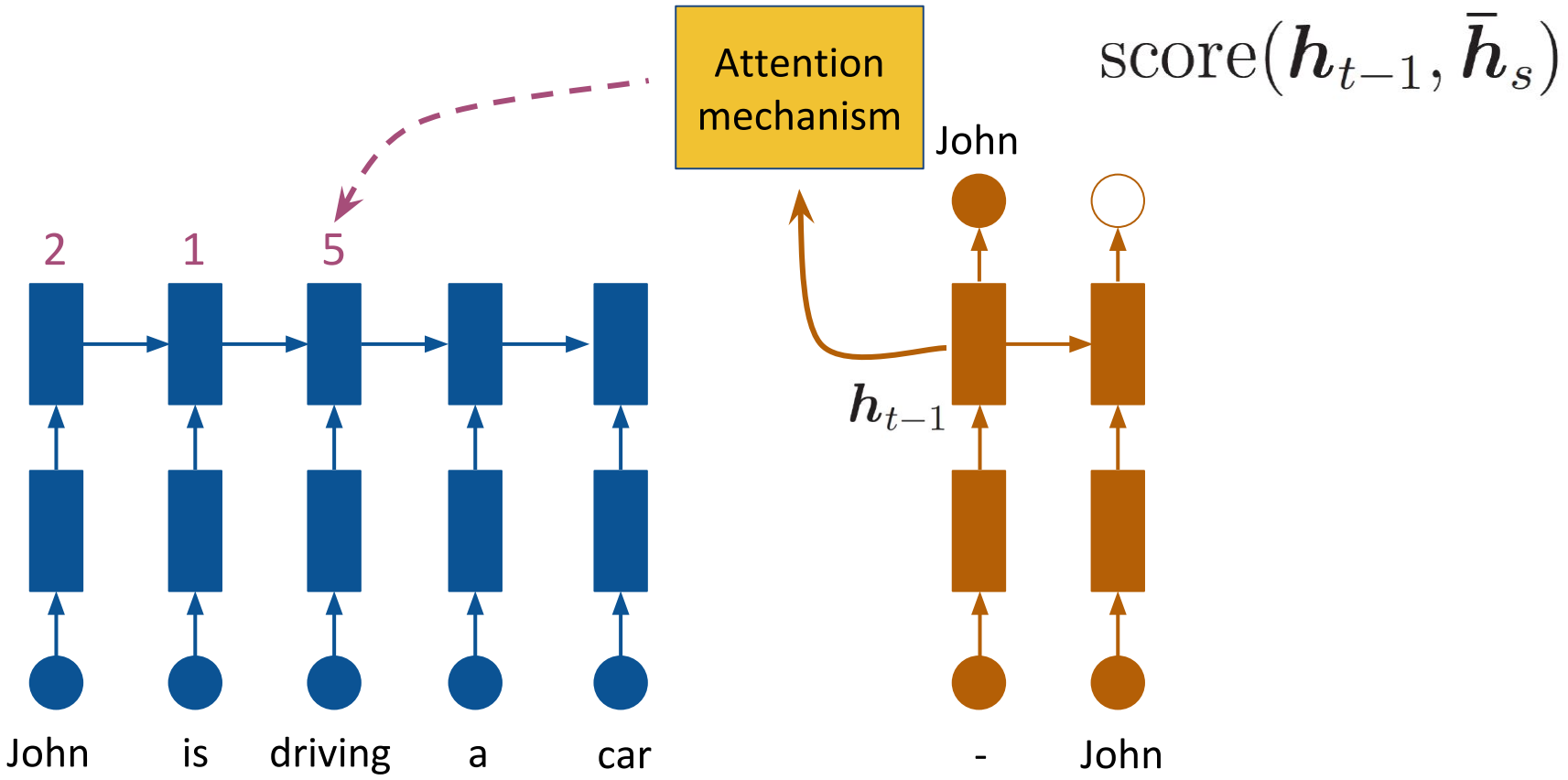
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



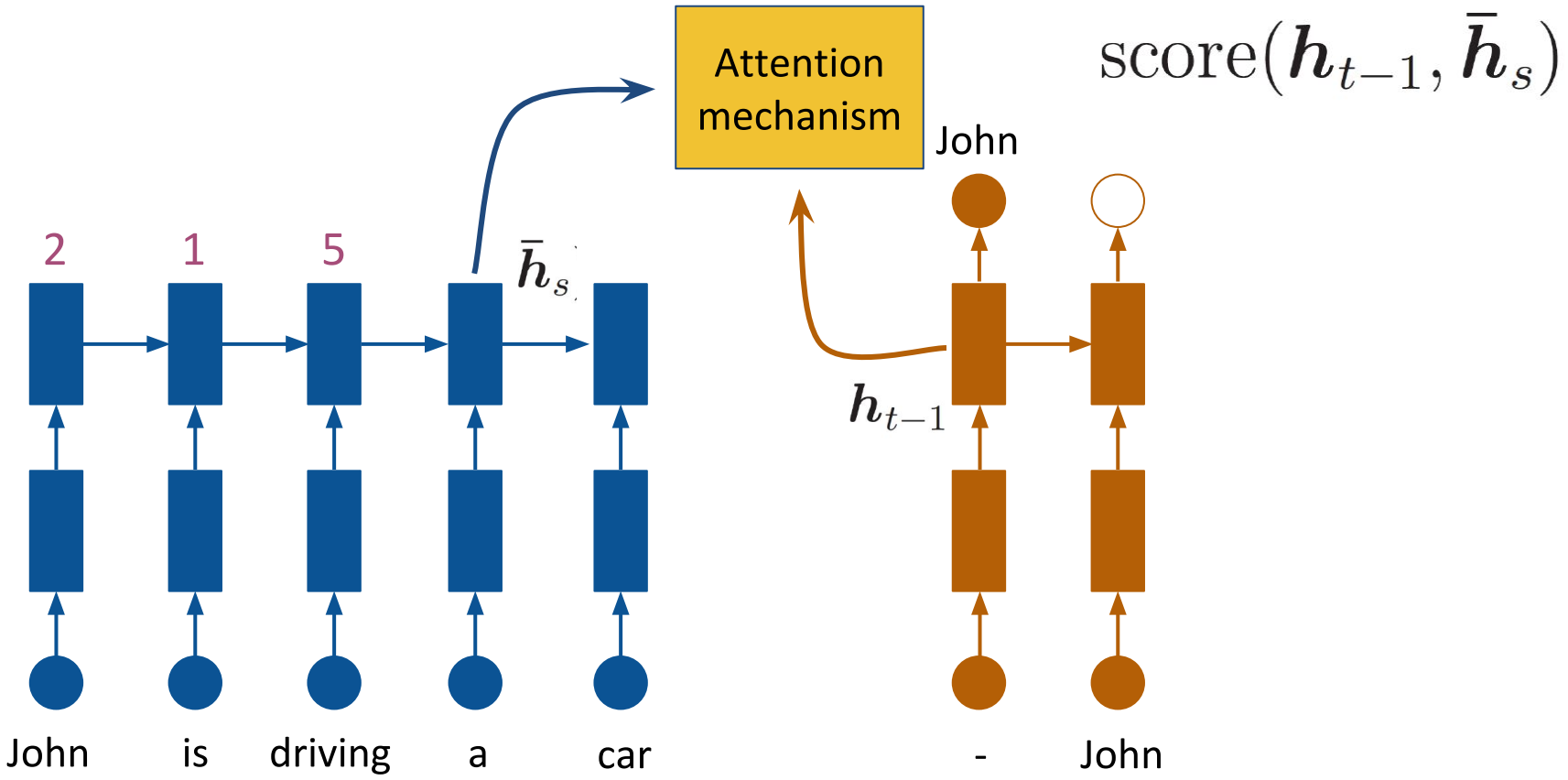
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



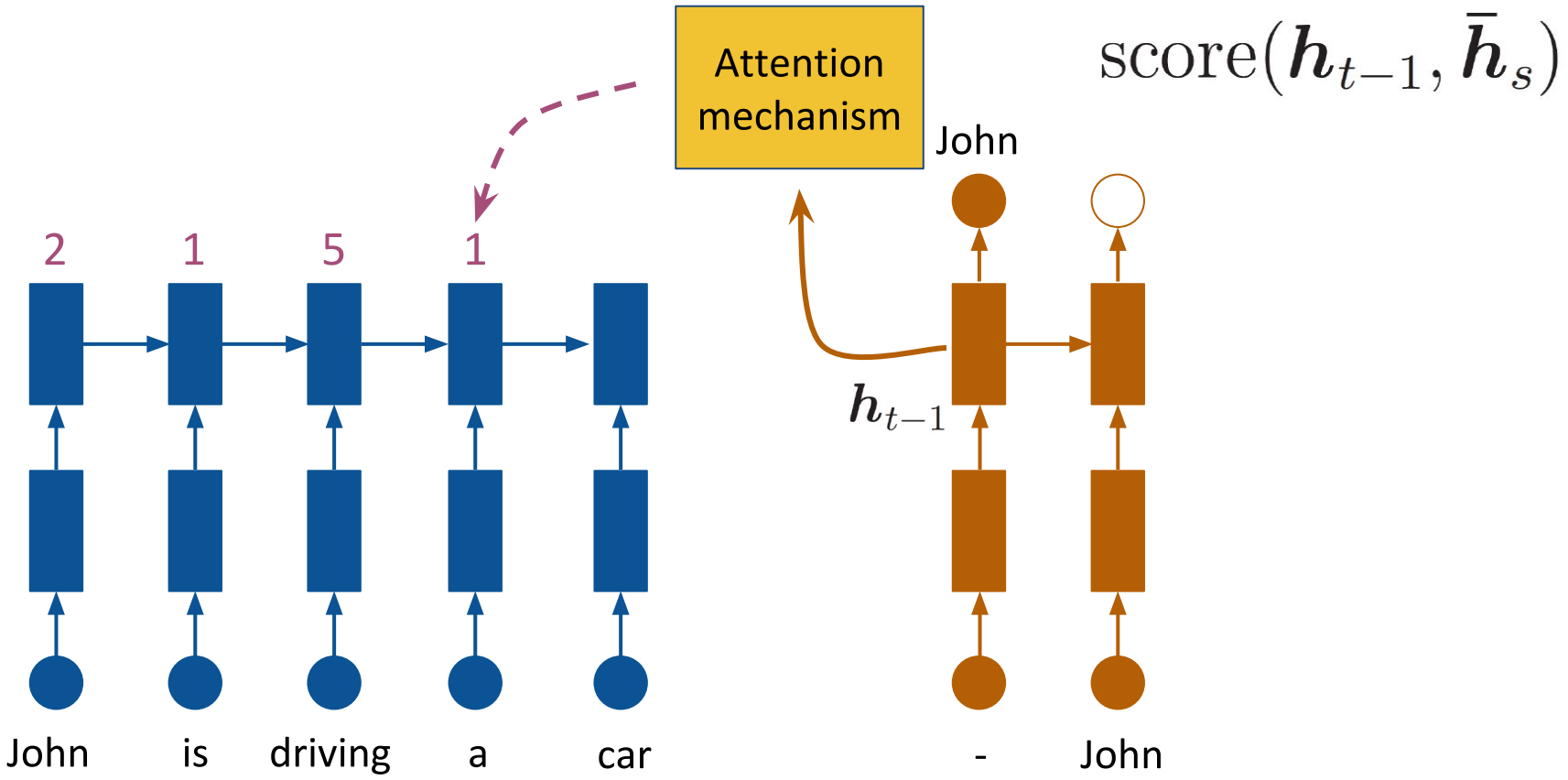
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



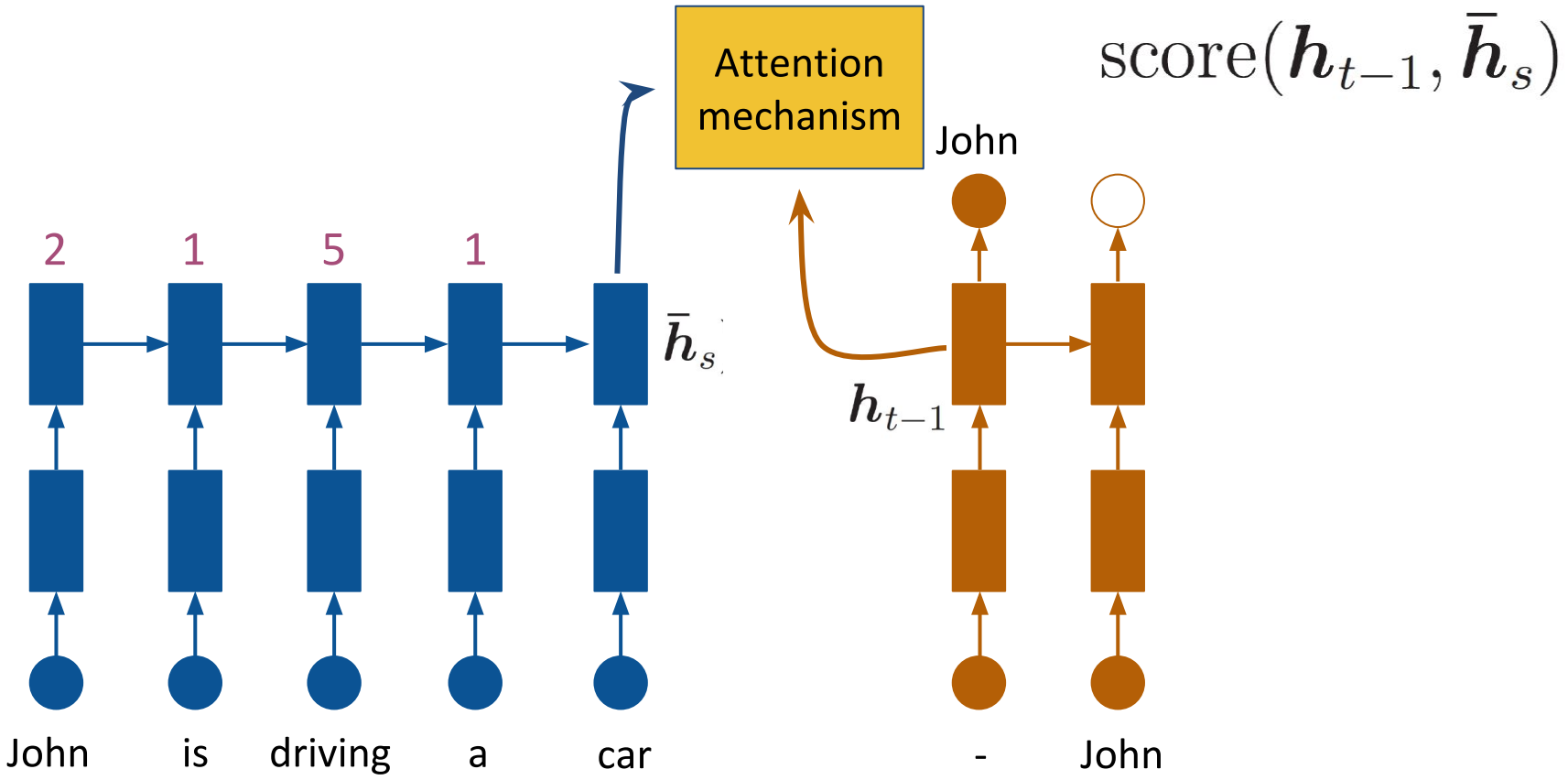
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



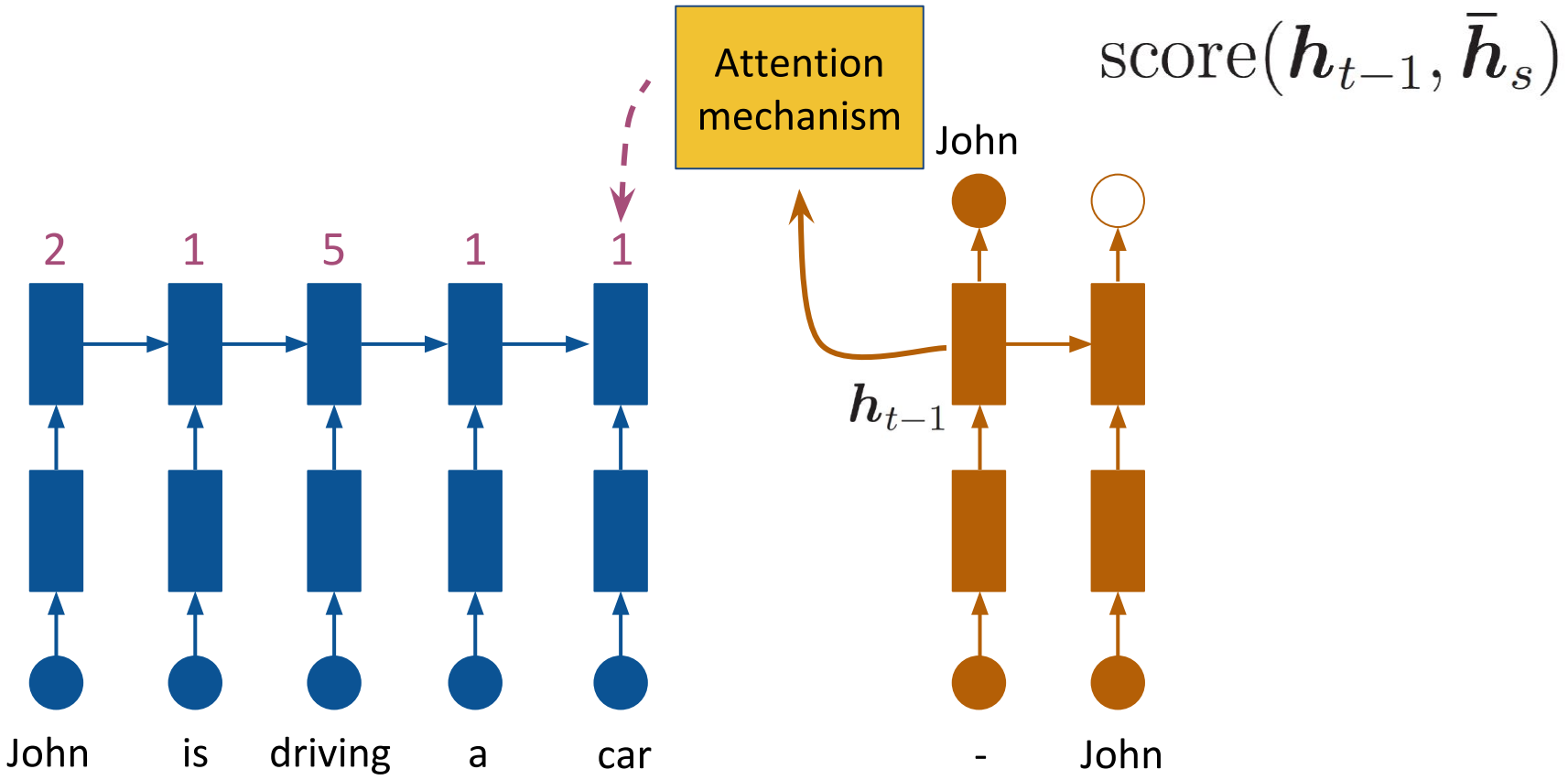
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



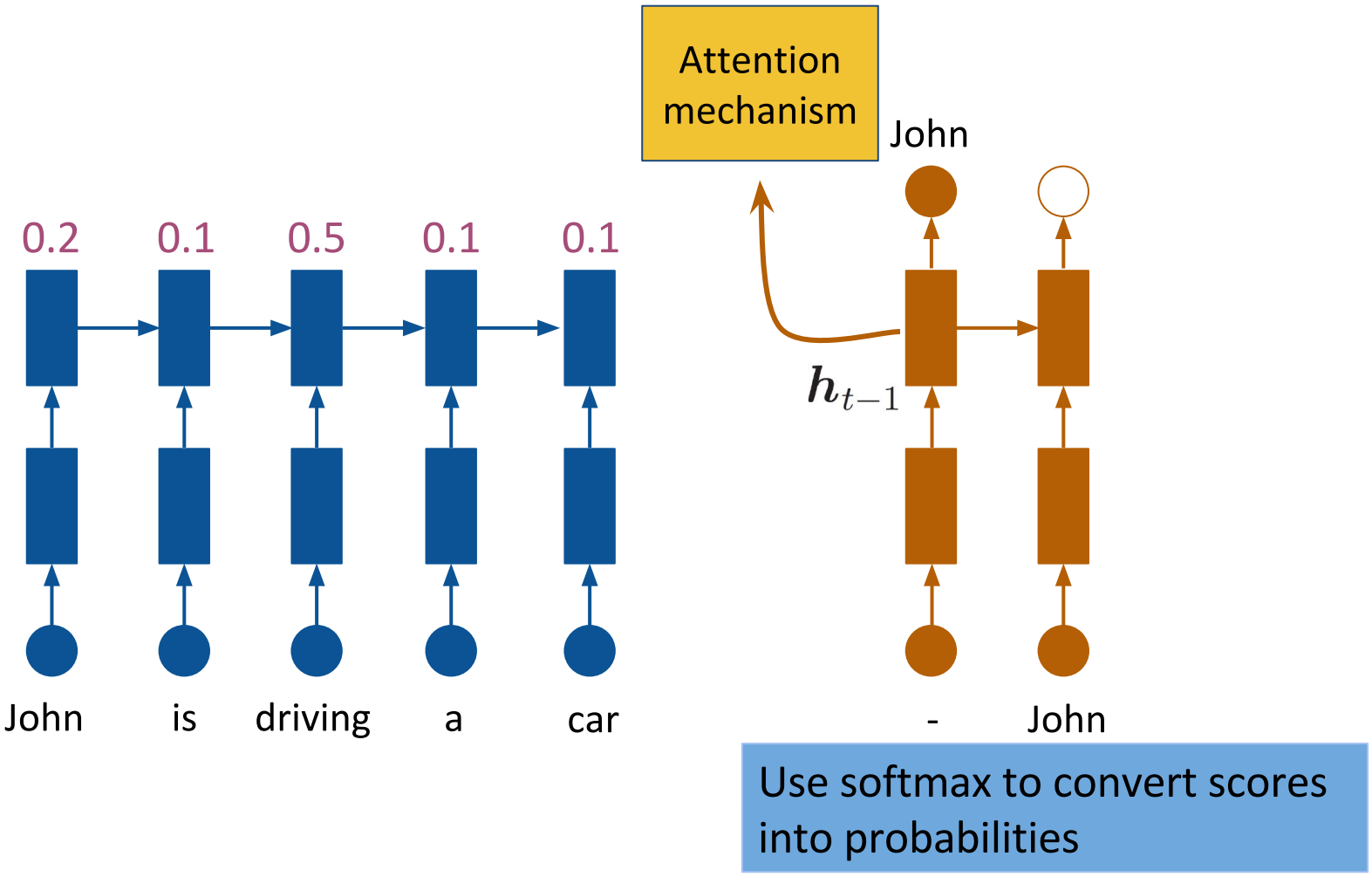
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



Neural Machine Translation by Luong, Cho and Manning

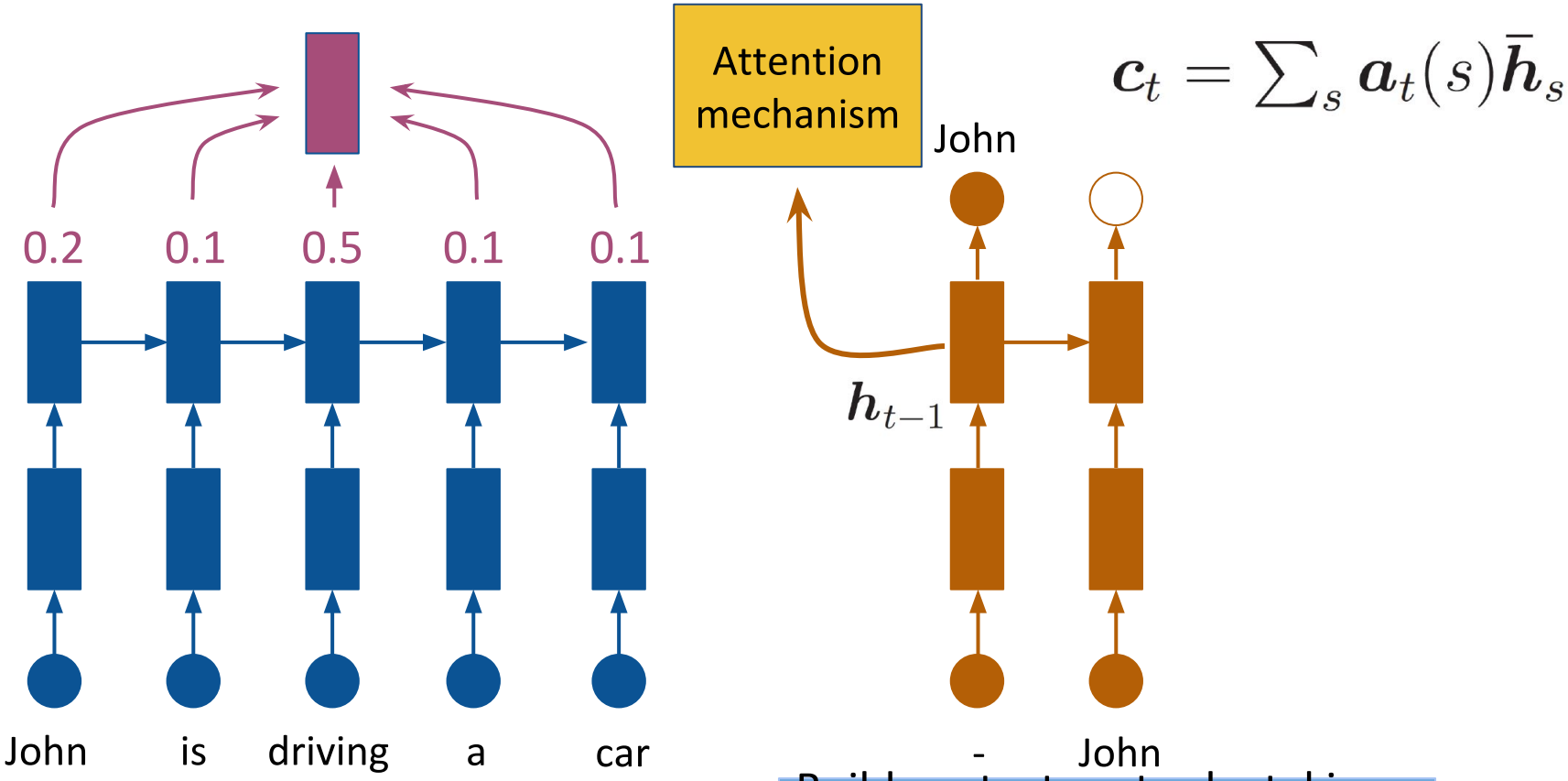
Attention Mechanism



Use softmax to convert scores into probabilities

Neural Machine Translation by Luong, Cho and Manning

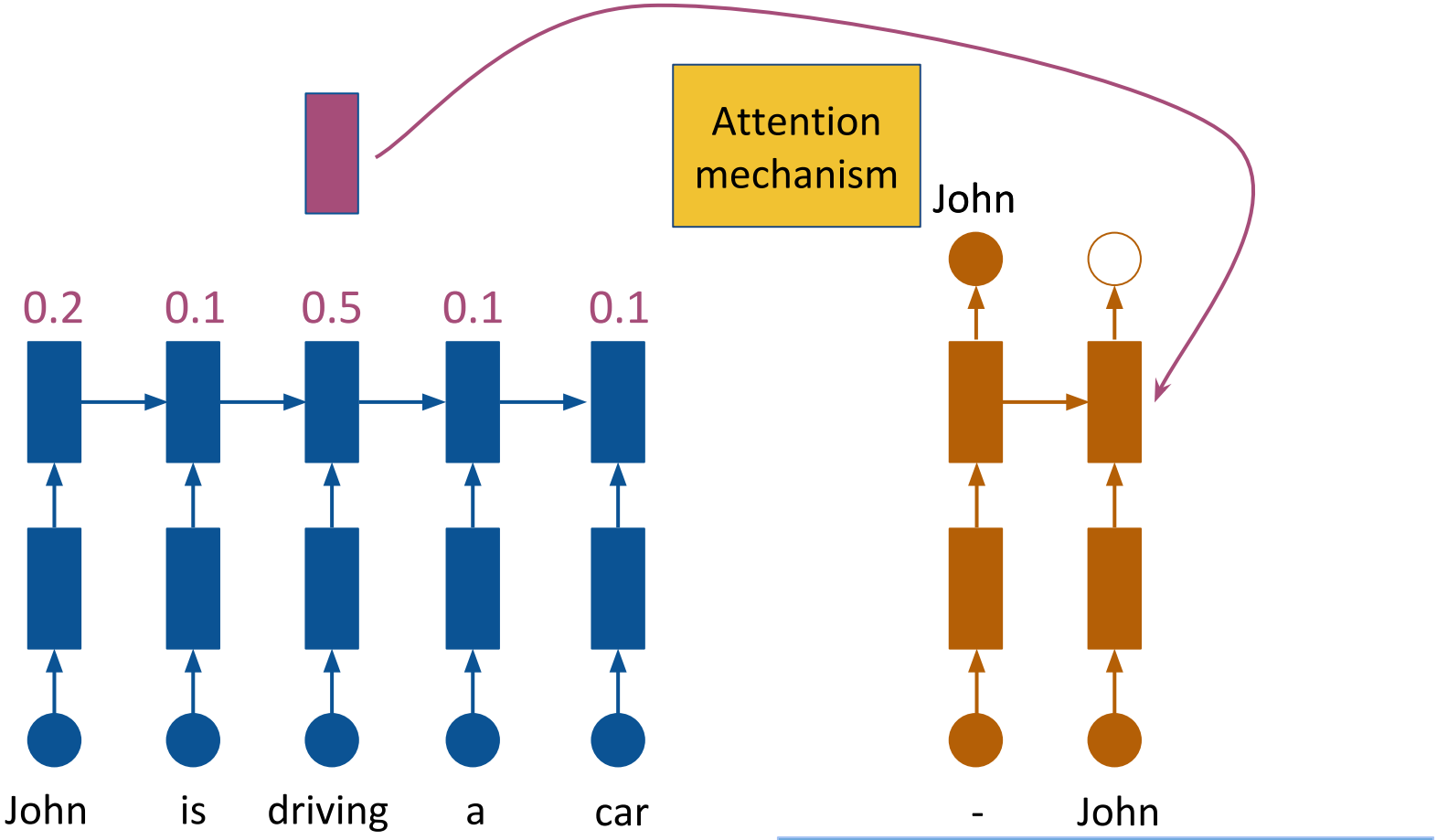
Attention Mechanism



Build context vector by taking a **weighted sum** over the source

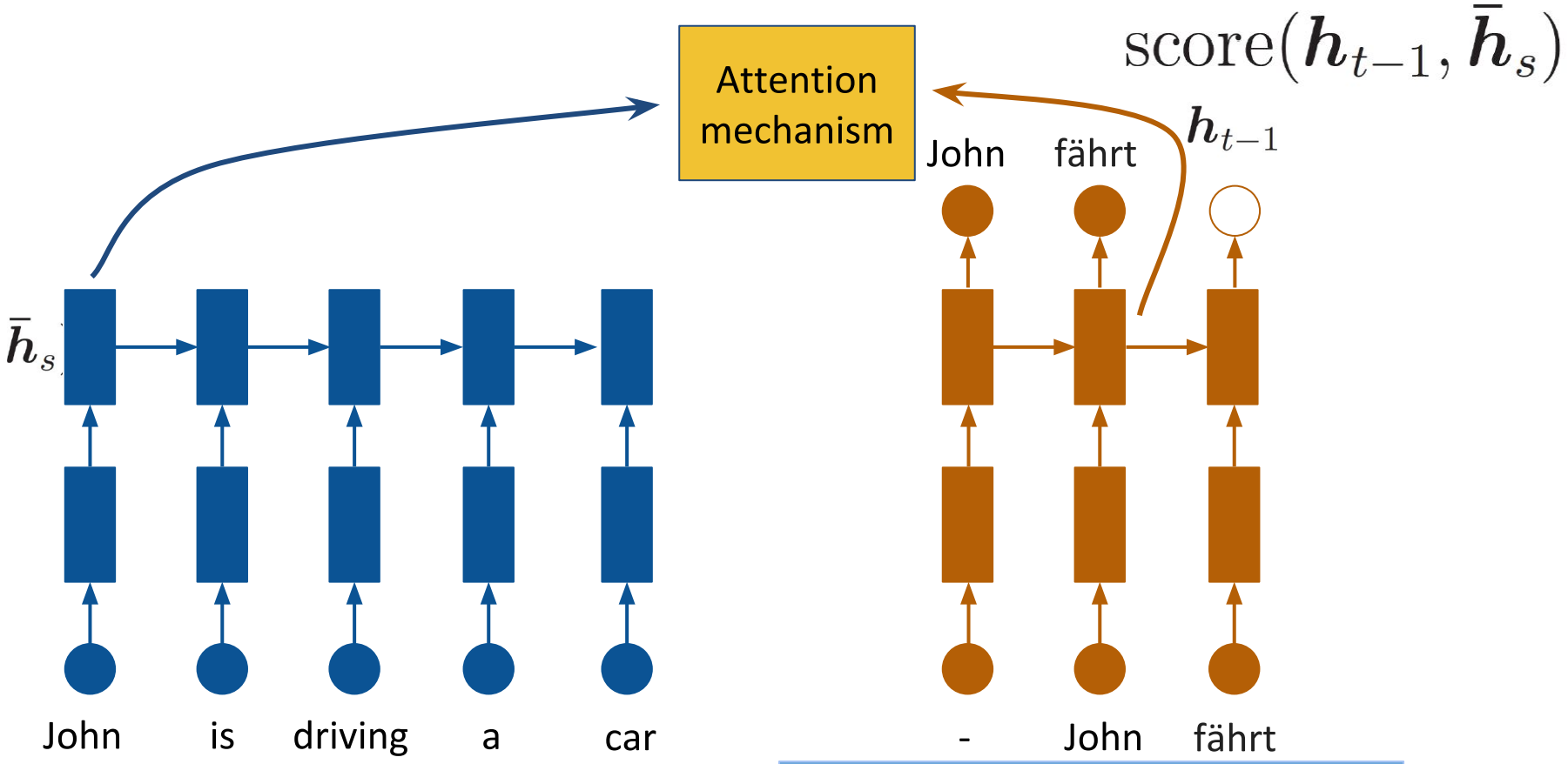
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



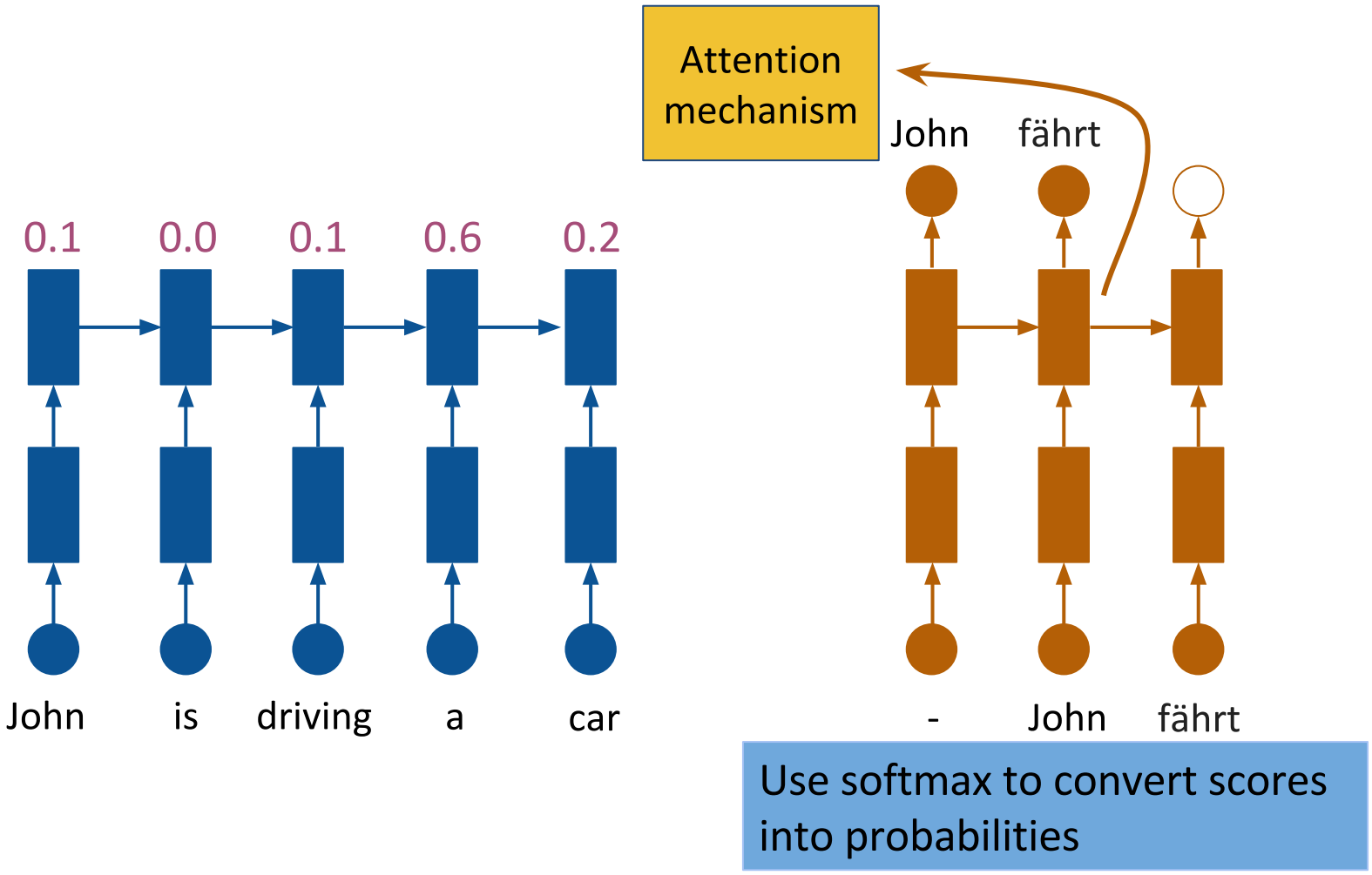
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



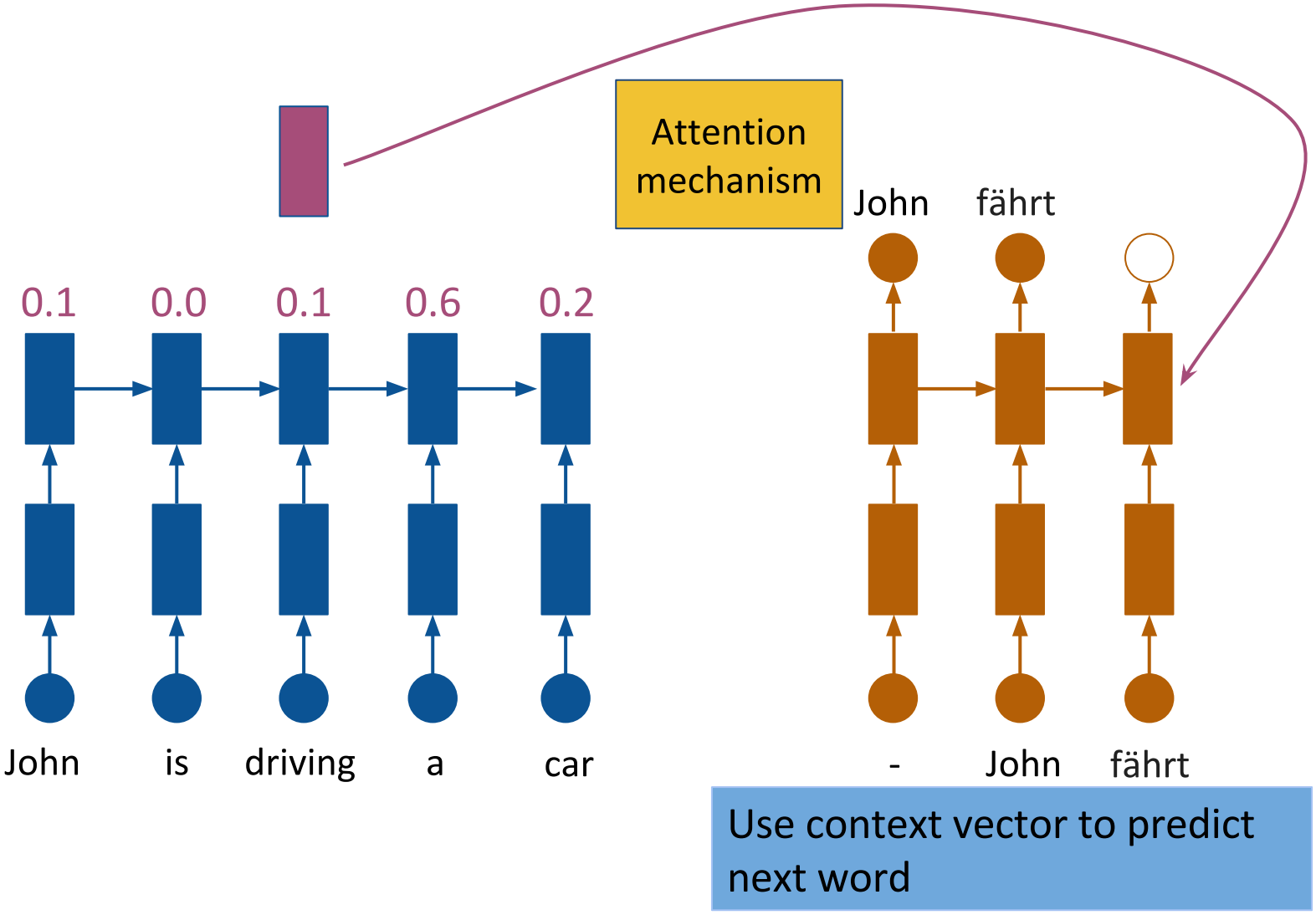
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



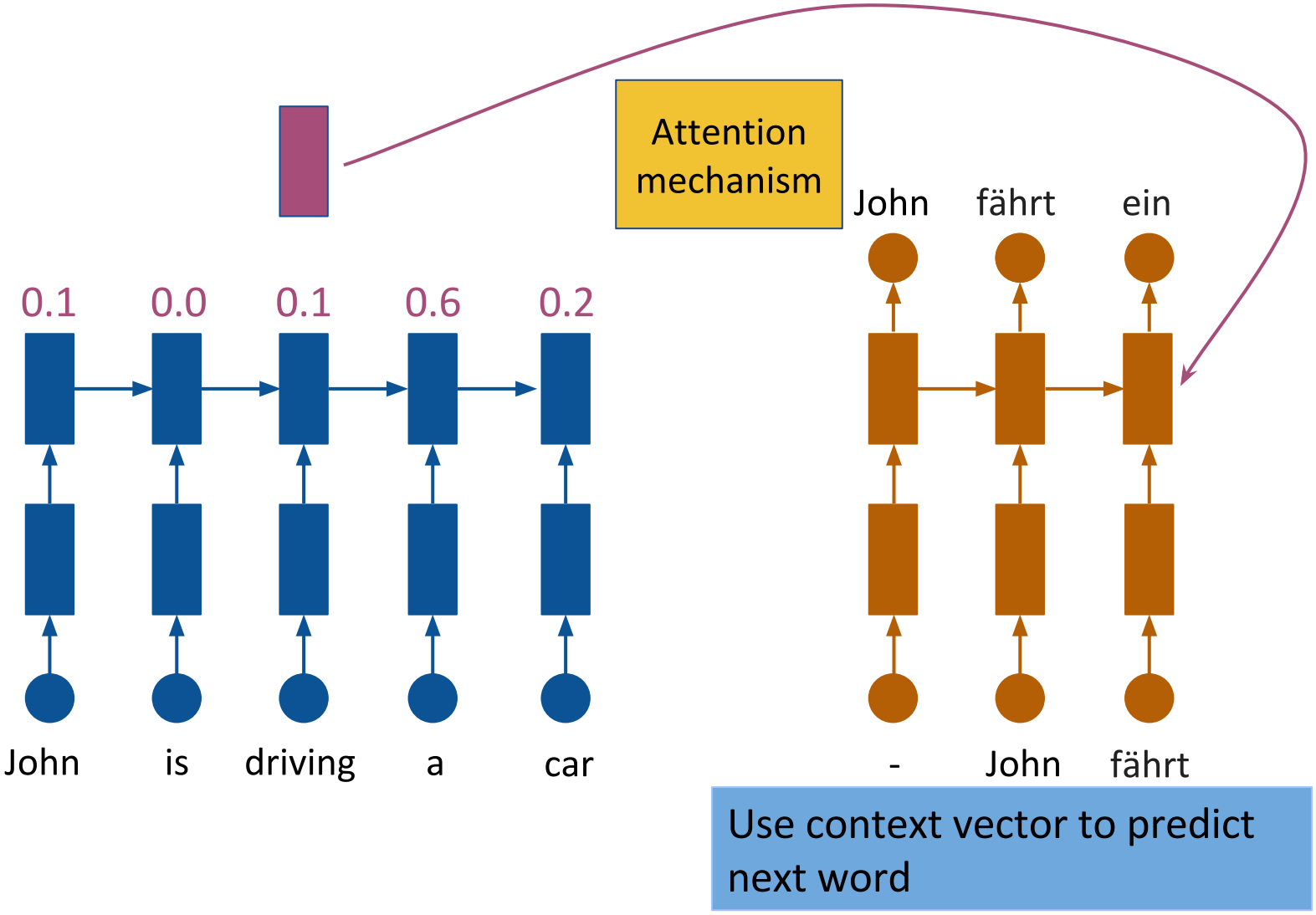
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



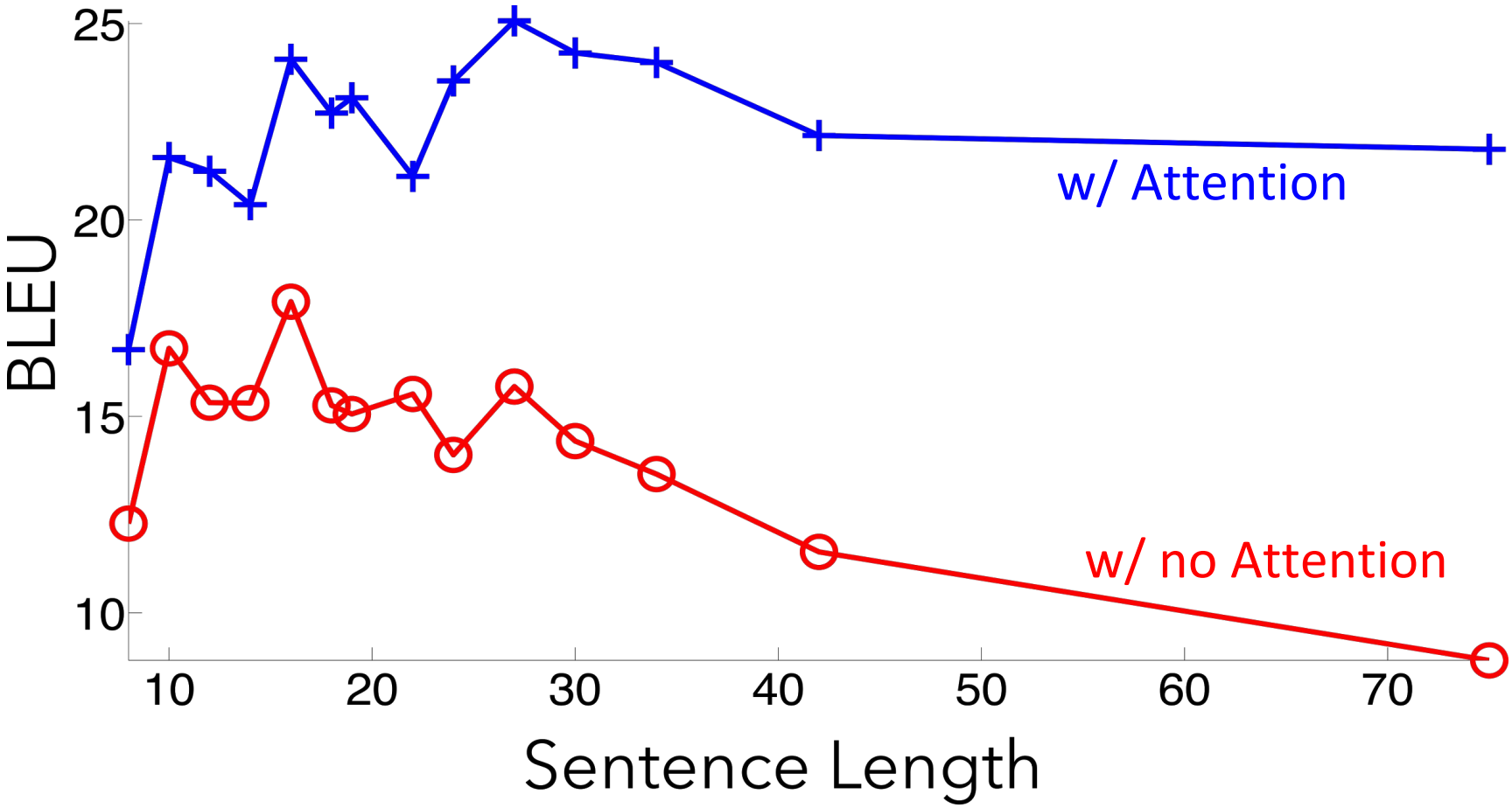
Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism



Neural Machine Translation by Luong, Cho and Manning

Empirical Evaluation of Attention

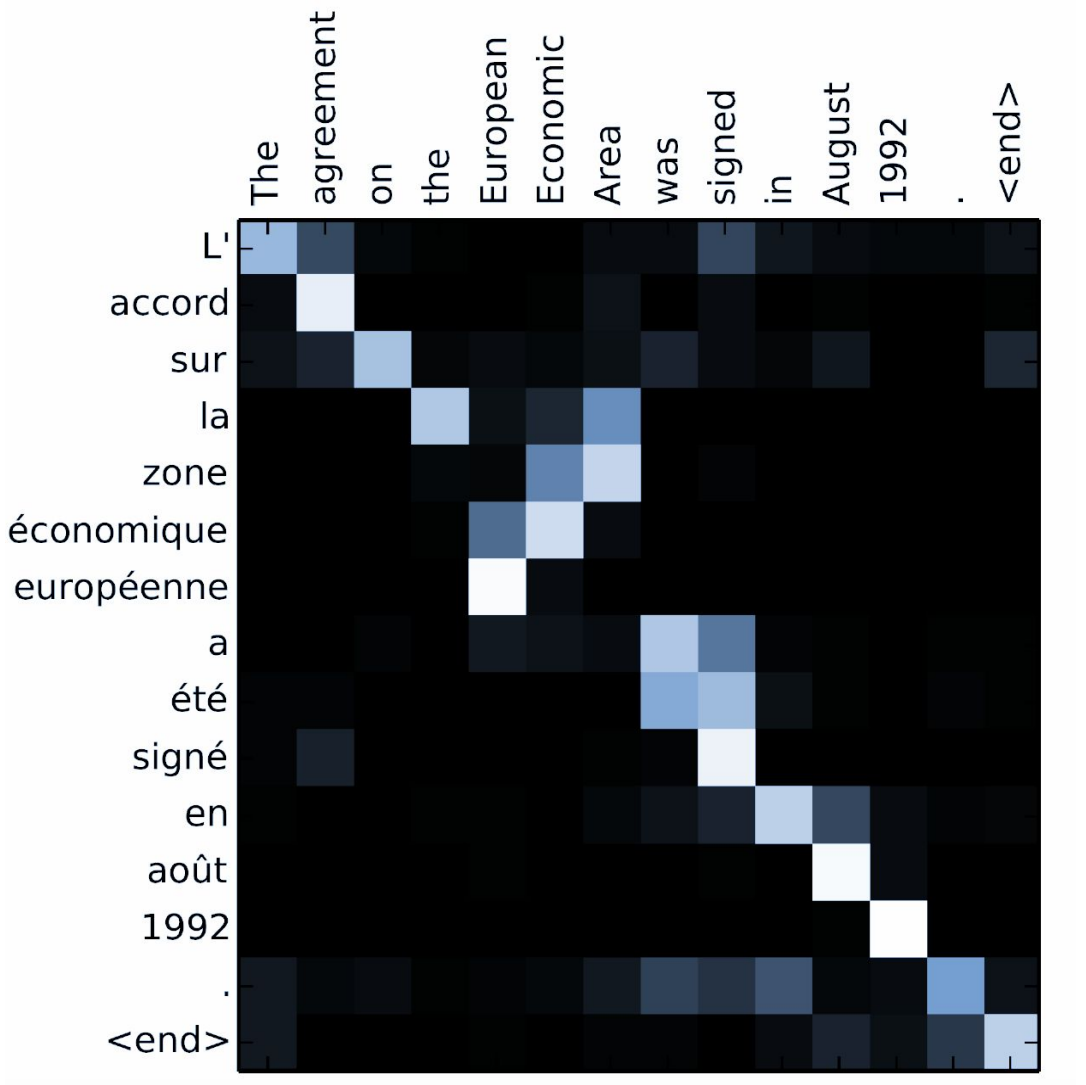


Neural Machine Translation by Luong, Cho and Manning

Attention Mechanism - Visual

Aligning words

If we plot the *source scores* for each target word, we can see what each target word is aligned to.



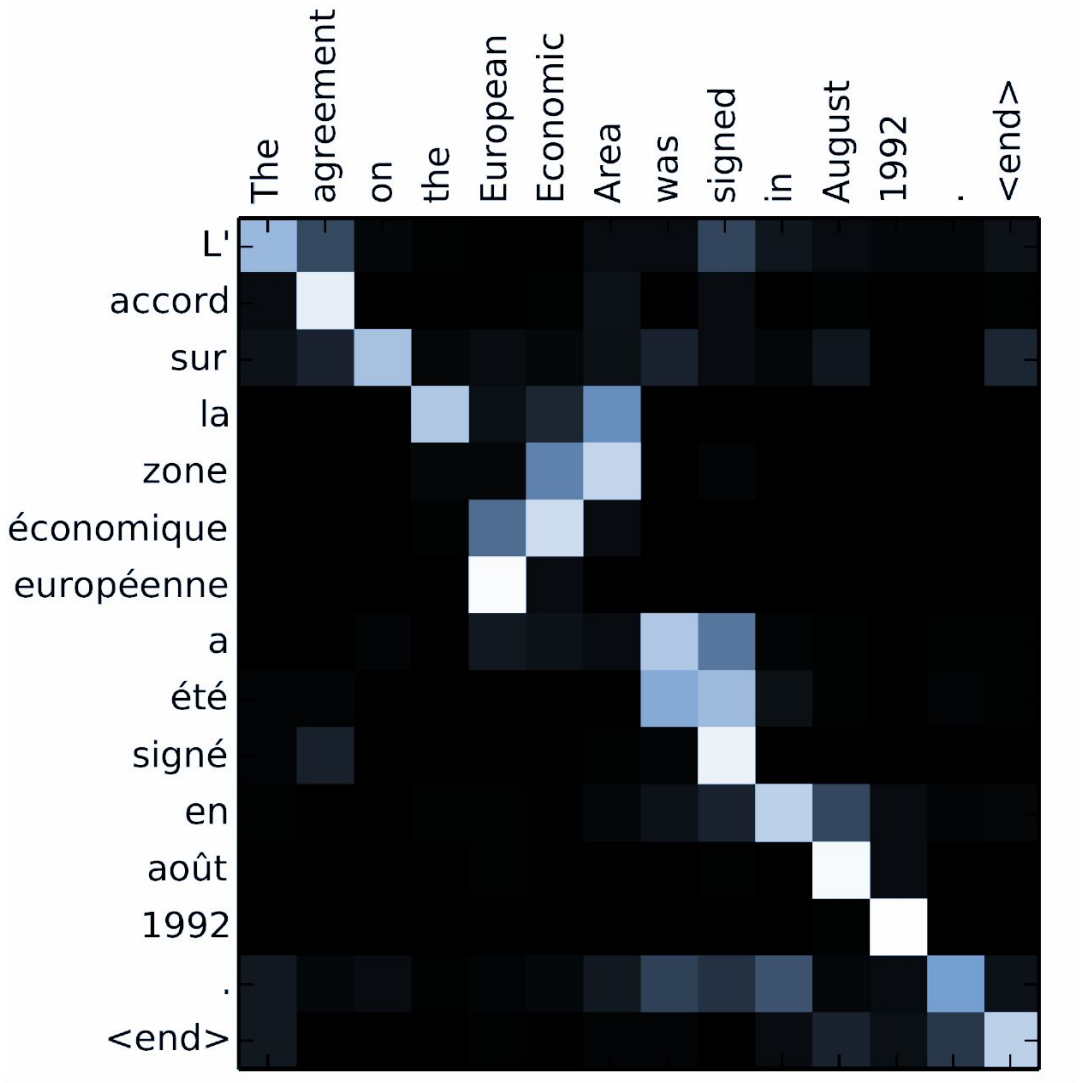
Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Translate and Align. ICLR'15

Attention Mechanism - Visual

Aligning words

If we plot the *source scores* for each target word, we can see what each target word is aligned to.

Note the reordering in this particular example



Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Translate and Align. ICLR'15

Summary

- Bilingual LSTM translates from one language to another language
- Encoder-Decoder model helps us *encode* a sequence into a summary vector and use that to *decode* another sequence
- Attention mechanism learns soft alignment between source and target words