Recurrent Networks for Sequence Generation (seq2seq)

9 април 2025 г.

Acknowledgments

Slides adapted from

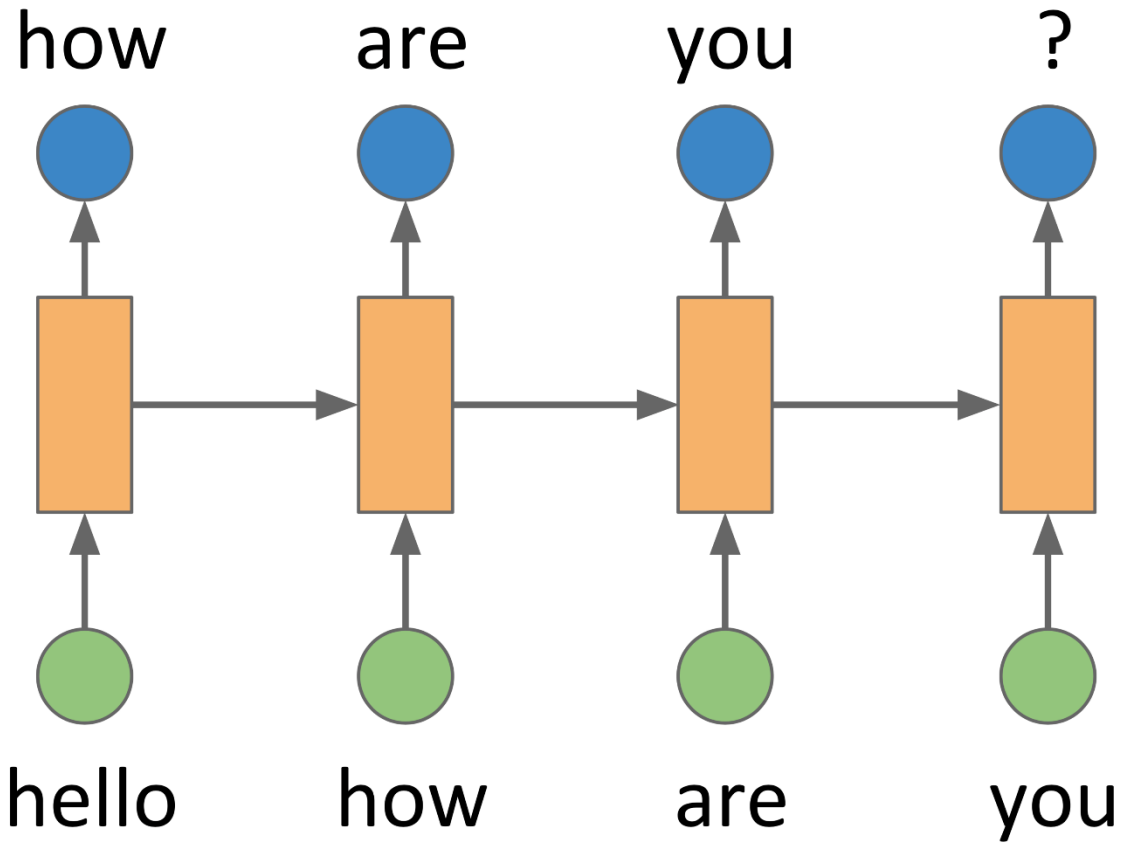
Fahim Dalvi, Heng Ji, Kevin Knight, Hassan Sajjad,

Abigail See, Rico Sennrich, Elena Voita

RNNs Recap

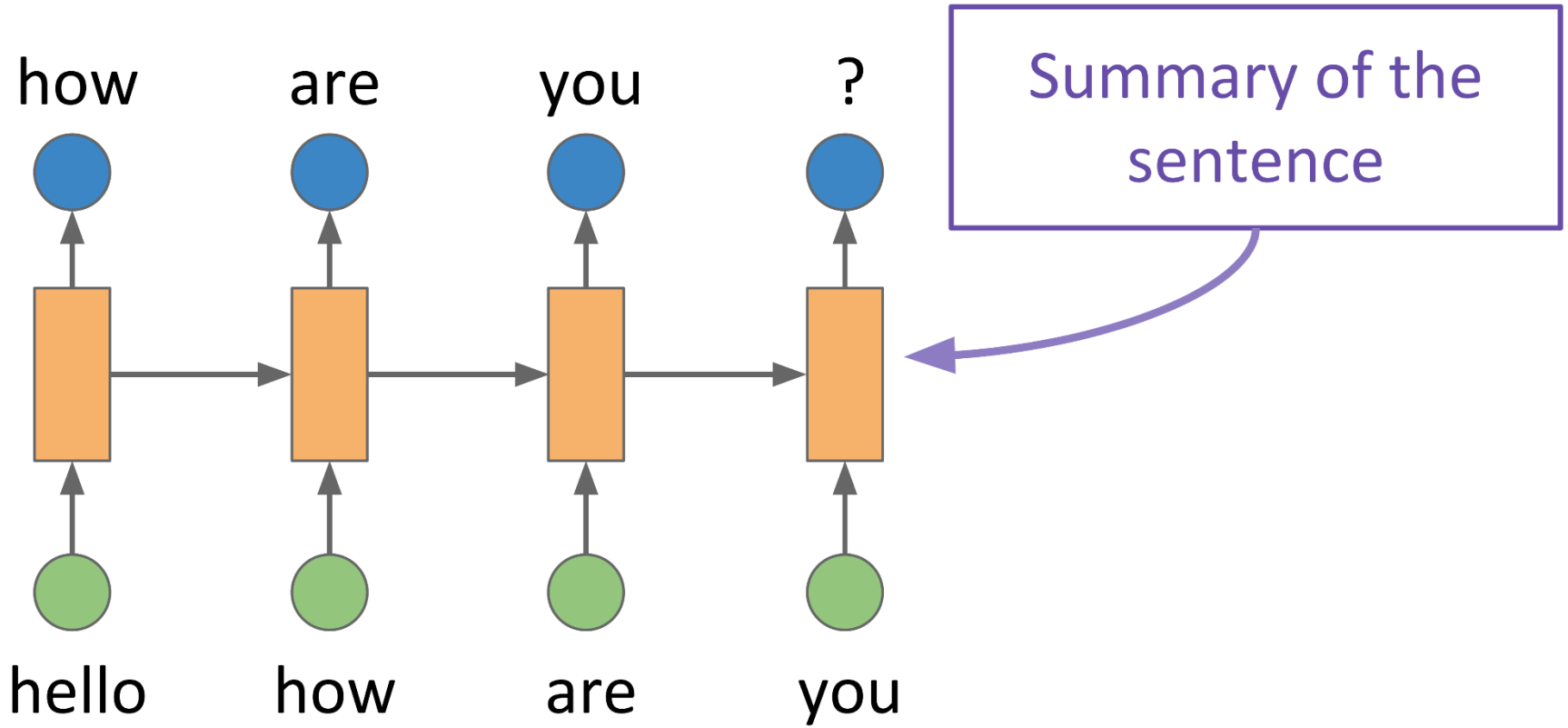
Recap

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



Recap

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



Sequence to Sequence Models (Seq2seq)

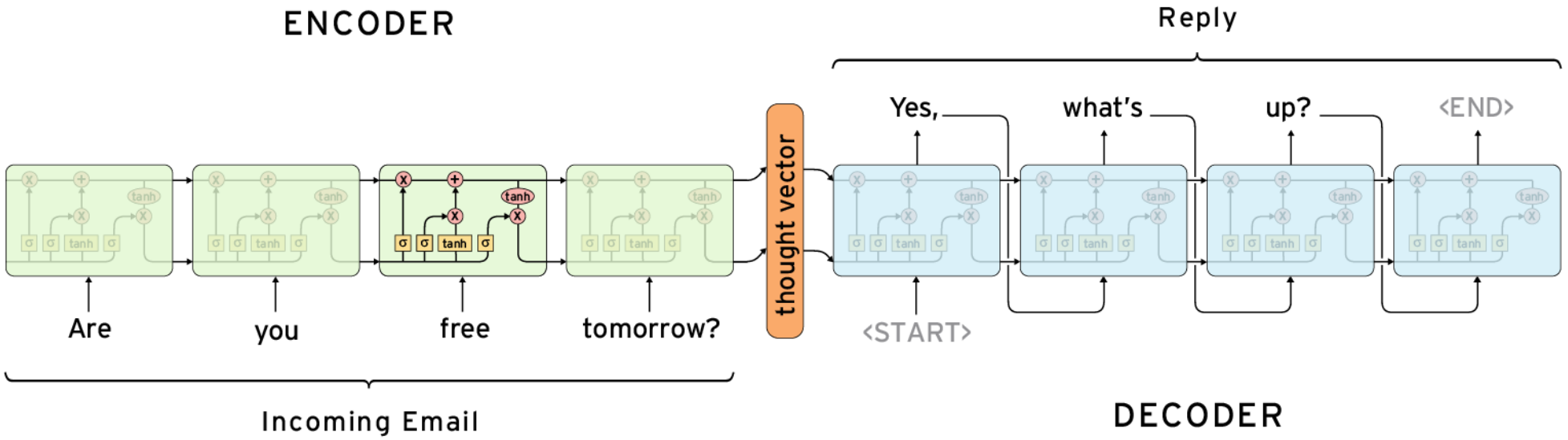
Sequence-to-Sequence Models (Seq2seq)

- Seq2Seq models take RNN a step further

- Seq2Seq models take a sequence and predict another sequence
  - Translation: Source sentence to target sentence
    - sequence of words → sequence of words in a different language
  - Chatbots: question to response answer
    - sequence of words → sequence of words in the same language
  - Speech Recognition: Audio waves to transcription
    - sequence of audio signals → sequence of words
  - Image Captioning: Images to text sequence
    - “sequence” of pixels → sequence of words
  - …

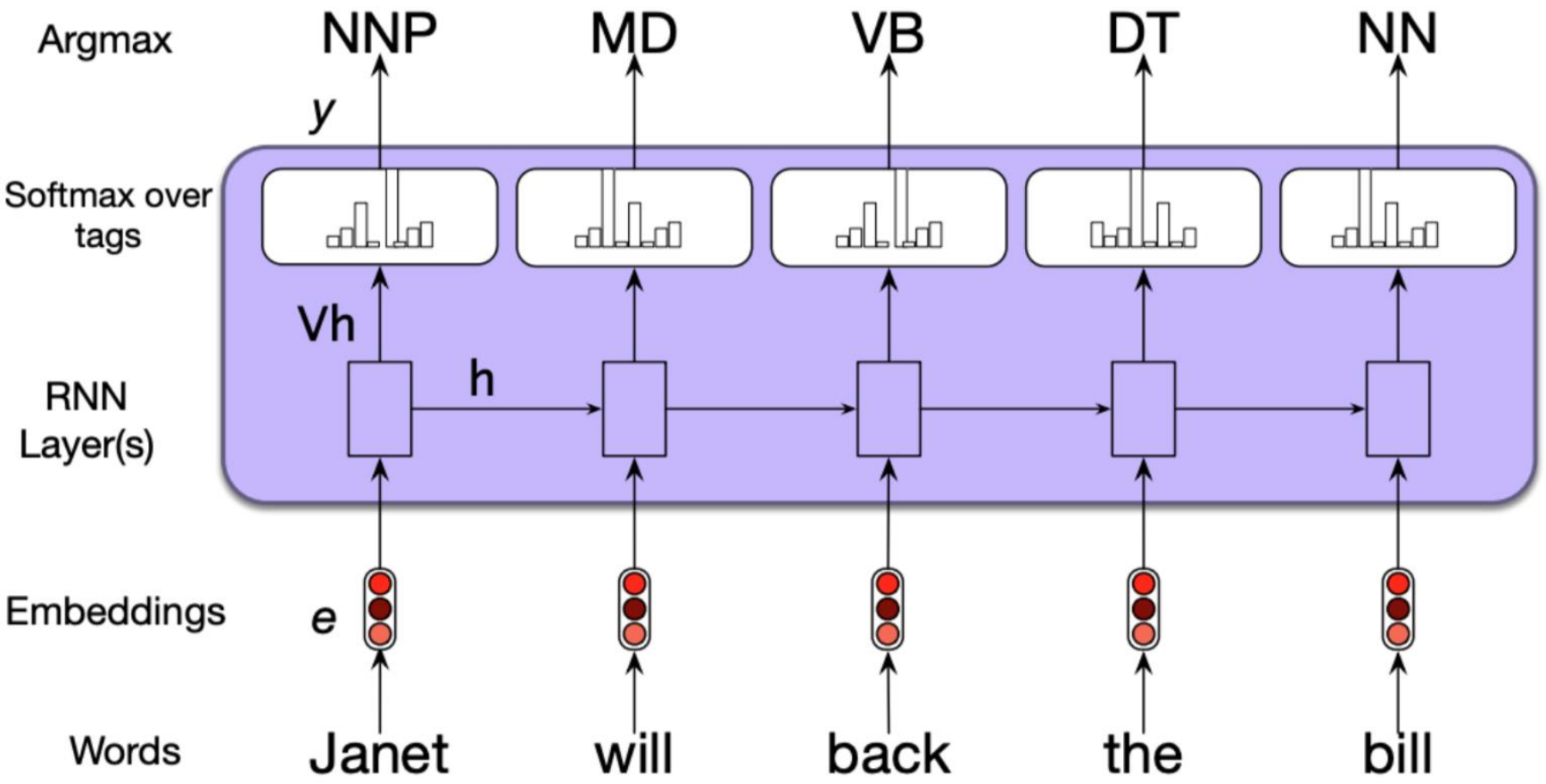
Sequence-to-Sequence Models (Seq2seq)

- Seq2seq model
  1. Reads a source sequence completely
  2. Generates a target sequence



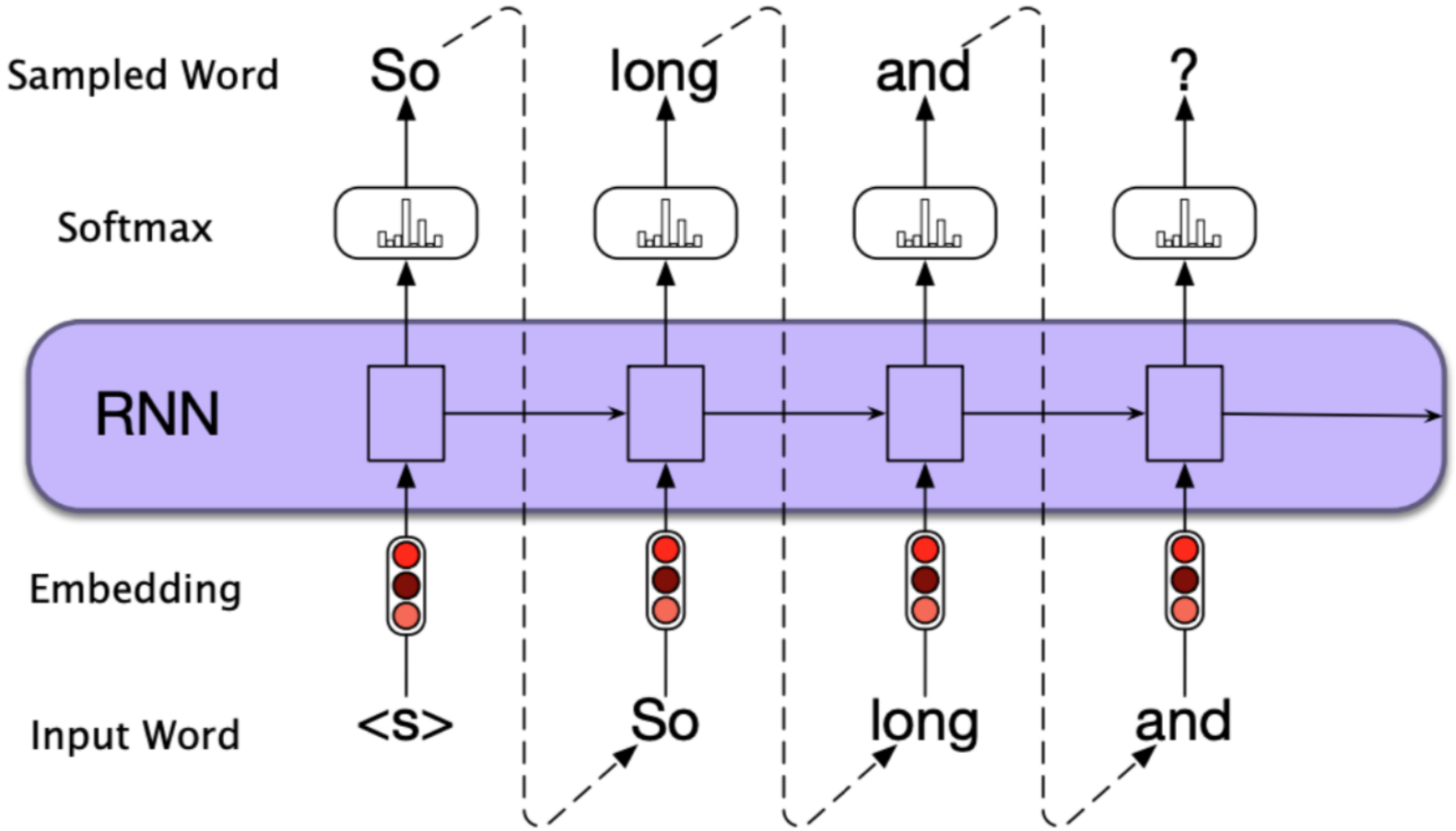
Sequence-to-Sequence Models (Seq2seq)

Seq2Seq is different from per timestep prediction using RNNs



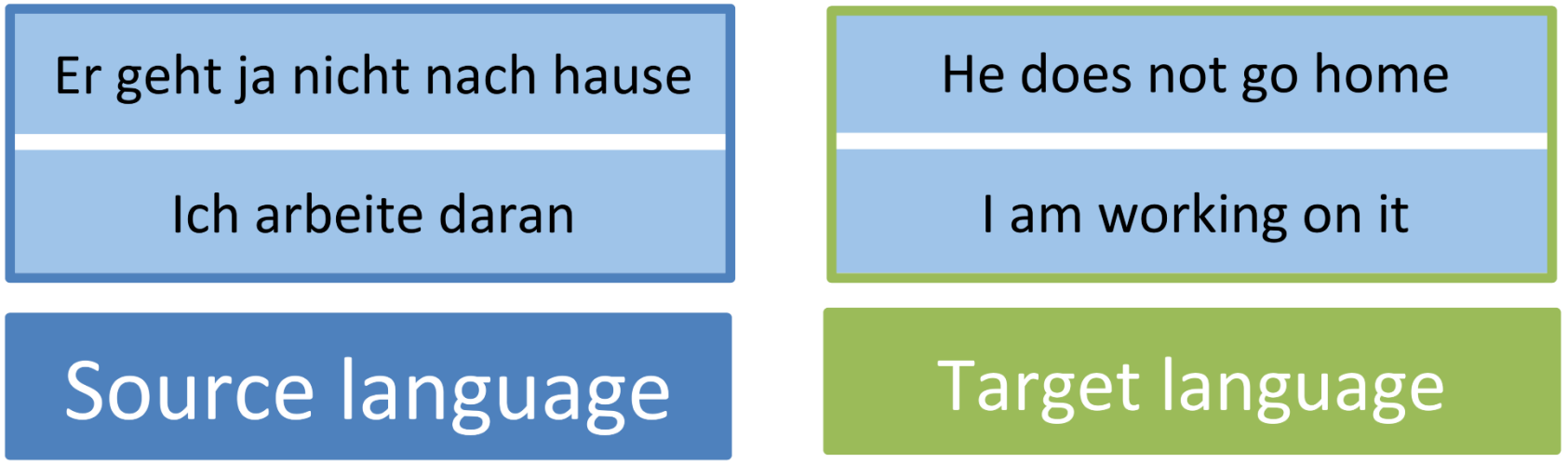
Sequence-to-Sequence Models (Seq2seq)

Seq2Seq is different from sequence generation using RNNs



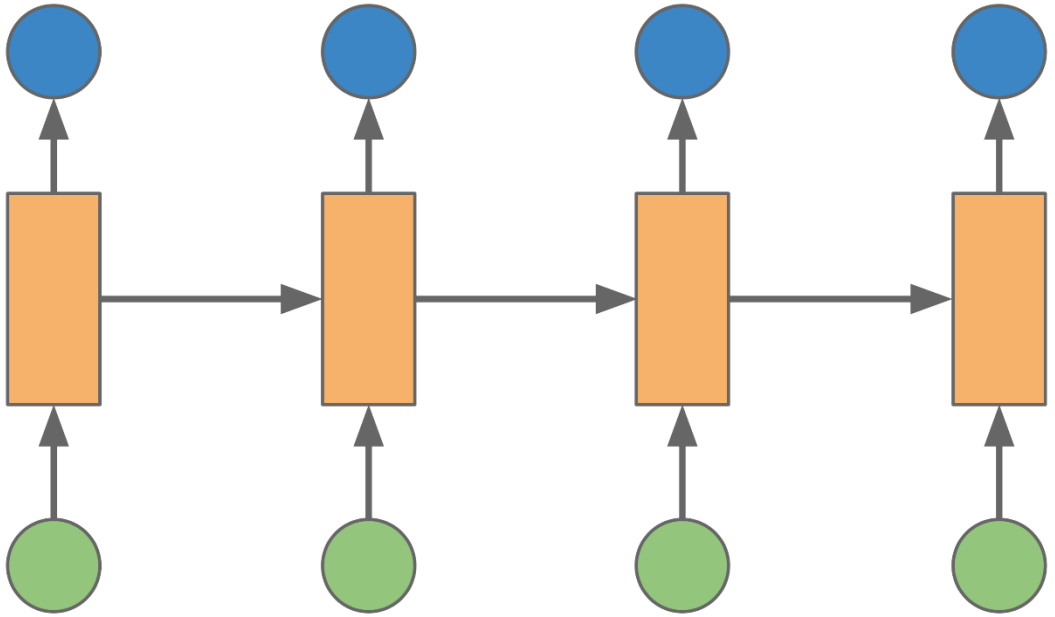
Sequence-to-Sequence Models (Seq2seq)

- Machine translation as an example
  - Pairs of sentences in two languages
  - Given a source sequence of words, generate a target sequence of words
  - The target sequence can only be generated after processing the entire source sequence



Sequence-to-Sequence Models (Seq2seq)

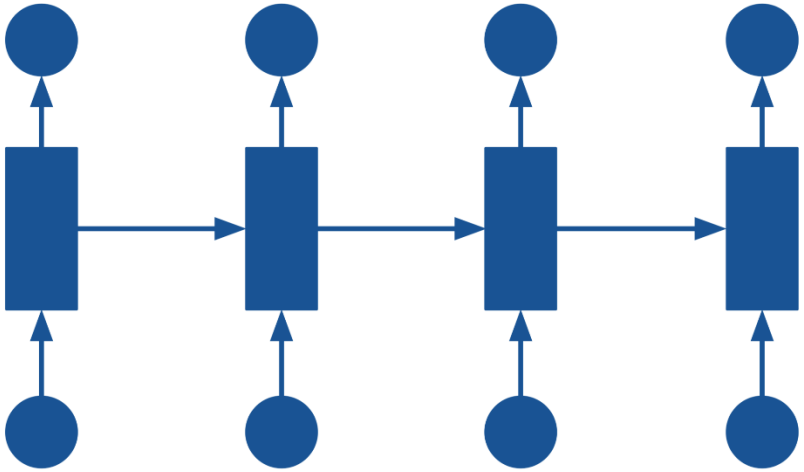
- So far, we have seen an RNN with one language:



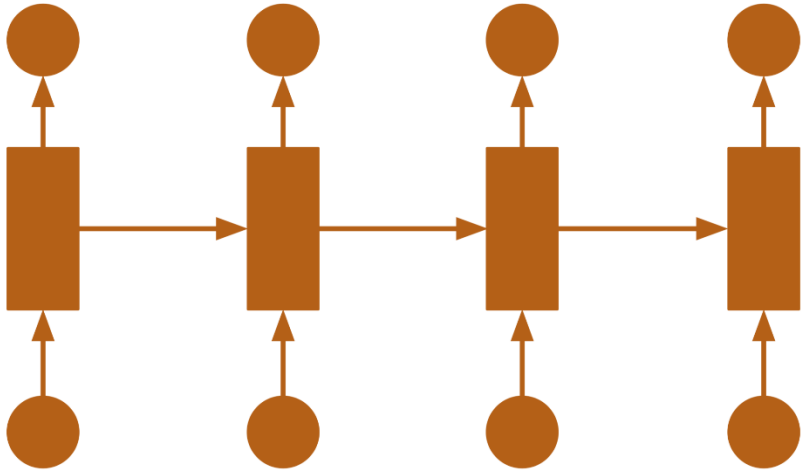
English

Sequence-to-Sequence Models (Seq2seq)

- Seq2seq models can be seen as bilingual RNNs
  - Consider two language models



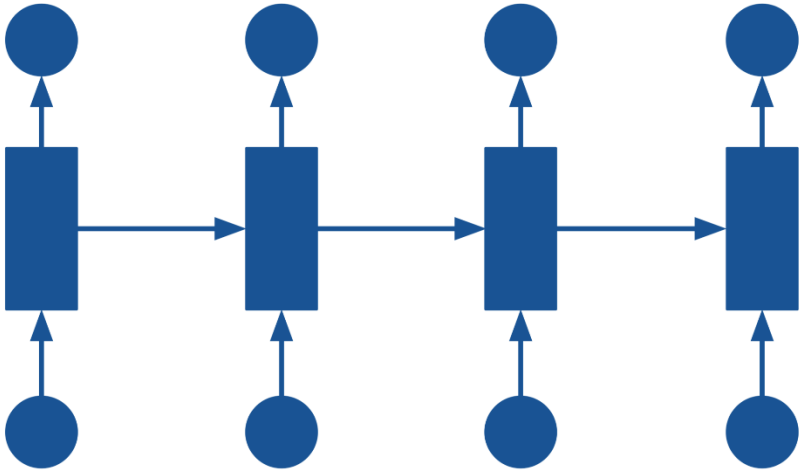
English



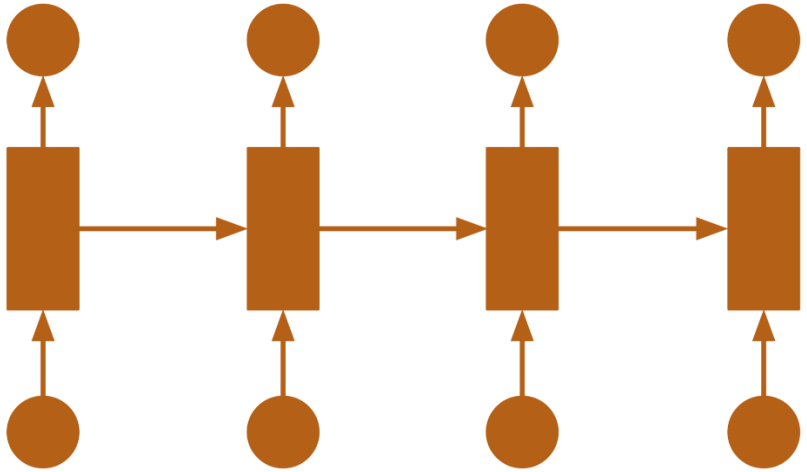
German

Sequence-to-Sequence Models (Seq2seq)

Recap: a language model *can be used to* predict the next word given a sequence of words



English



German

Sequence-to-Sequence Models (Seq2seq)

Now, consider a bilingual RNN:

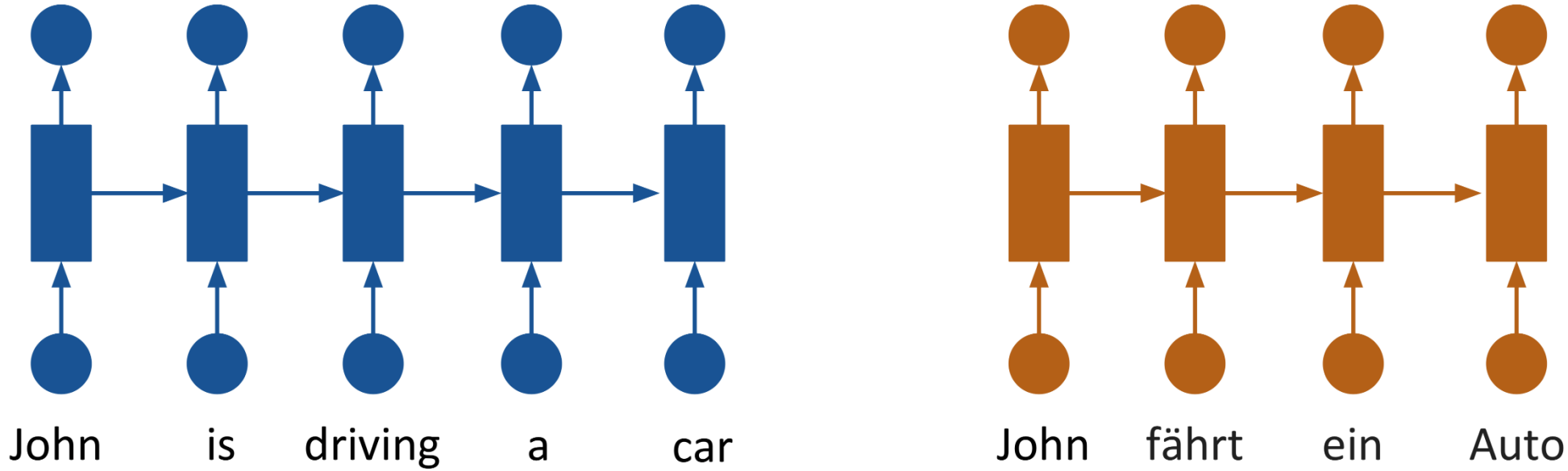
- Imagine English and German sentences as a single sequence of strings
- No explicit information about the source and the target language

John is driving a car . John fährt ein Auto .

Boundary symbols to mark source and target sentence endings

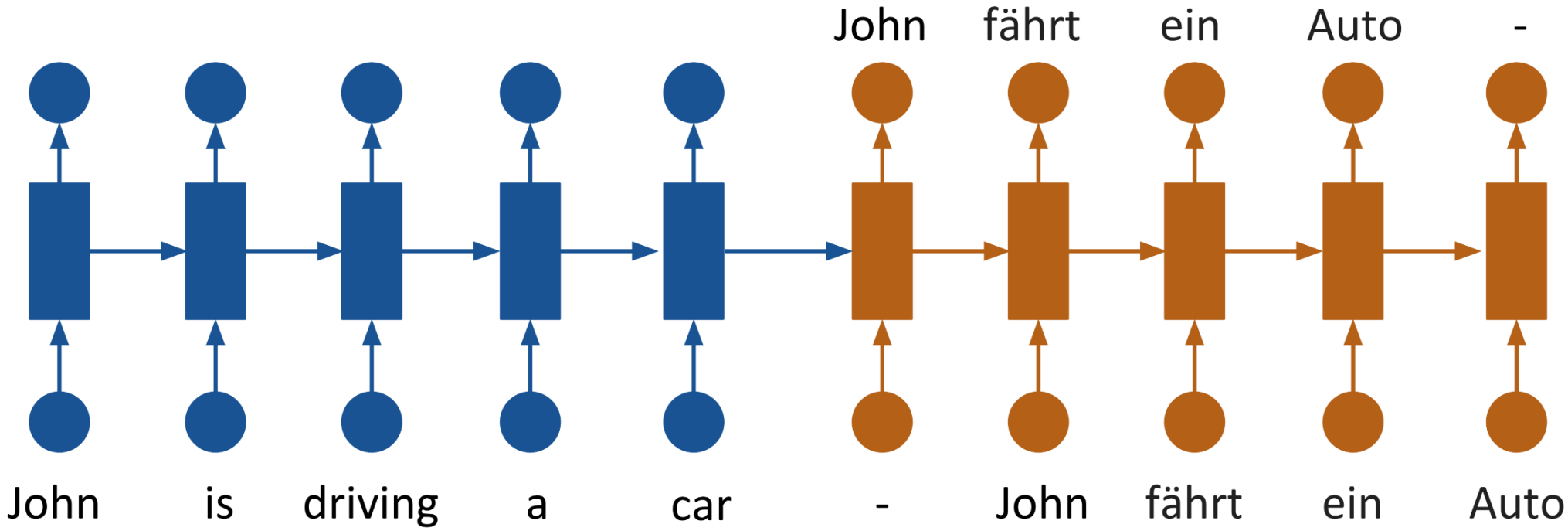
Sequence-to-Sequence Models (Seq2seq)

Now, let us consider the two languages as a single one



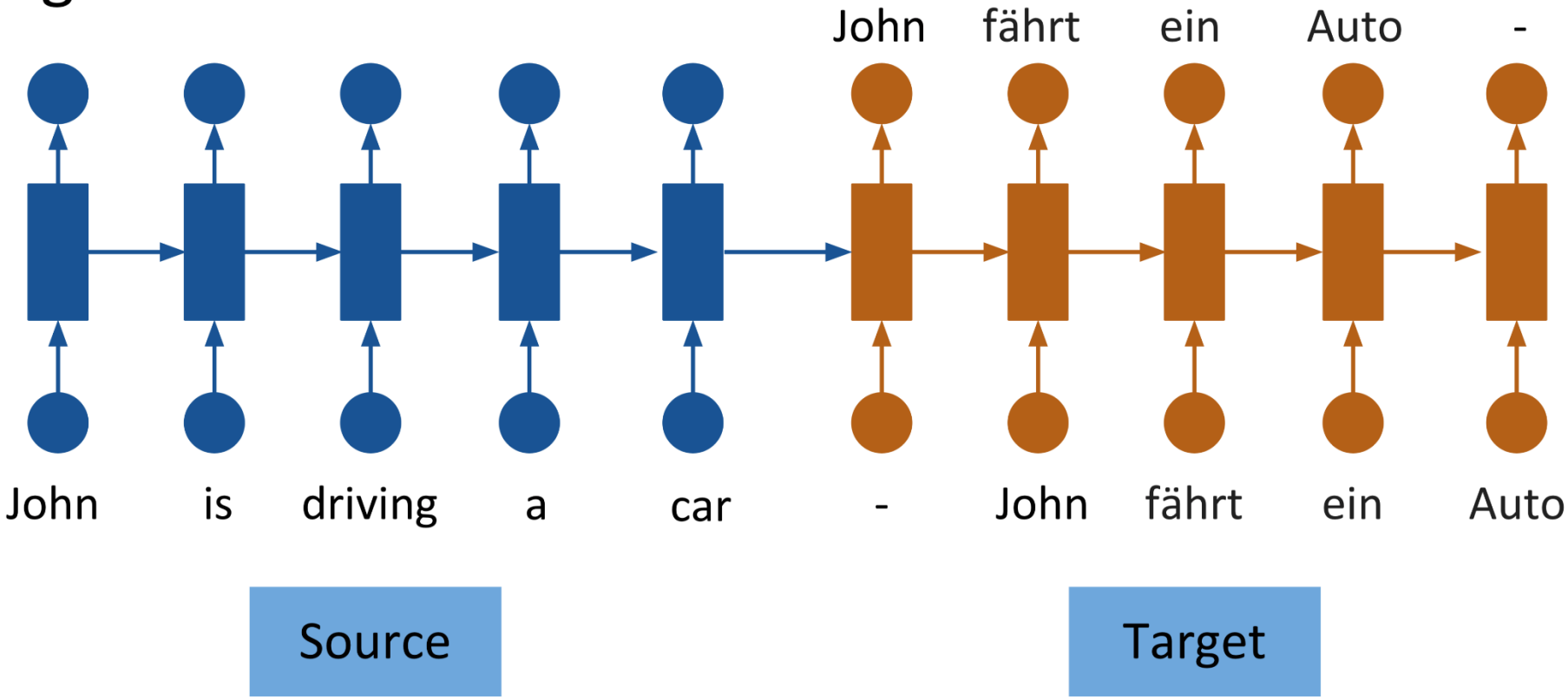
Sequence-to-Sequence Models (Seq2seq)

Now, let us consider the two languages as a single one



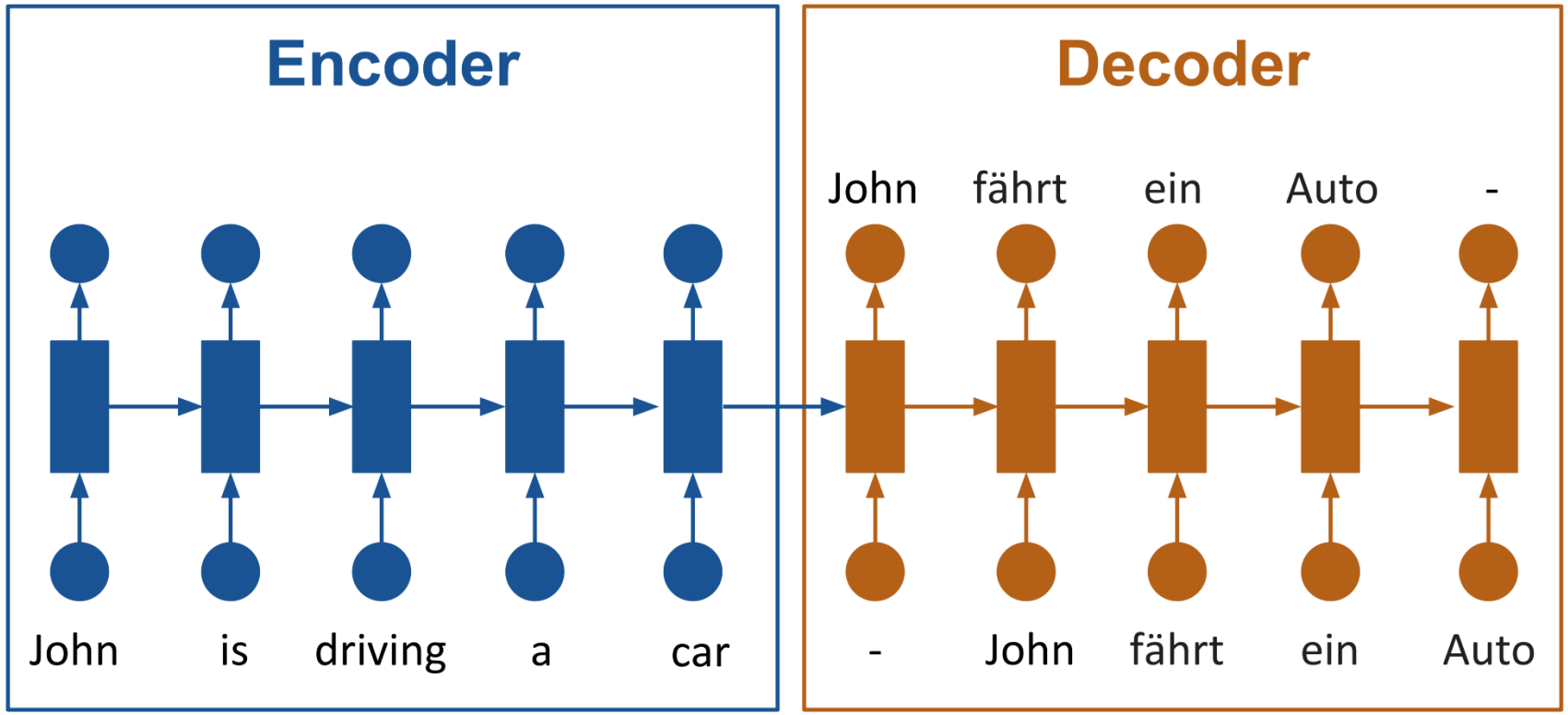
Sequence-to-Sequence Models (Seq2seq)

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



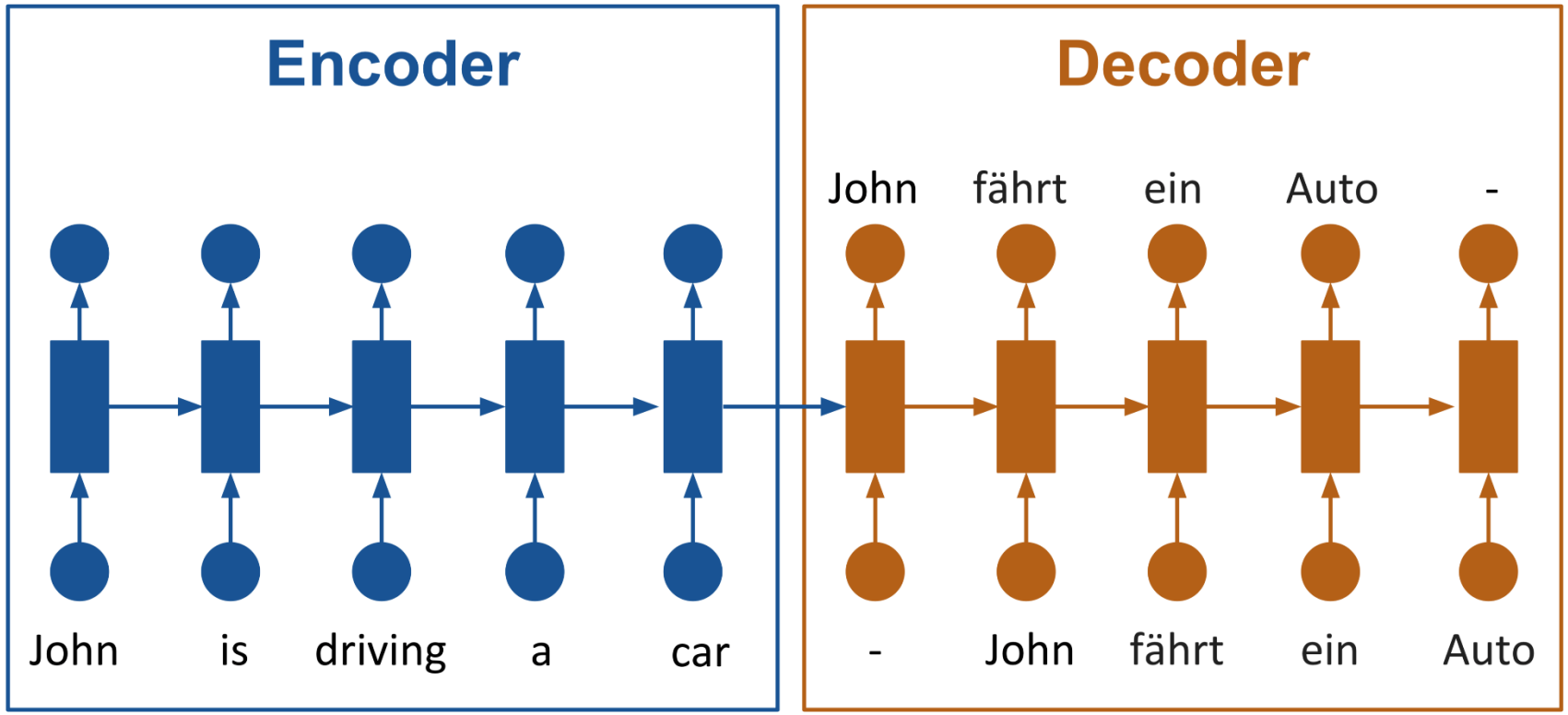
Sequence-to-Sequence Models (Seq2seq)

This is called the Encode-Decoder Architecture



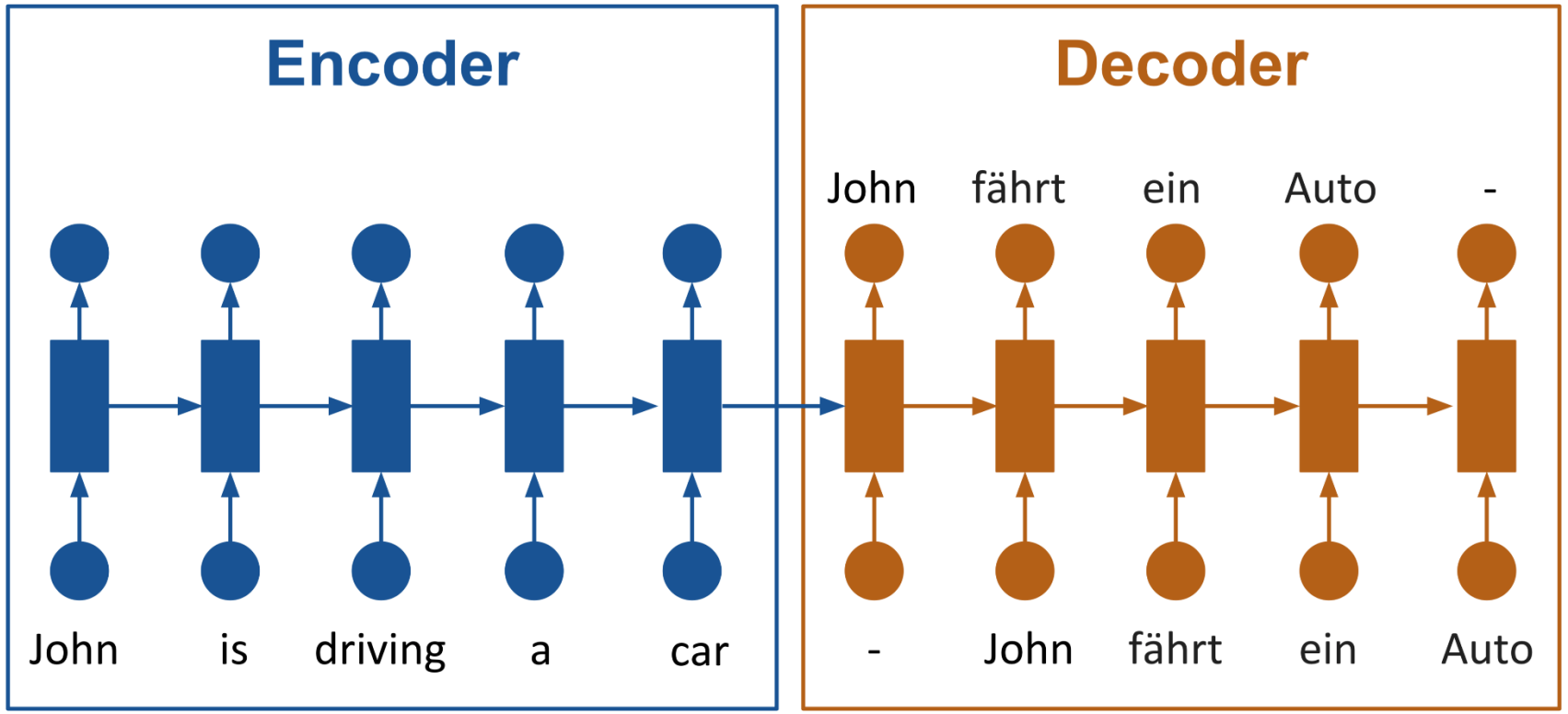
Sequence-to-Sequence Models (Seq2seq)

The model learns to read a source sequence,
and then predict the corresponding target sequence



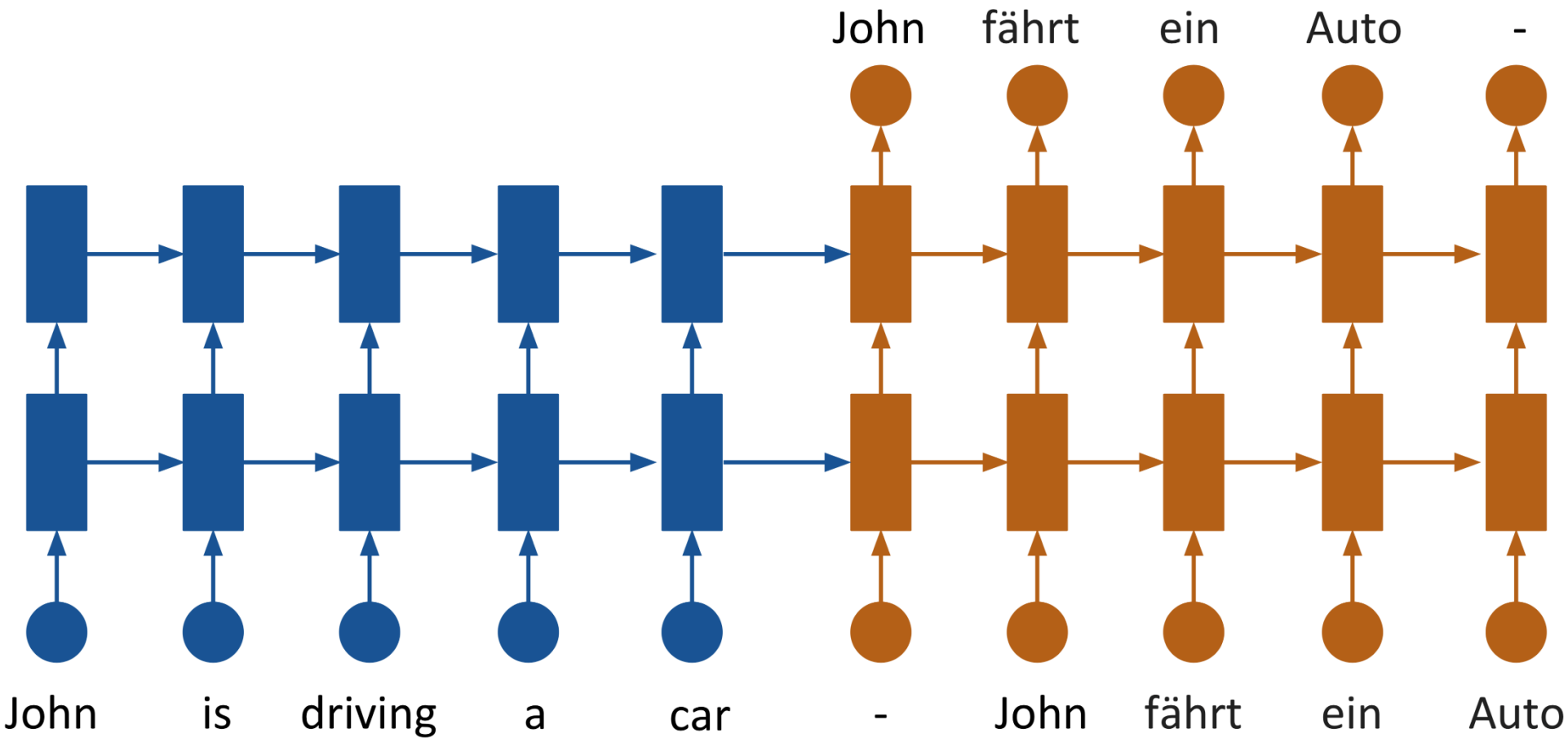
Sequence-to-Sequence Models (Seq2seq)

We generally do not produce any output in the Encoder, as we are interested in generating the second sequence

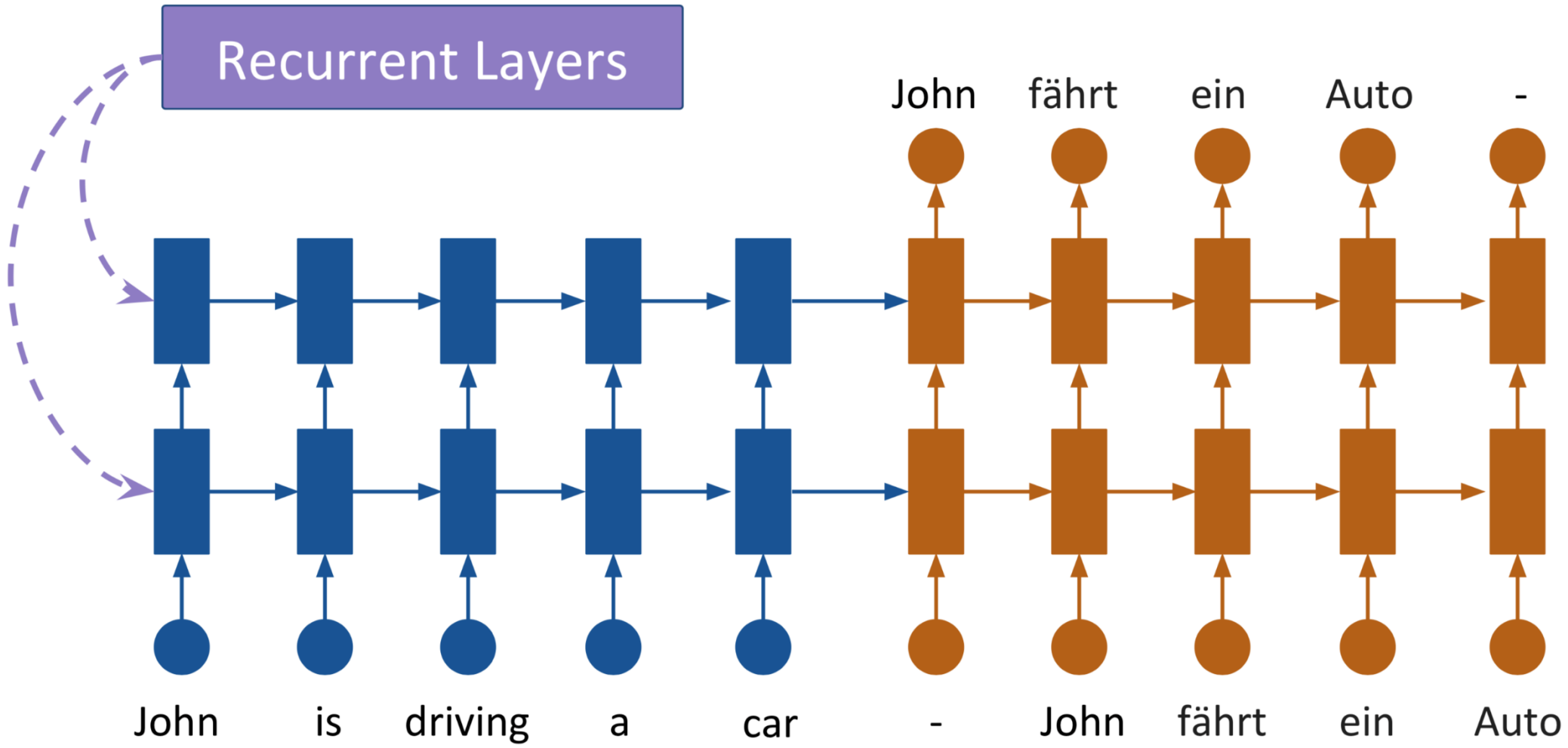


The Anatomy of a Seq2Seq Model

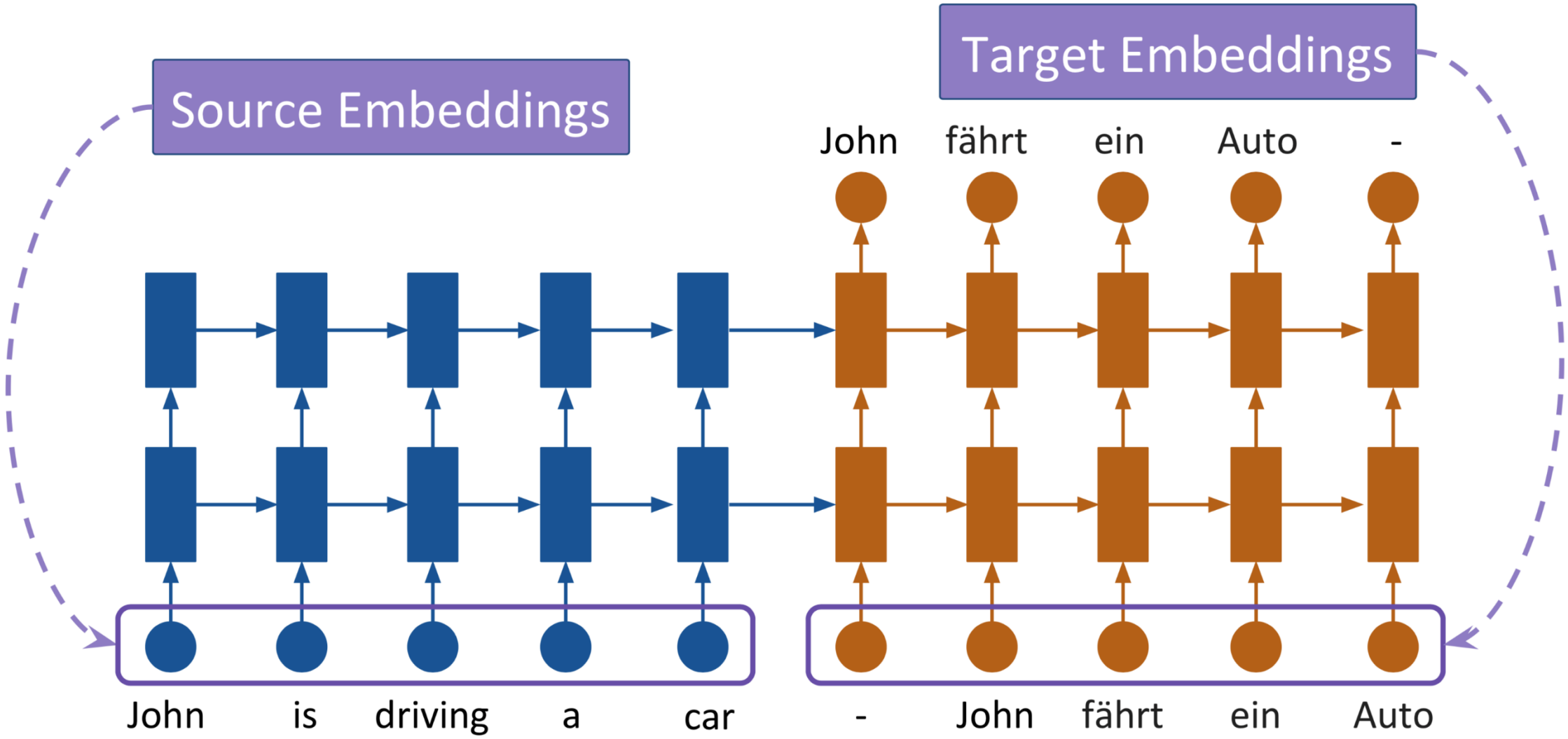
The Anatomy of a Seq2Seq Model



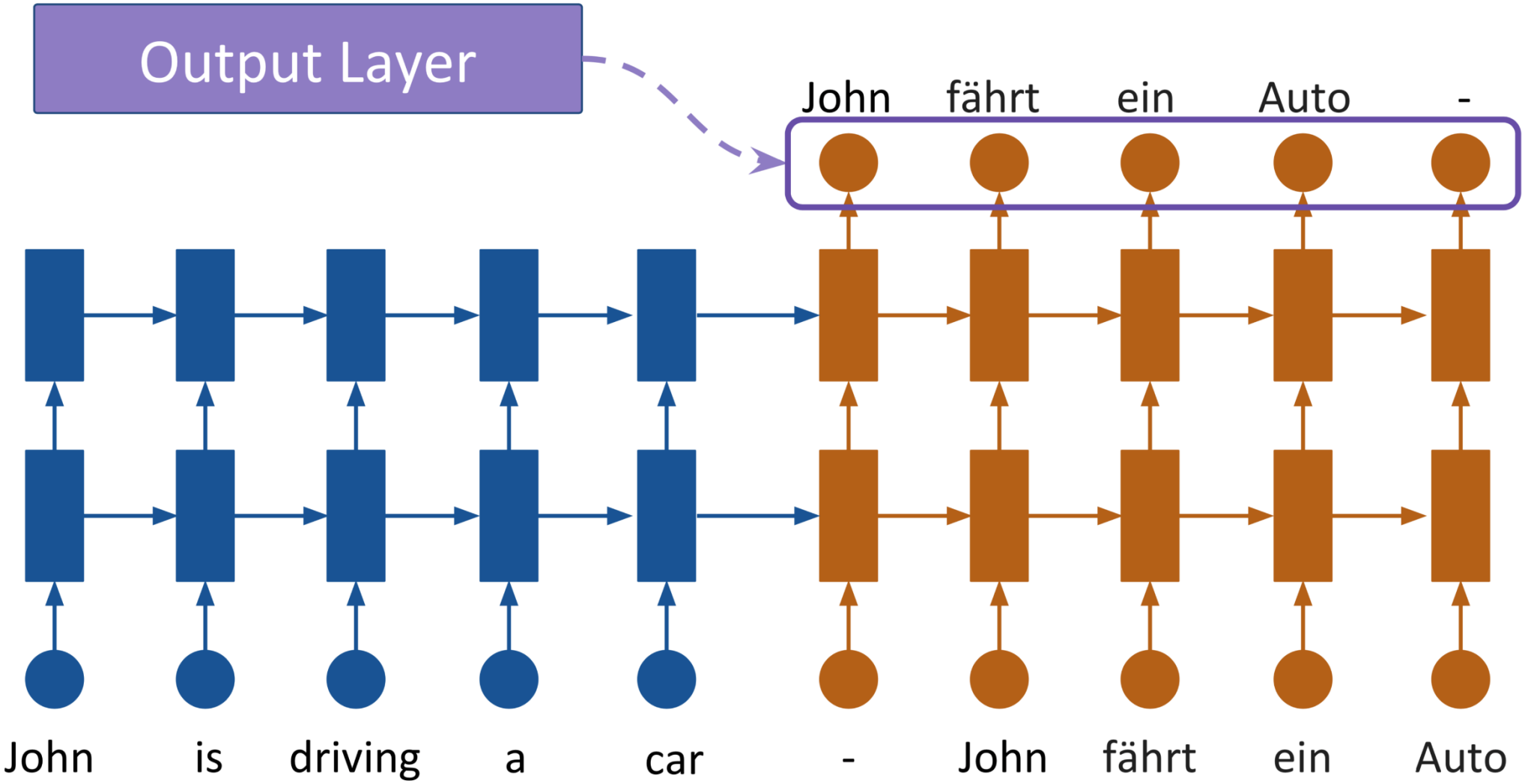
The Anatomy of a Seq2Seq Model



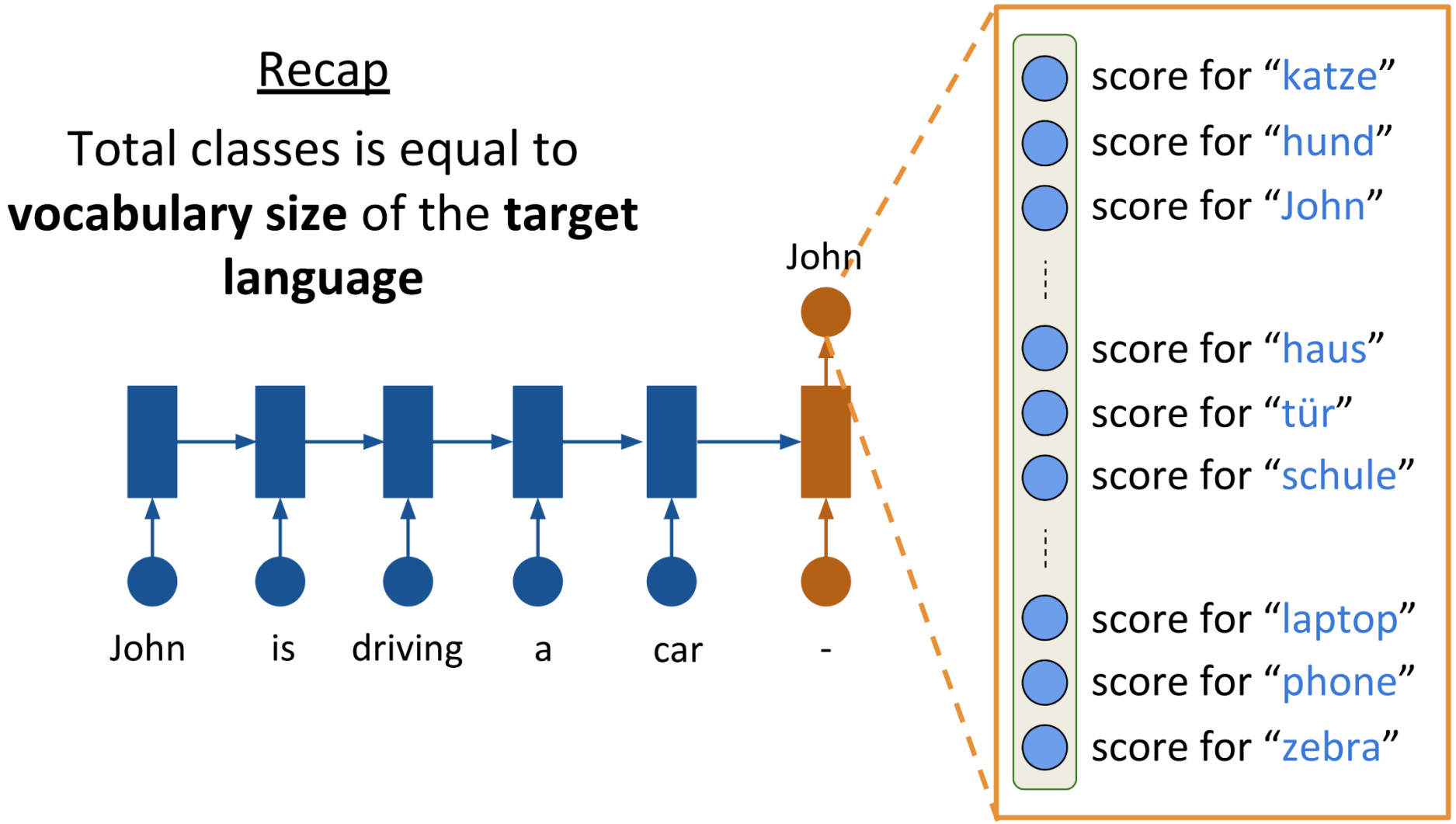
The Anatomy of a Seq2Seq Model



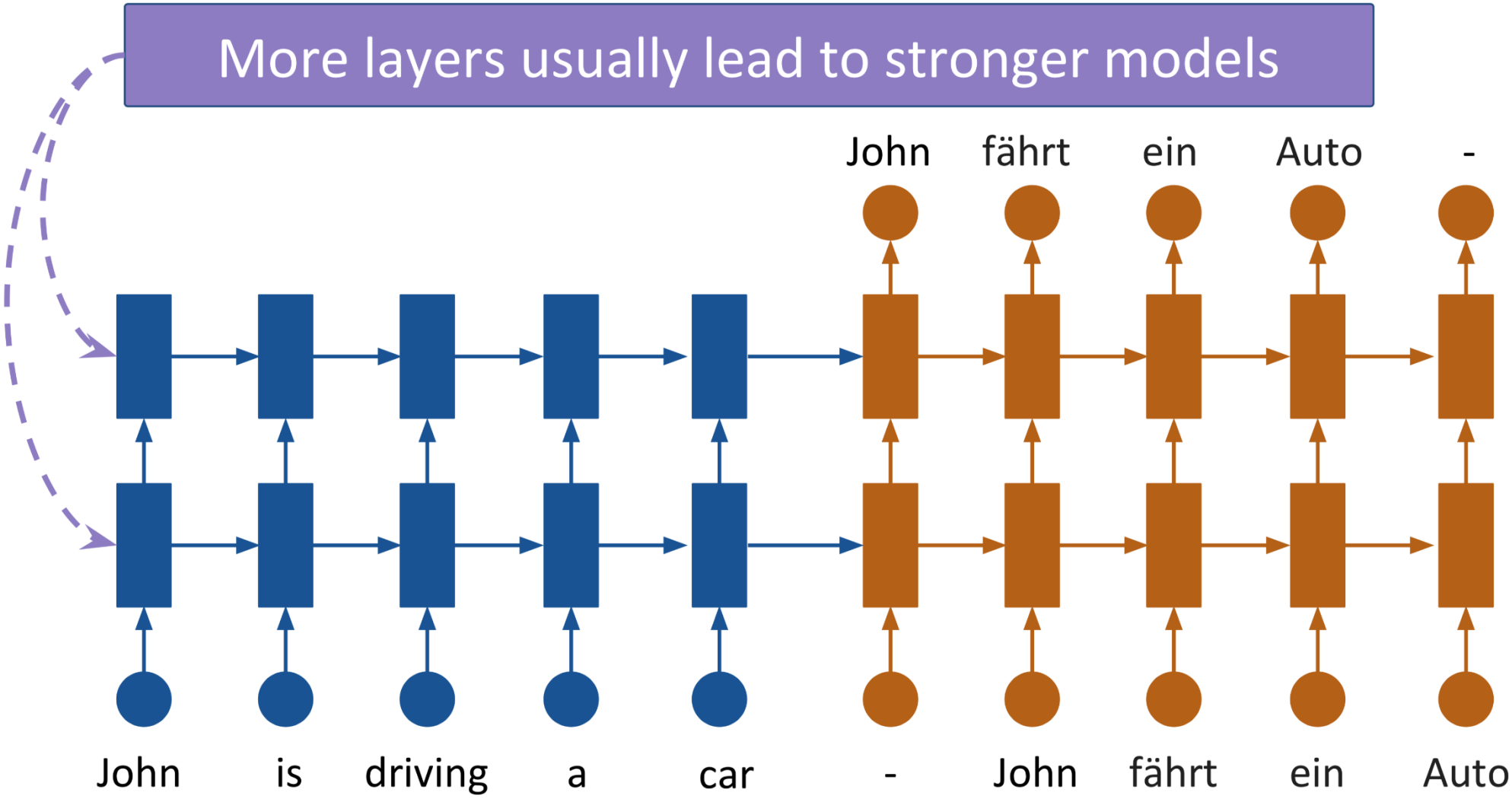
The Anatomy of a Seq2Seq Model



The Anatomy of a Seq2Seq Model



The Anatomy of a Seq2Seq Model



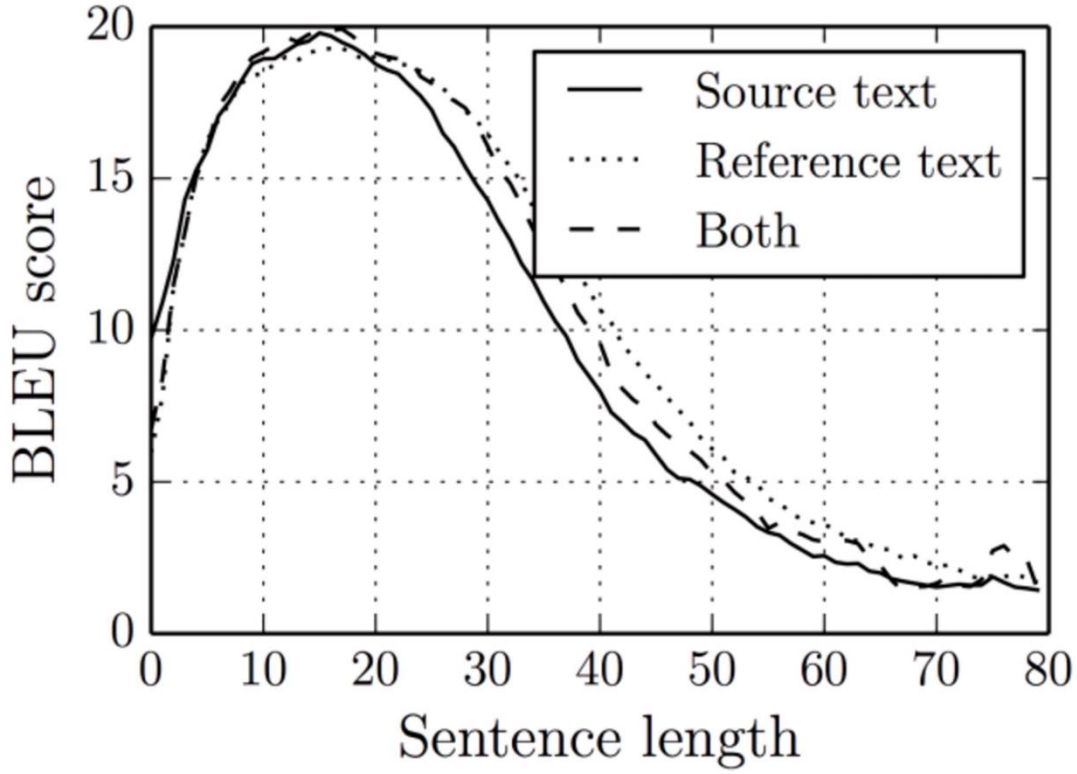
Seq2Seq Models

- End-to-end training
  - one loss to optimize all parameters
- Better use of context
  - use source sentences and previously predicted target words
- Distributed word representations
  - semantic similarities

Issues with Seq2Seq Models

Issues with Seq2Seq Models

Sequence to sequence models perform poorly when translating long sentences

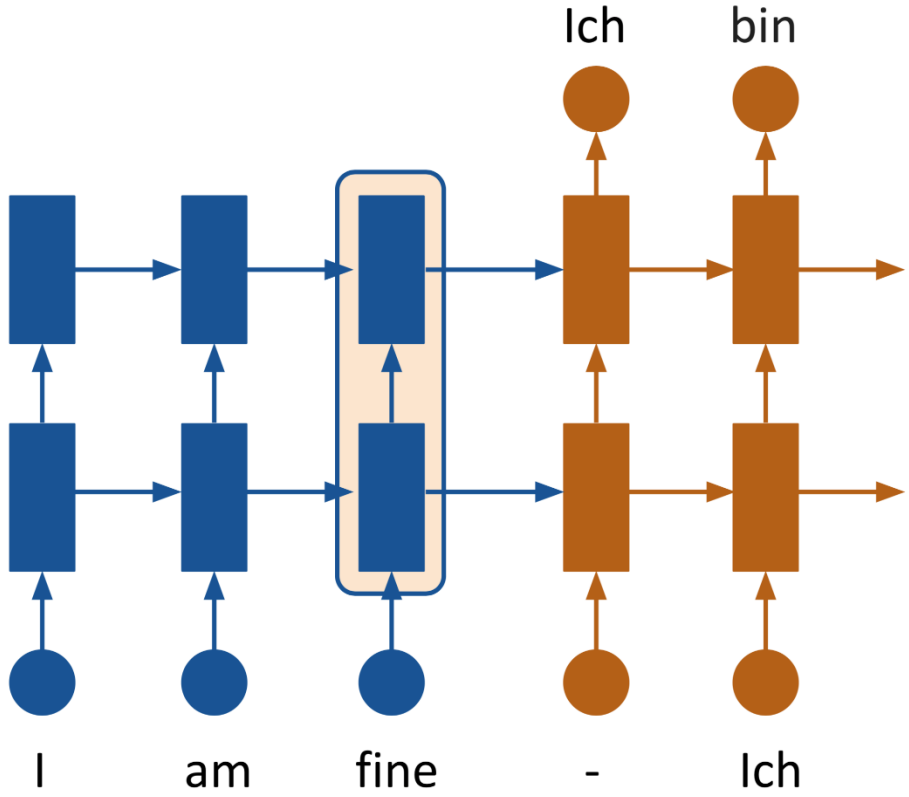


https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Issues with Seq2Seq Models

- Sequence to sequence models perform poorly when translating long sentences

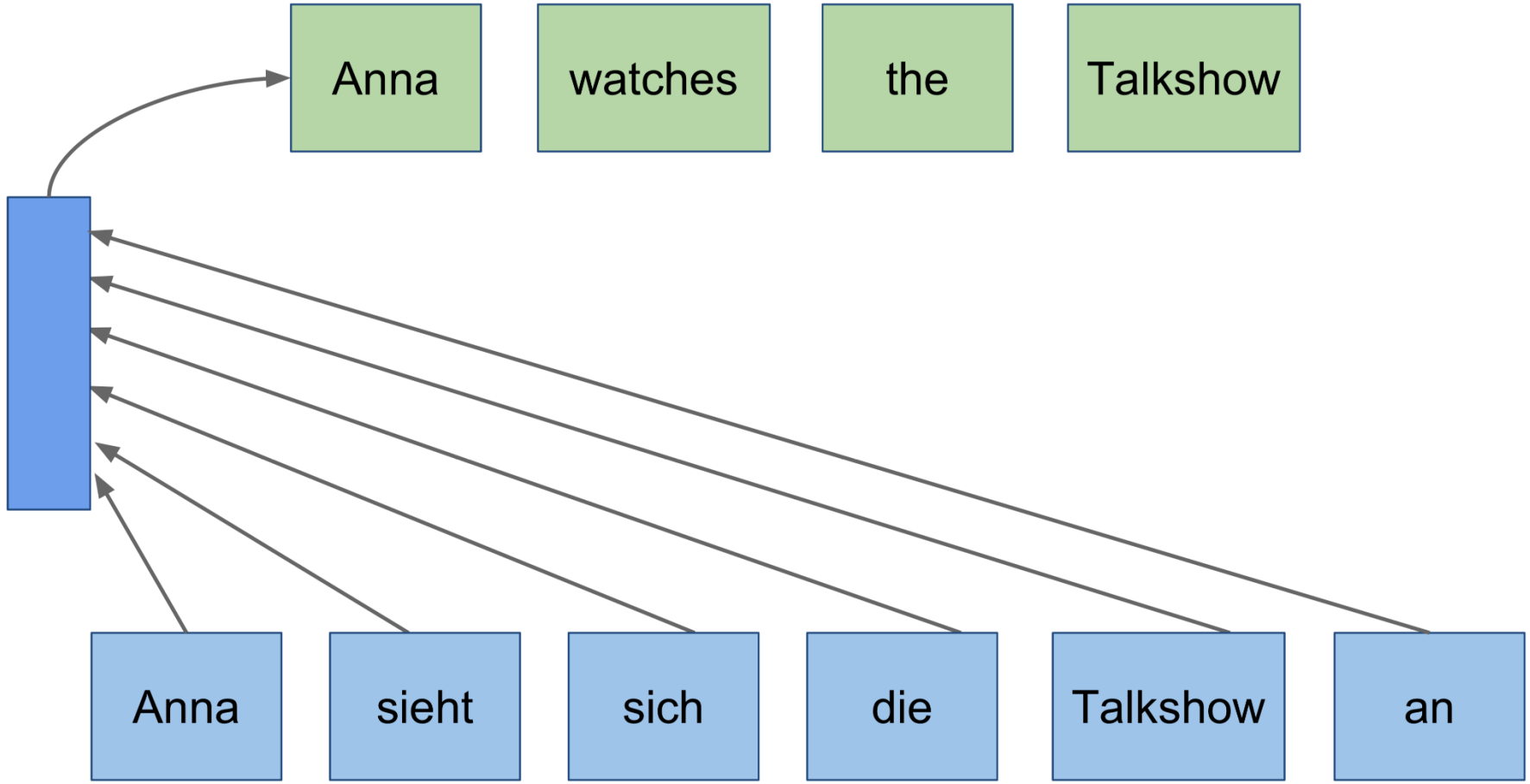
- **Issue**: a sentence is represented as a fixed vector
- Relationship between source and target words is very abstract
- All words are not equally important to predict a target word



Attention Mechanism

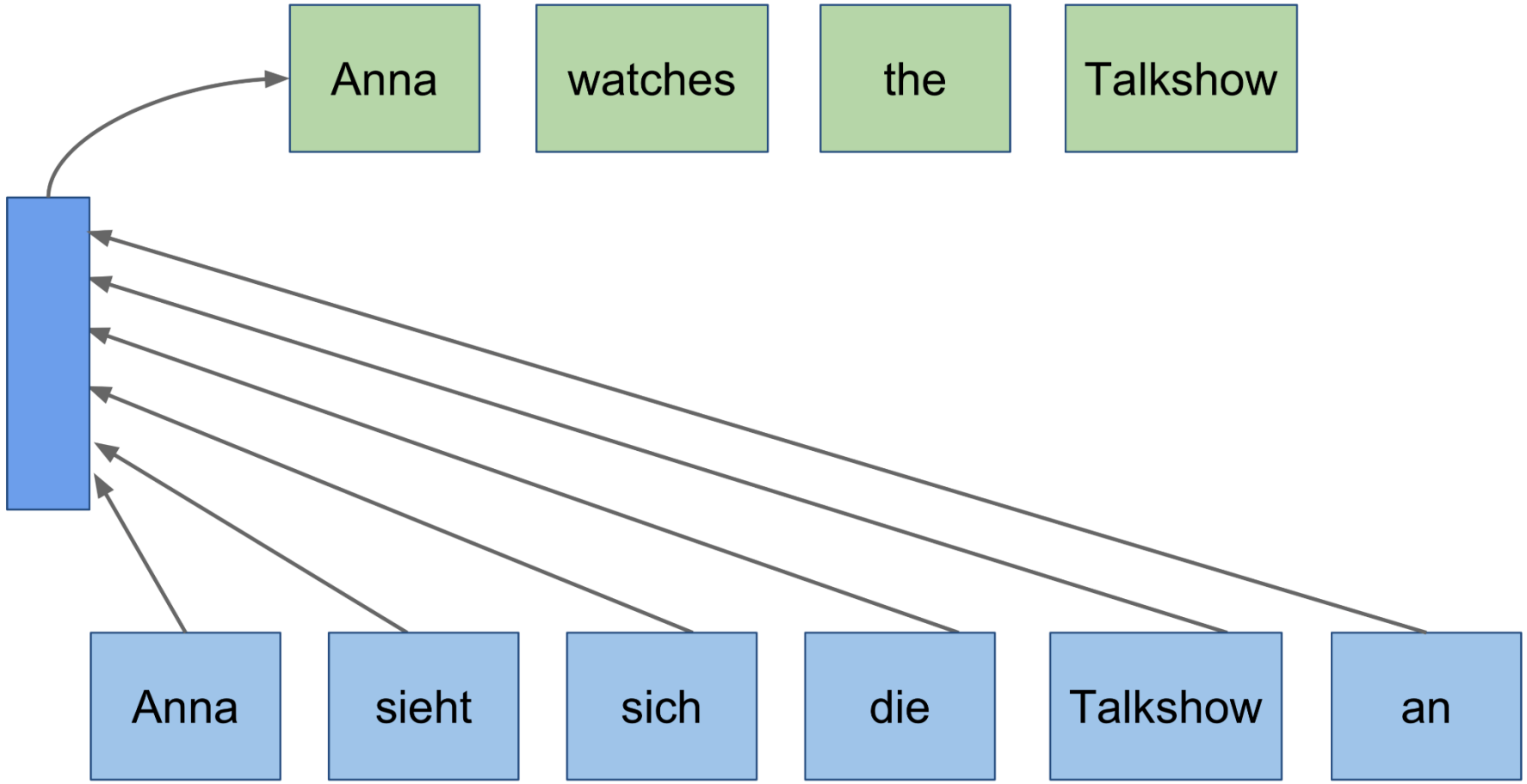
Attention Mechanism

One vector represents all source words



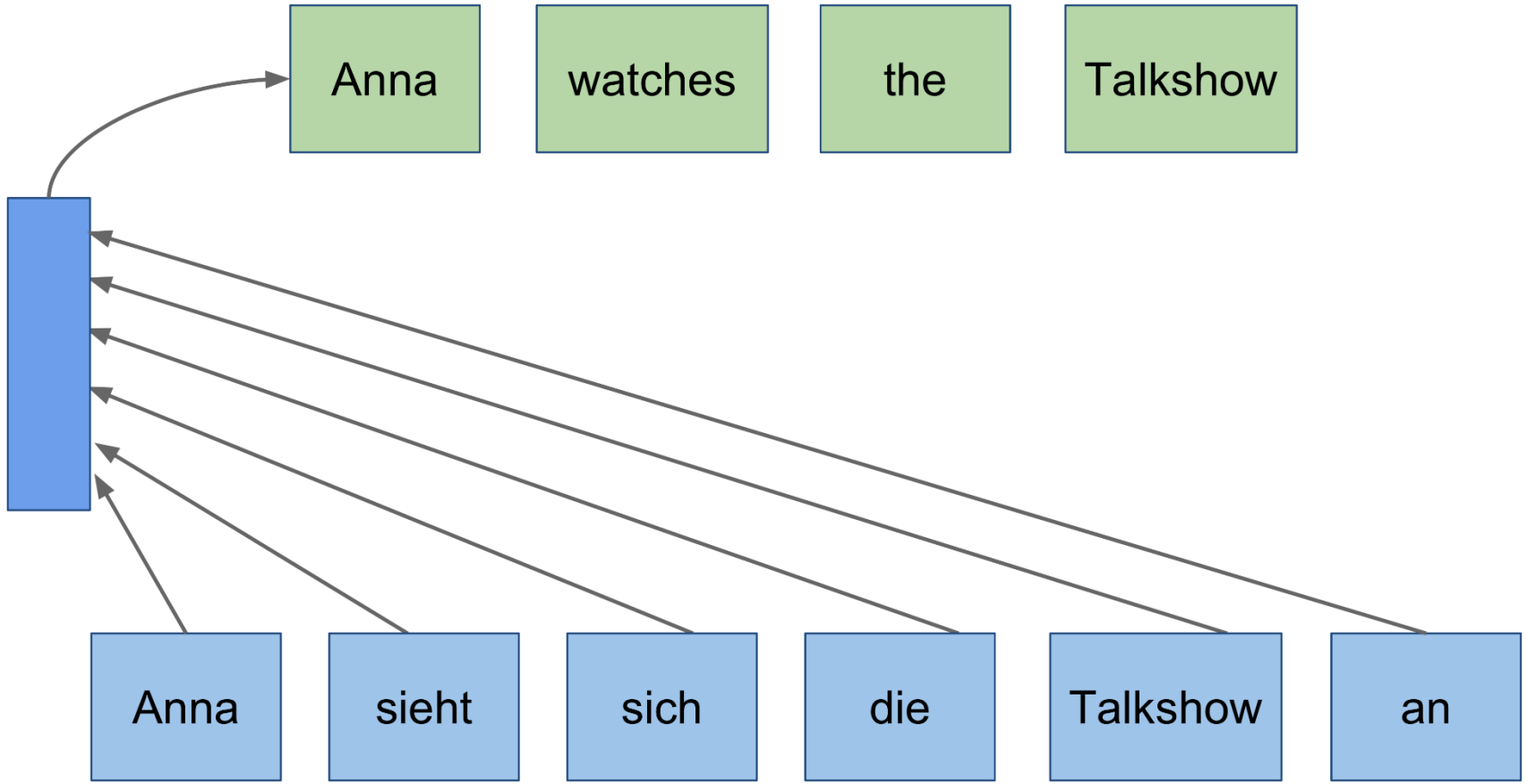
Attention Mechanism

Source and target words inherently have relationships



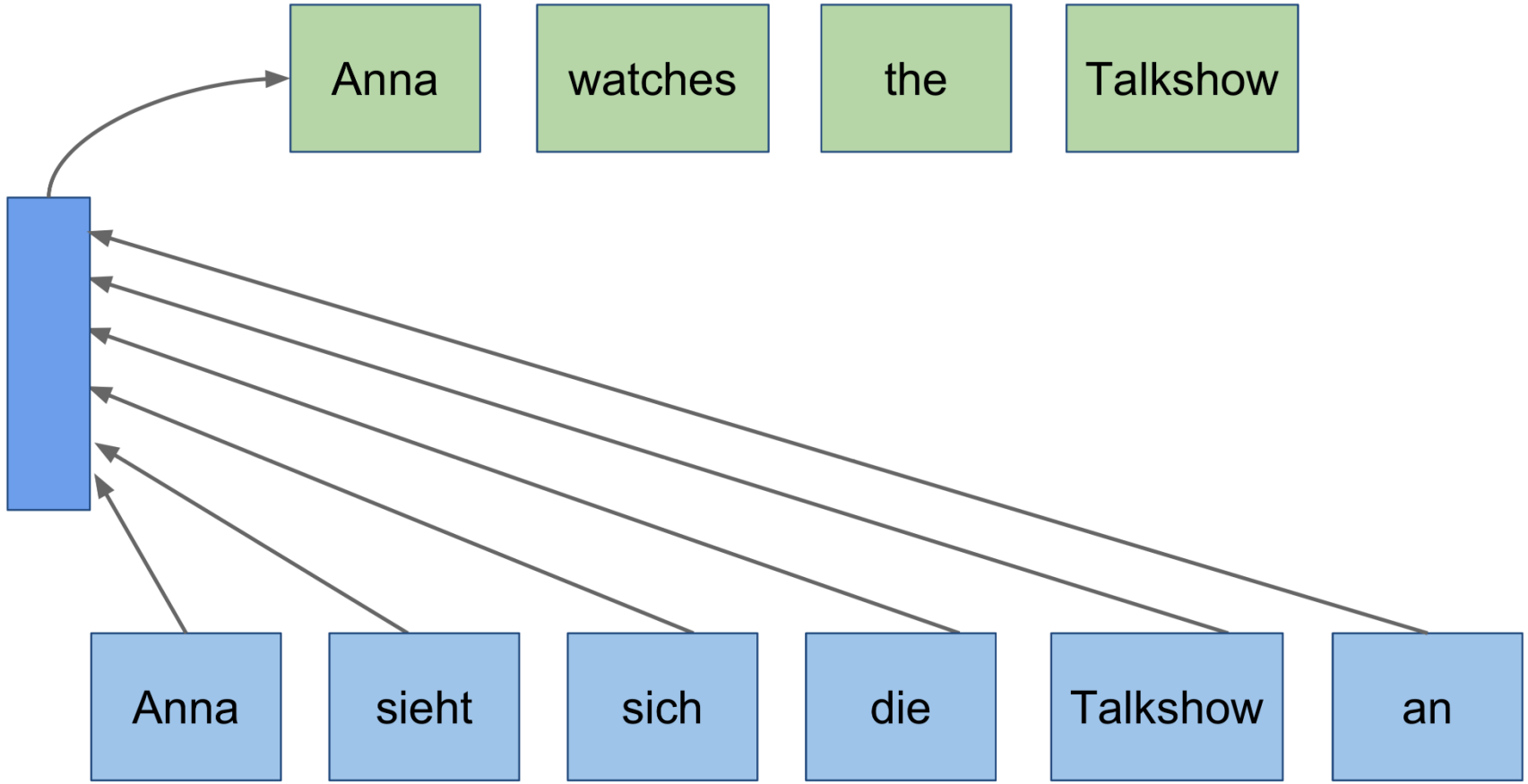
Attention Mechanism

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



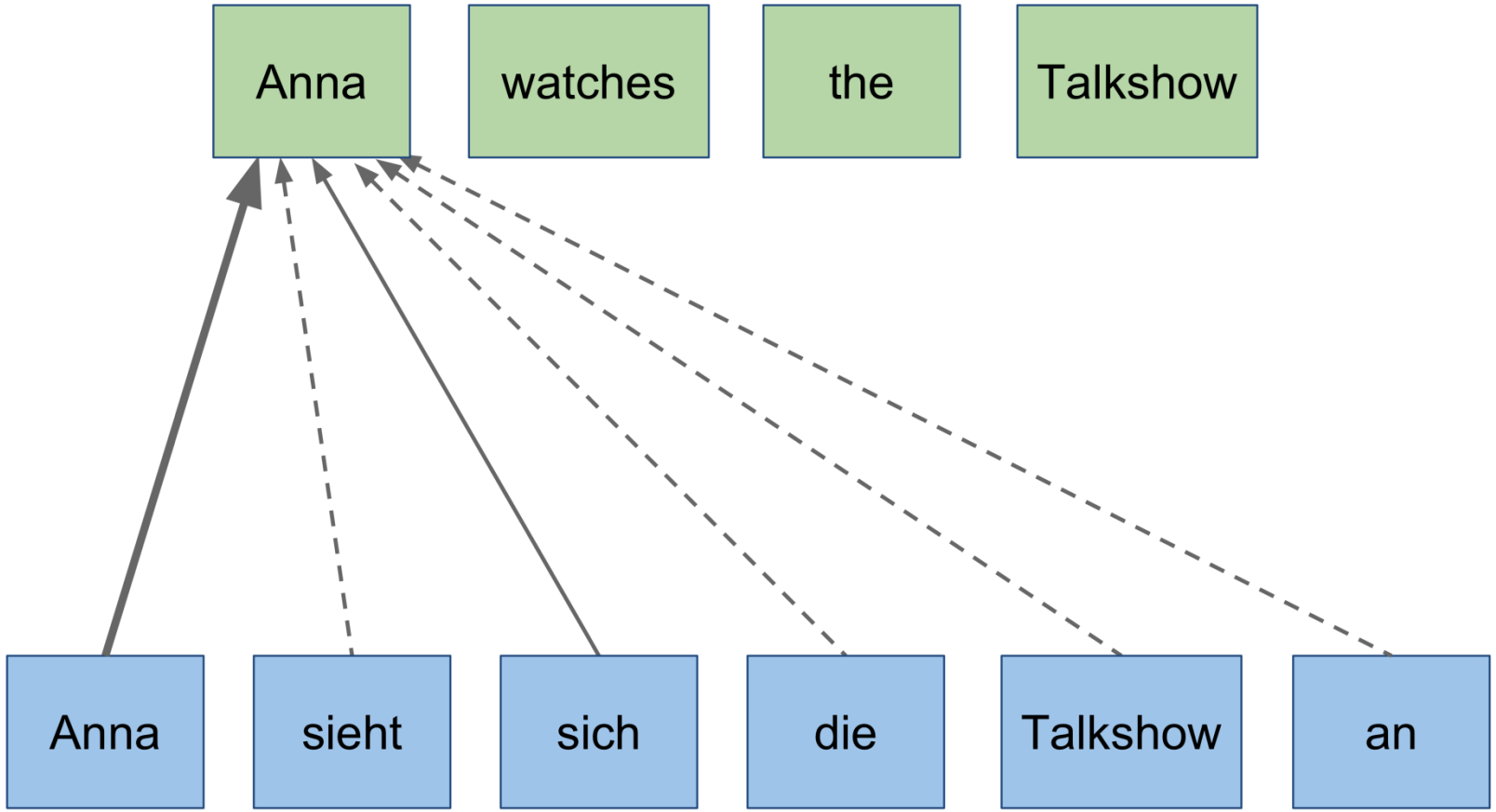
Attention Mechanism

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



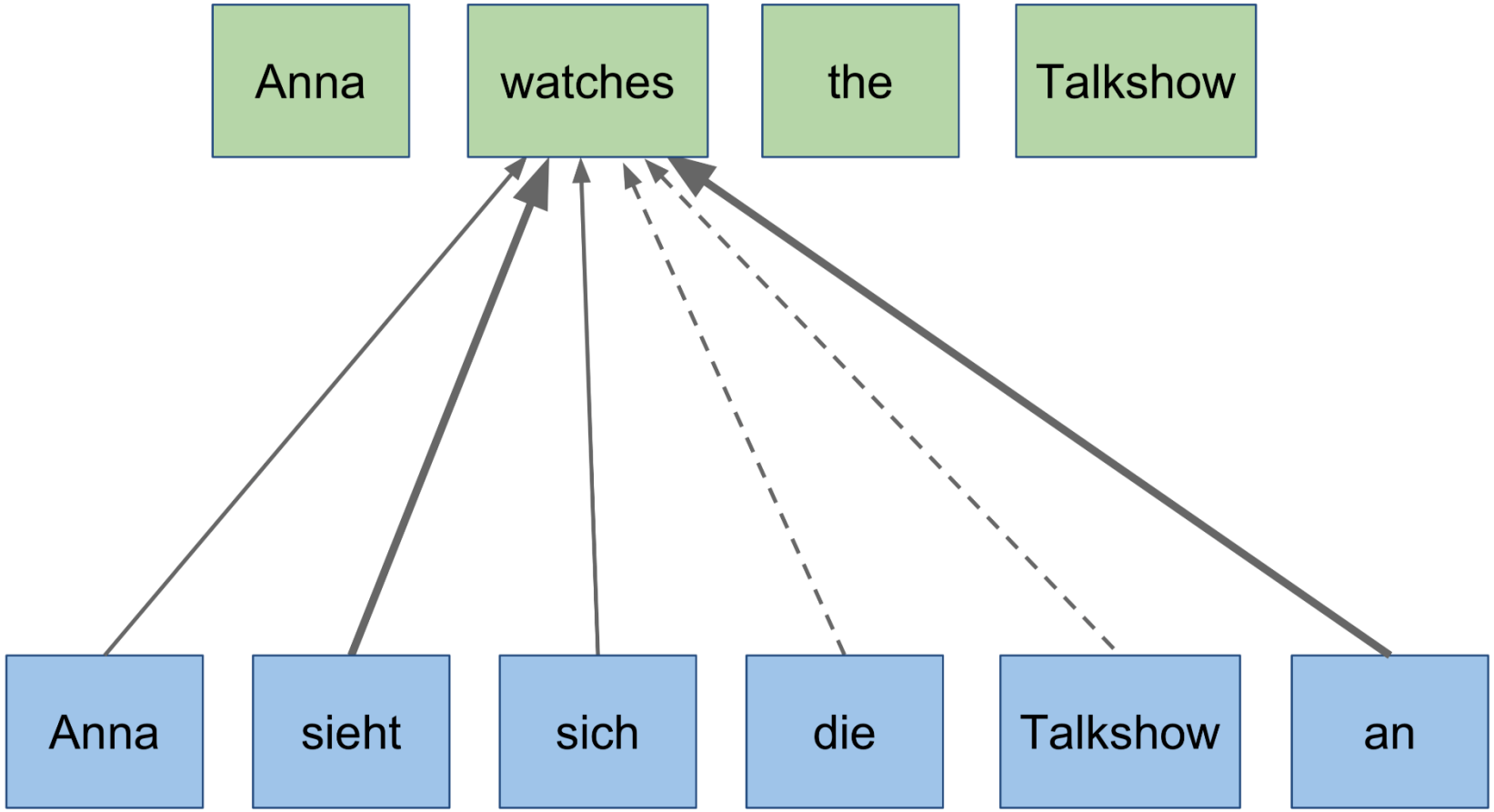
Attention Mechanism

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



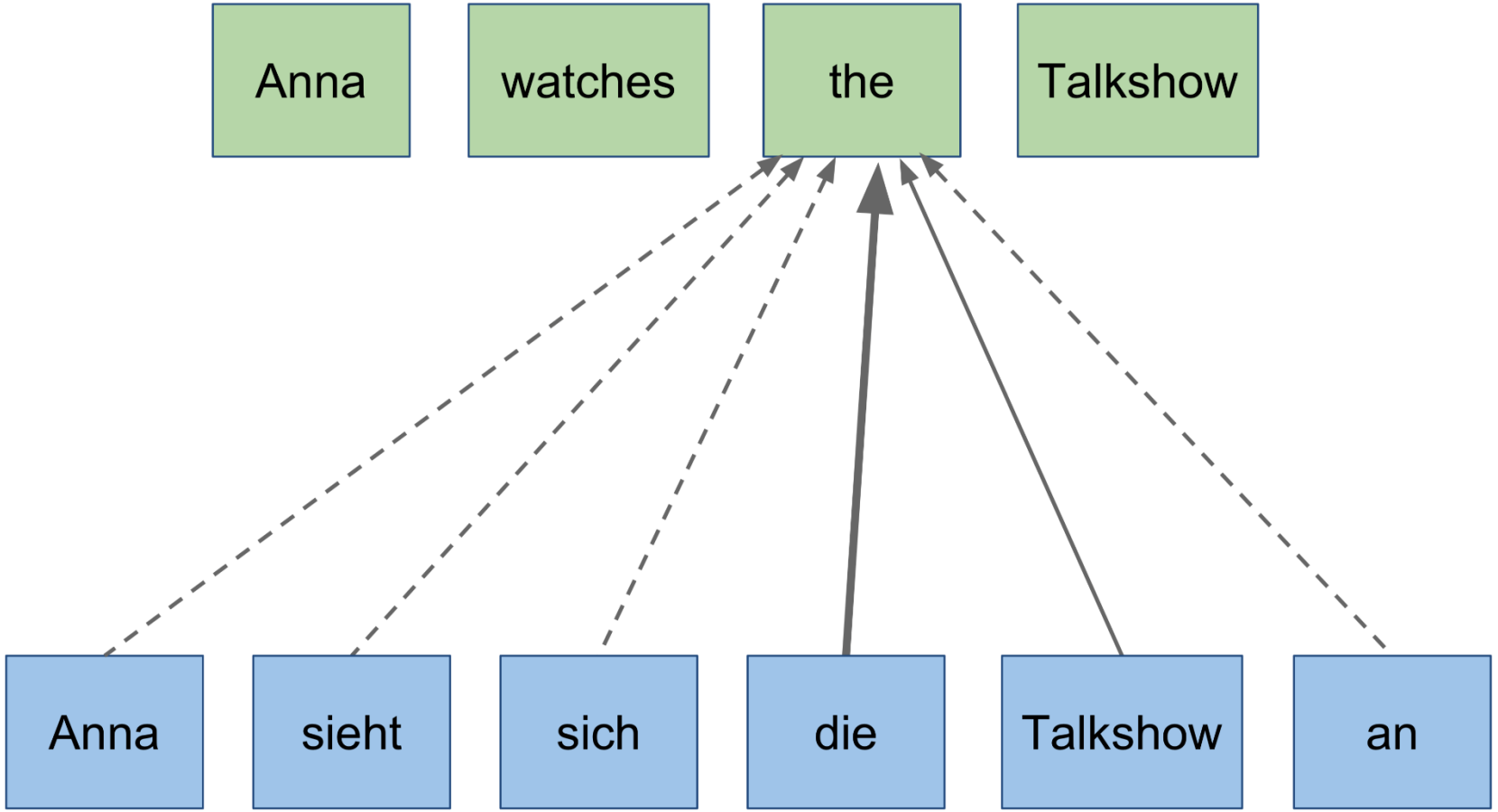
Attention Mechanism

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



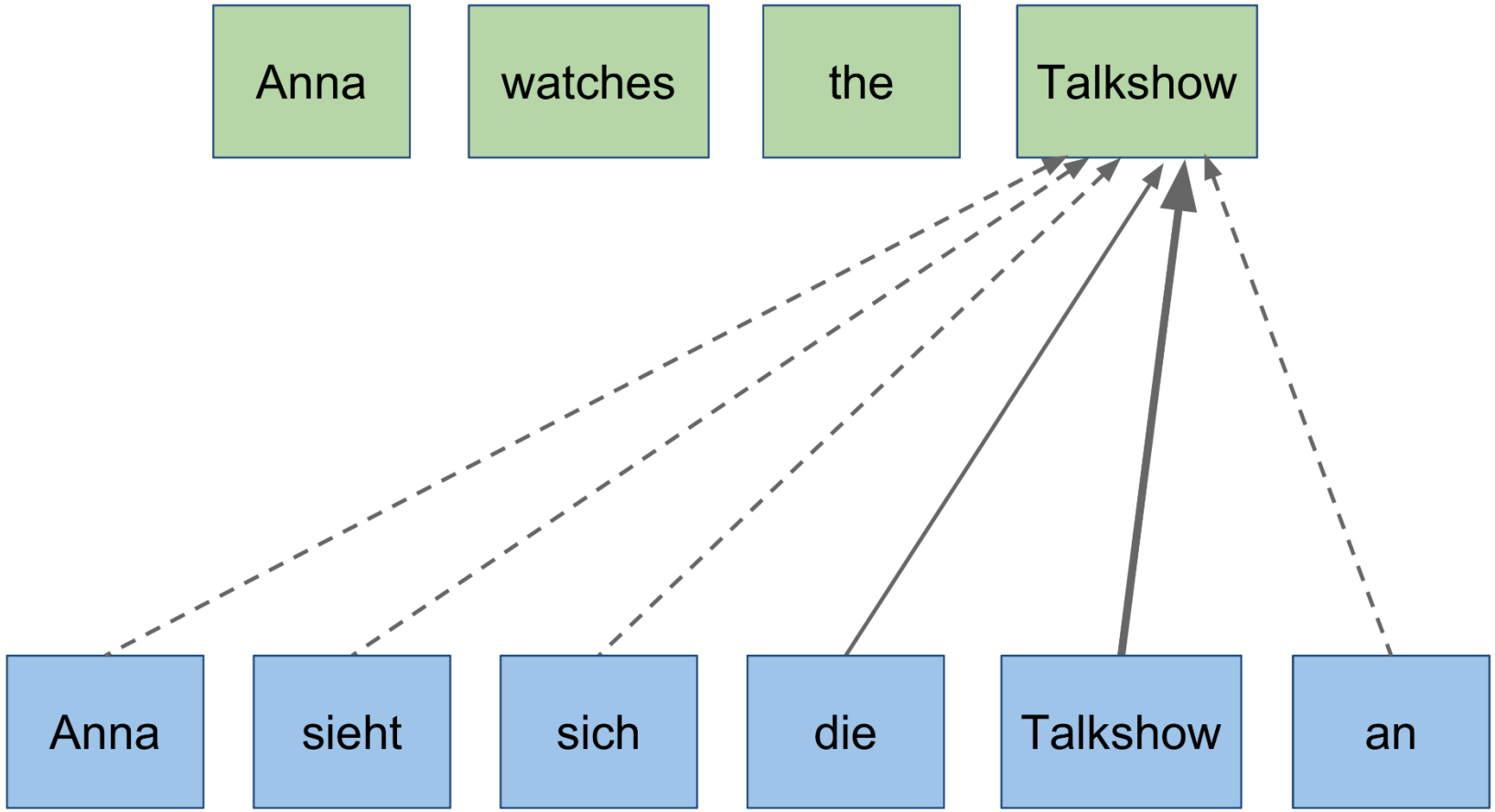
Attention Mechanism

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



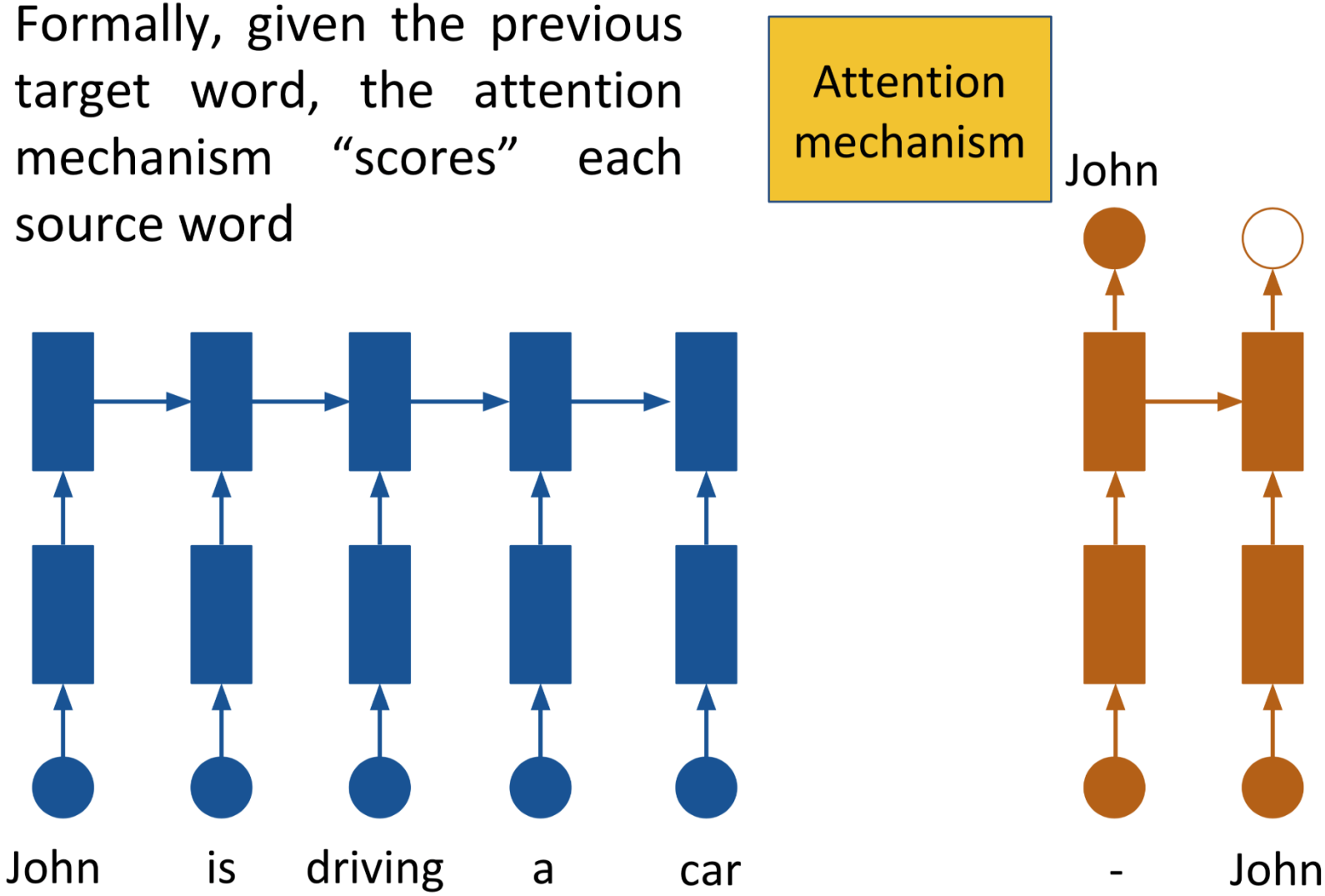
Attention Mechanism

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

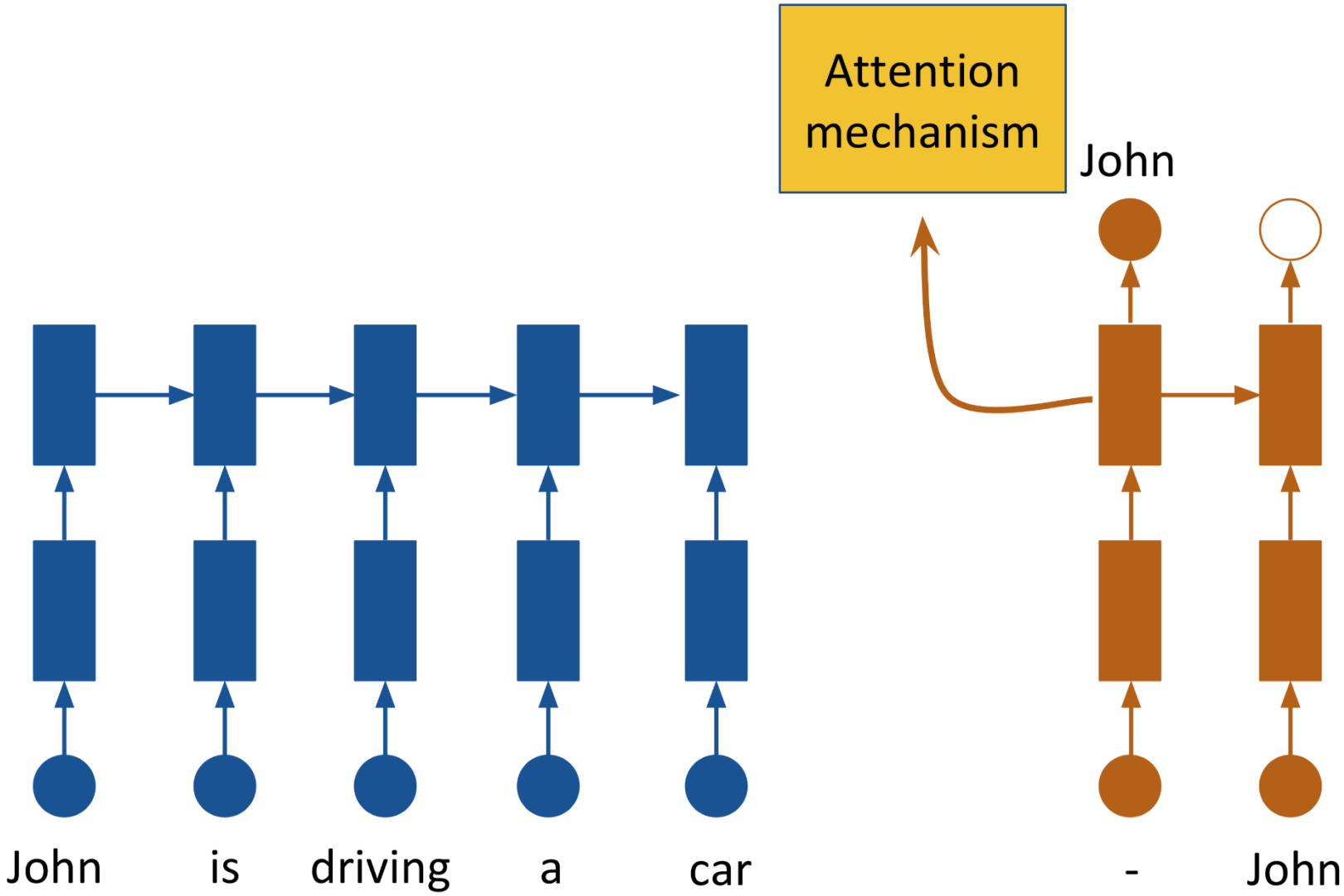


Attention Mechanism

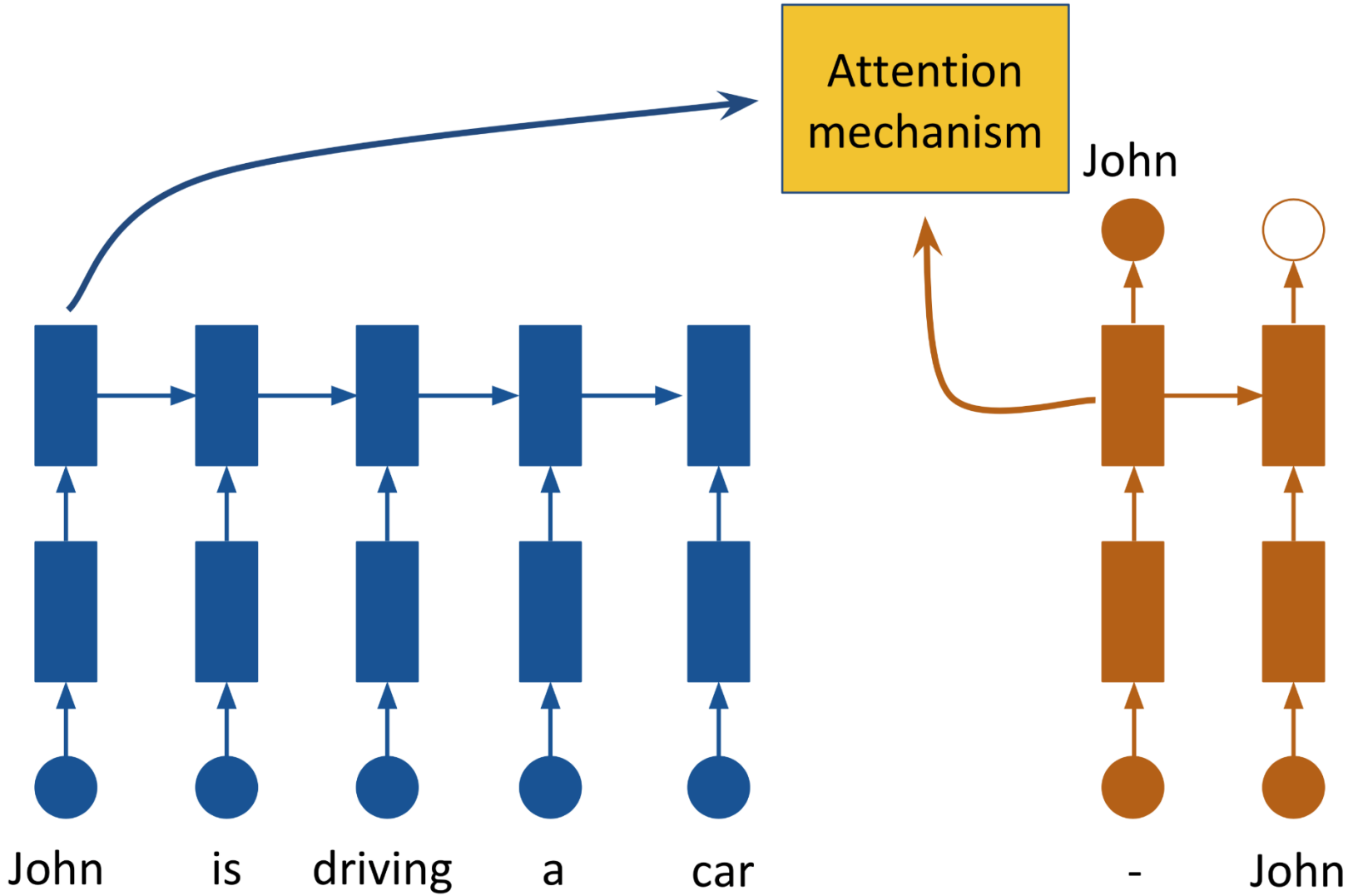
Formally, given the previous target word, the attention mechanism "scores" each source word



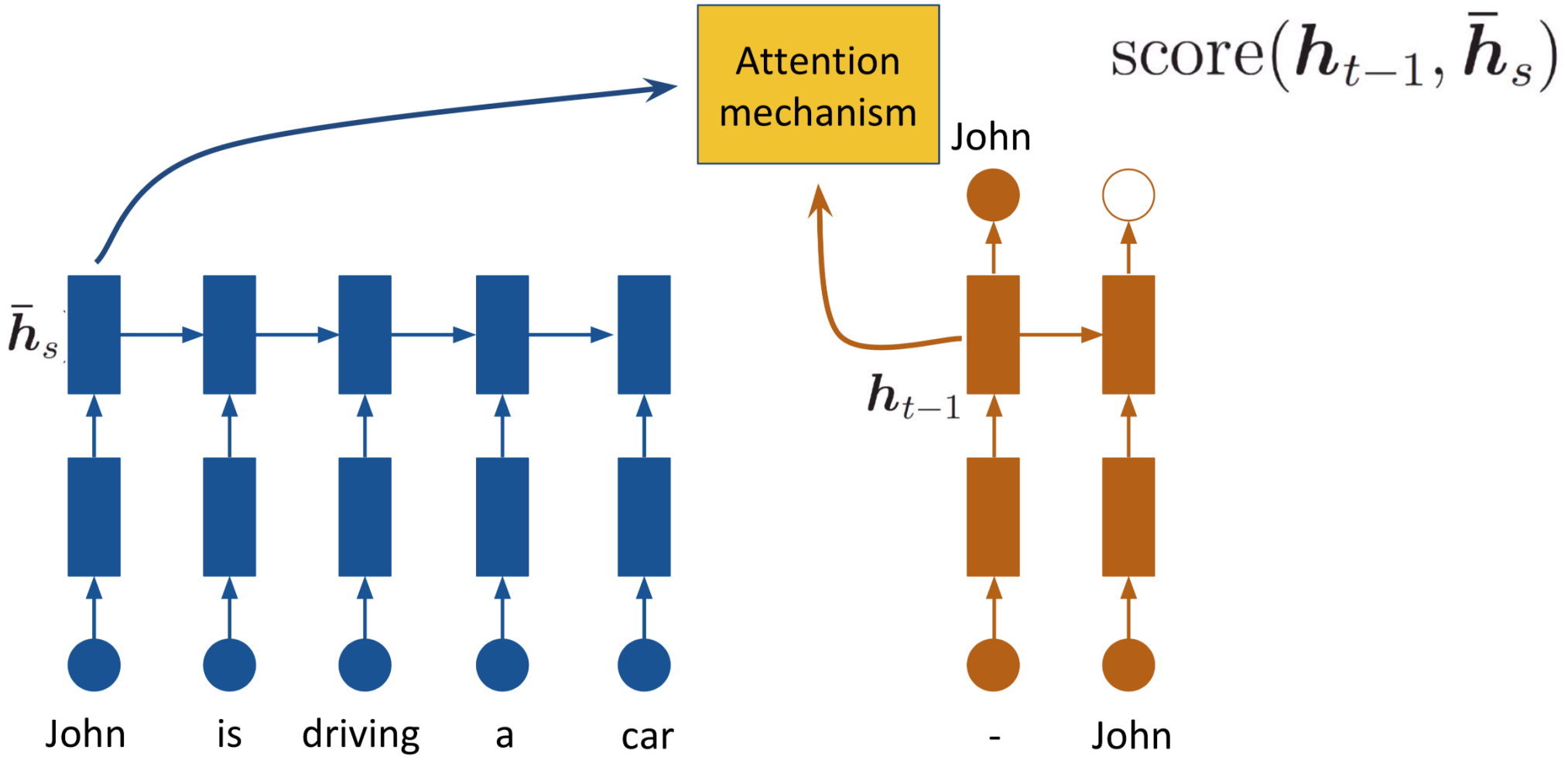
Attention Mechanism



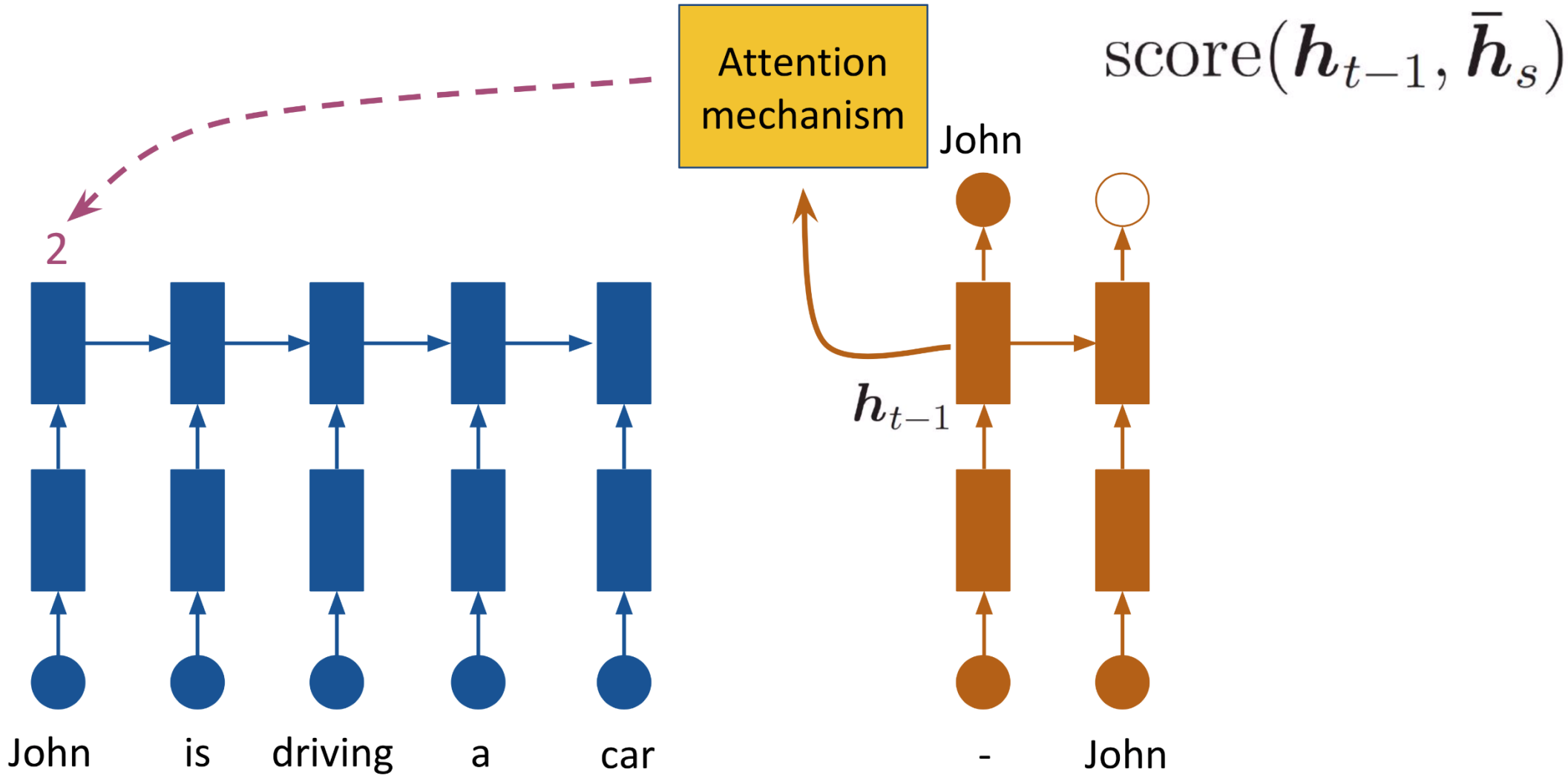
Attention Mechanism



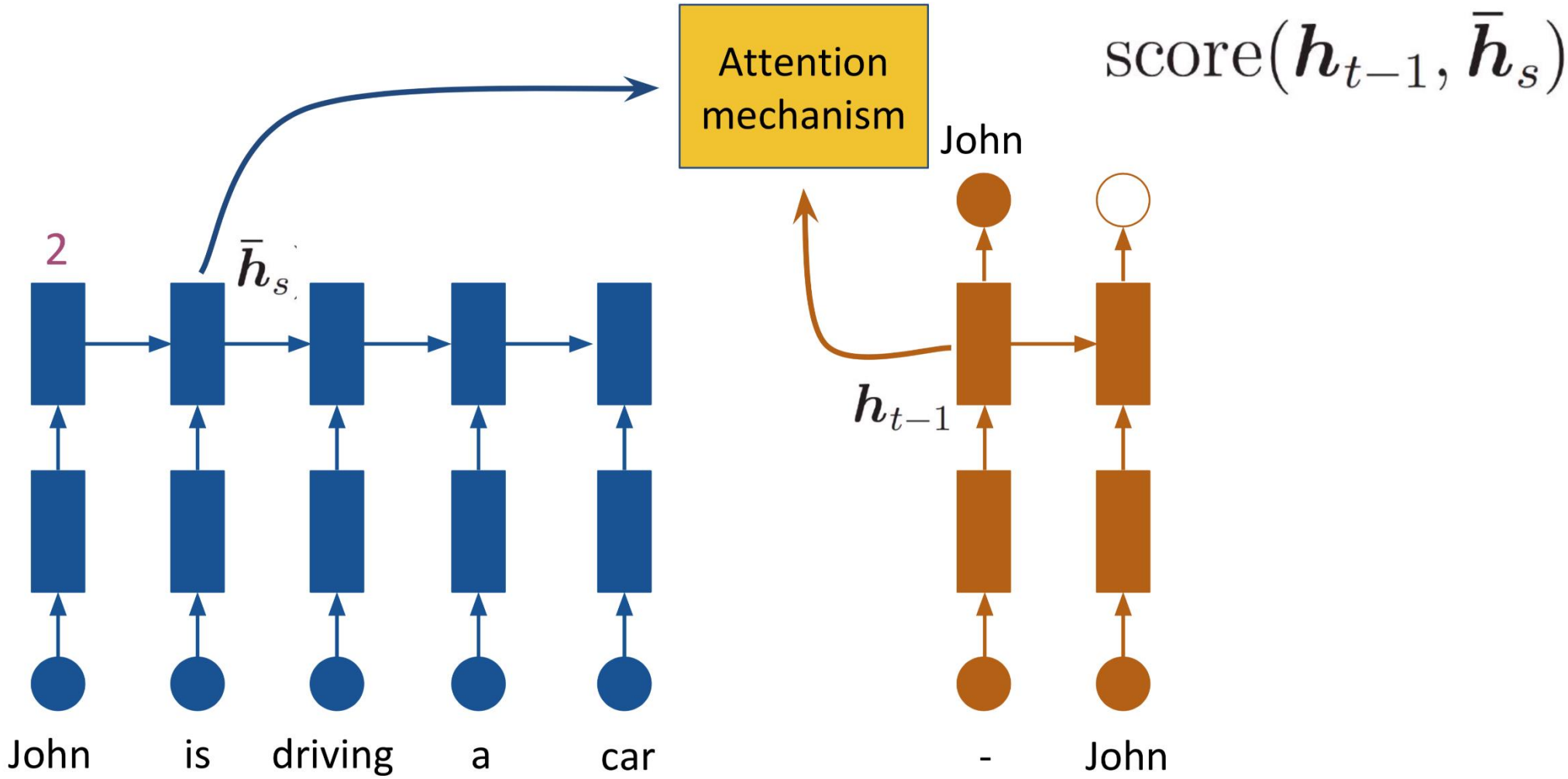
Attention Mechanism



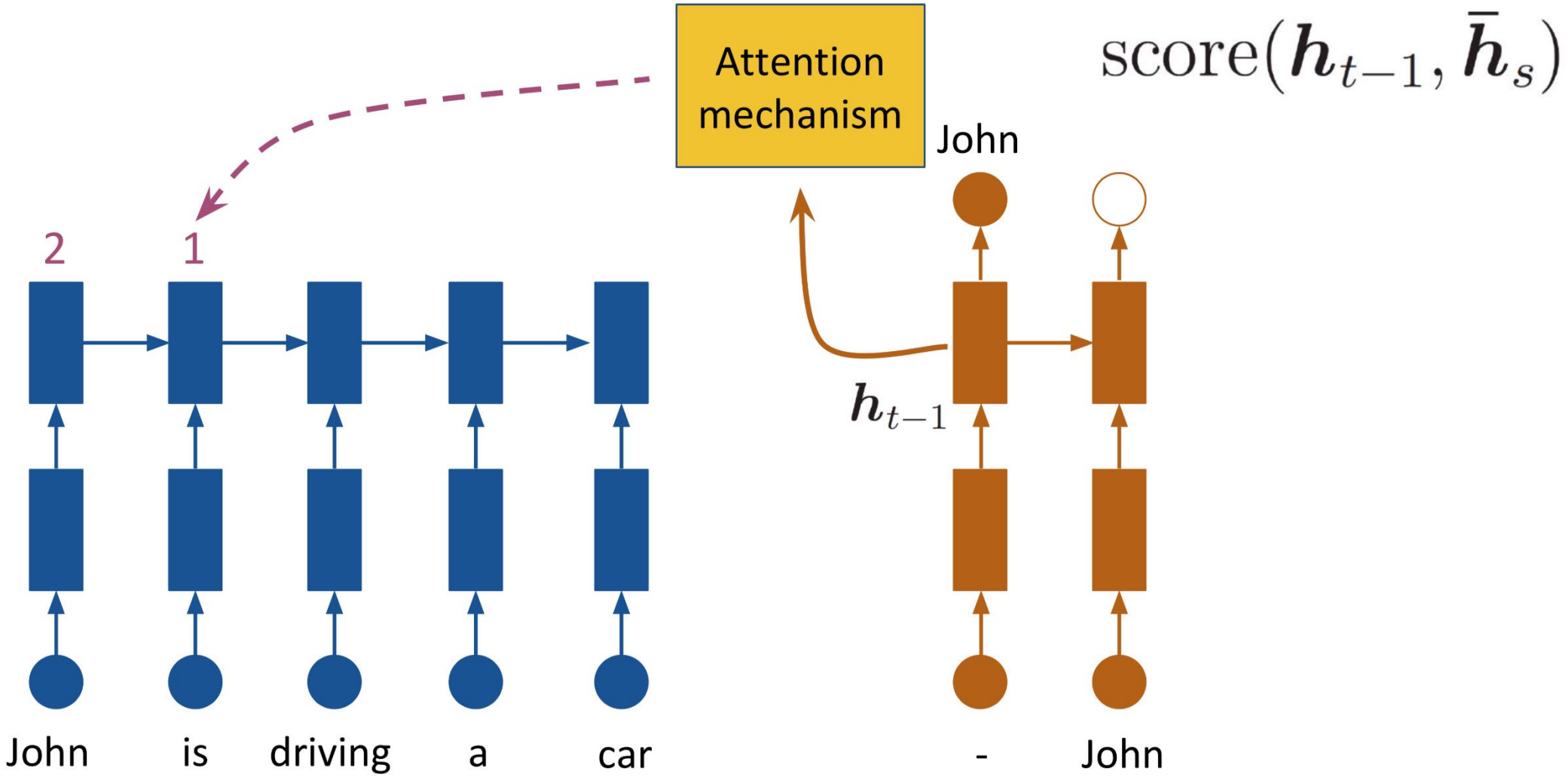
Attention Mechanism



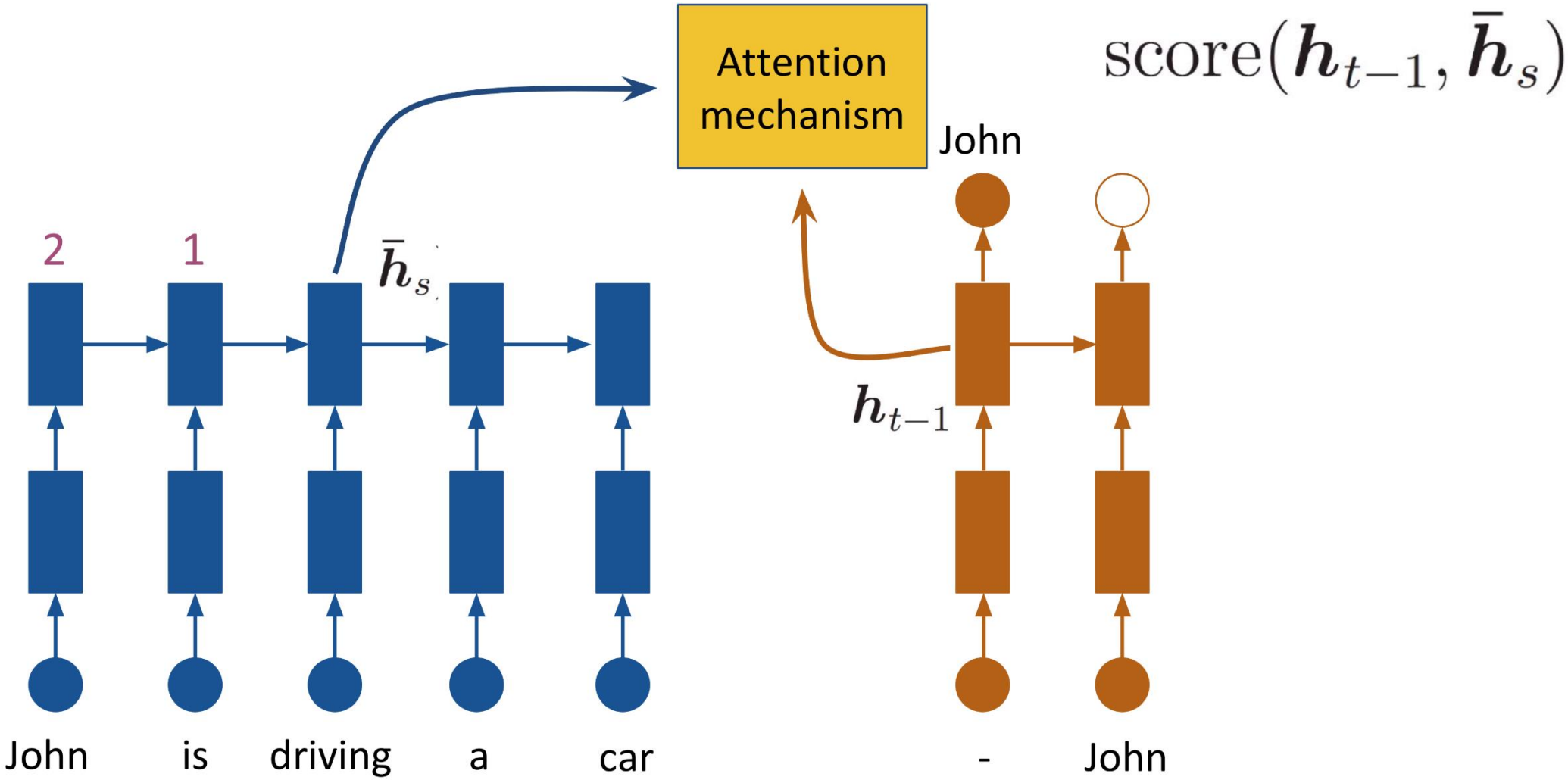
Attention Mechanism



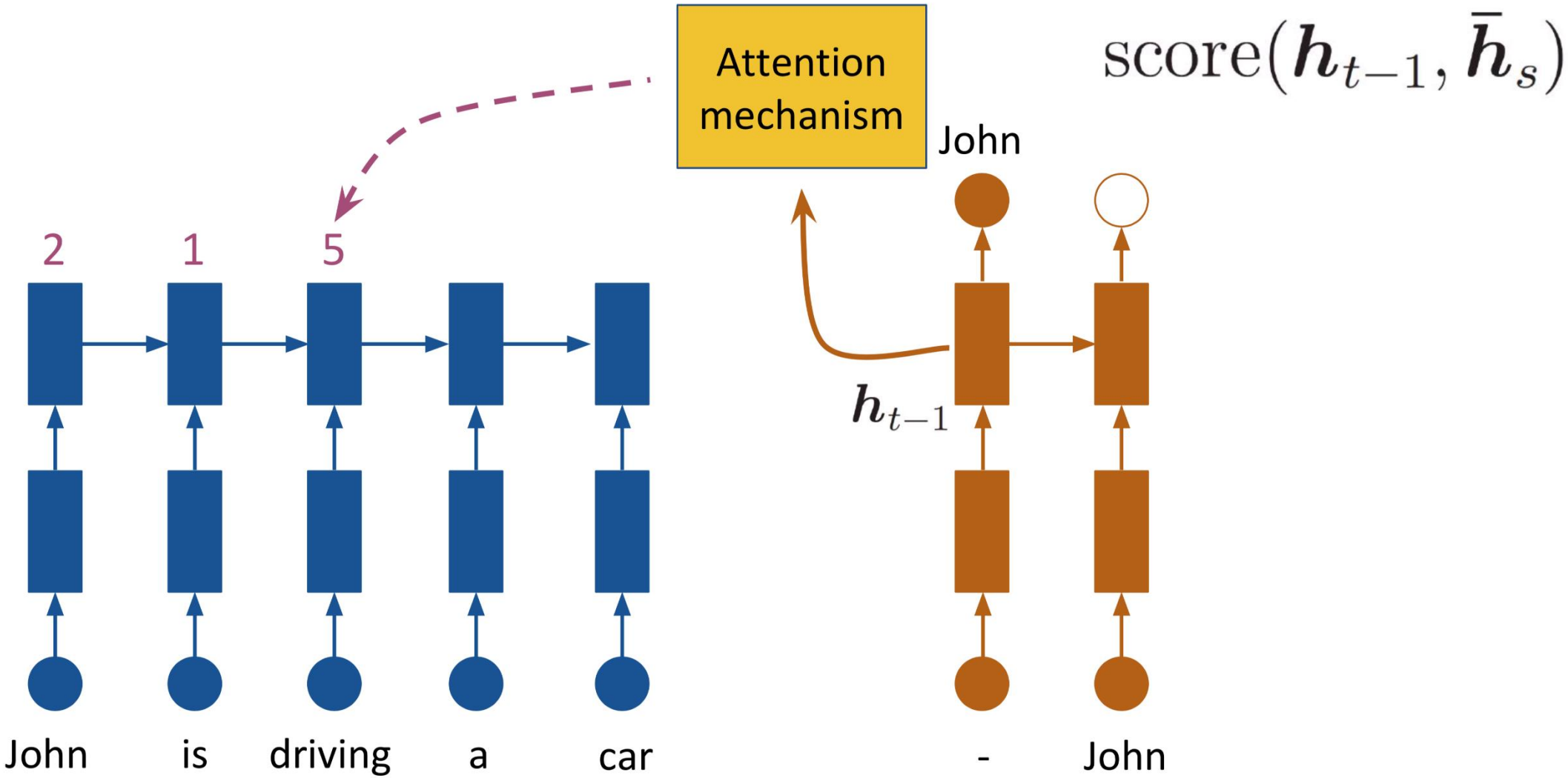
Attention Mechanism



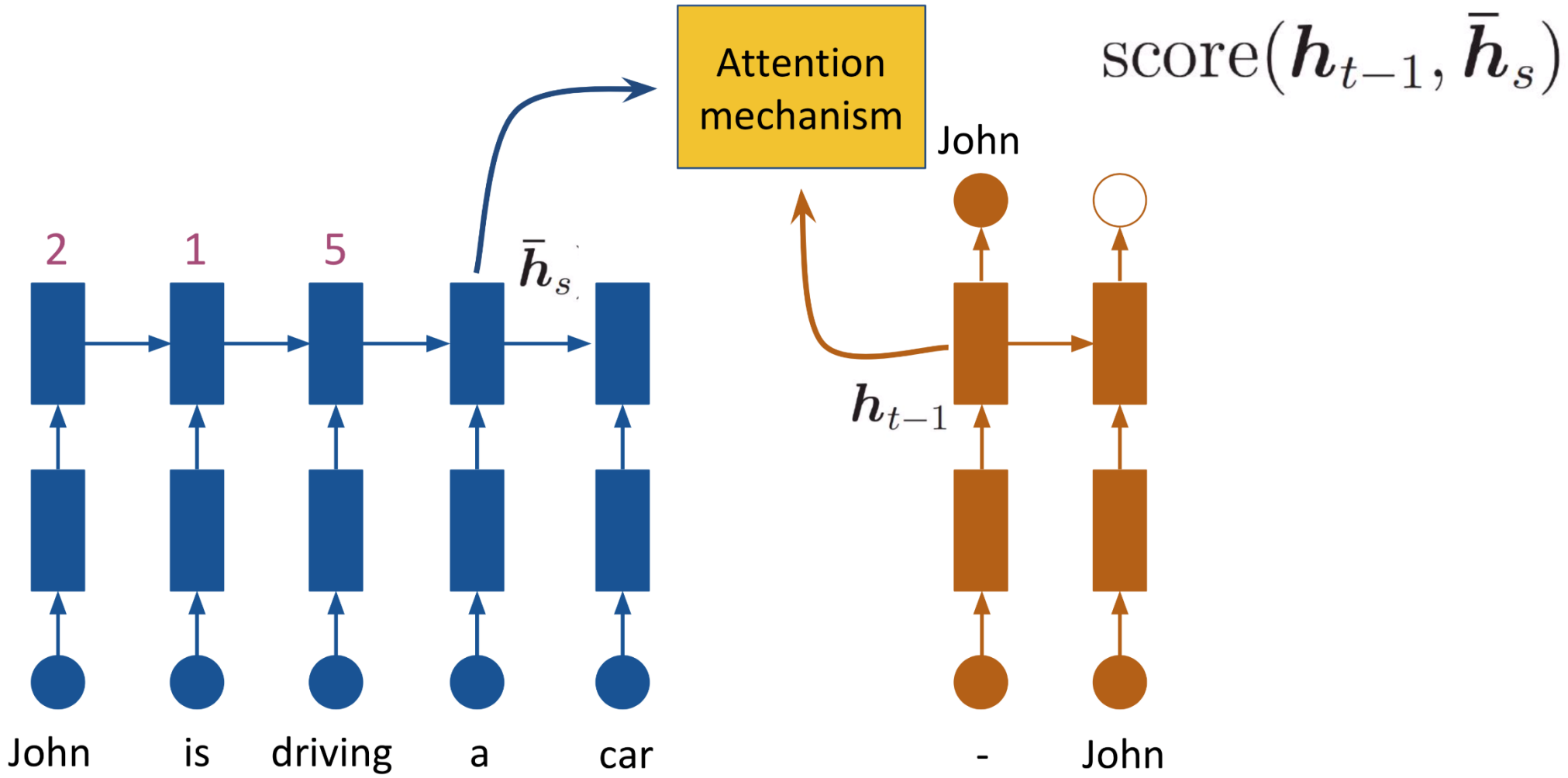
Attention Mechanism



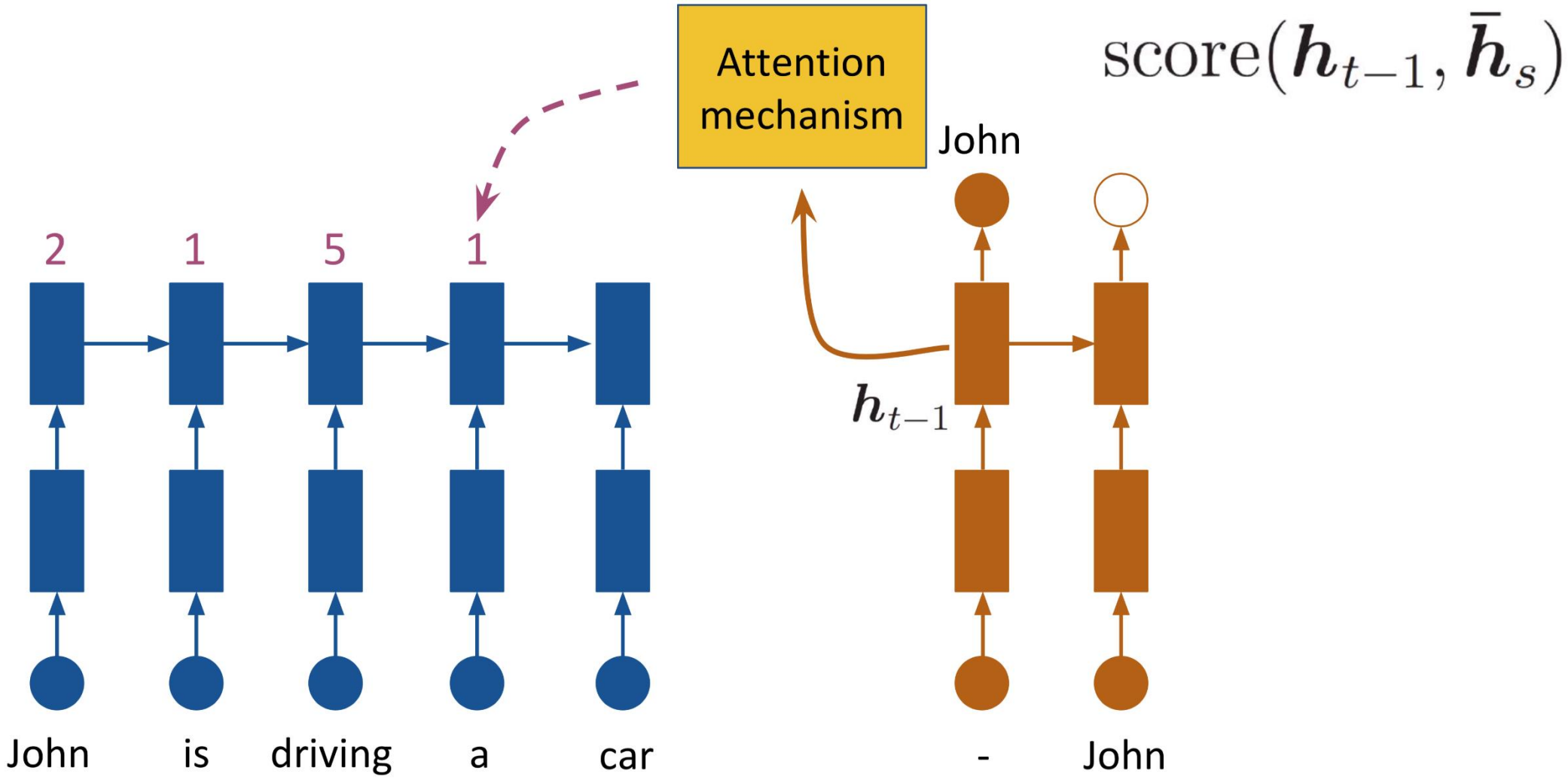
Attention Mechanism



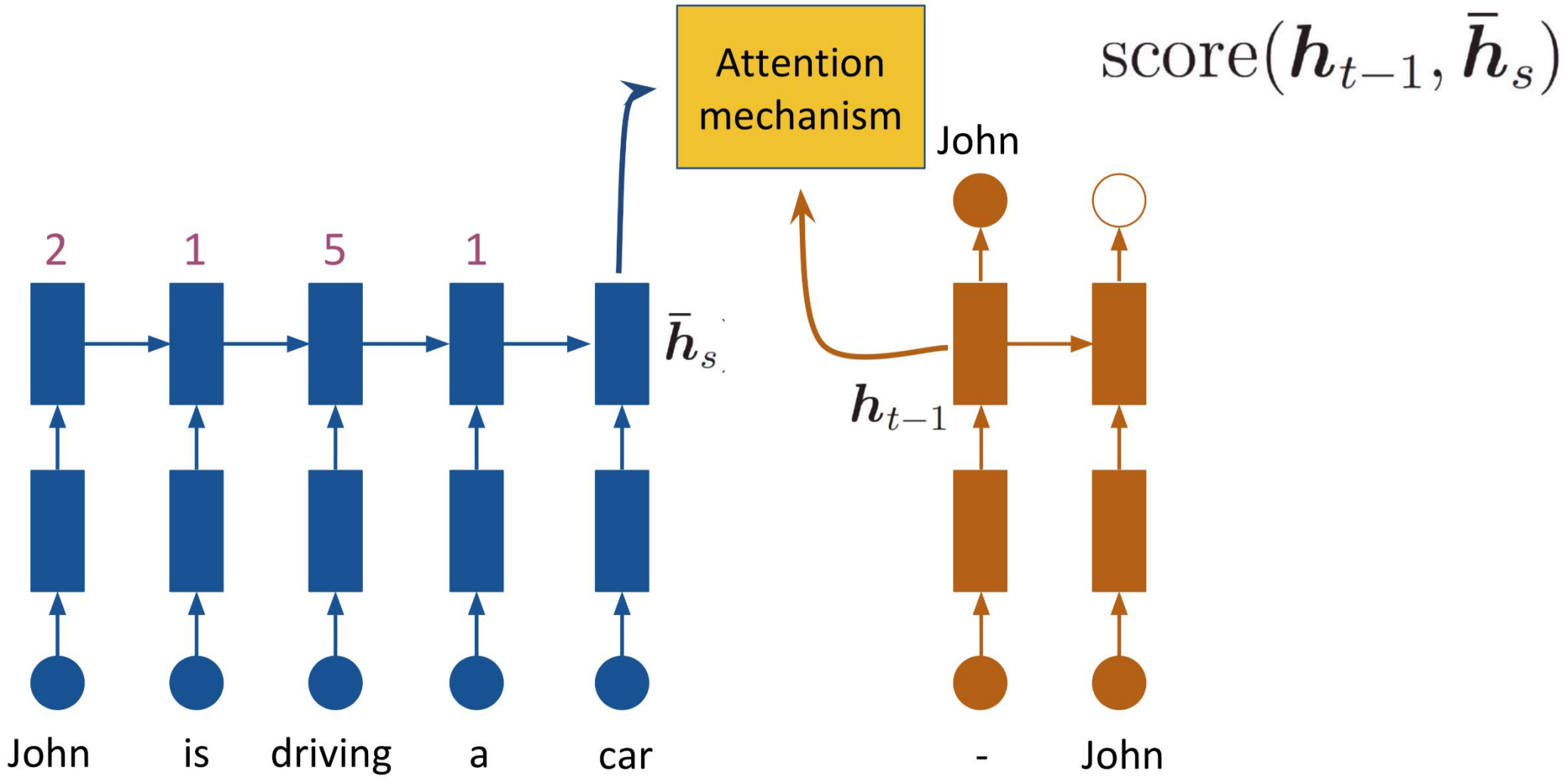
Attention Mechanism



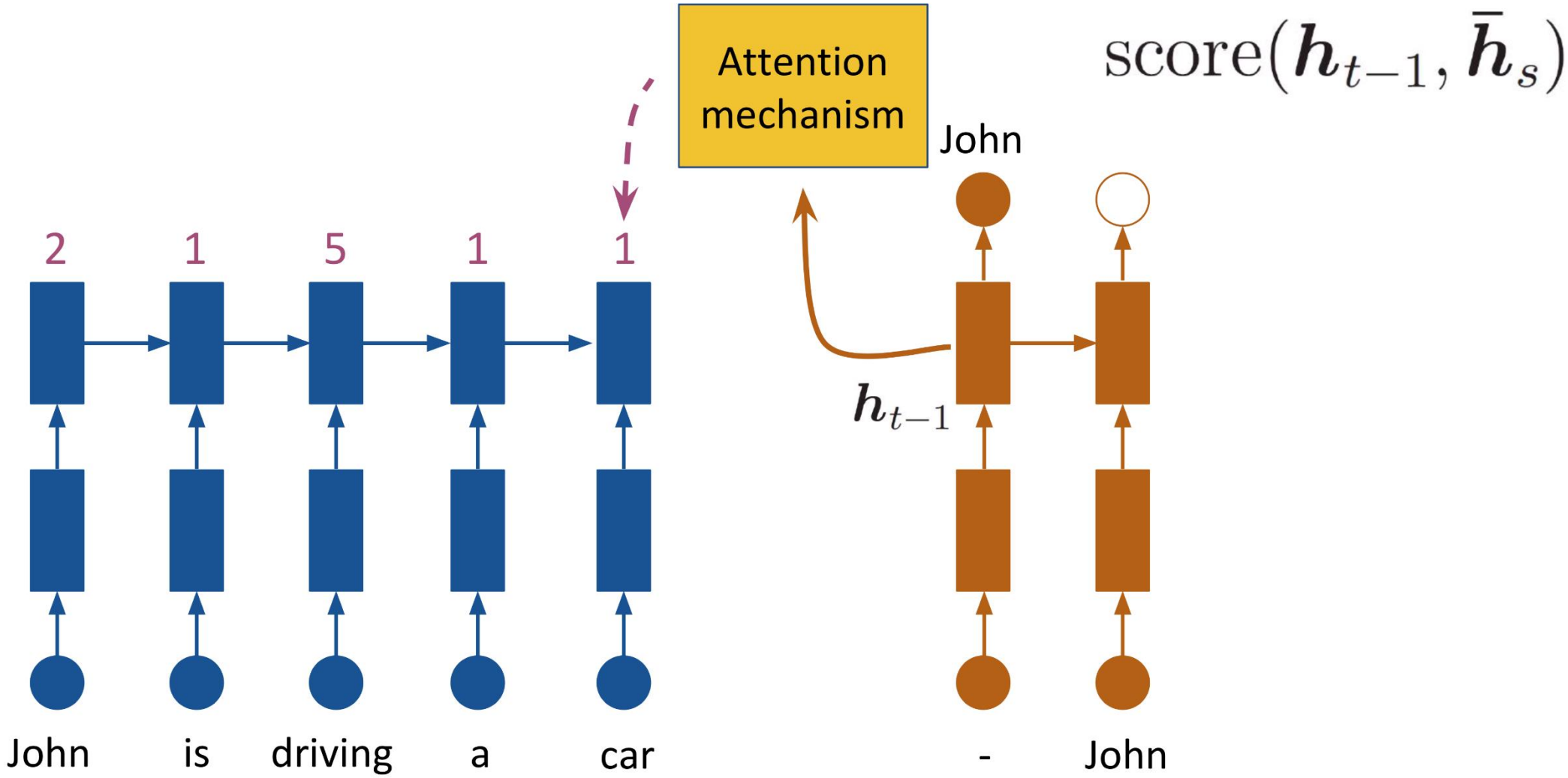
Attention Mechanism



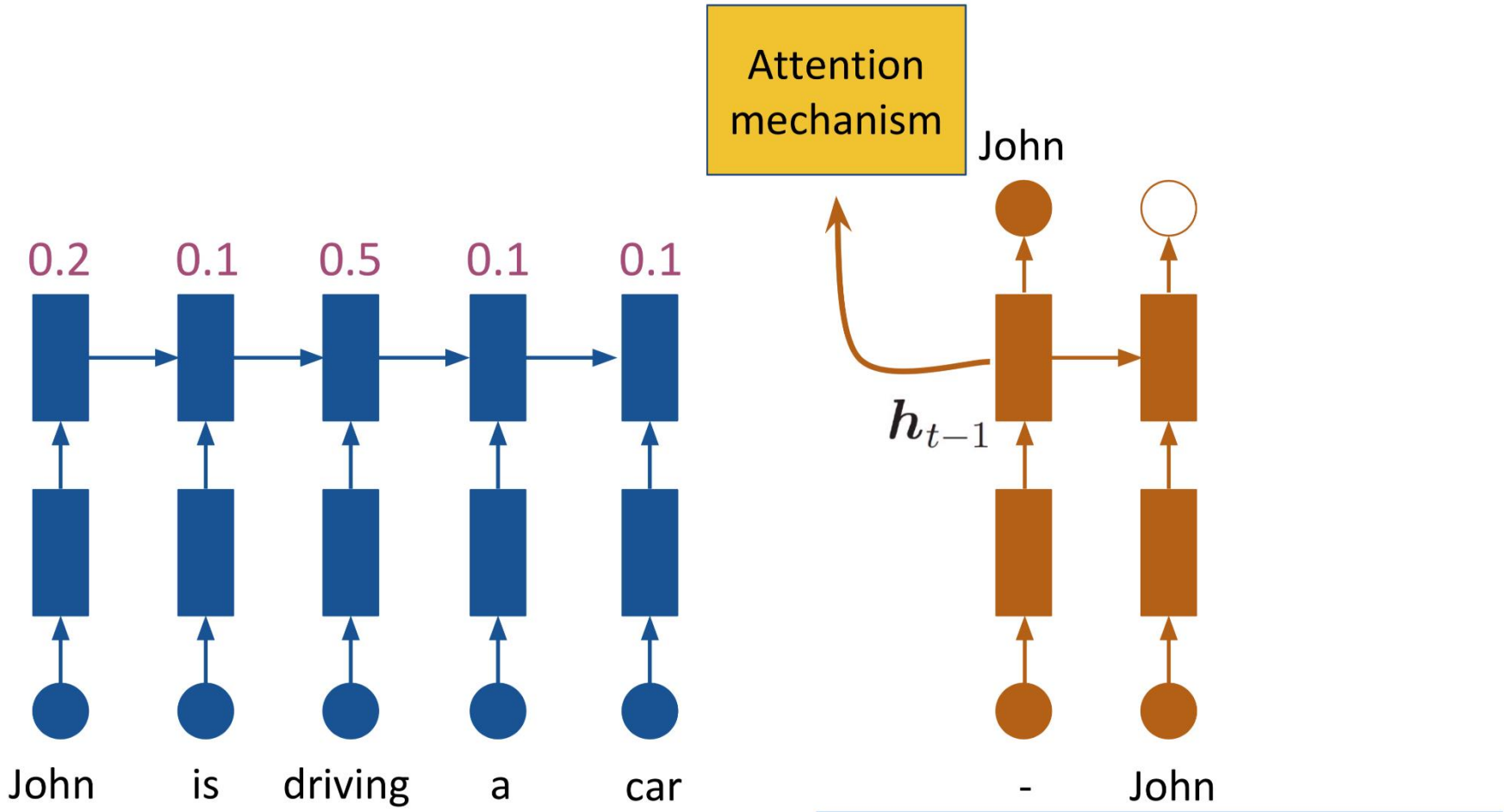
Attention Mechanism



Attention Mechanism

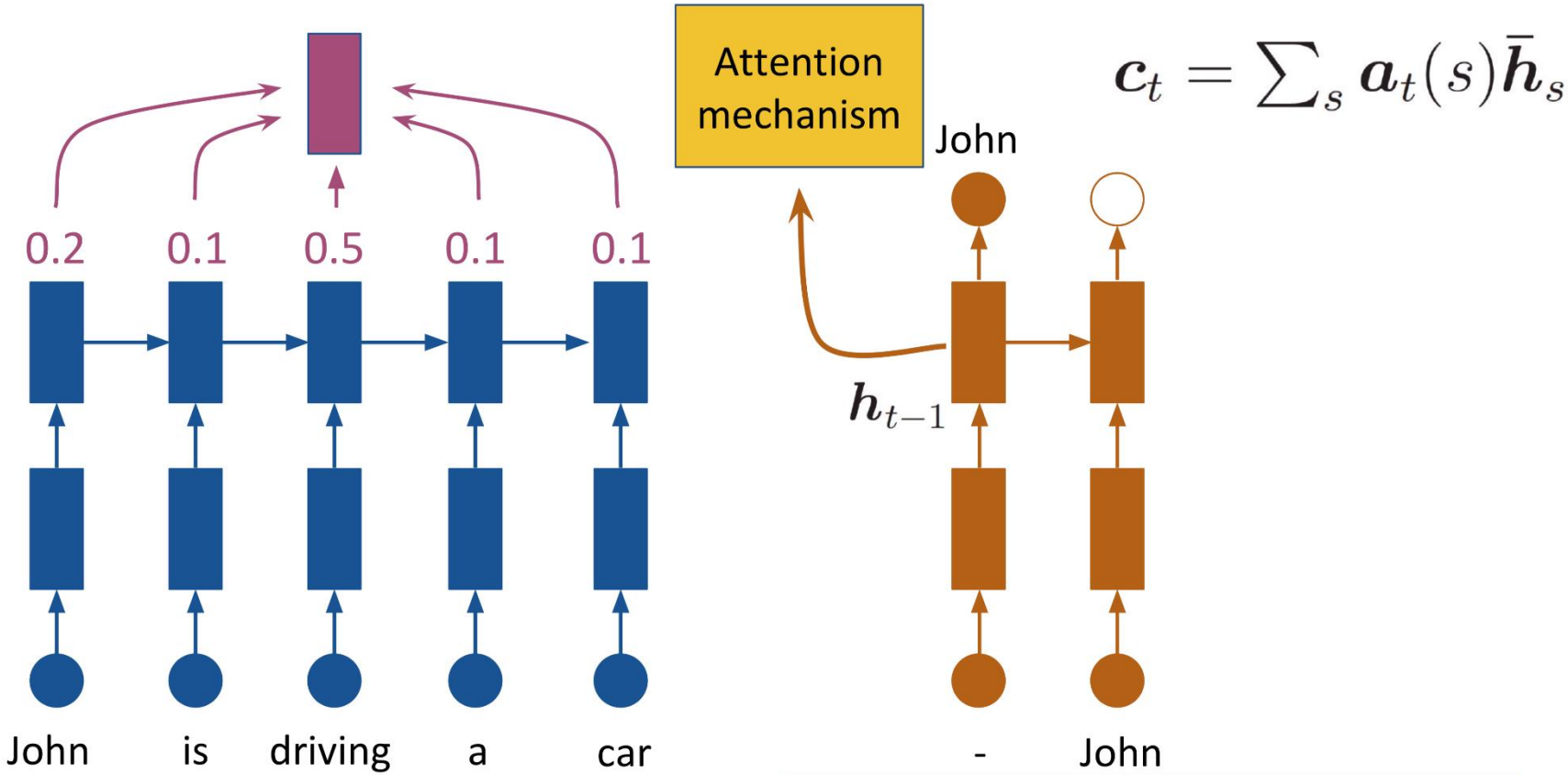


Attention Mechanism

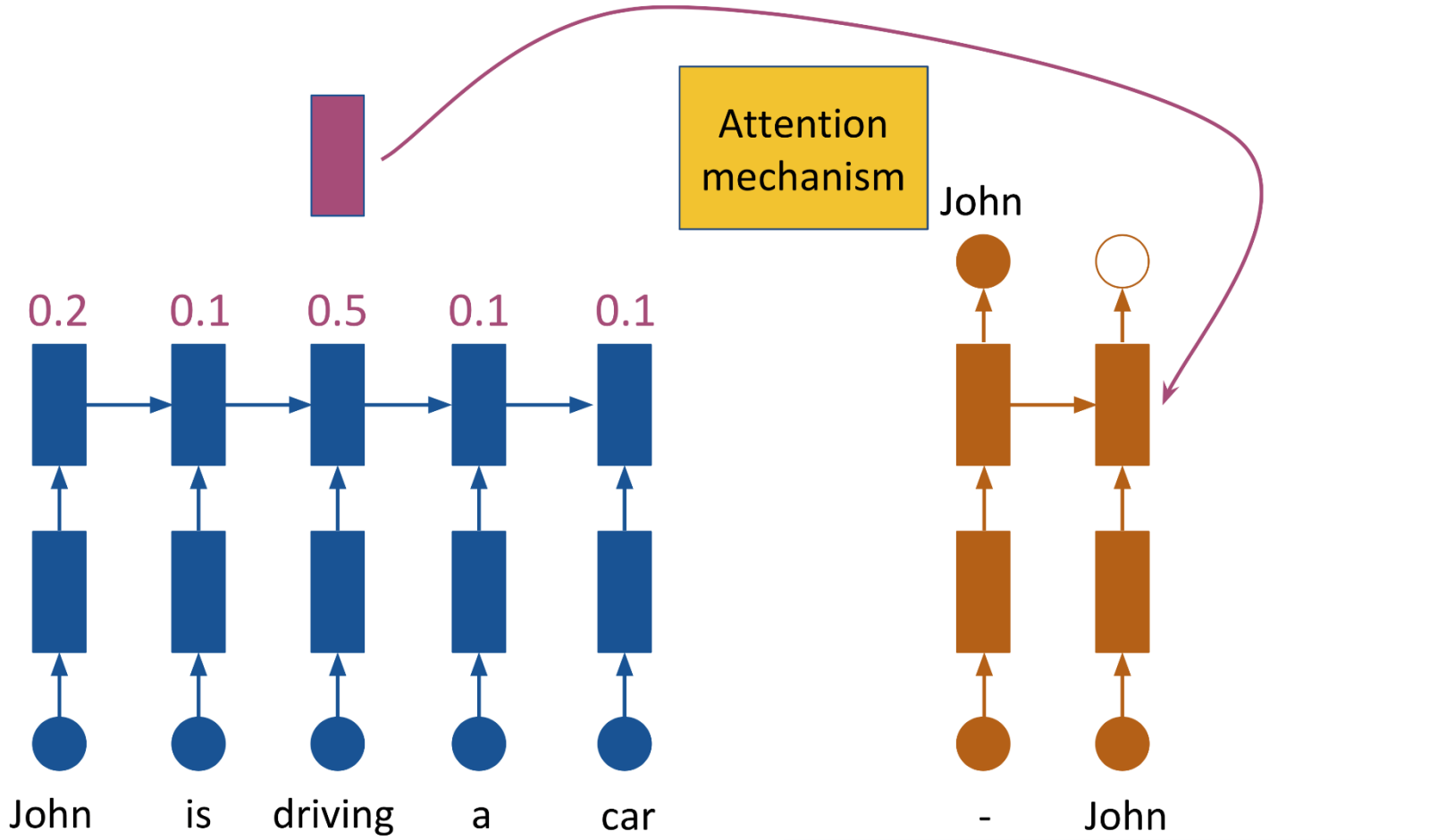


Use softmax to convert scores into probabilities

Attention Mechanism

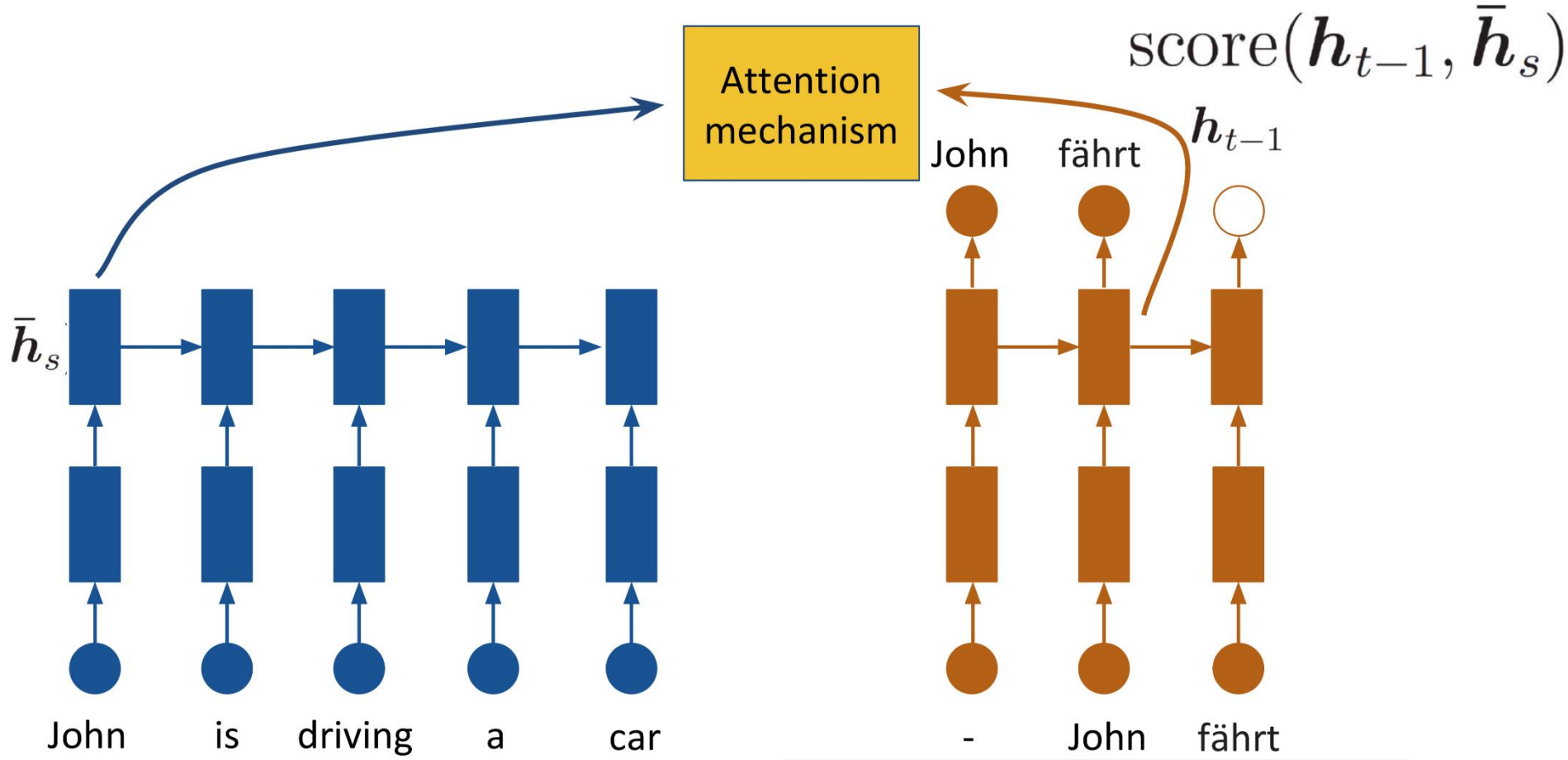


Attention Mechanism



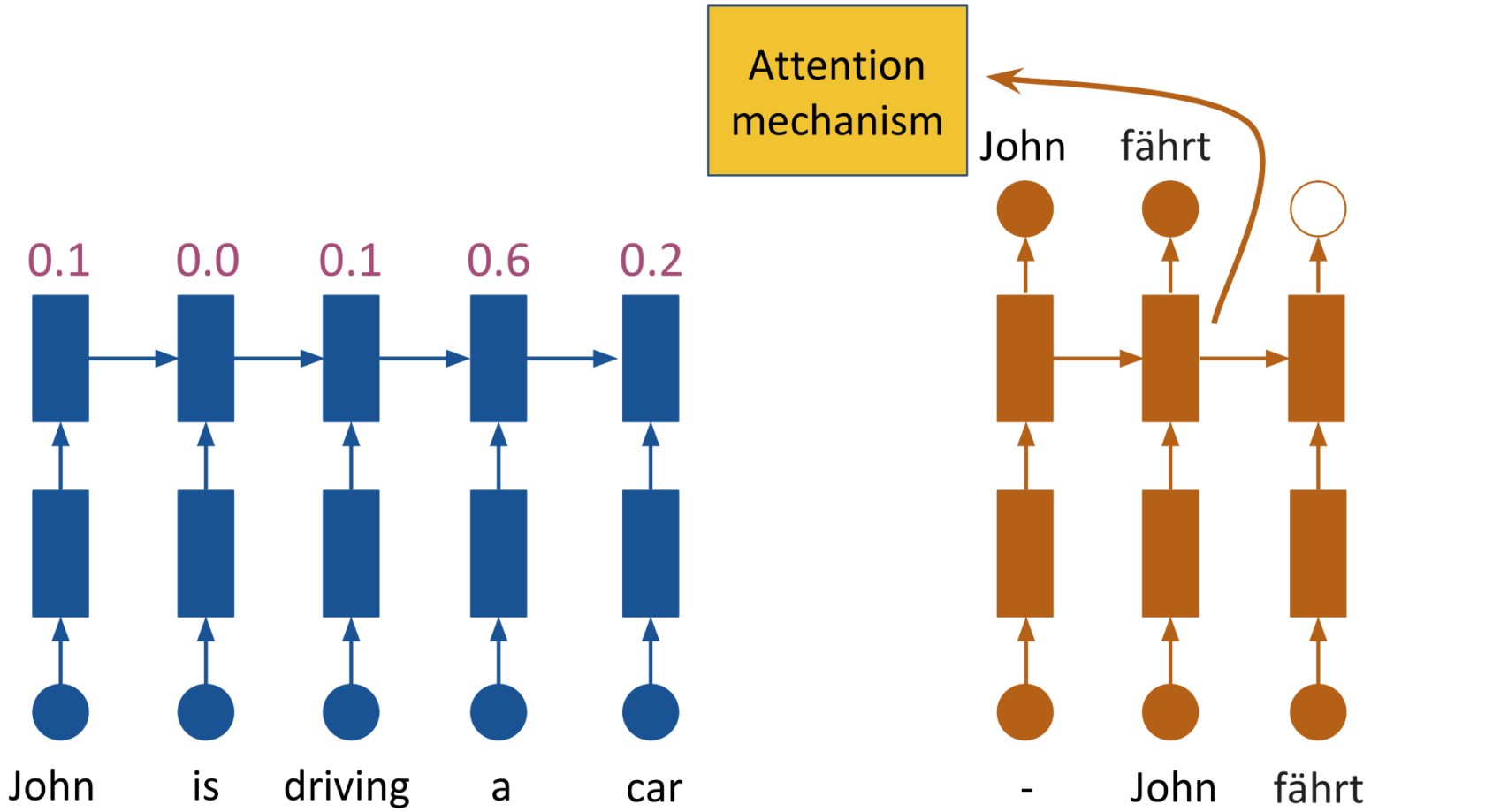
Use context vector to predict next word

Attention Mechanism

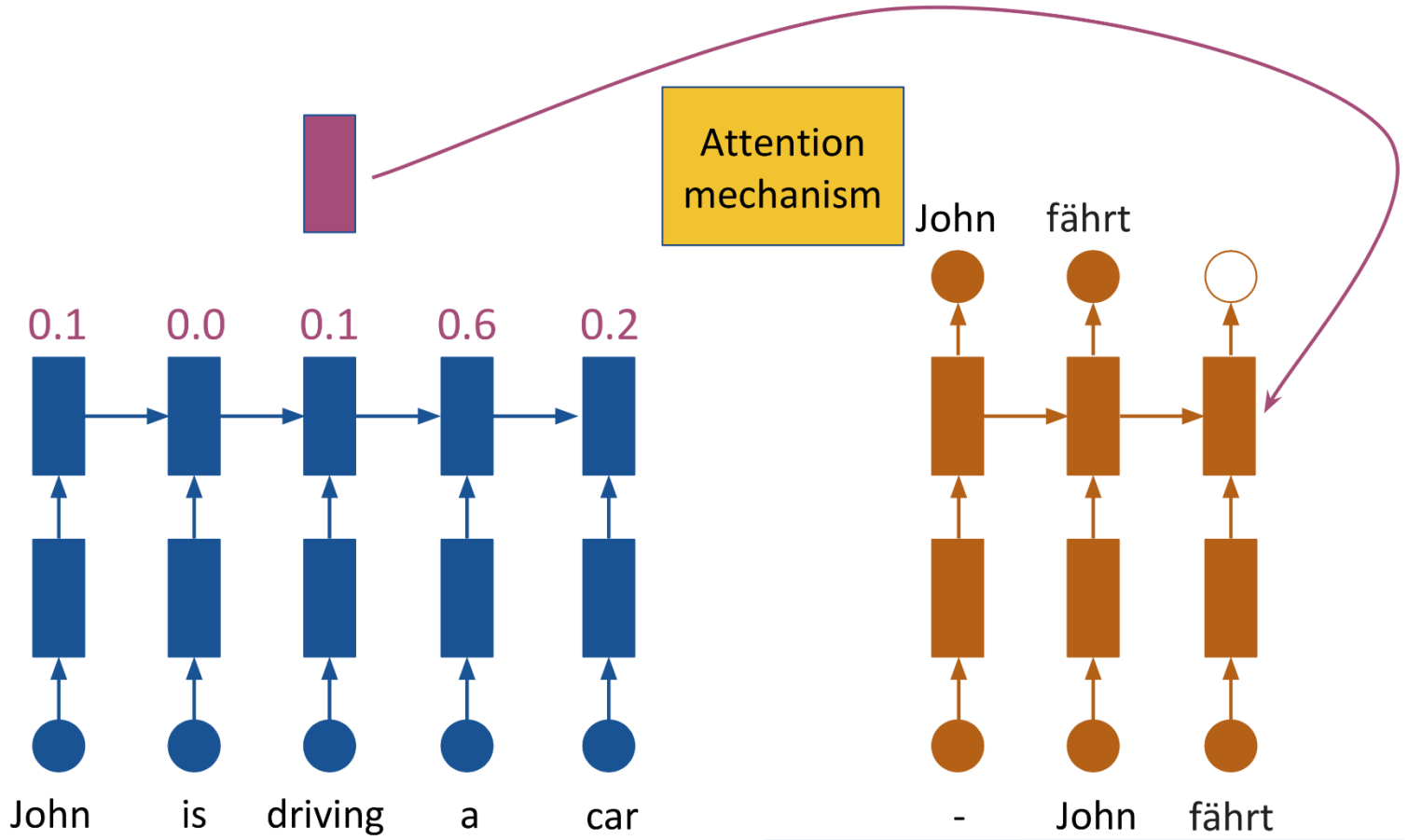


Similarly, get *source scores* for the next prediction

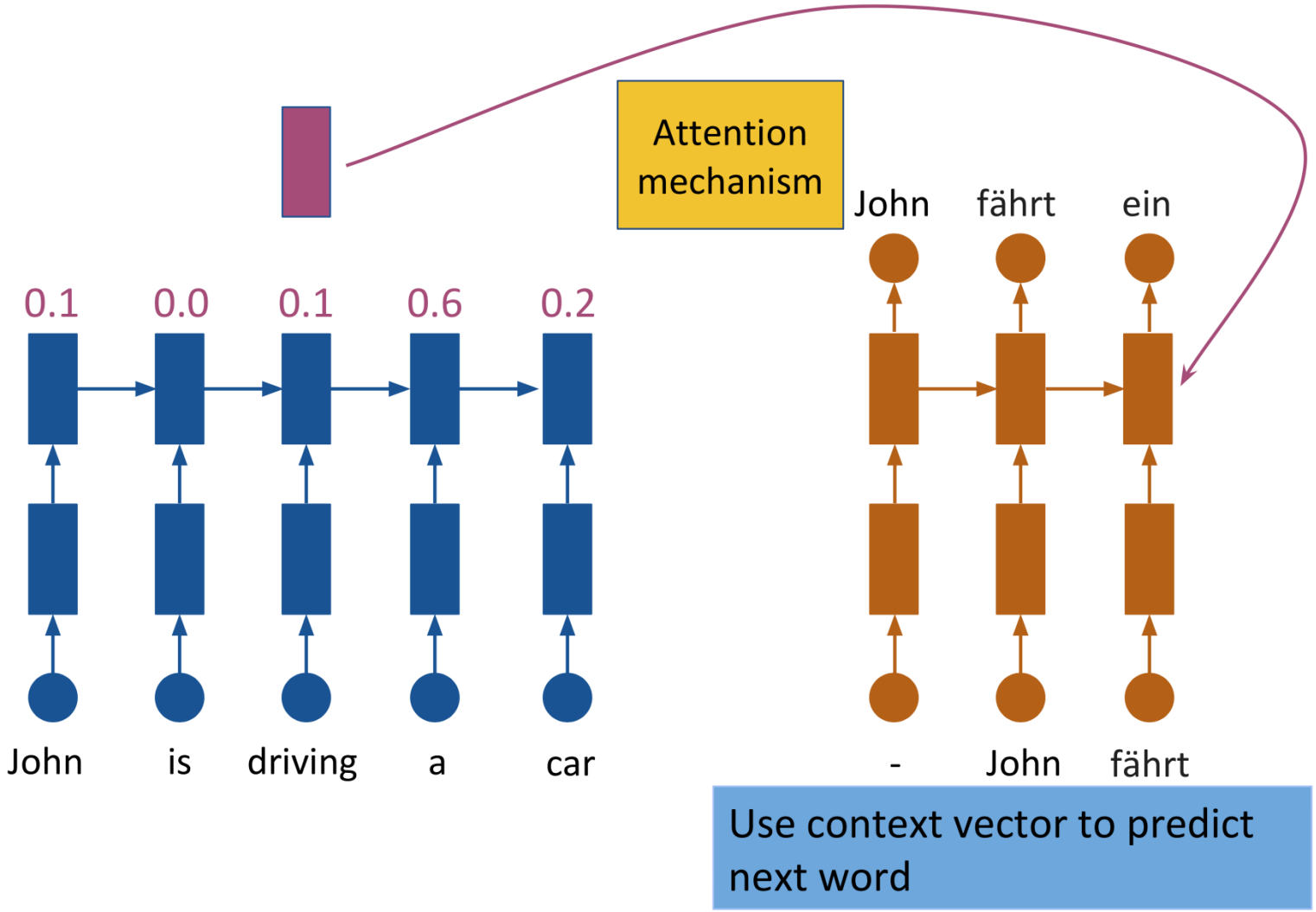
Attention Mechanism



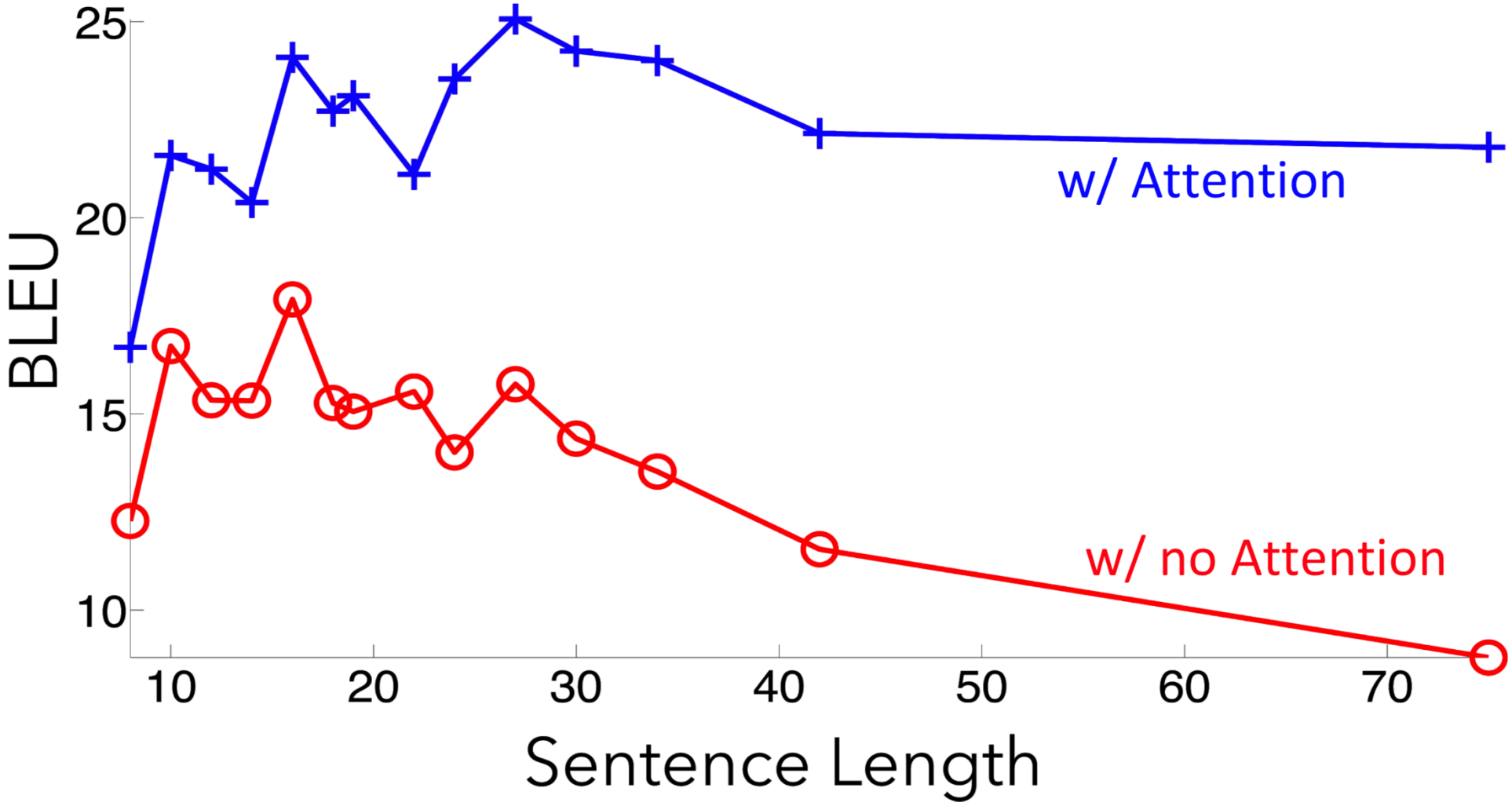
Attention Mechanism



Attention Mechanism



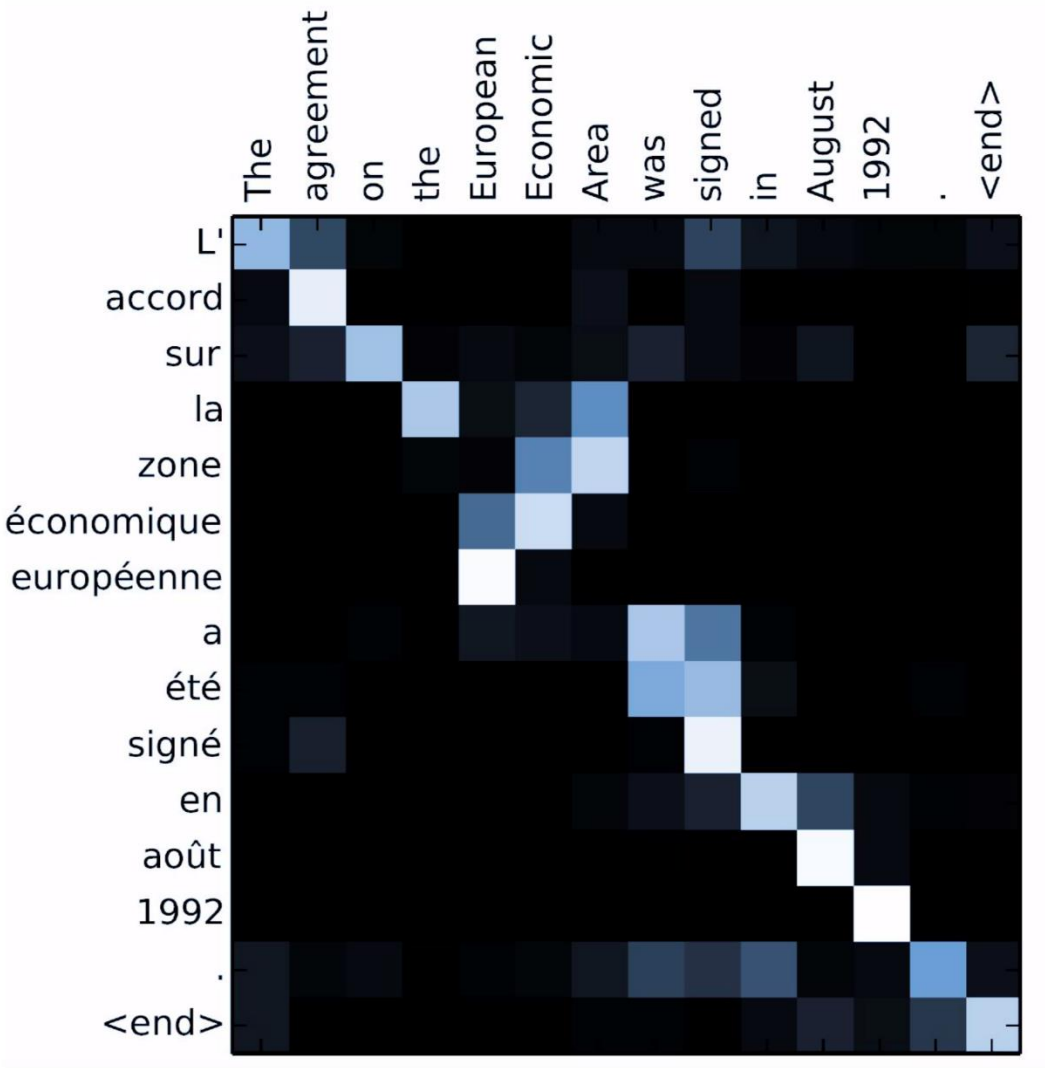
Attention Mechanism



Attention Mechanism Induces Word Alignments

Aligning words

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



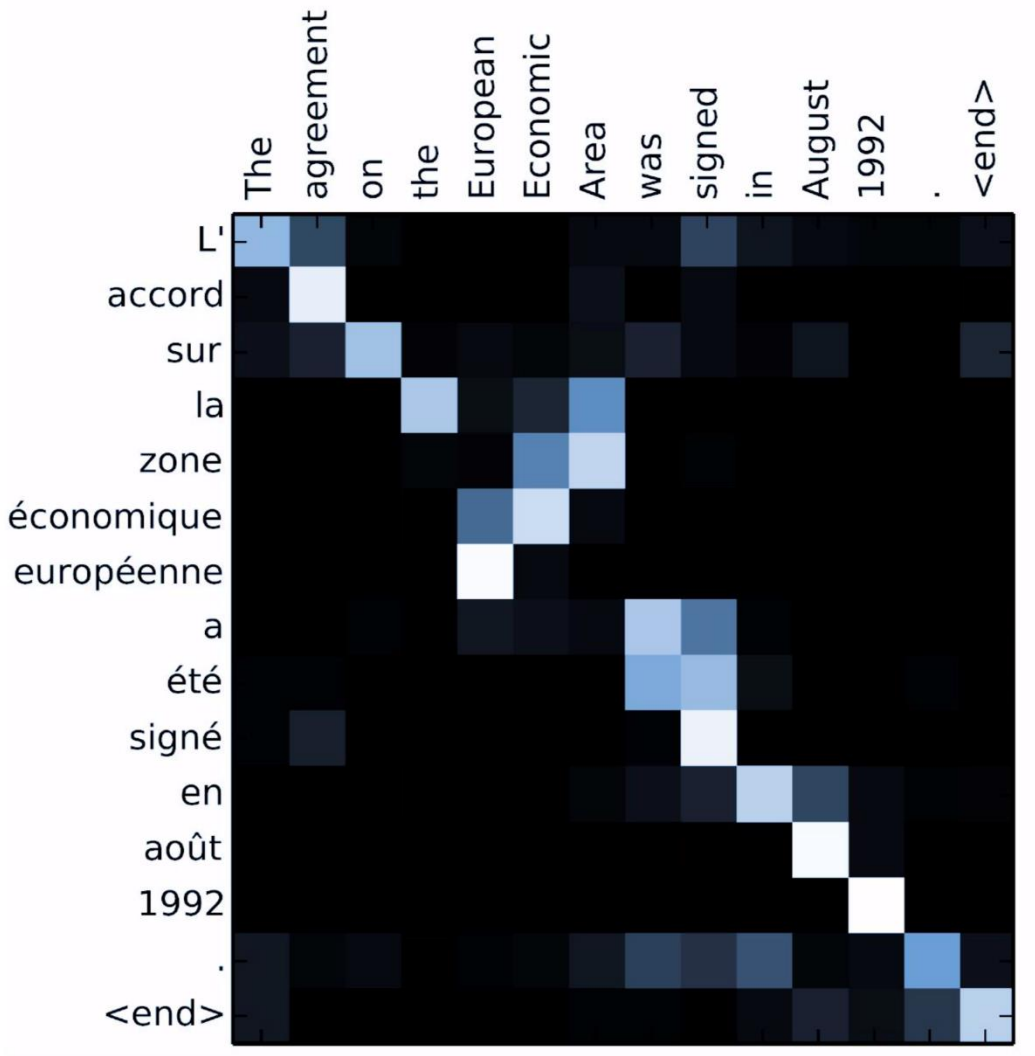
Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Translate and Align. ICLR'15

Attention Mechanism Induces Word Alignments

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

- Seq2Seq (aka encoder-decoder) transforms one sequence into another sequence
  - encodes a sequence into a summary vector
  - uses it to decode another sequence

- Attention mechanism
  - learns soft alignment between source and target words

Evaluating Machine Translation

BLEU (Papineni et al, 2002)

Reference (human) translation:

The U.S. island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself the Saudi Arabian Osama bin Laden and threatening a biological/chemical attack against public places such as the airport .

Machine translation:

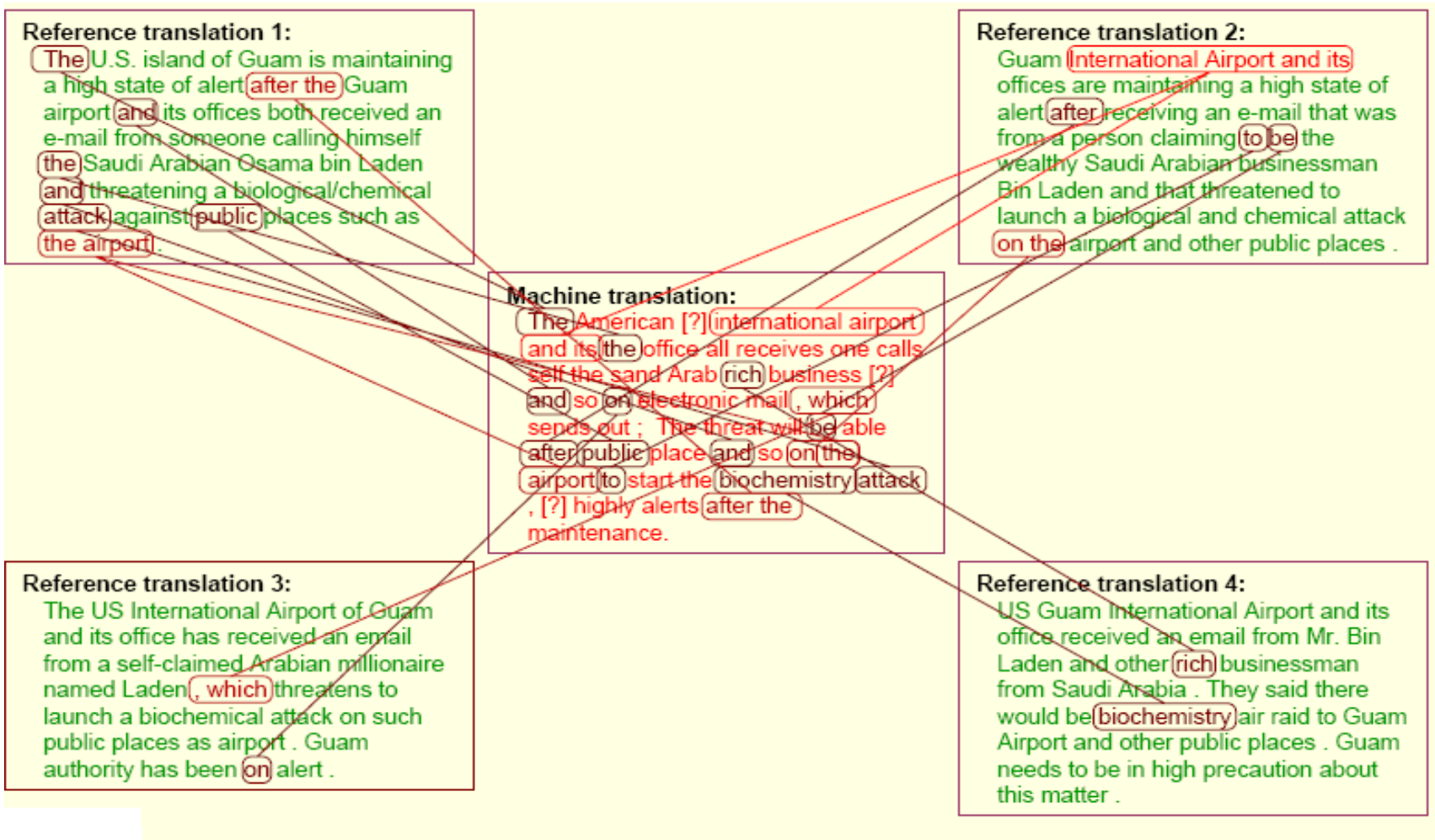
The American [?] international airport and its the office all receives one calls self the sand Arab rich business [?] and so on electronic mail , which sends out ; The threat will be able after public place and so on the airport to start the biochemistry attack , [?] highly alerts after the maintenance.

BLEU4 formula
(counts n-grams up to length 4)

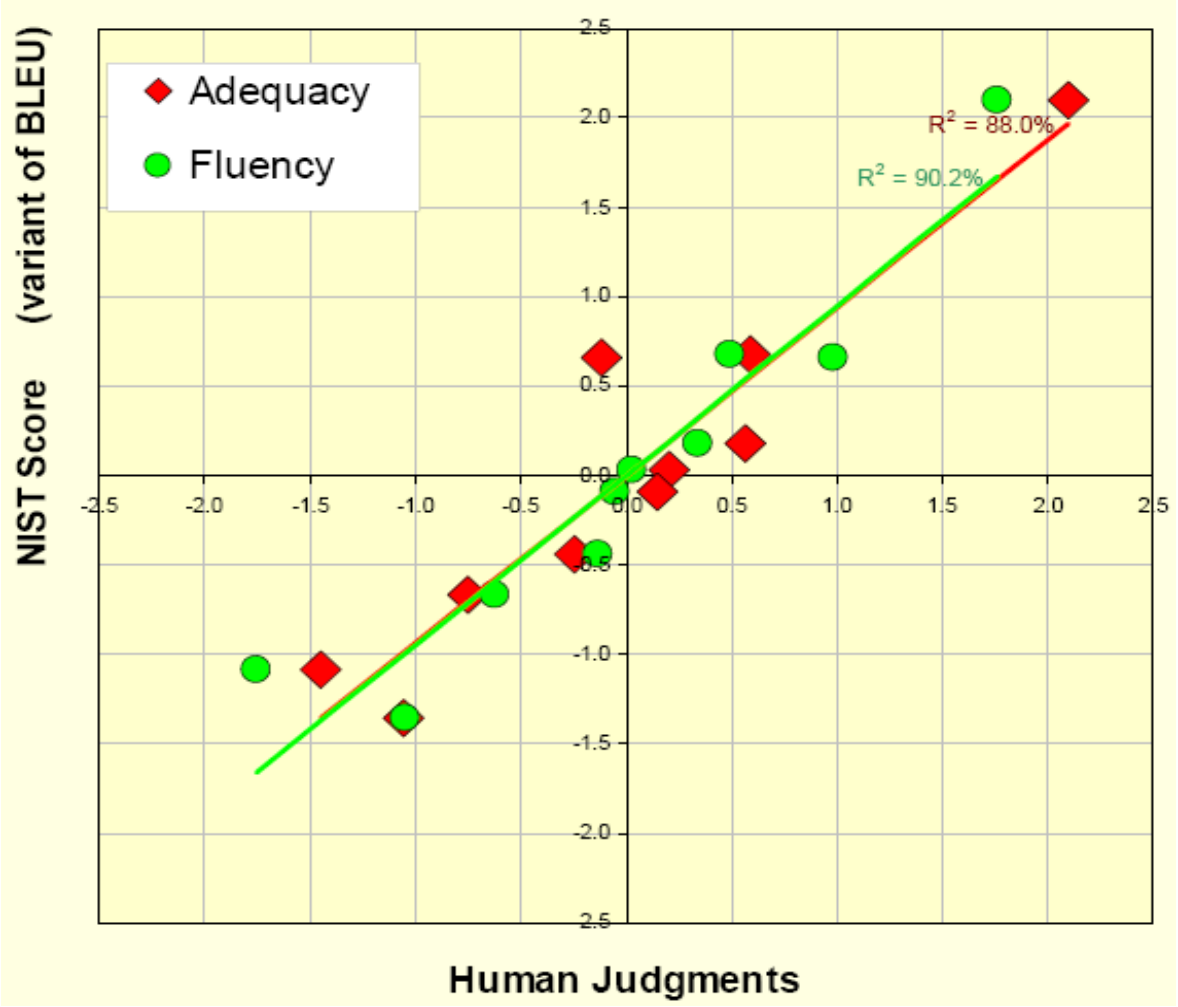
exp (1.0 * log p1 +
0.5 * log p2 +
0.25 * log p3 +
0.125 * log p4 –
max(words-in-reference / words-in-machine – 1,
0)

p1 = 1-gram precision
p2 = 2-gram precision
p3 = 3-gram precision
p4 = 4-gram precision

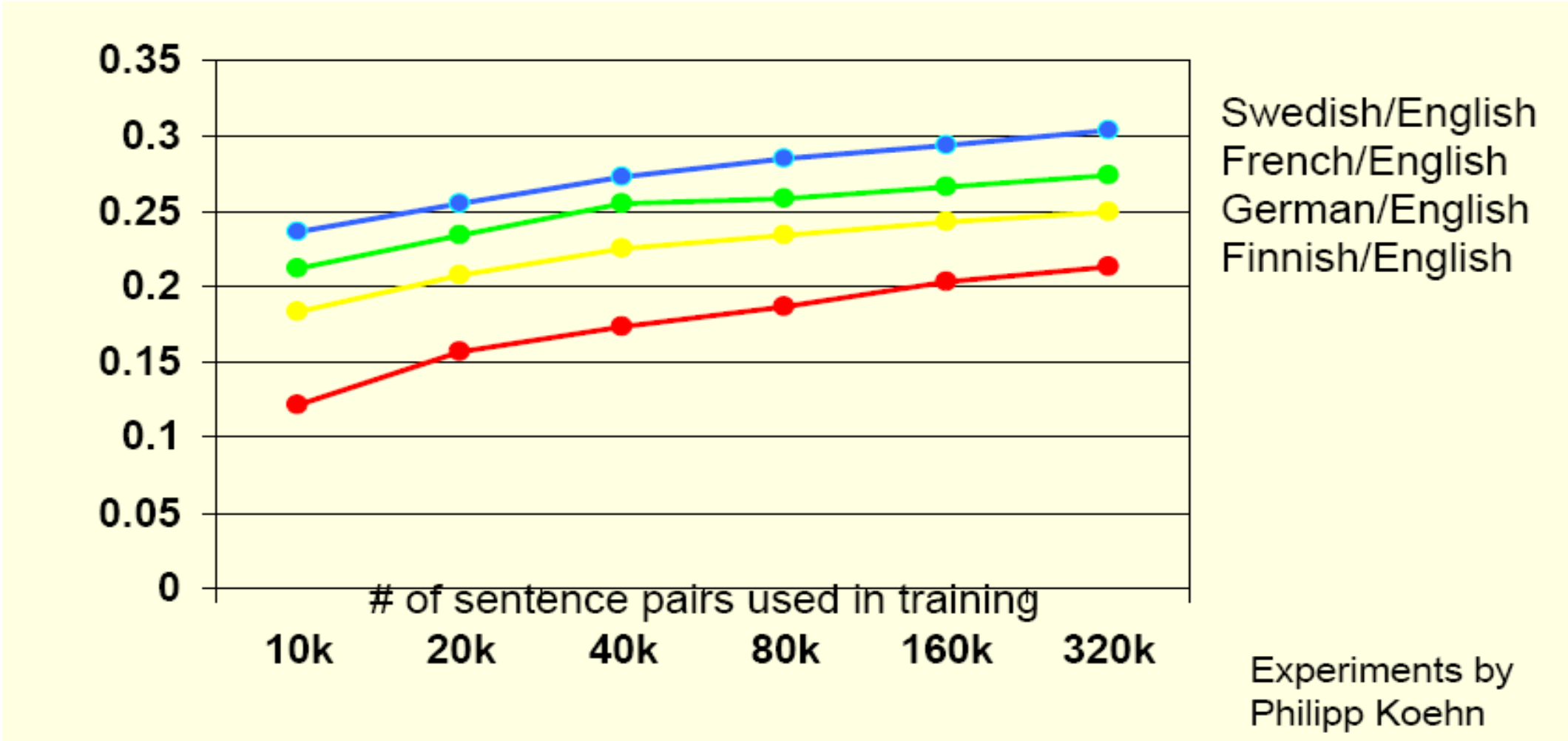
BLEU with Multiple References



BLEU: Correlation with Human Judgments



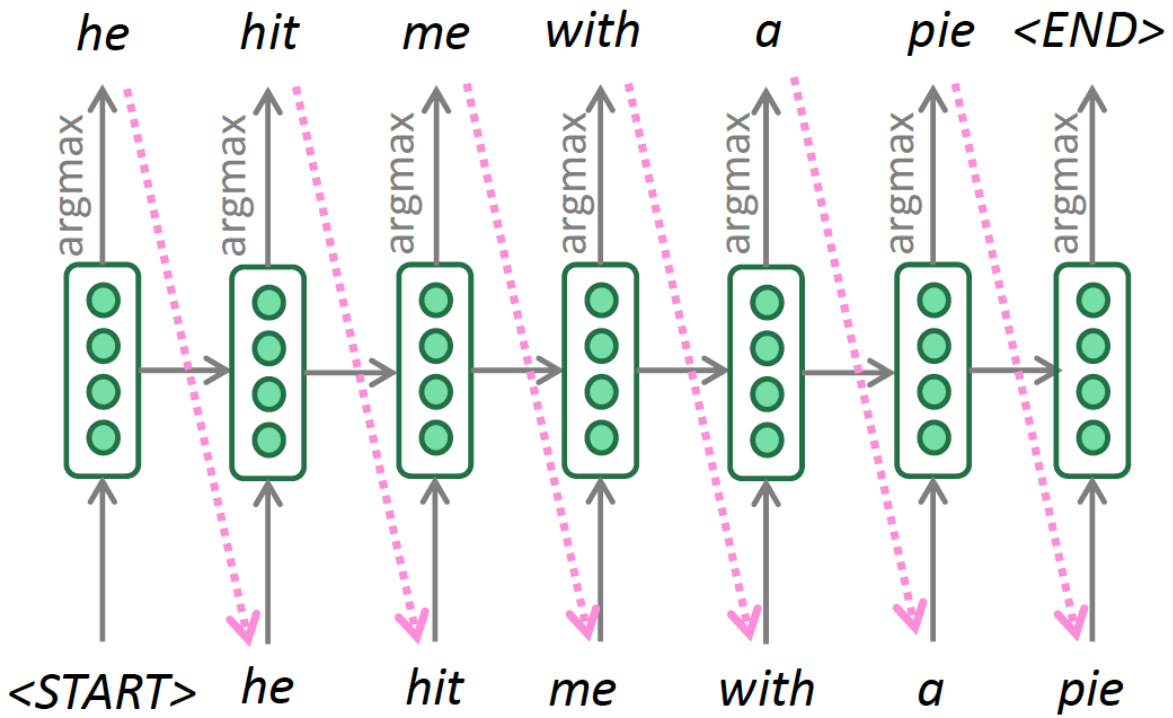
BLEU for Different Training Sizes



Decoding Strategies in Seq2Seq

Decoding Algorithm: Greedy Decoding

- At each step, take the most probable word (argmax)
- Use that as the next word, feed it as input on the next step
- Keep going until you produce the end token



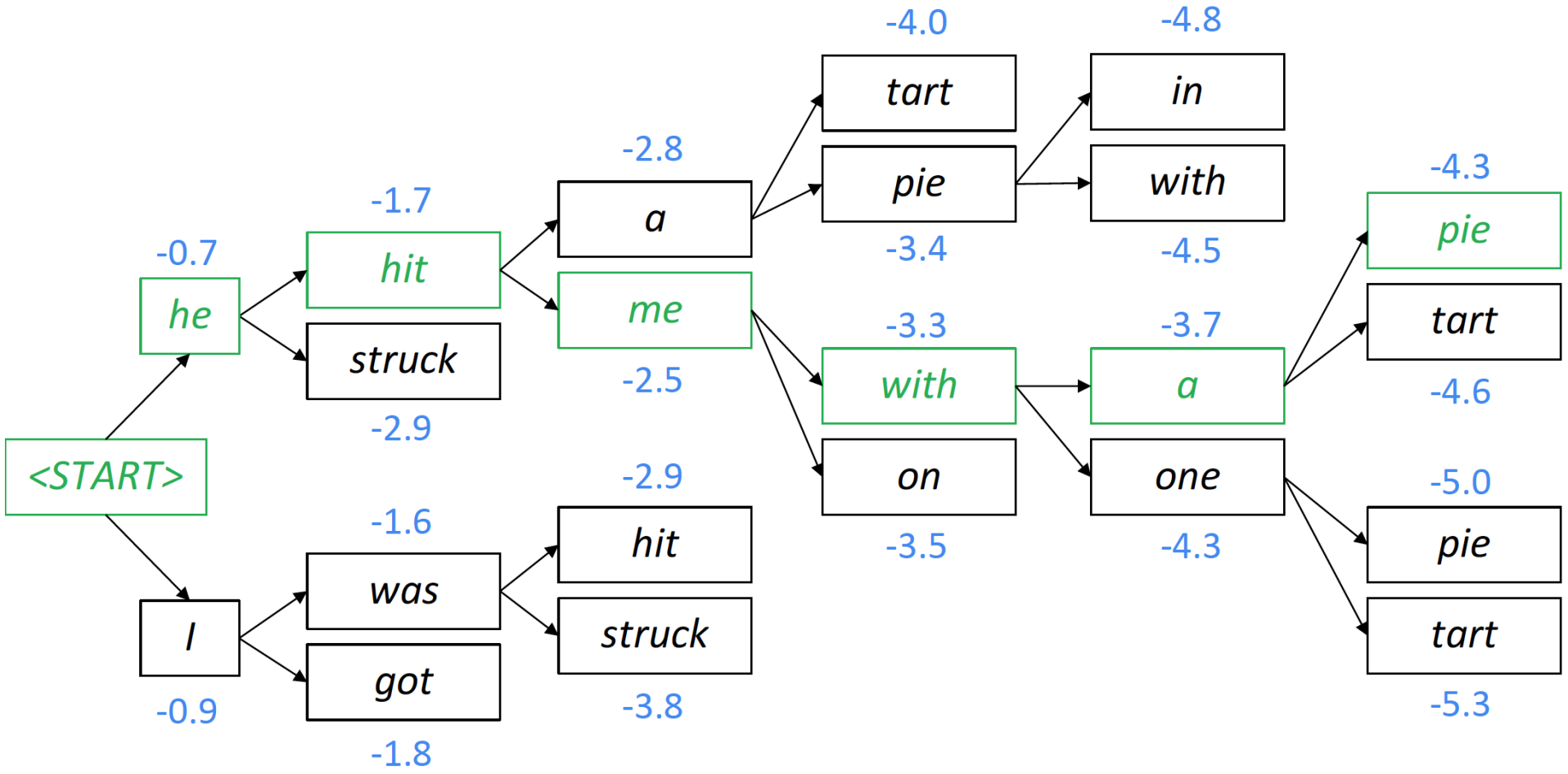
Decoding Algorithm: Beam Search

- A search algorithm that aims to find a high-probability sequence (not necessarily the optimal sequence, though) by tracking multiple possible sequences at once

- Core idea
  - On each step of decoder, keep track of the *k* (beam size) most probable partial sequences (which we call hypotheses)
  - After you reach some stopping criterion, choose the sequence with the highest probability (factoring in some adjustment for the length)

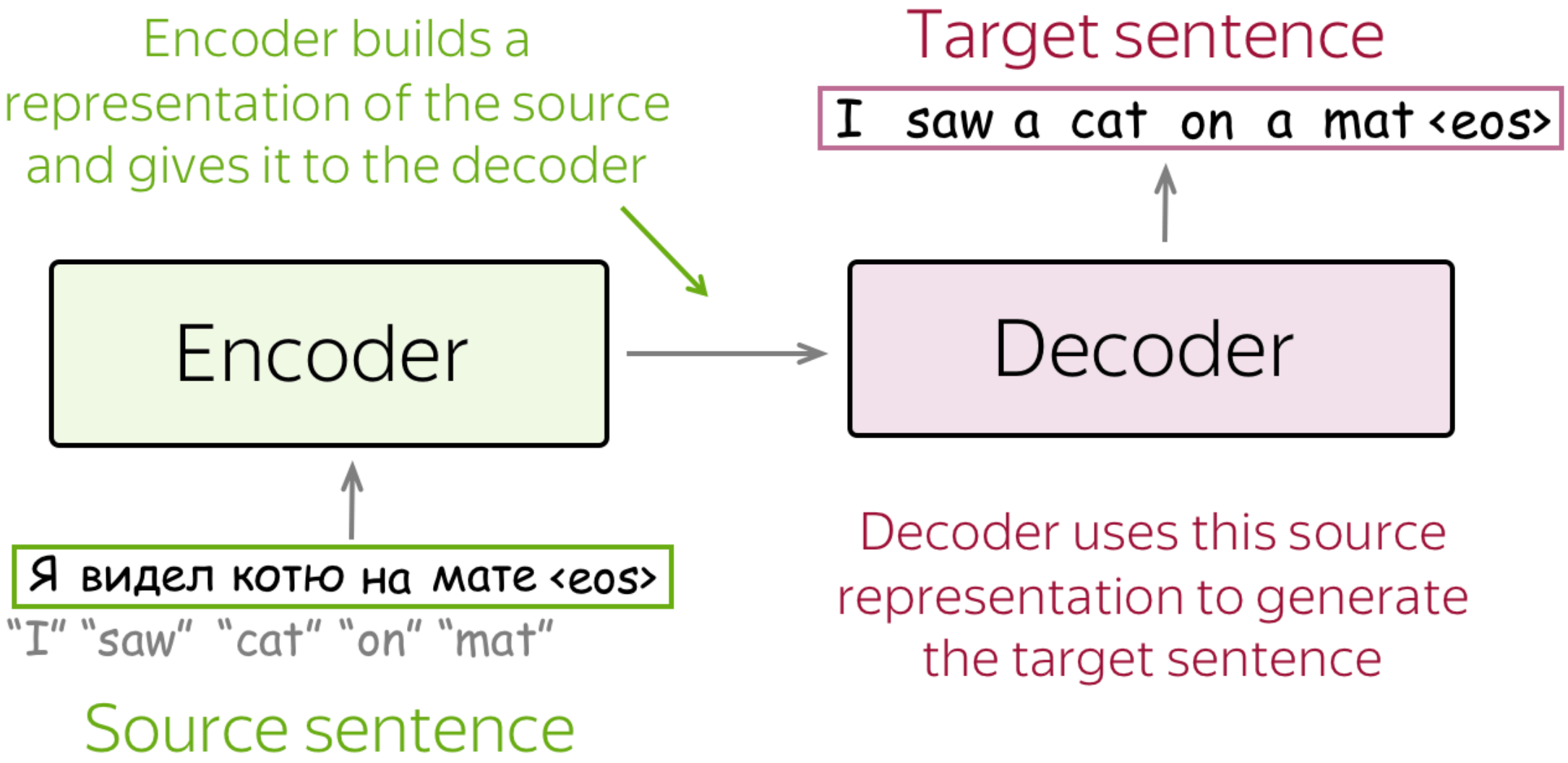
Decoding Algorithm: Beam Search

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Ilustrarted Example

Encoder-Decoder Framework

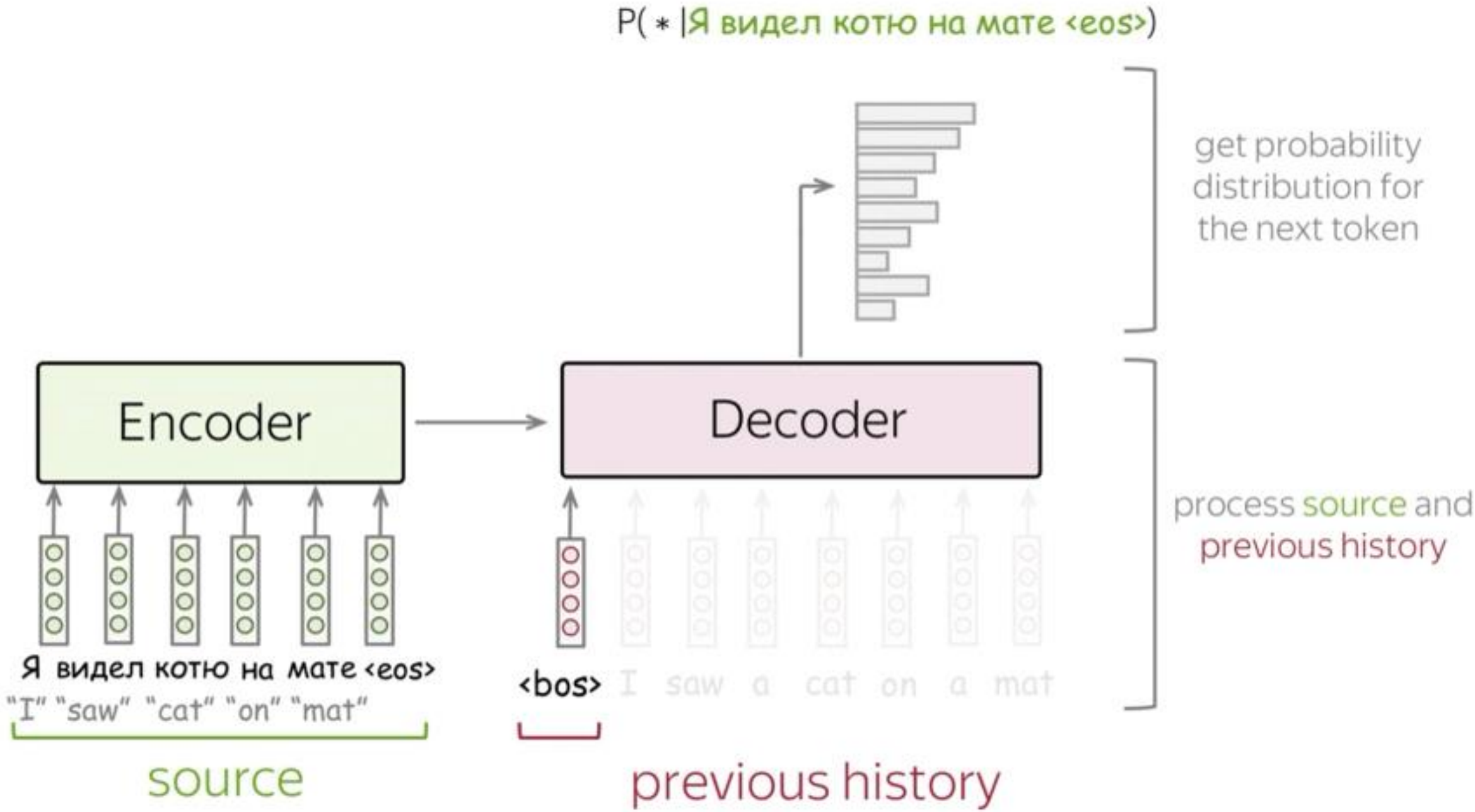


Conditional Language Models

Language Models: $P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$

Conditional Language Models: $P(y_1, y_2, \ldots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$

condition on source x

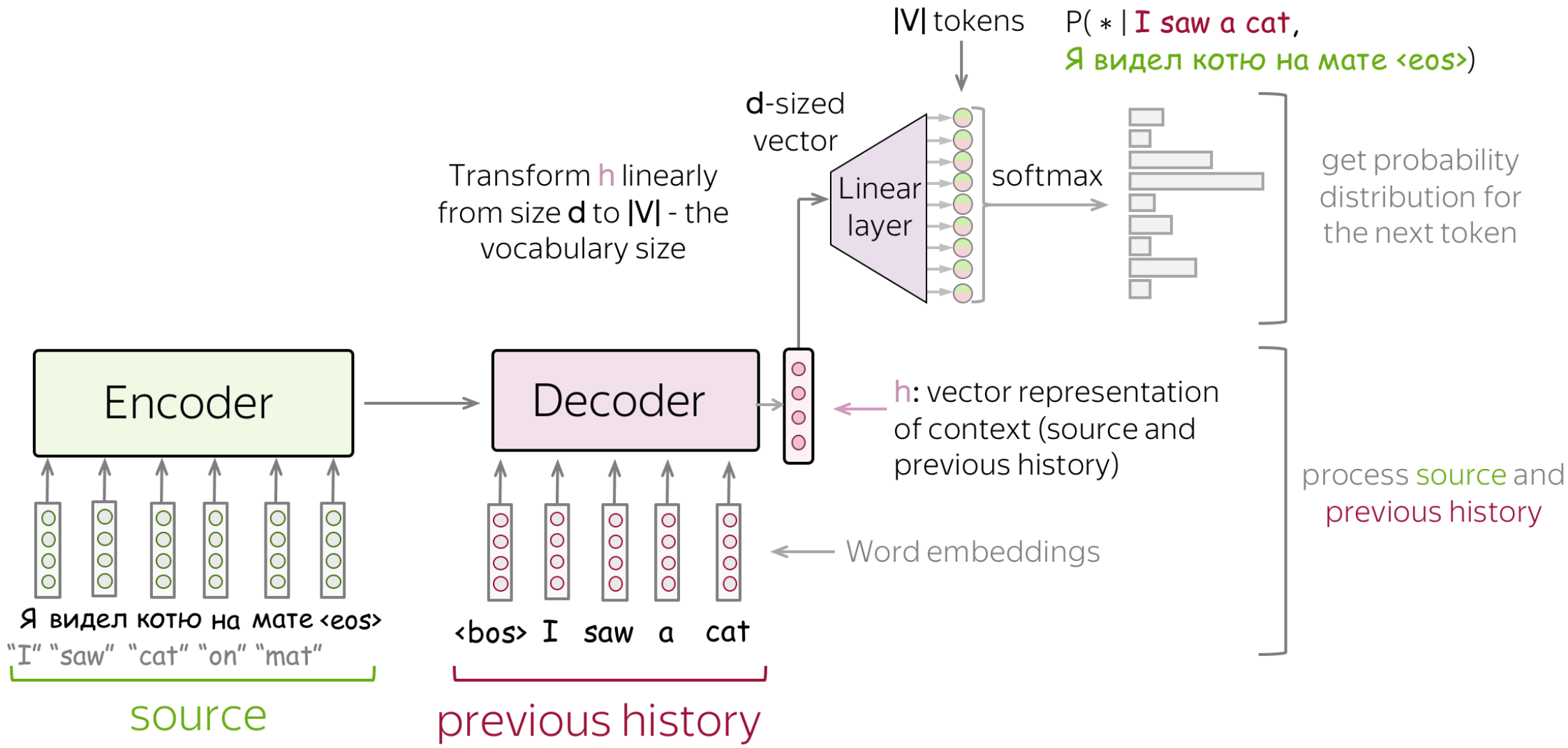

P(* |Я видел котю на мате <eos>)
get probability distribution for the next token
Encoder
Decoder
process source and previous history
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos>
I saw a cat on a mat
source
previous history

Conditional Language Models

Since the only difference from LMs is the presence of source $\boldsymbol{x}$, the modeling and training is very similar to language models. In particular, the high-level pipeline is as follows:

- feed source and previously generated target words into a network;
- get vector representation of context (both source and previous target) from the networks decoder;
- from this vector representation, predict a probability distribution for the next token.

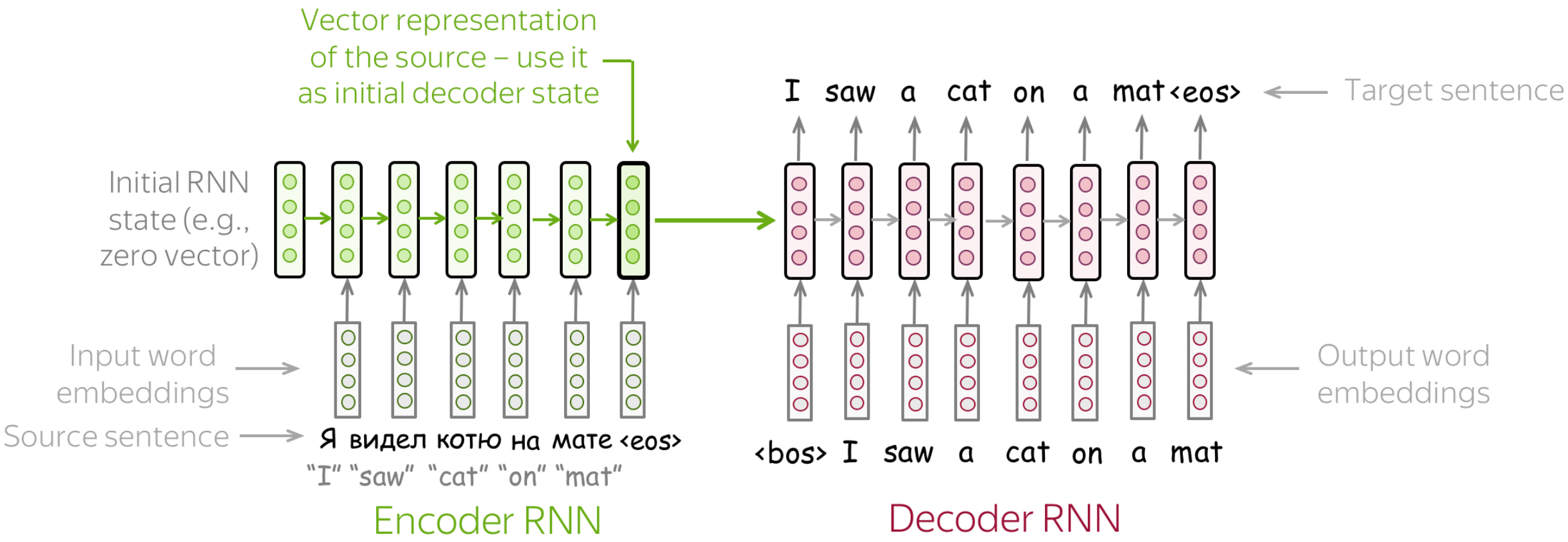
Conditional Language Models



Conditional Language Models

Similarly to neural classifiers and language models, we can think about the classification part (i.e., how to get token probabilities from a vector representation of a text) in a very simple way. Vector representation of a text has some dimensionality d, but in the end, we need a vector of size $|V|$ (probabilities for $|V|$ tokens/classes). To get a $|V|$-sized vector from a d-sized, we can use a linear layer. Once we have a $|V|$-sized vector, all is left is to apply the softmax operation to convert the raw numbers into token probabilities.

Encoder-Decoder with RNNs: Simplest Version

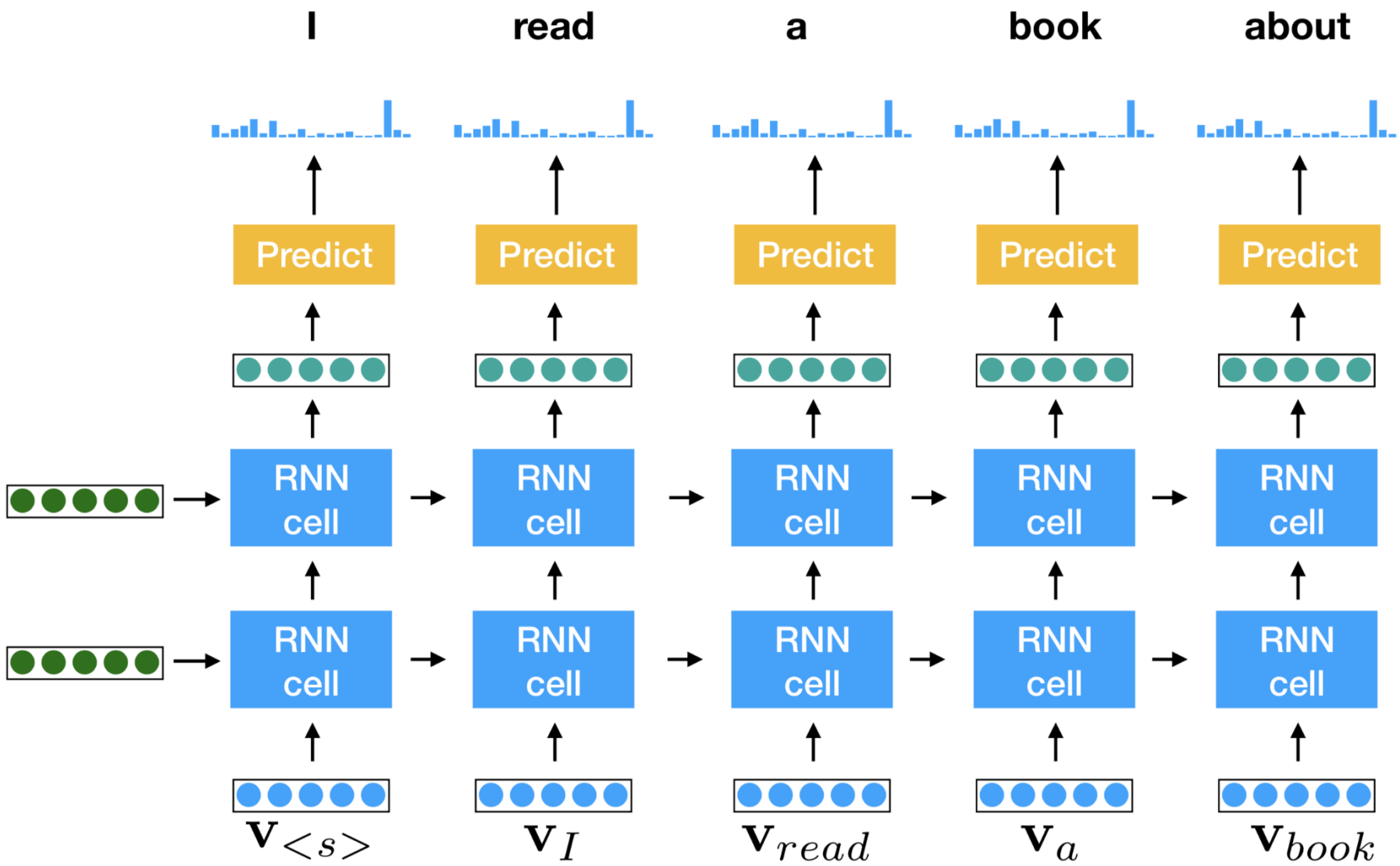


Useful Tutorial

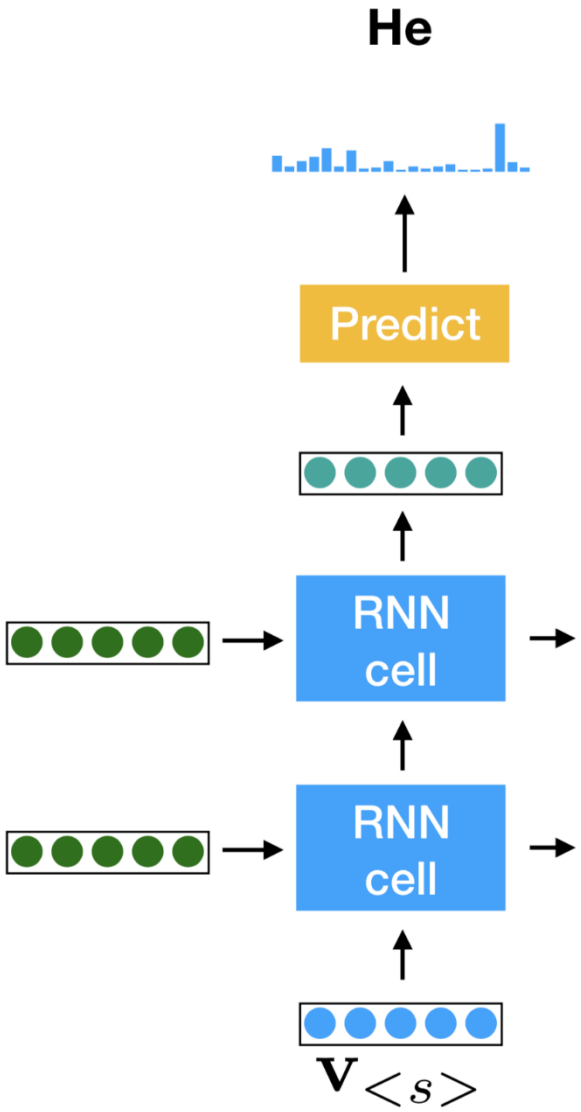
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Conditioned Generation

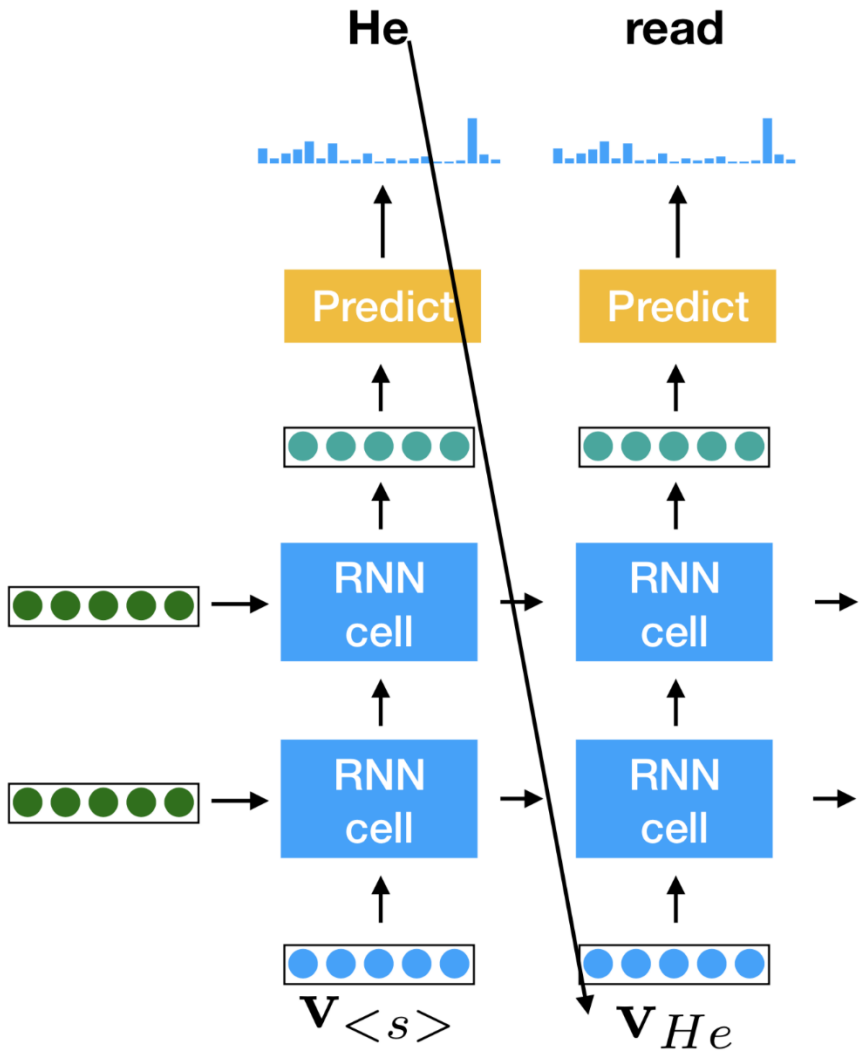
Training


I
read
a
book
about
Predict
Predict
Predict
Predict
Predict
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_I$
$\mathbf{v}_{read}$
$\mathbf{v}_a$
$\mathbf{v}_{book}$

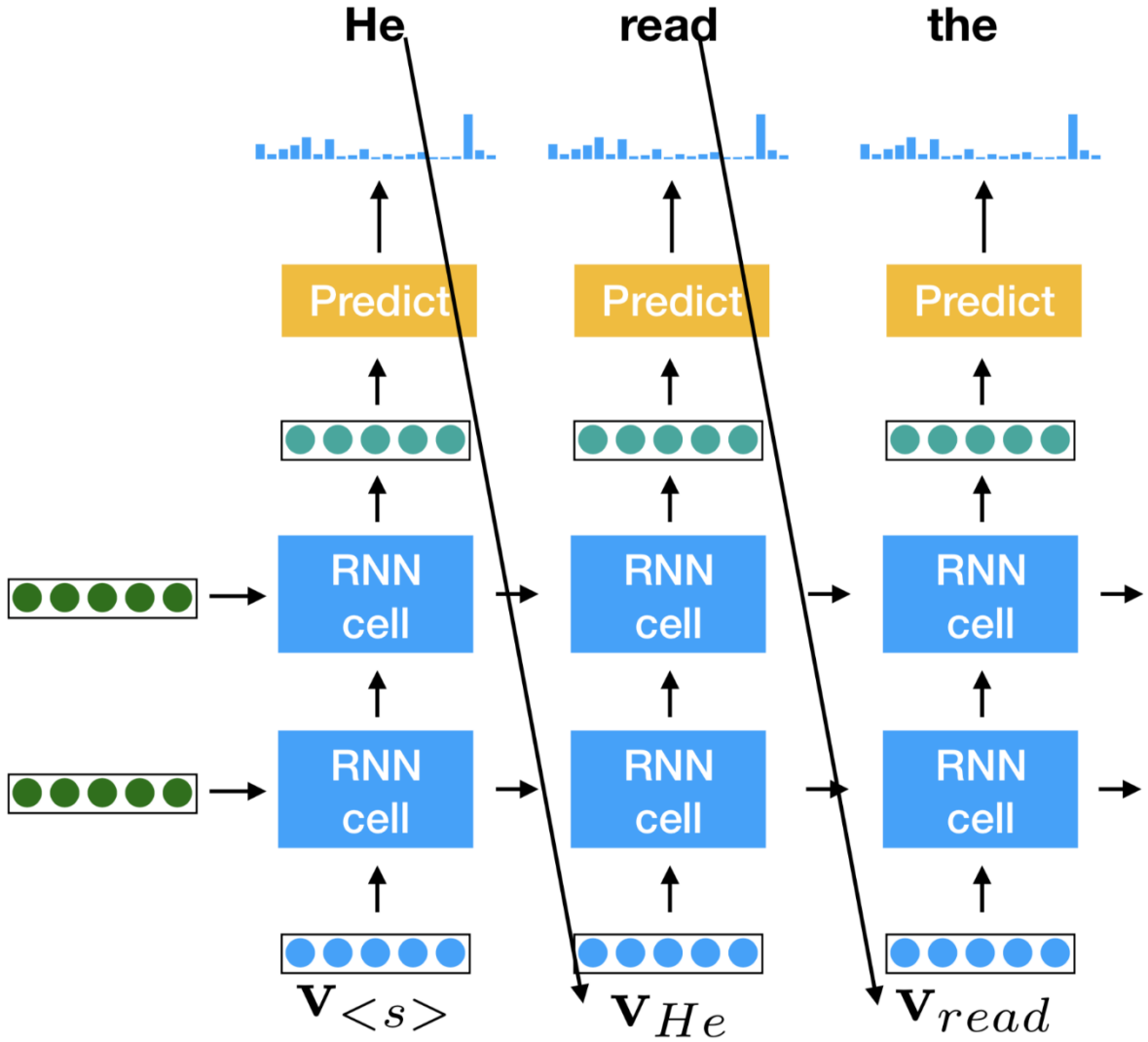
Generation



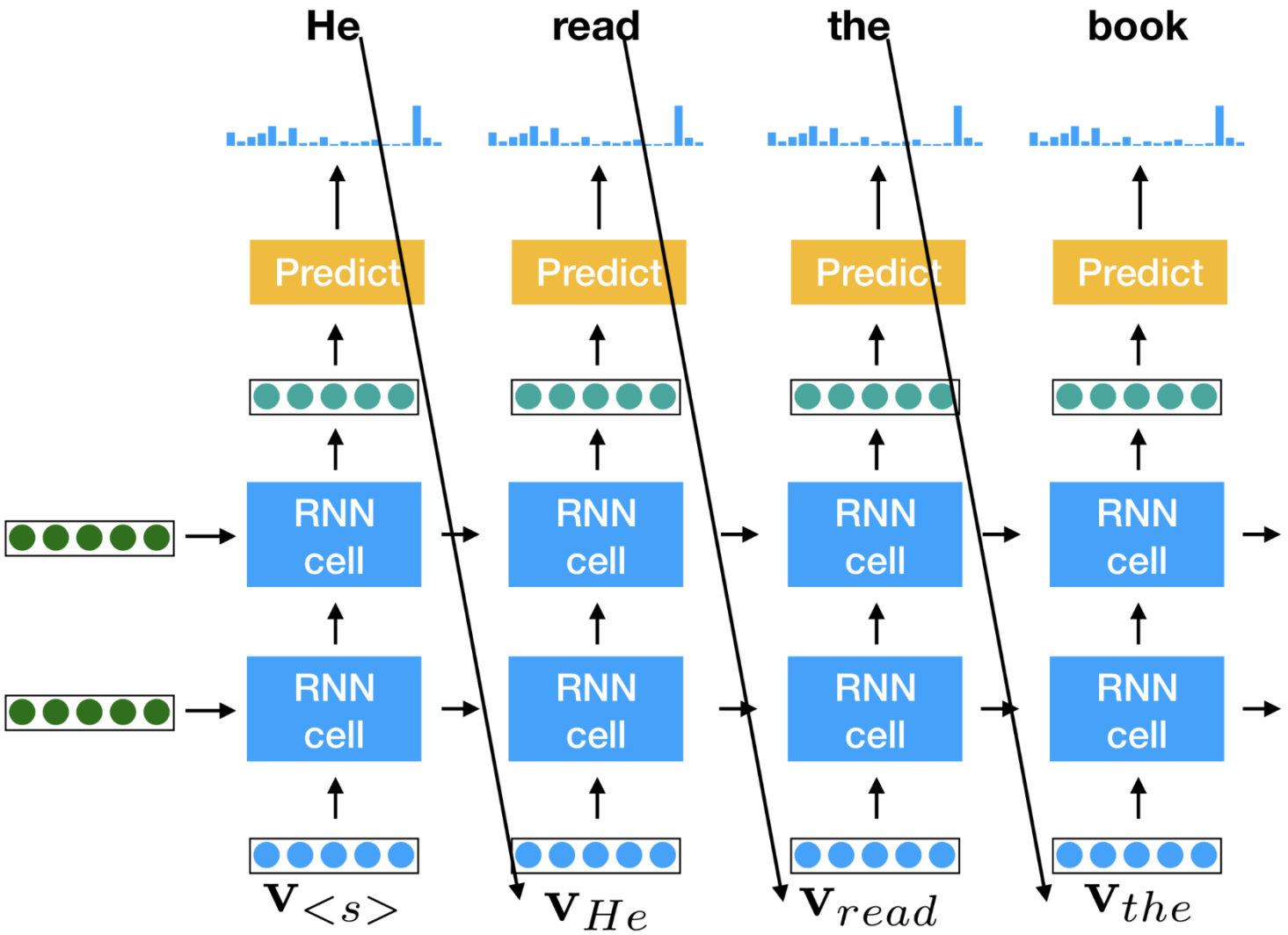
Generation


He
read
Predict
Predict
RNN cell
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$

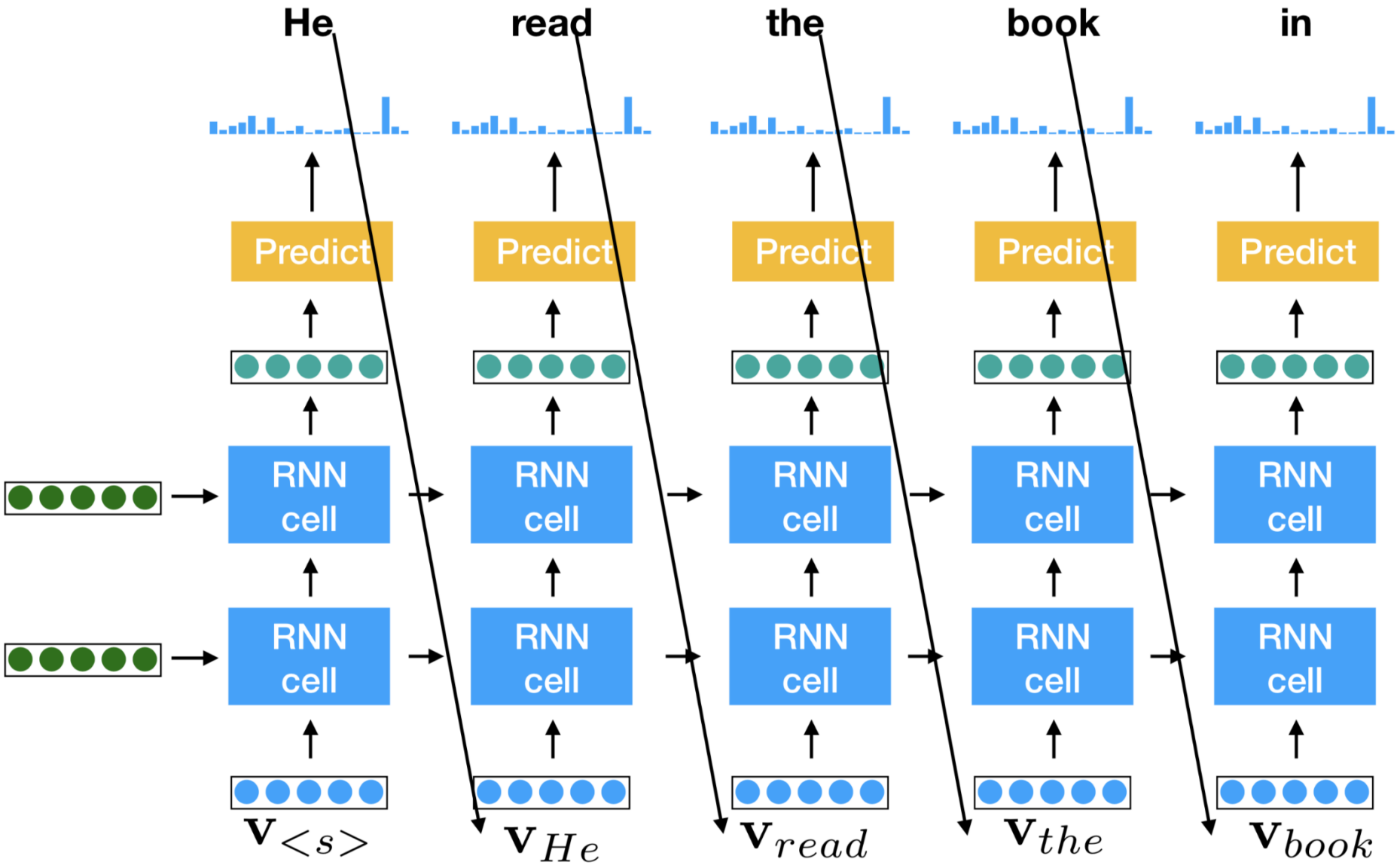
Generation


He
read
the
Predict
Predict
Predict
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
$\mathbf{v}_{read}$

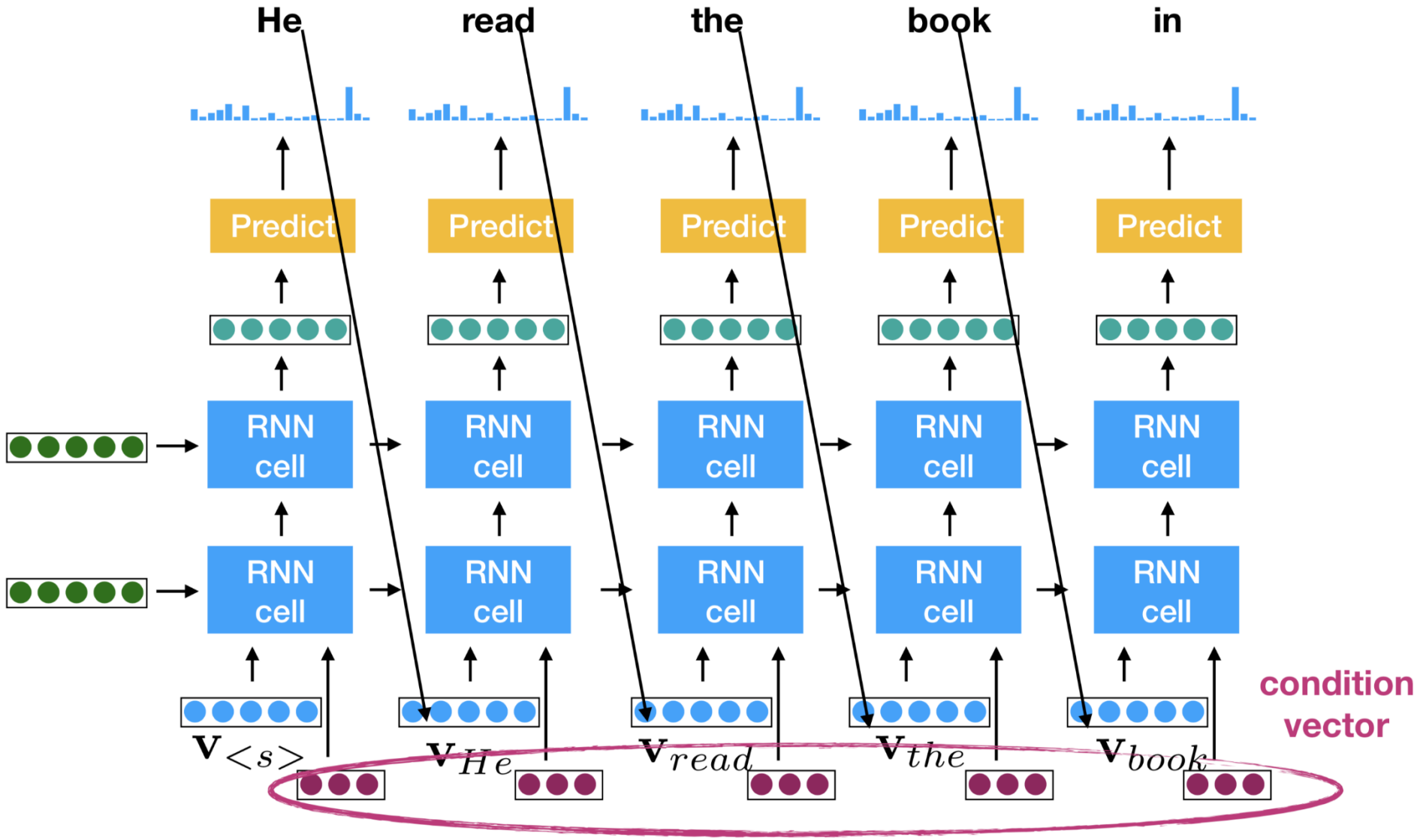
Generation


He
read
the
book
Predict
Predict
Predict
Predict
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
$\mathbf{v}_{read}$
$\mathbf{v}_{the}$

Generation

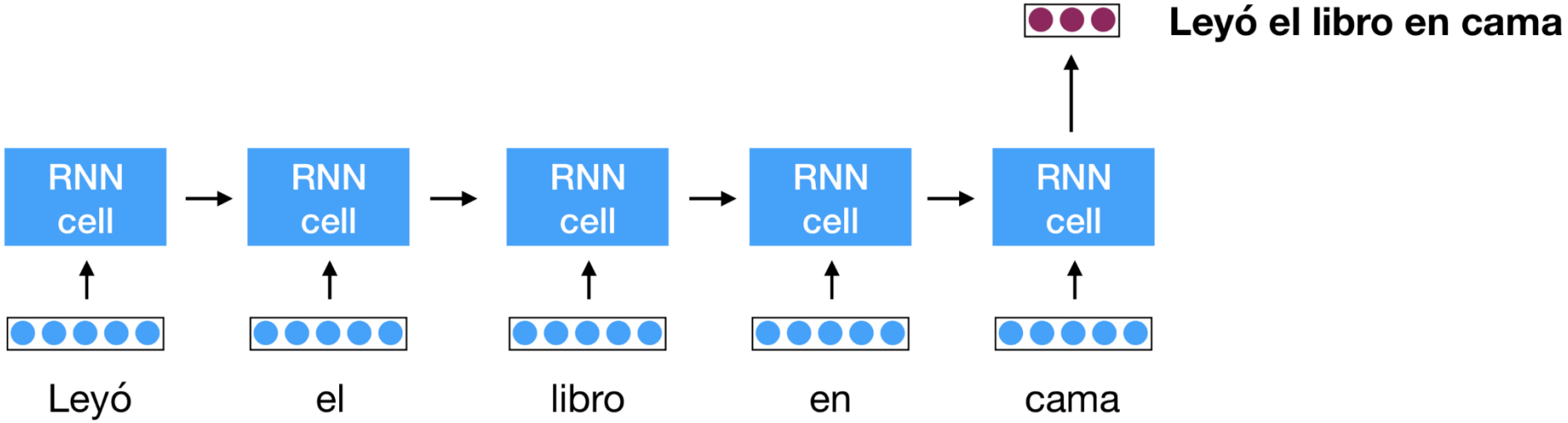

He
read
the
book
in
Predict
Predict
Predict
Predict
Predict
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
$\mathbf{v}_{read}$
$\mathbf{v}_{the}$
$\mathbf{v}_{book}$

Conditioned Generation

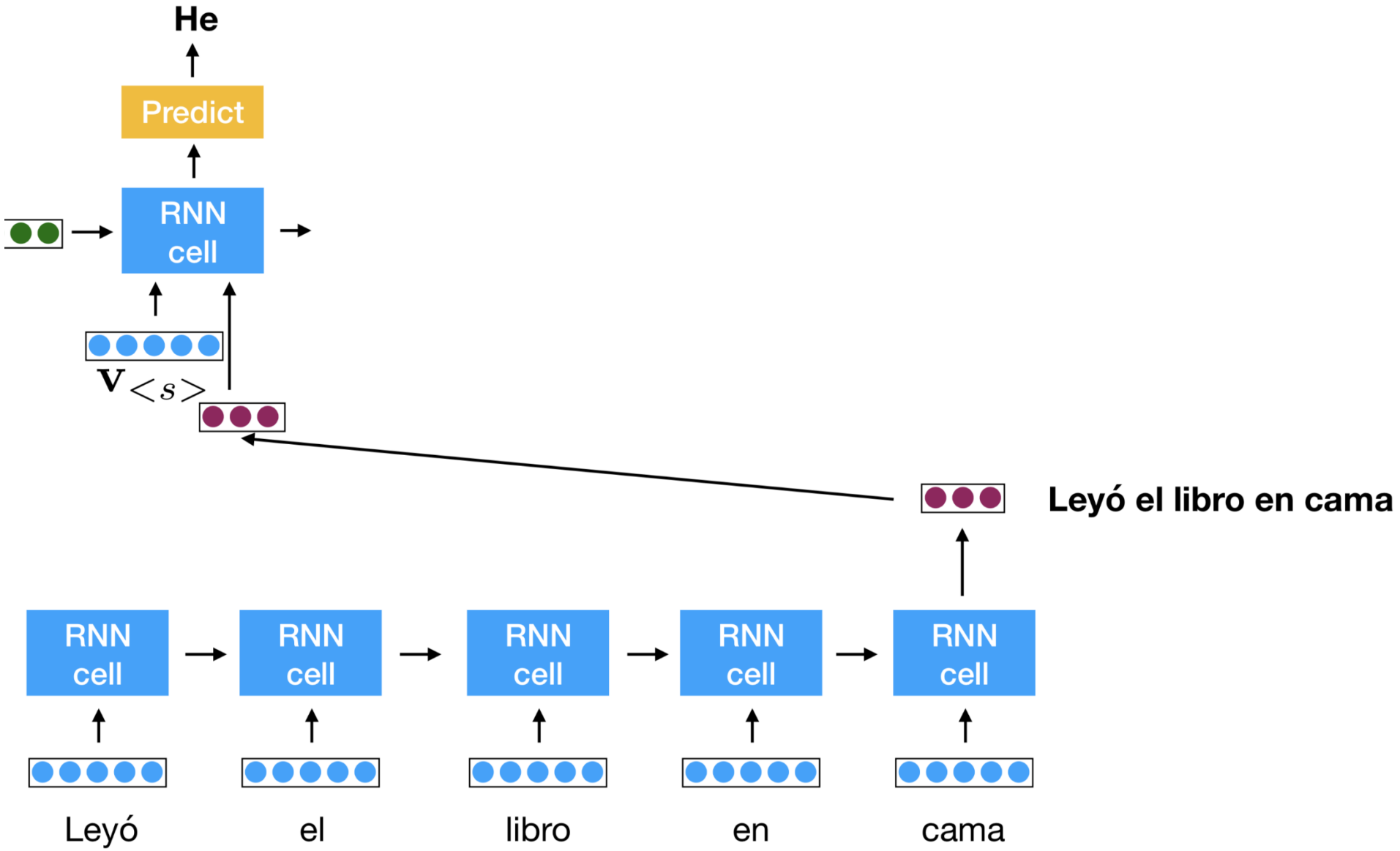


Attention in Detail

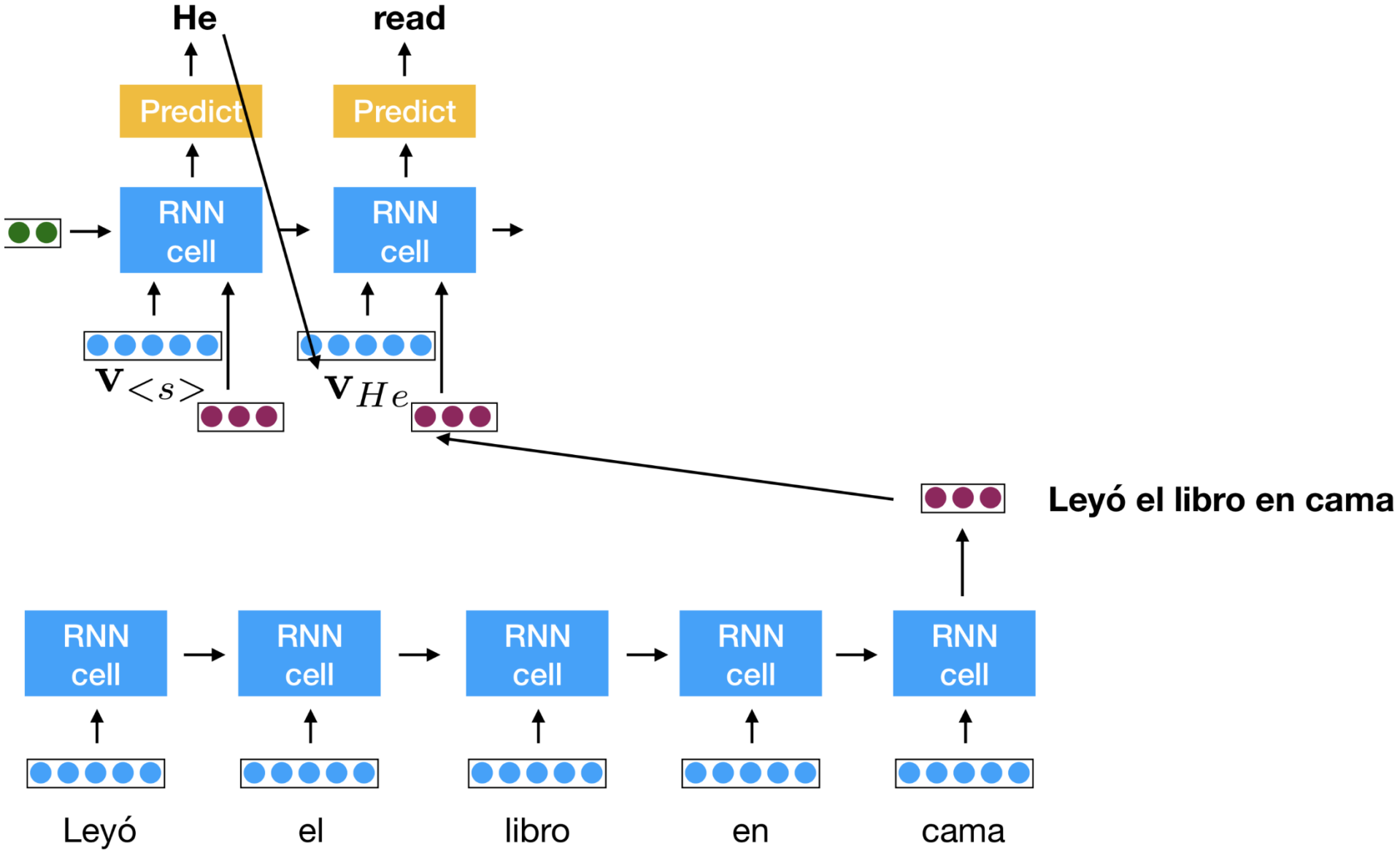
Seq2Seq


Leyó el libro en cama
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
Leyó
el
libro
en
cama

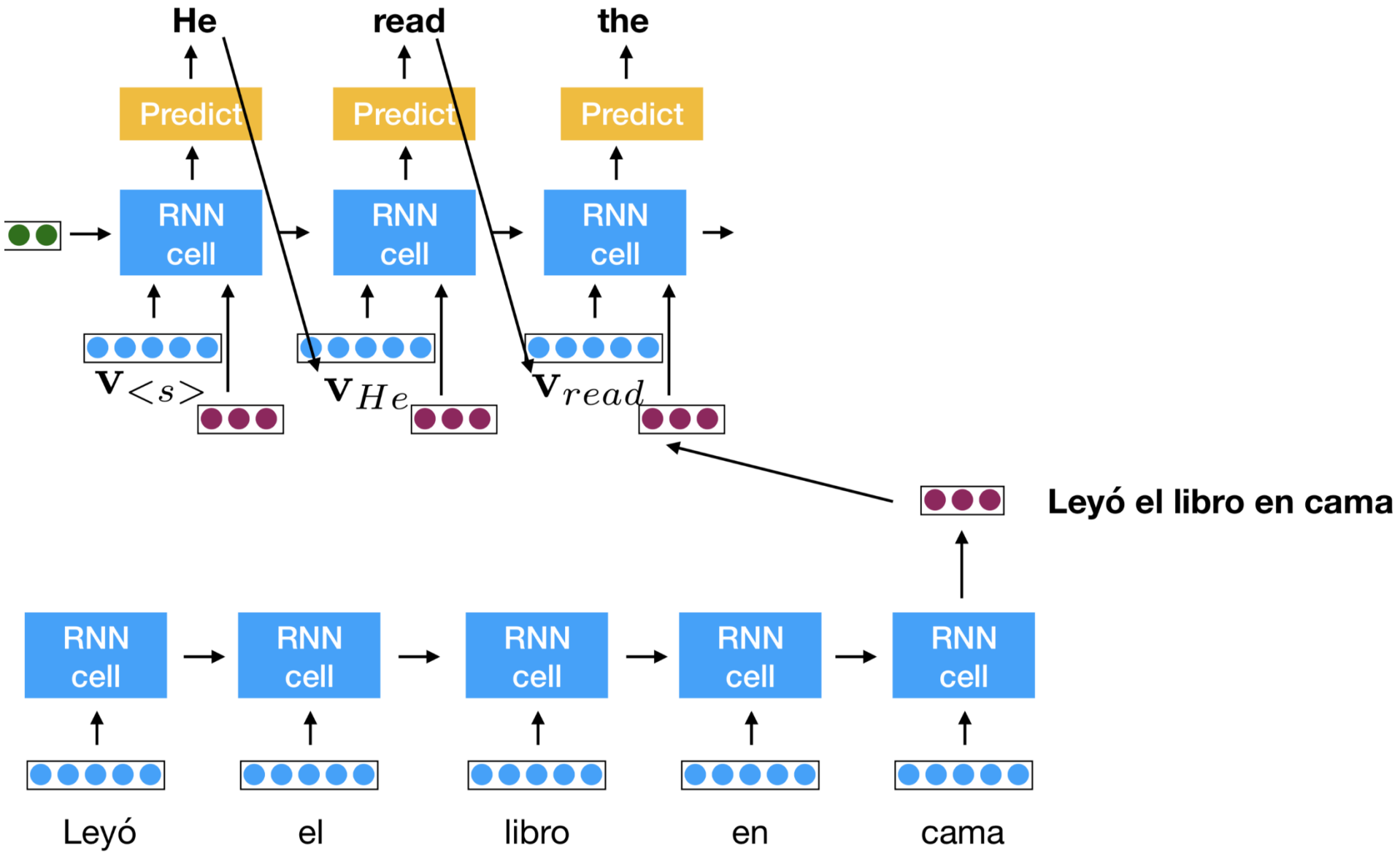
Seq2Seq


He
Predict
RNN cell
$\mathbf{V}_{<s>}$
Leyó el libro en cama
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
Leyó
el
libro
en
cama

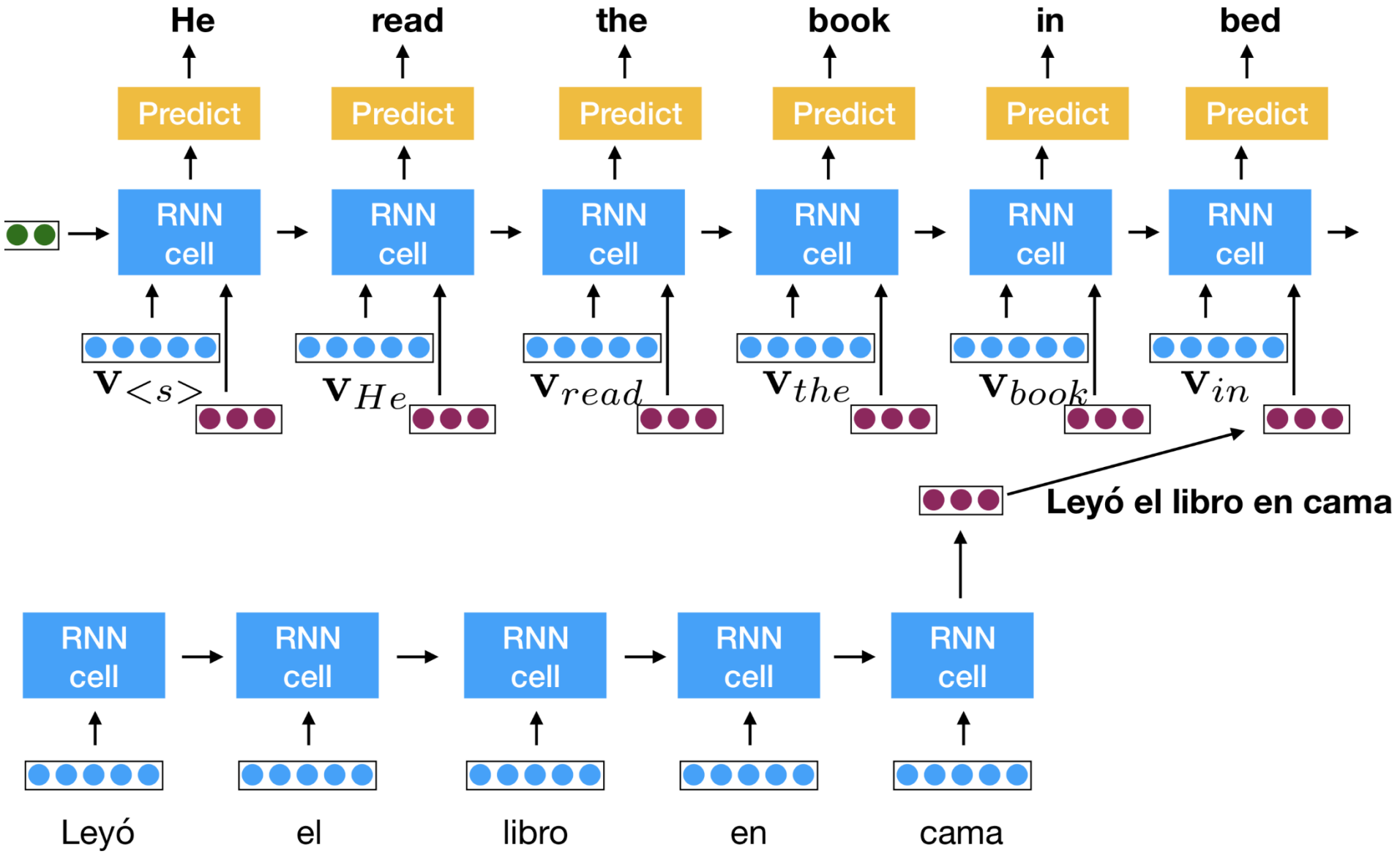
Seq2Seq


He
read
Predict
Predict
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
Leyó el libro en cama
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
Leyó
el
libro
en
cama

Seq2Seq

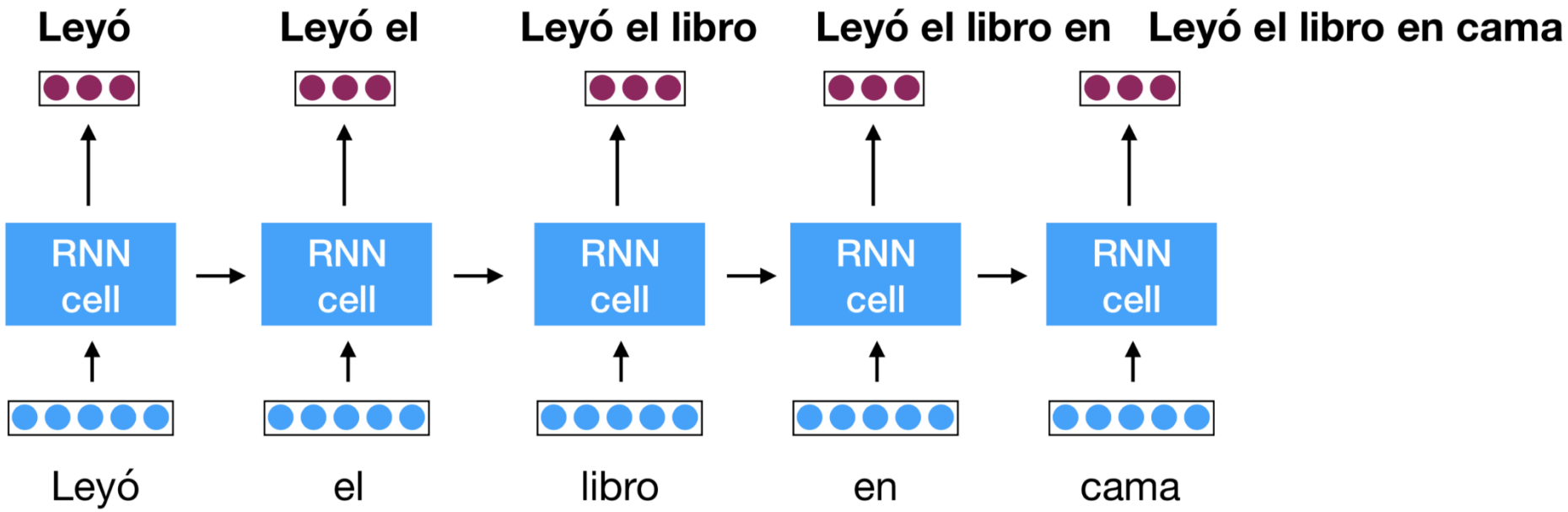

He
read
the
Predict
Predict
Predict
RNN cell
RNN cell
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
$\mathbf{v}_{read}$
Leyó el libro en cama
RNN cell
RNN cell
RNN cell
RNN cell
RNN cell
Leyó
el
libro
en
cama

Seq2Seq


He
read
the
book
in
bed
Predict
RNN cell
$\mathbf{v}_{<s>}$
$\mathbf{v}_{He}$
$\mathbf{v}_{read}$
$\mathbf{v}_{the}$
$\mathbf{v}_{book}$
$\mathbf{v}_{in}$
Leyó el libro en cama
Leyó
el
libro
en
cama

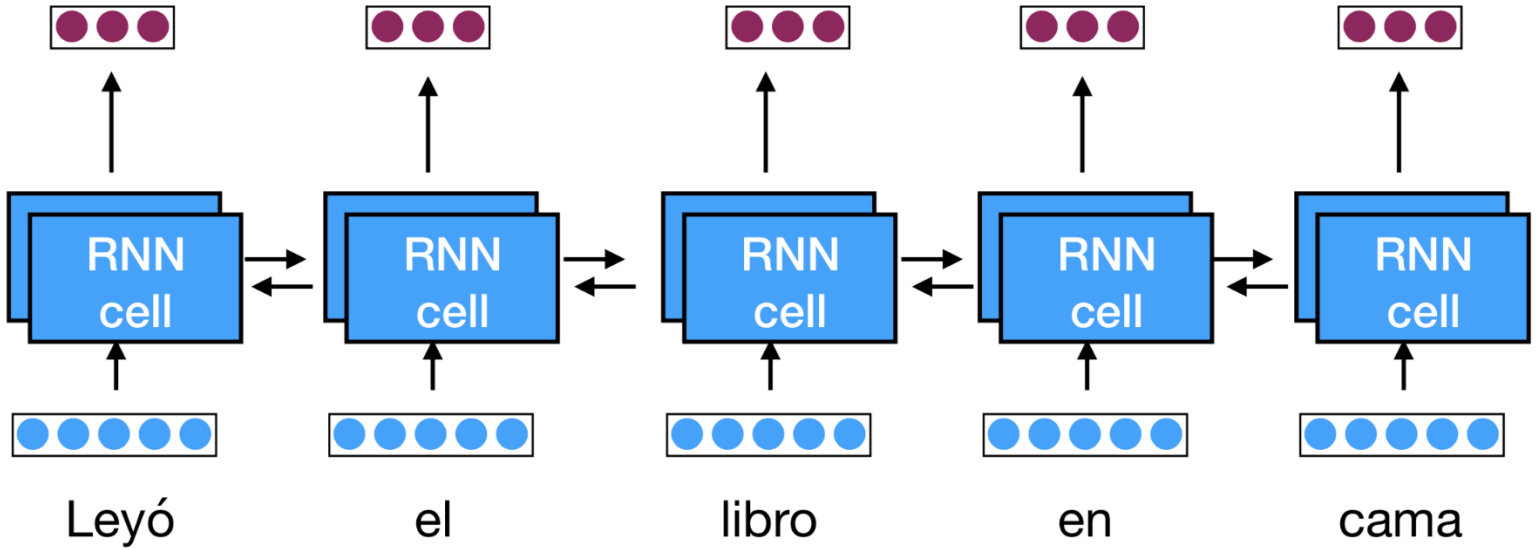
Seq2Seq with Attention

keep intermediate vectors



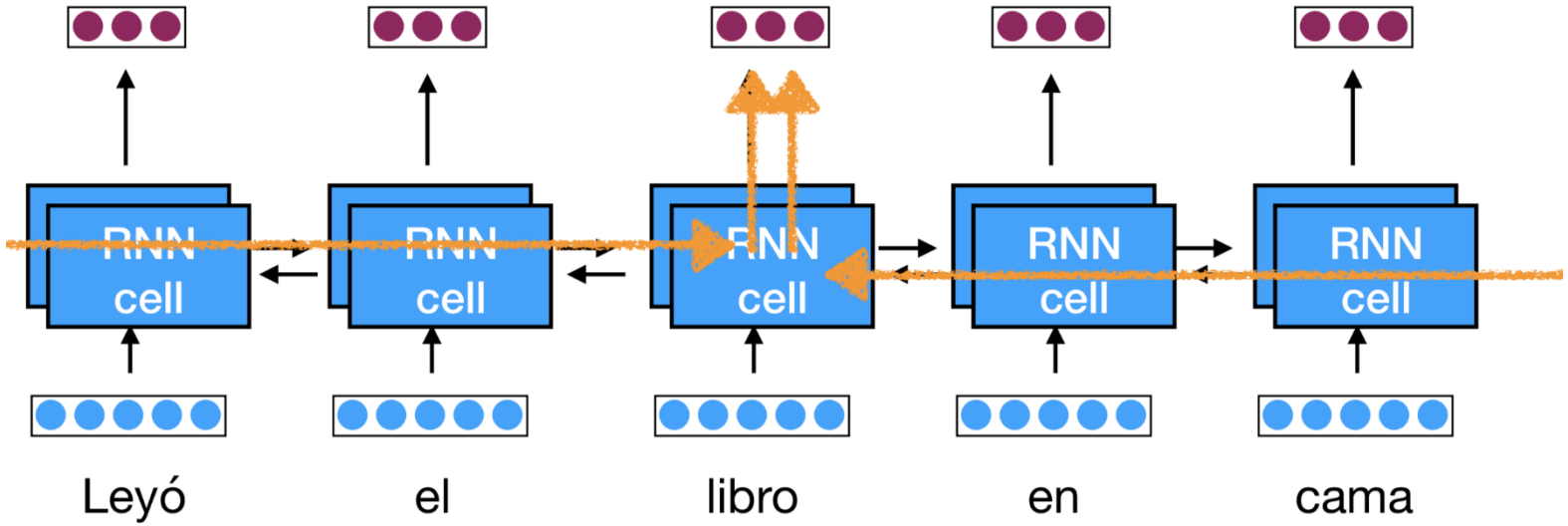
Seq2Seq with Attention

add right-to-left RNN (bi-RNN)



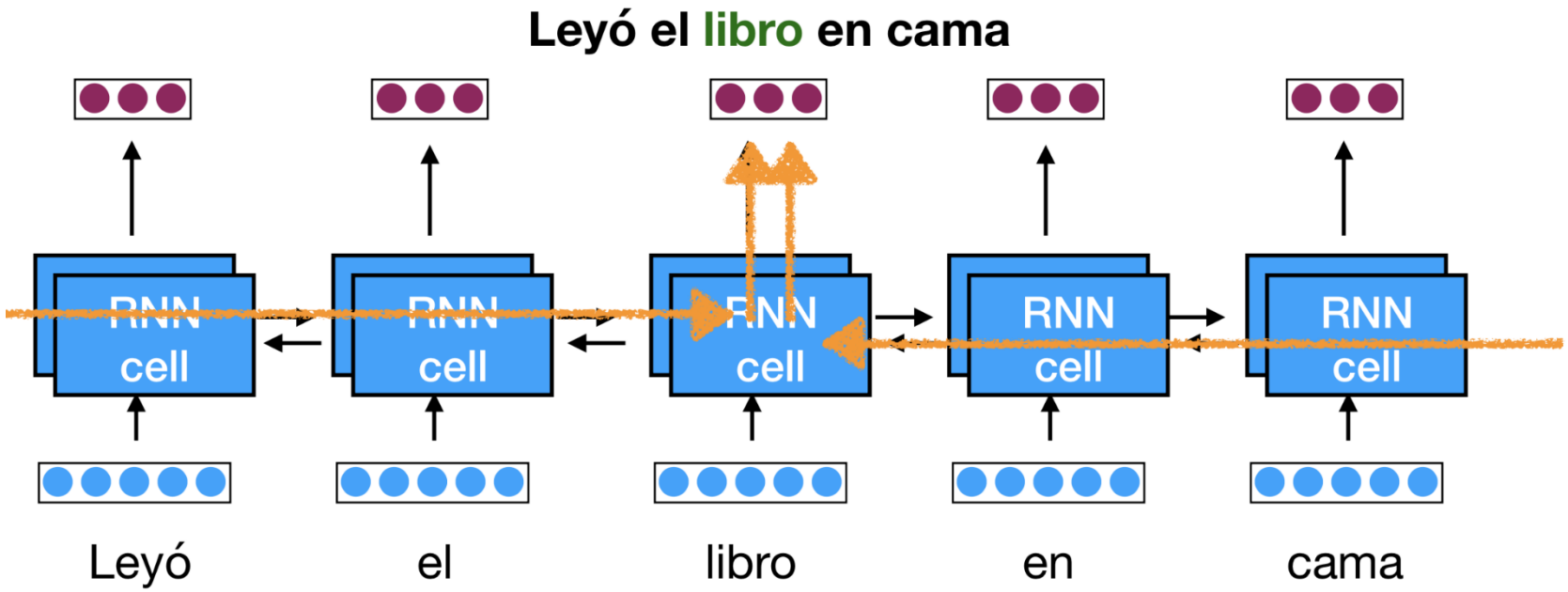
Seq2Seq with Attention

add right-to-left RNN (bi-RNN)



Seq2Seq with Attention

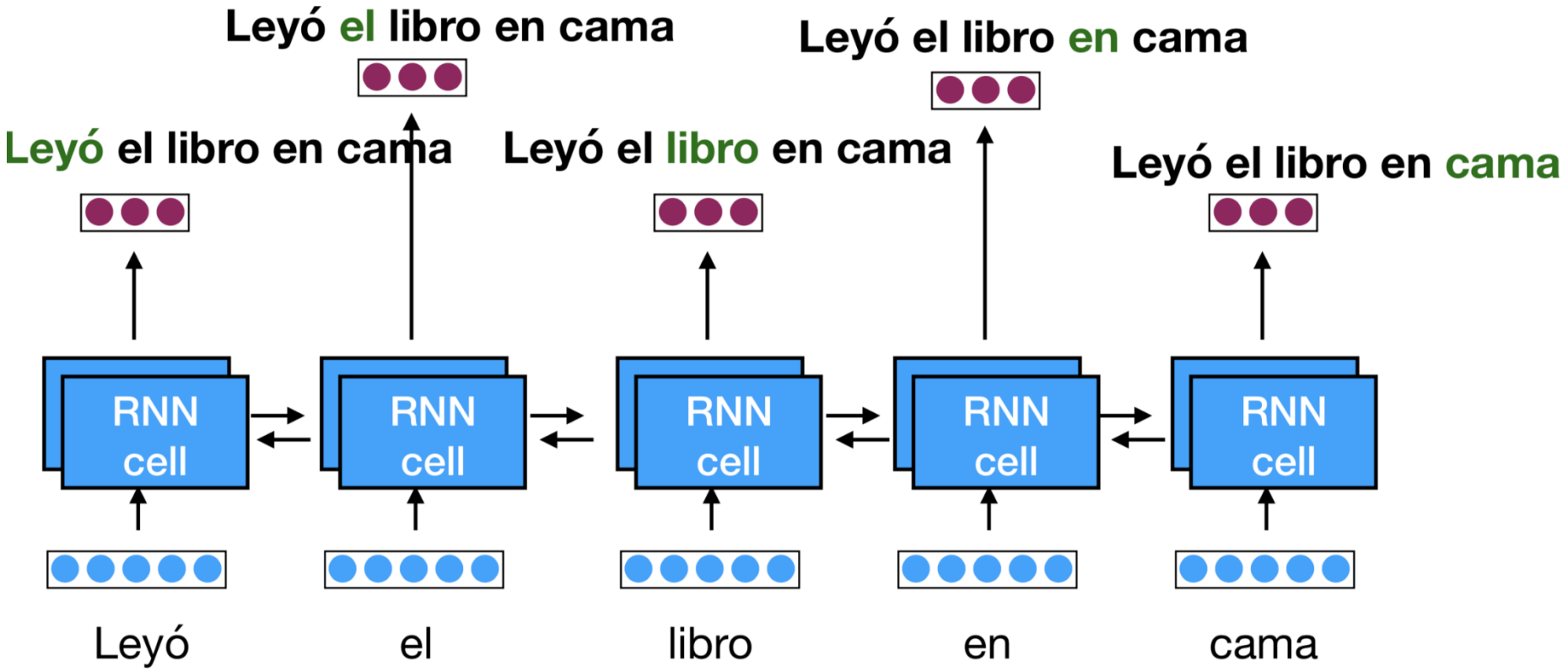
add right-to-left RNN (bi-RNN)



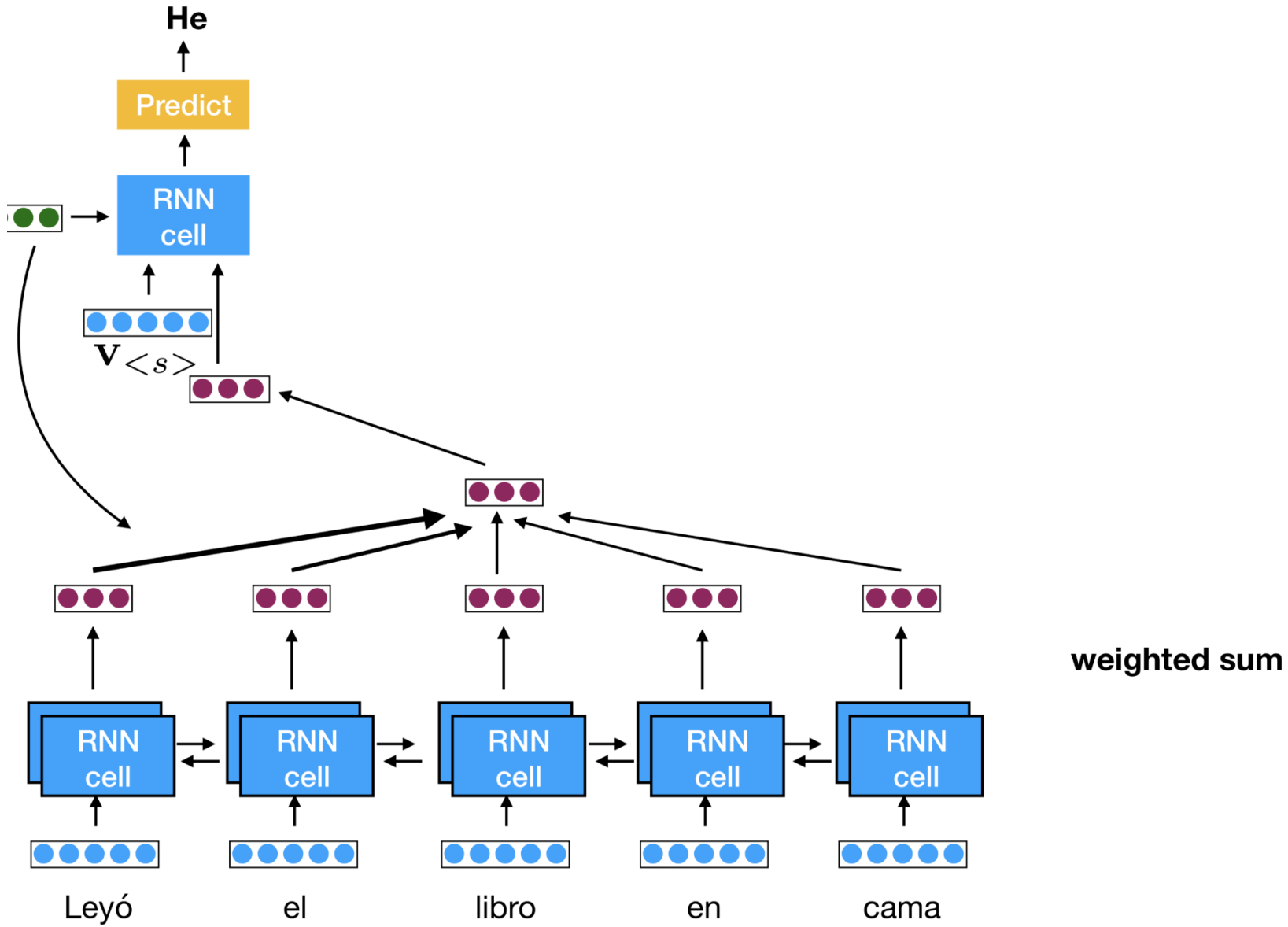
Seq2Seq with Attention

a representation of a word in context.

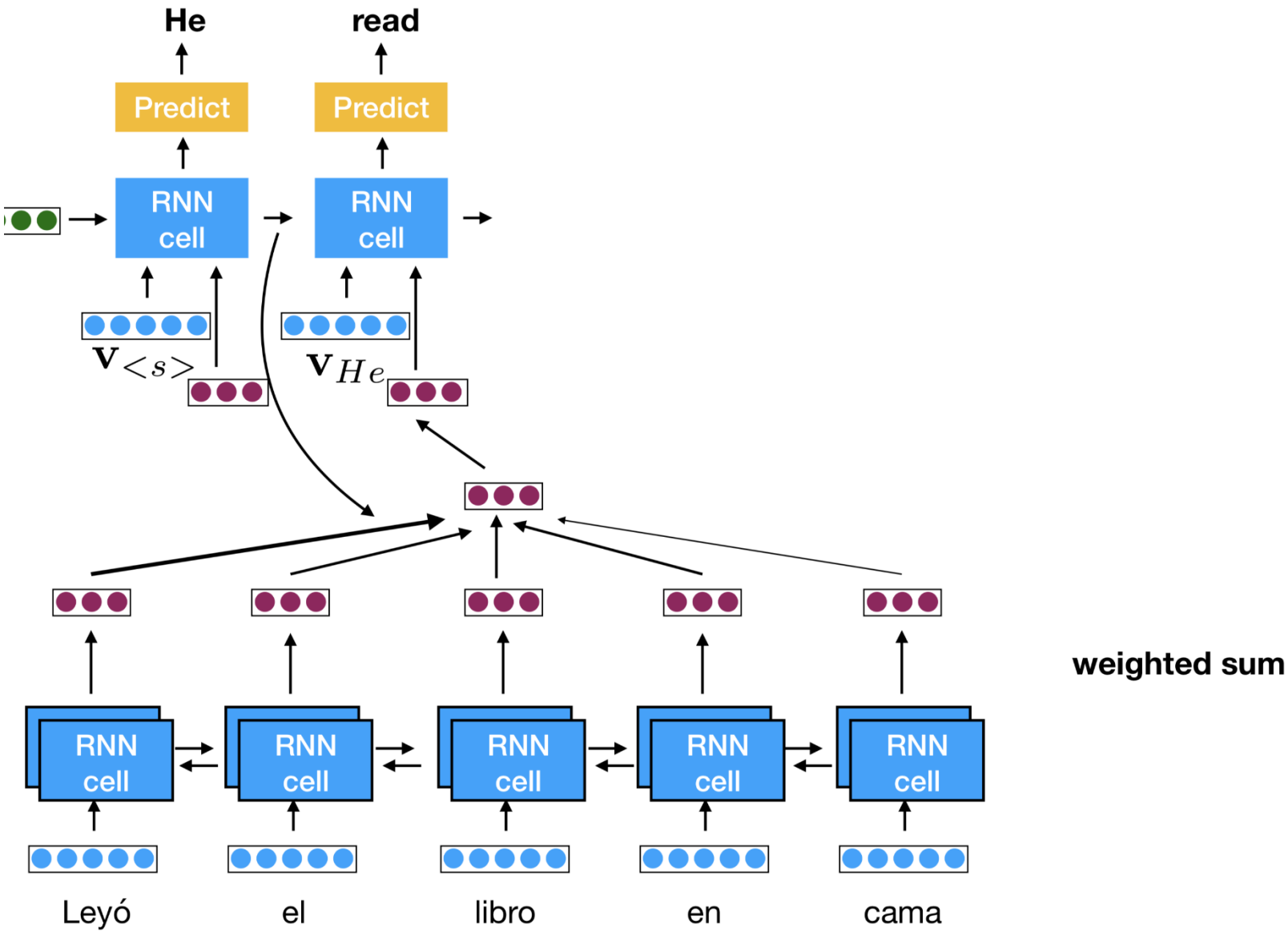
add right-to-left RNN (bi-RNN)



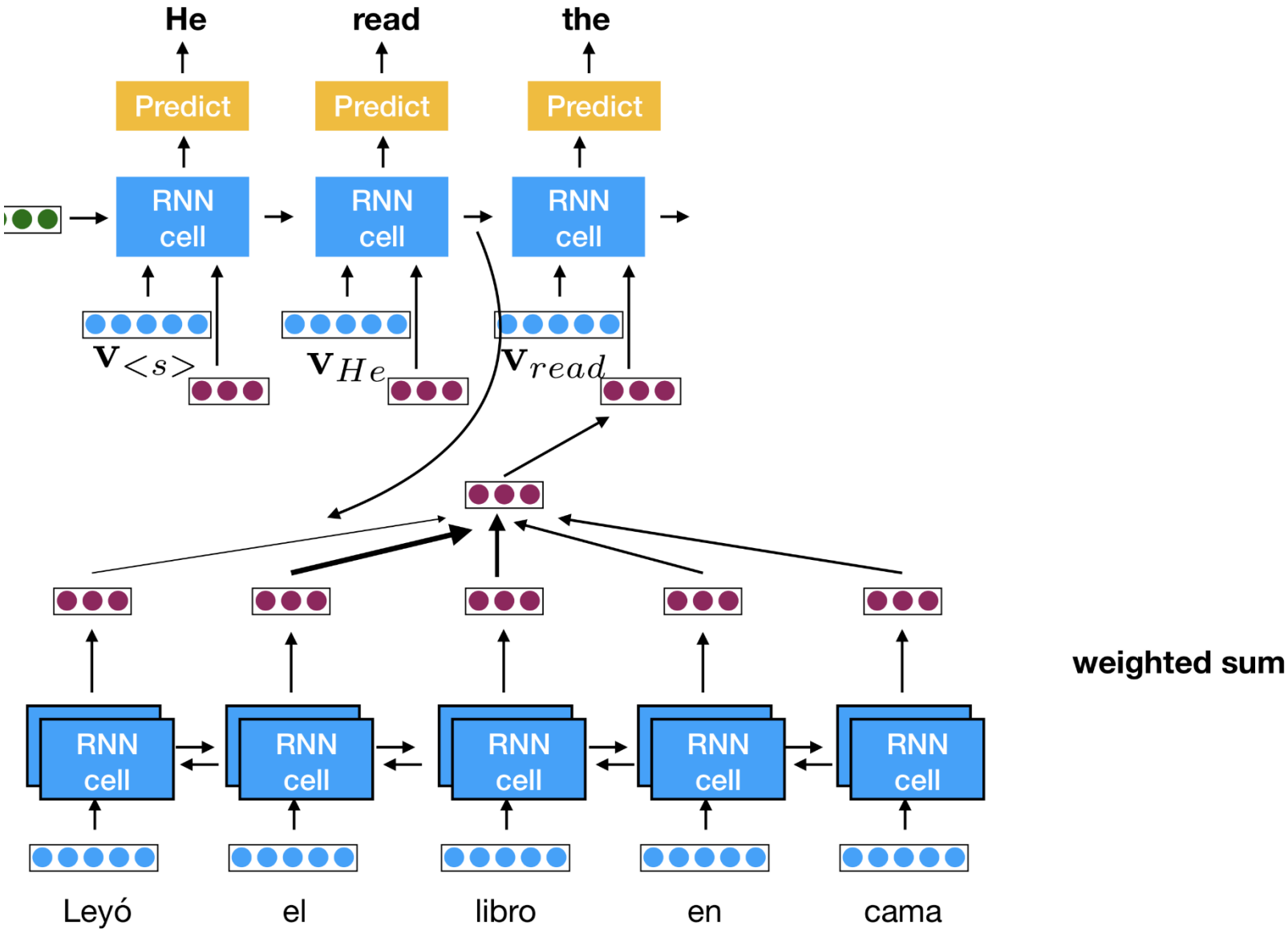
Seq2Seq with Attention



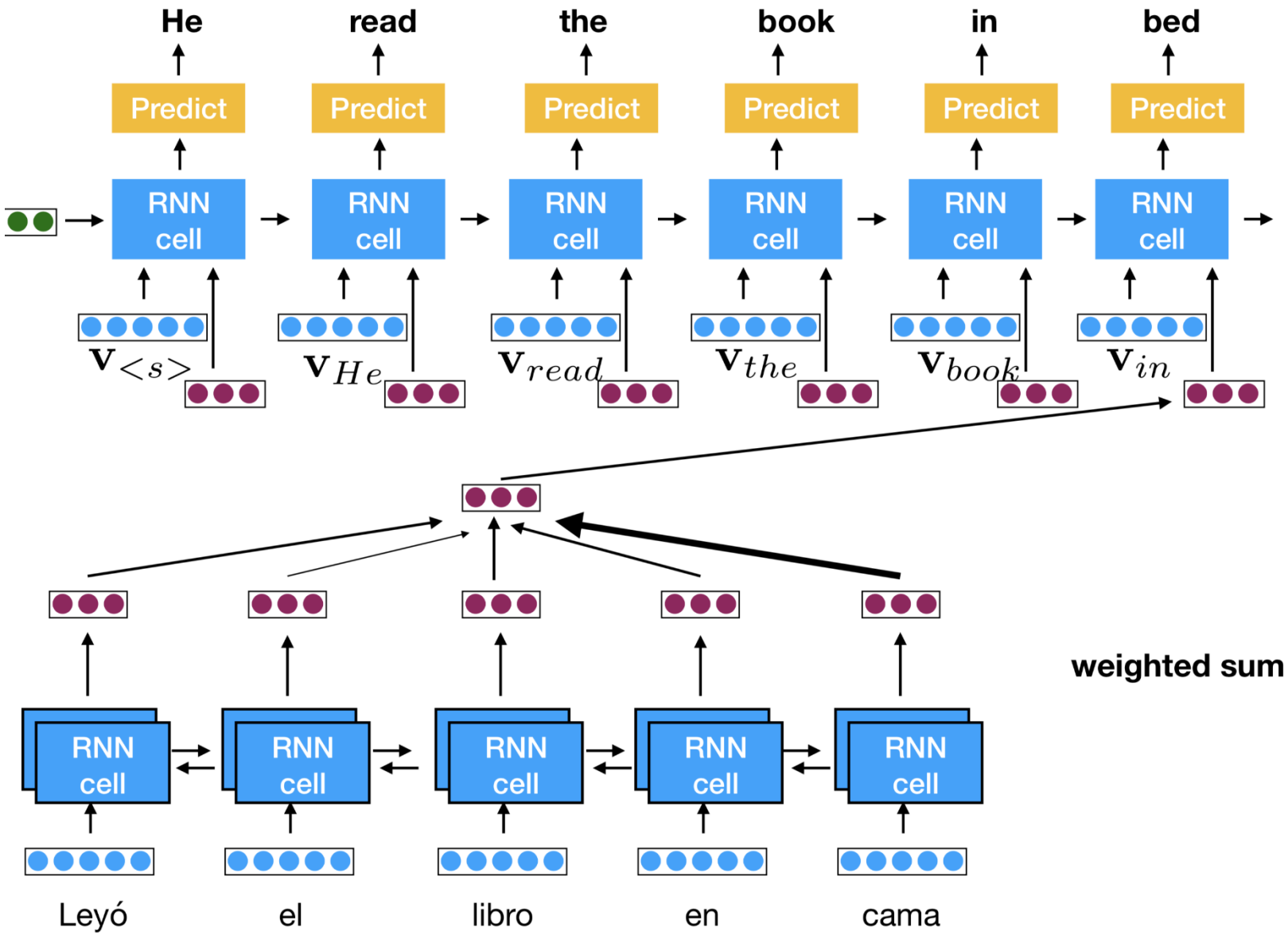
Seq2Seq with Attention



Seq2Seq with Attention



Seq2Seq with Attention



Attention: Use

- 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: The Math

- 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:

There are multiple ways to do this

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

  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*)

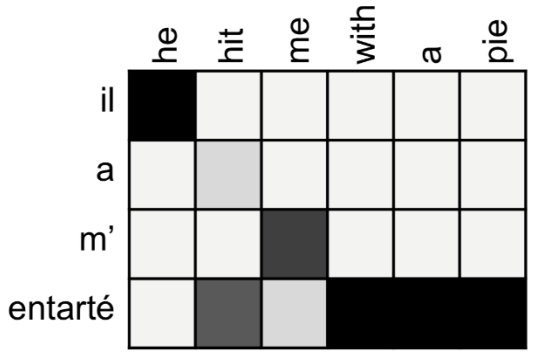
Attention: Variants

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- 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

Attention

- 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 helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Vocabulary Size Limitations

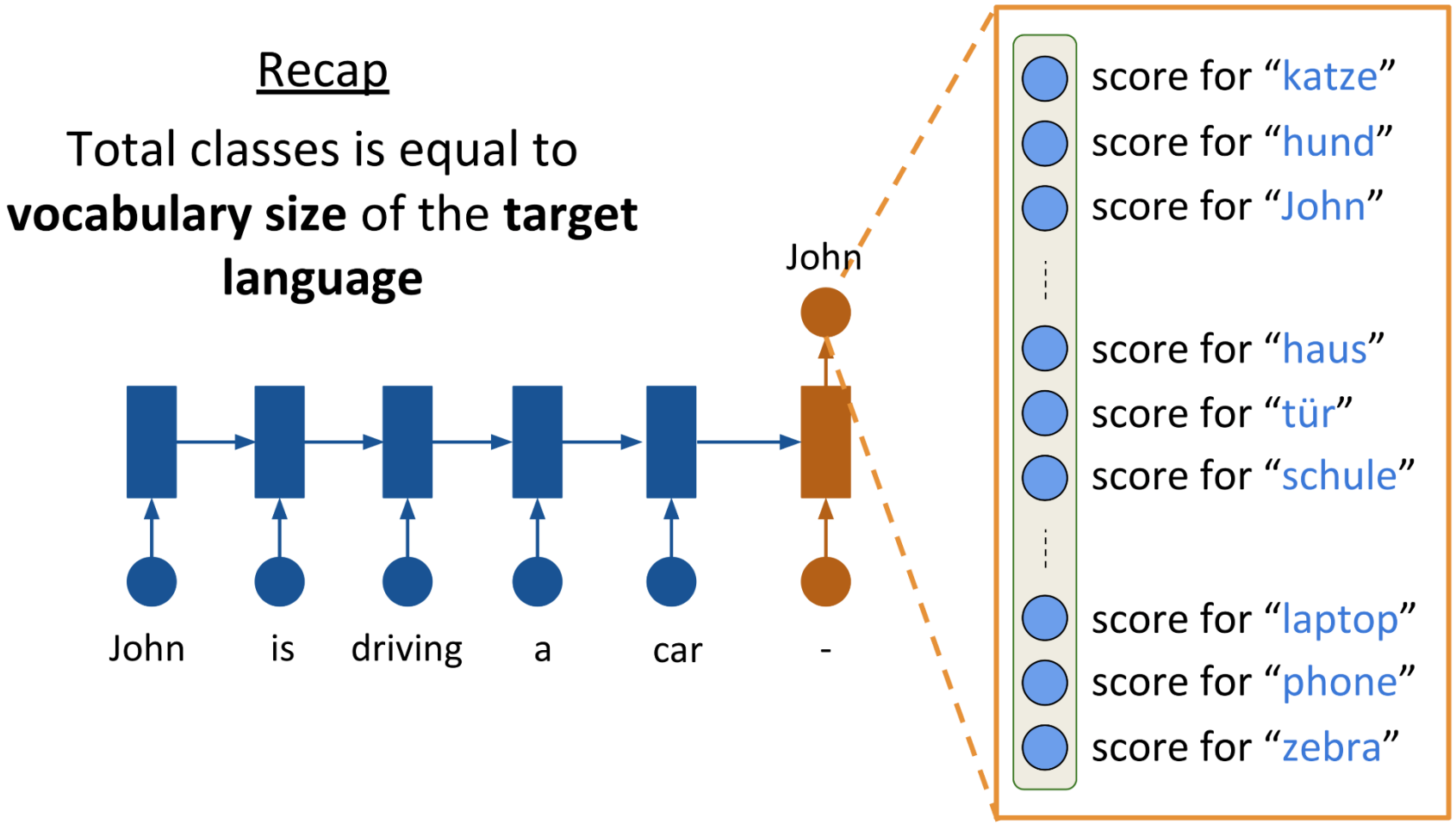
Vocabulary Size Limitations

MT is an open-vocabulary problem

- compounding and other productive morphological processes
  - they charge a carry-on bag fee.
  - sie erheben eine Hand|gepäck|gebühr.
- names
  - Obama(English; German)
  - Обама (Russian)
  - オバマ (o-ba-ma) (Japanese)
- technical terms, numbers, etc.

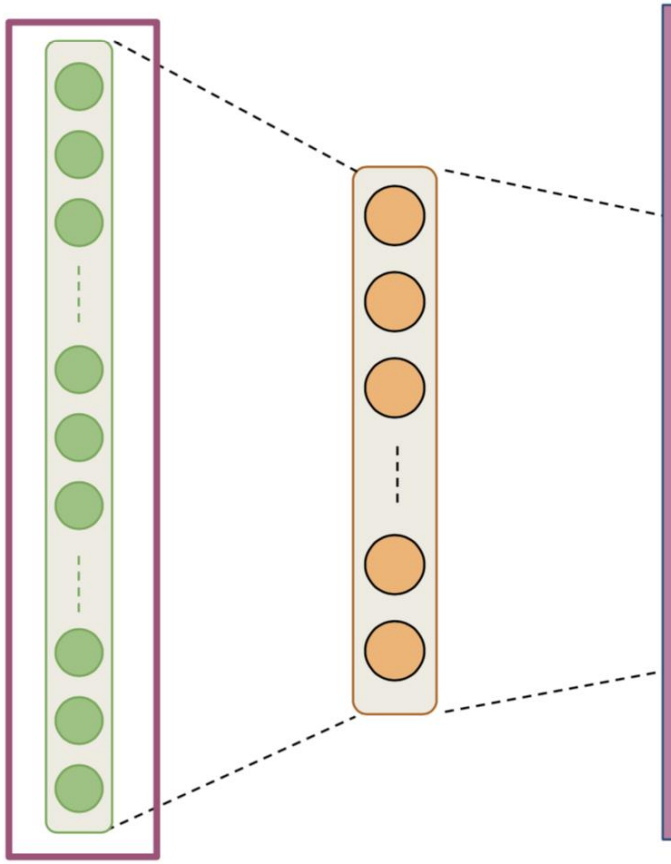
Vocabulary Size Limitations

The training time is sensitive to vocabulary size



Vocabulary Size Limitations: Source Side

- Training time of neural models is very sensitive to vocabulary size



Larger source side vocabulary means a longer one-hot vector!

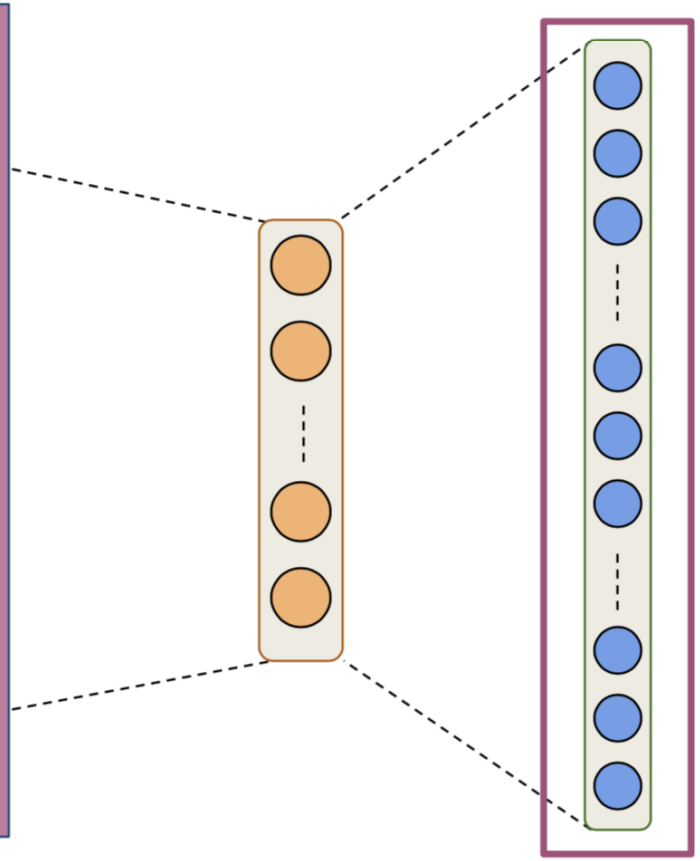
Number of parameters increases, and so does computation time for the associated multiplications

Vocabulary Size Limitations: Target Side

- Training time of neural models is very sensitive to vocabulary size

Larger target side vocabulary means a lot more scores and probabilities to compute!

Softmax is notoriously slow for large vectors, slowing down the entire training process



Vocabulary Size Limitations

- **Word-based** models suffer from data sparseness

Play Playing Played

- The above words are represented as three vocabulary units, when they mean very similar things
- Explicit or implicit handling of various language phenomena can reduce data sparseness

Easy Solution: Use <UNK>

- replace out-of-vocabulary words with UNK
- a vocabulary of 50 000 words covers 95% of text

this gets you 95% of the way...
... if you only care about automatic metrics

why 95% is not enough

rare outcomes have high self-information

source	The **indoor temperature** is very pleasant.
reference	Das **Raumklima** ist sehr angenehm.
[Bahdanau et al., 2015]	Die **UNK** ist sehr angenehm. ✗

Easy Solution: Use <UNK>

- A typical solution
  - only keep the most frequent source and target vocabulary words
  - replace the infrequent words with a unique token, say <UNK>

- Issues
  - Increased number of unknowns during testing
  - No explicit handling of unknown words that were not present in the training corpus

Approximative Softmax

approximative softmax [Jean et al., 2015]

compute softmax over "active" subset of vocabulary
→ smaller weight matrix, faster softmax

- at training time: vocabulary based on words occurring in training set partition
- at test time: determine likely target words based on source text (using cheap method like translation dictionary)

limitations

- allows larger vocabulary, but still not open
- network may not learn good representation of rare words

Backoff Models

back-off models [Jean et al., 2015, Luong et al., 2015]

- replace rare words with UNK at training time
- when system produces UNK, align UNK to source word, and translate this with back-off method

source	The **indoor temperature** is very pleasant.	
reference	Das **Raumklima** ist sehr angenehm.	
[Bahdanau et al., 2015]	Die **UNK** ist sehr angenehm.	✗
[Jean et al., 2015]	Die **Innenpool** ist sehr angenehm.	✗

limitations

- compounds: hard to model 1-to-many relationships
- morphology: hard to predict inflection with back-off dictionary
- names: if alphabets differ, we need transliteration
- alignment: attention model unreliable

Alternative: Subword-Based Methods

Words

Play Playing Played

Common Sub-word

Play

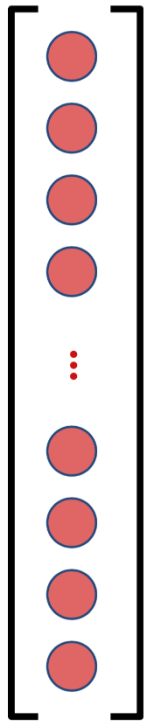
Subword Representations

- Character based
- Character-LSTM based
- Character-CNN based
- Frequency based sub-word segmentation
  - BPE
  - WordPiece
- Hybrid representations

Character-Based Modeling

Character-Based Modeling

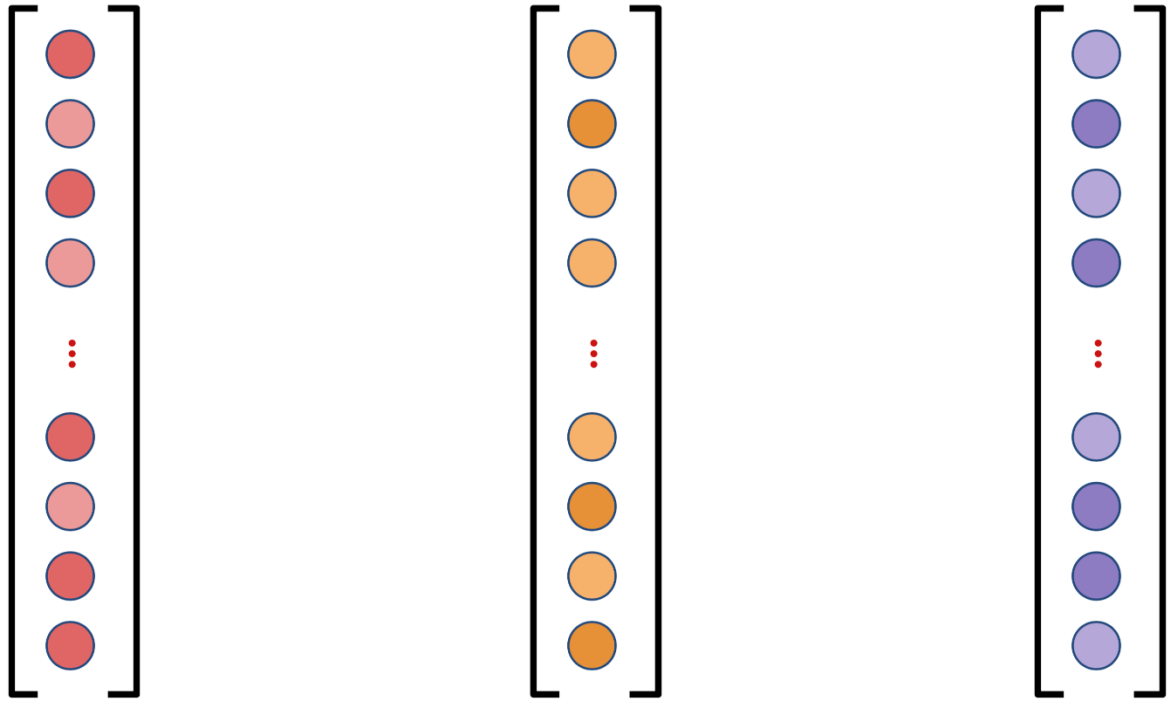
A *word embedding* is represented as a vector per word:



Embedding of
“car”

Character-Based Modeling

A *character embedding* is represented as a vector per character:



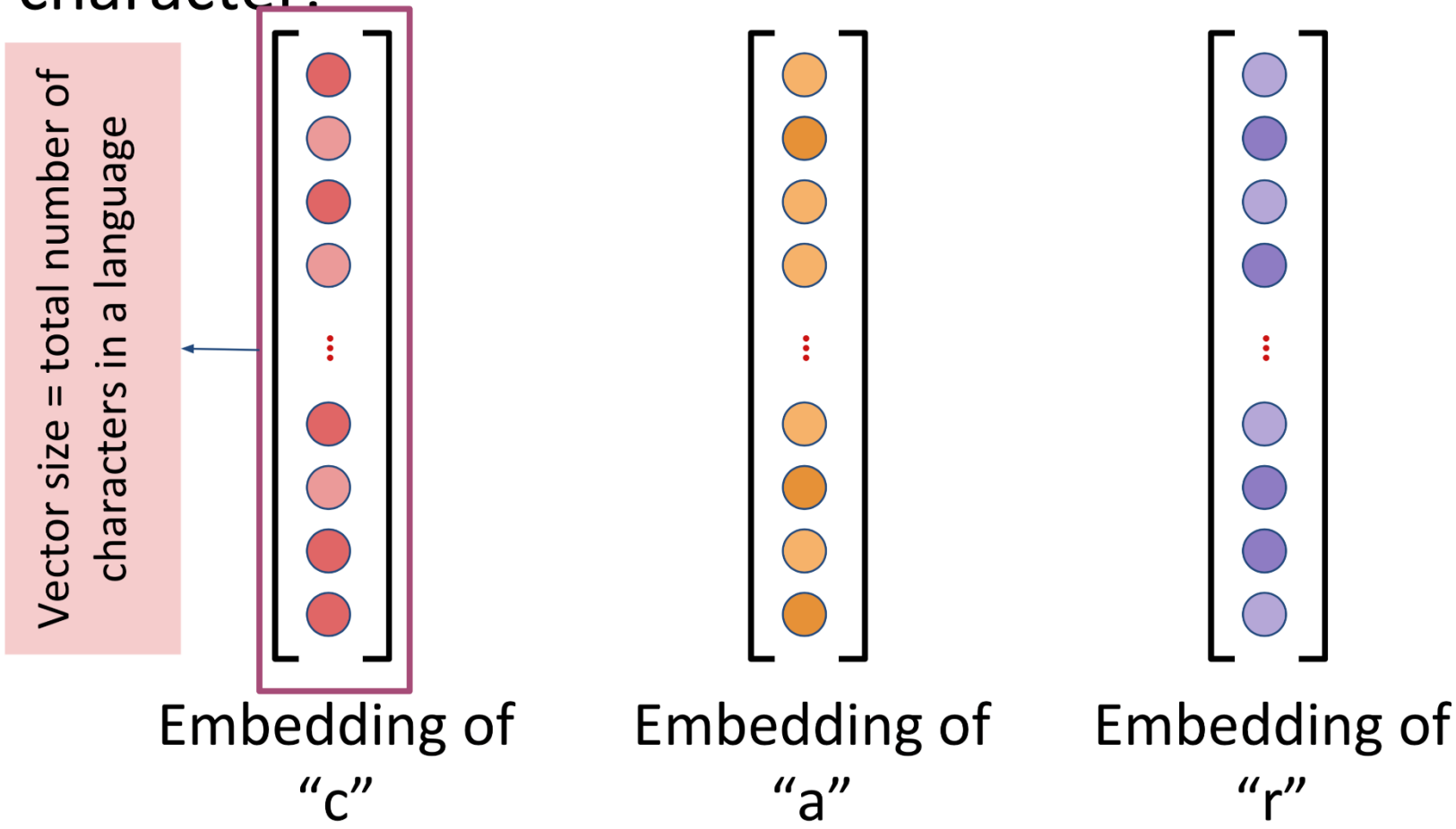
Embedding of "c"

Embedding of "a"

Embedding of "r"

Character-Based Modeling

A *character embedding* is represented as a vector per character:



Character-Based Modeling

Q: How can we use these practically in our models?

Character-Based Modeling

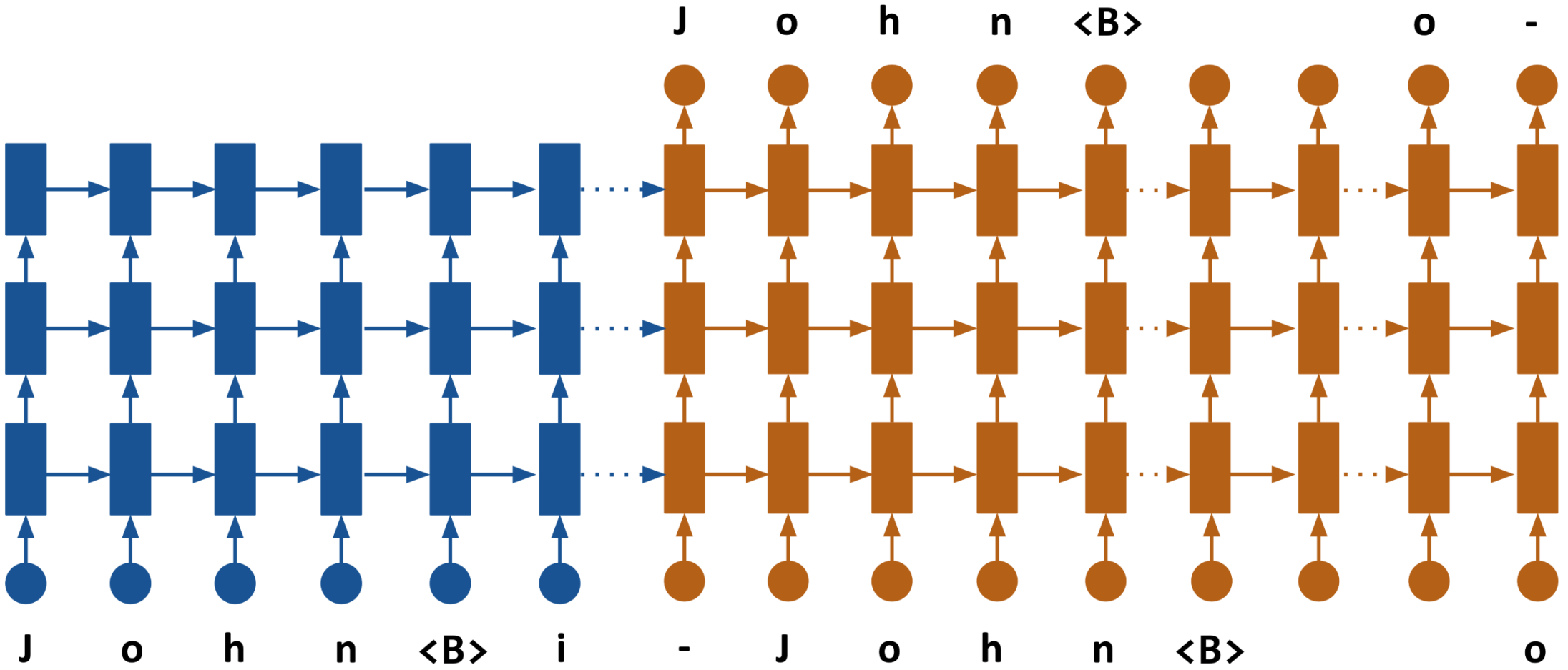
Q: How can we use these practically in our models?

A: Just split the words into characters, and use a special symbol to keep track of word boundaries

J o h n <B> i s <B> d r i v i n g <B>

Keep everything else the same, our vocabulary will now be all characters and the special symbol!

Character-Based Modeling



Character-Based Modeling

- Pros
  - Vocabulary size is just the **total unique characters** in the corpus
  - Model learns to handle morphological variations
    - play, playing, played
- Cons
  - Conceptually, characters do not represent a semantic unit in contrast to words
  - Long sequence length
    - makes prediction at train and test time very slow
    - a **large number** of softmax operations (albeit smaller operations)

Character-Based Modeling

- advantages:
  - (mostly) open-vocabulary
  - no heuristic or language-specific segmentation
  - neural network can conceivably learn from raw character sequences
- drawbacks:
  - increasing sequence length slows training/decoding
    (reported x2–x4 increase in training time)
  - naive char-level encoder-decoders are currently resource-limited
    [Luong and Manning, 2016]
- open questions
  - on which level should we represent meaning?
  - on which level should attention operate?

Mixed Character-Word Models

Hierarchical Backoff Model

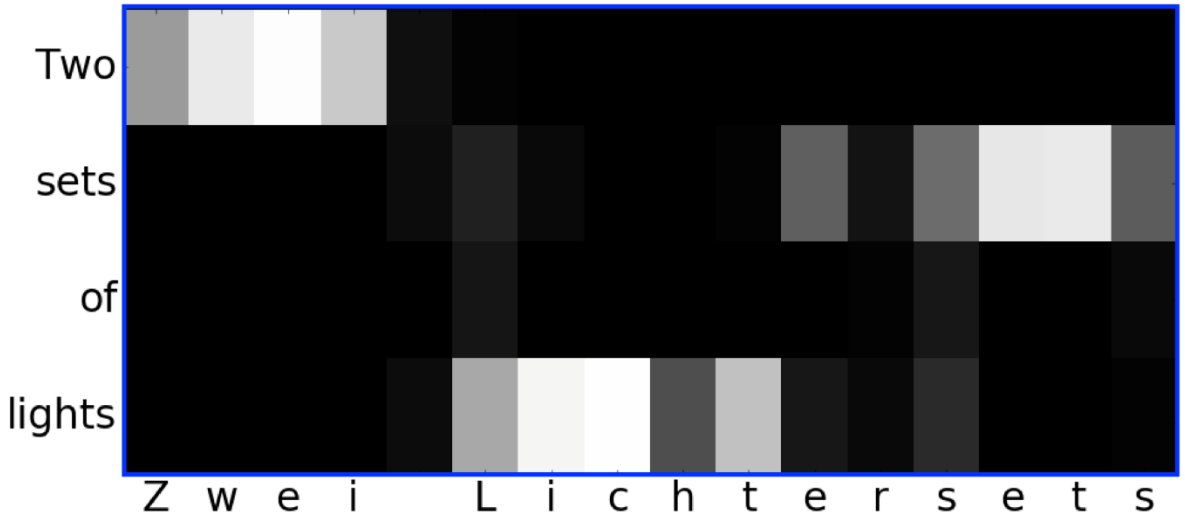
hierarchical model: back-off revisited [Luong and Manning, 2016]

- word-level model produces UNKs
- for each UNK, character-level model predicts word based on word hidden state
- pros:
  - prediction is more flexible than dictionary look-up
  - more efficient than pure character-level translation
- cons:
  - independence assumptions between main model and backoff model

Character-Level Output

character-level output [Chung et al., 2016]

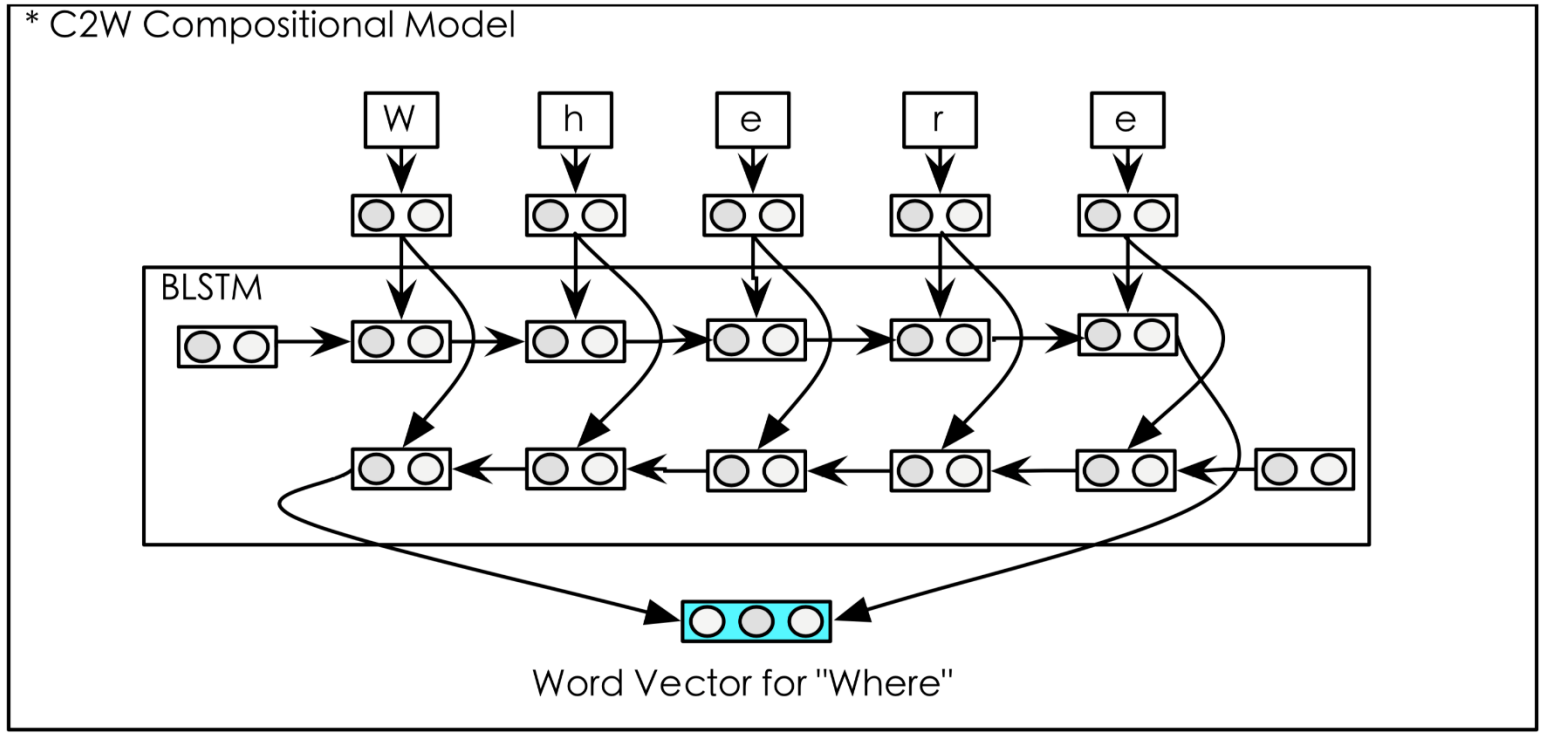
- no word segmentation on target side
- encoder is BPE-level
- good results for EN→{DE,CS,RU,FI}
- long training time (≈ x2 compared to BPE-level model)



Character-Level Input LSTM

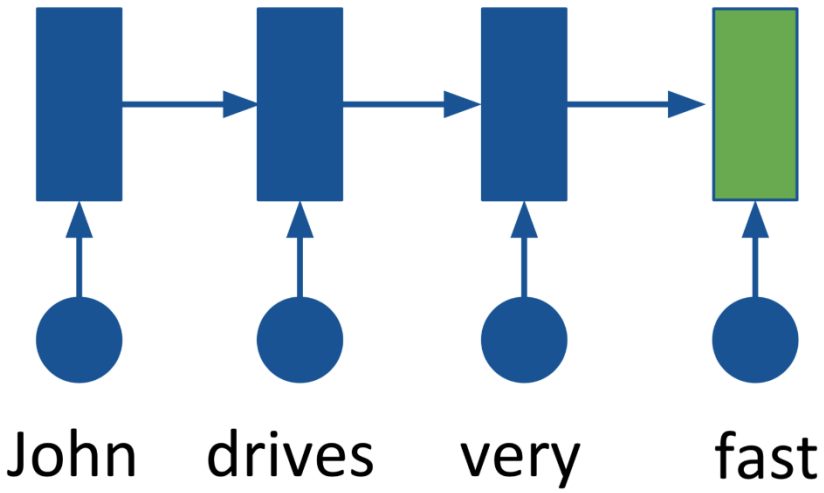
character-level input [Ling et al., 2015]

hierarchical representation: RNN states represent words, but their representation is computed from character-level LSTM



Character-Level Input LSTM

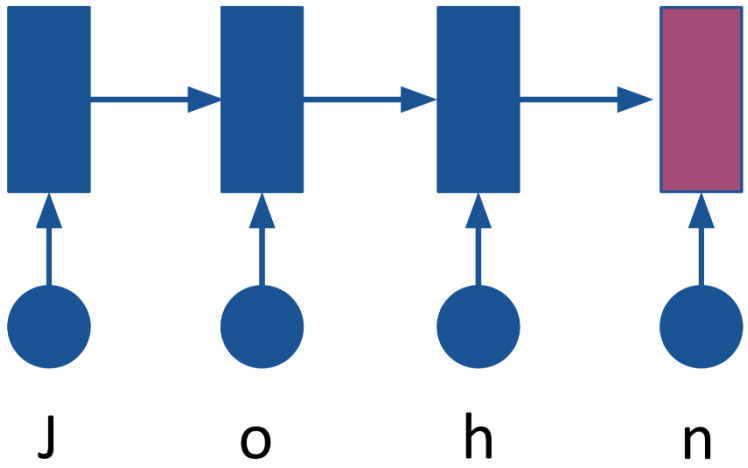
We can learn word embeddings based on character LSTMs



Summary vector with summary of all the **words**

Character-Level Input LSTM

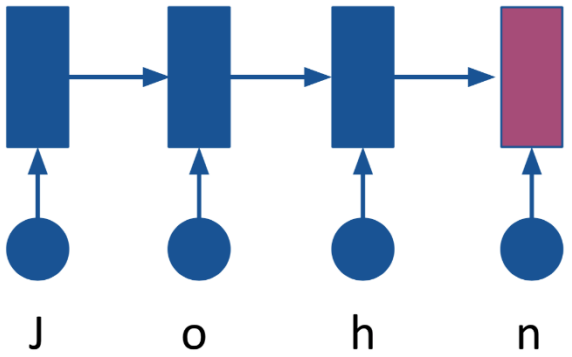
We can learn word embeddings based on character LSTMs



Summary vector with summary of all the **characters**

Character-Level Input LSTM

We can learn word embeddings based on character LSTMs

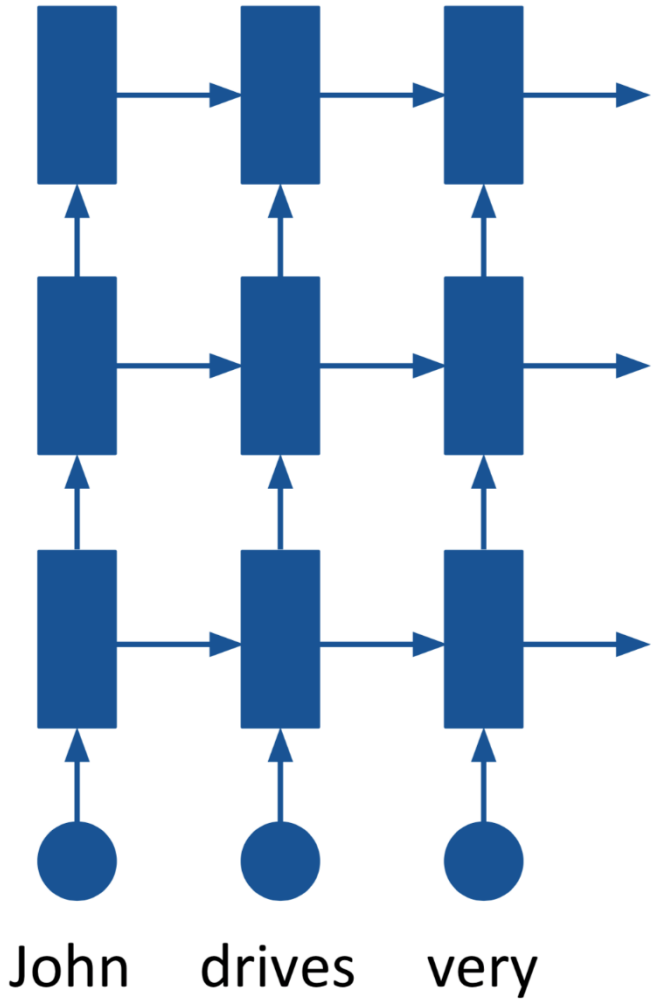


Summary vector with summary of all the **characters**

We can also consider this summary as an embedding of “John”

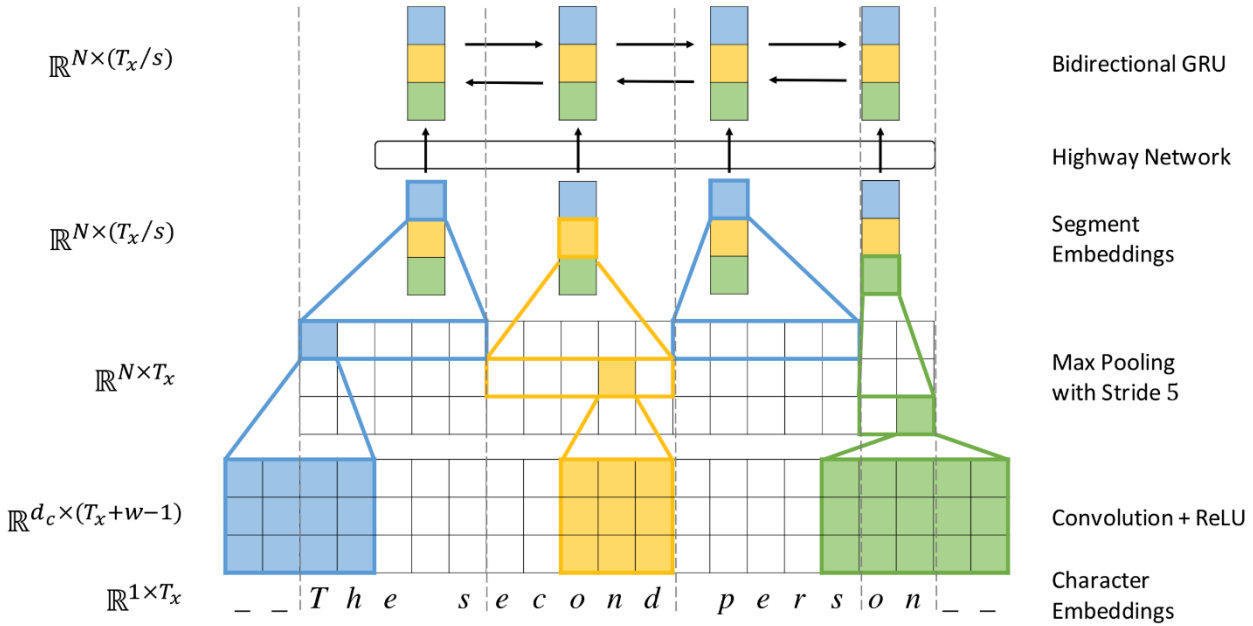
Character-Level Input LSTM

Usually, our first layer is a feed-forward layer which we treat as embeddings

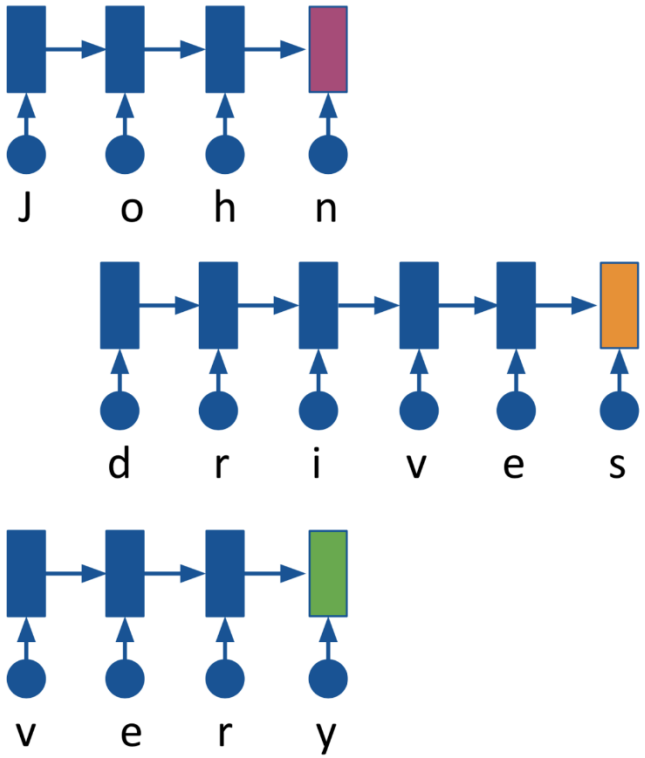


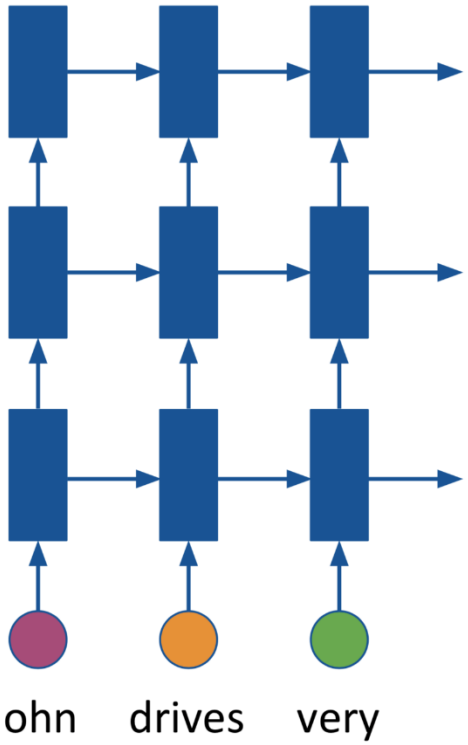
Fully Character-Level NMT [Lee et al., 2016]

- goal: get rid of word boundaries
- character-level RNN on target side
- source side: convolution and max-pooling layers



Character-Level Input LSTM





In this case, we will use our computed embeddings in our encoder or decoder

Mixed Character-Word Models

Pros

- Word representations using characters are richer than word-based embeddings
  - incorporate morphological information
    - play, playing, played, etc.
- Works well in translating morphologically rich languages
- No issue of vocabulary size
  - vocabulary is equal to total number of characters

Mixed Character-Word Models

Cons

- Significantly slower than normal feed-forward layers
- Number of parameters is higher

Sub-Words: Frequency-Based

Frequency-Based Subword Units

- Byte Pair Encoding
  - Sennrich, Haddow, Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016
- Word Piece Model
  - Wu et al. 2016, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Frequency-Based Subword Units

- Byte Pair Encoding
  - Sennrich, Haddow, Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016
- Word Piece Model
  - Wu et al. 2016, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Byte-Pair Encoding

- Frequency-based segmentation
  - split words to reduce vocabulary size
- Words have various overlapping segments
  - Play Playing Played Playstation
- Segment them based on frequency
  - play
  - ing
  - ed
  - station
- Essentially, mapping all forms of “Play” to the root form

Byte-Pair Encoding

bottom-up character merging

- starting point: character-level representation
  → computationally expensive
- compress representation based on information theory
  → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  → controls vocabulary size

Byte-Pair Encoding

"e s" comes 9 times together

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, s, t, i, d

Byte-Pair Encoding

"e s" comes 9 times together

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**

Byte-Pair Encoding

Similarly, “l o” comes 7 times together and “es t” comes 9 times together

Dictionary

5 **lo** w
2 **lo** w e r
6 n e w **est**
3 w i d **est**

Vocabulary

l, o, w, e, r, n, s, t, i, d, es, **lo**, **est**

Byte-Pair Encoding

Total number of merge operations done = 3

Dictionary

5	**lo** w
2	**lo** w e r
6	n e w **est**
3	w i d **est**

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**, **lo**, **est**

Byte-Pair Encoding

Vocabulary size = initial character size + number of merge operations

Dictionary

5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**, **lo**, **est**

Byte-Pair Encoding

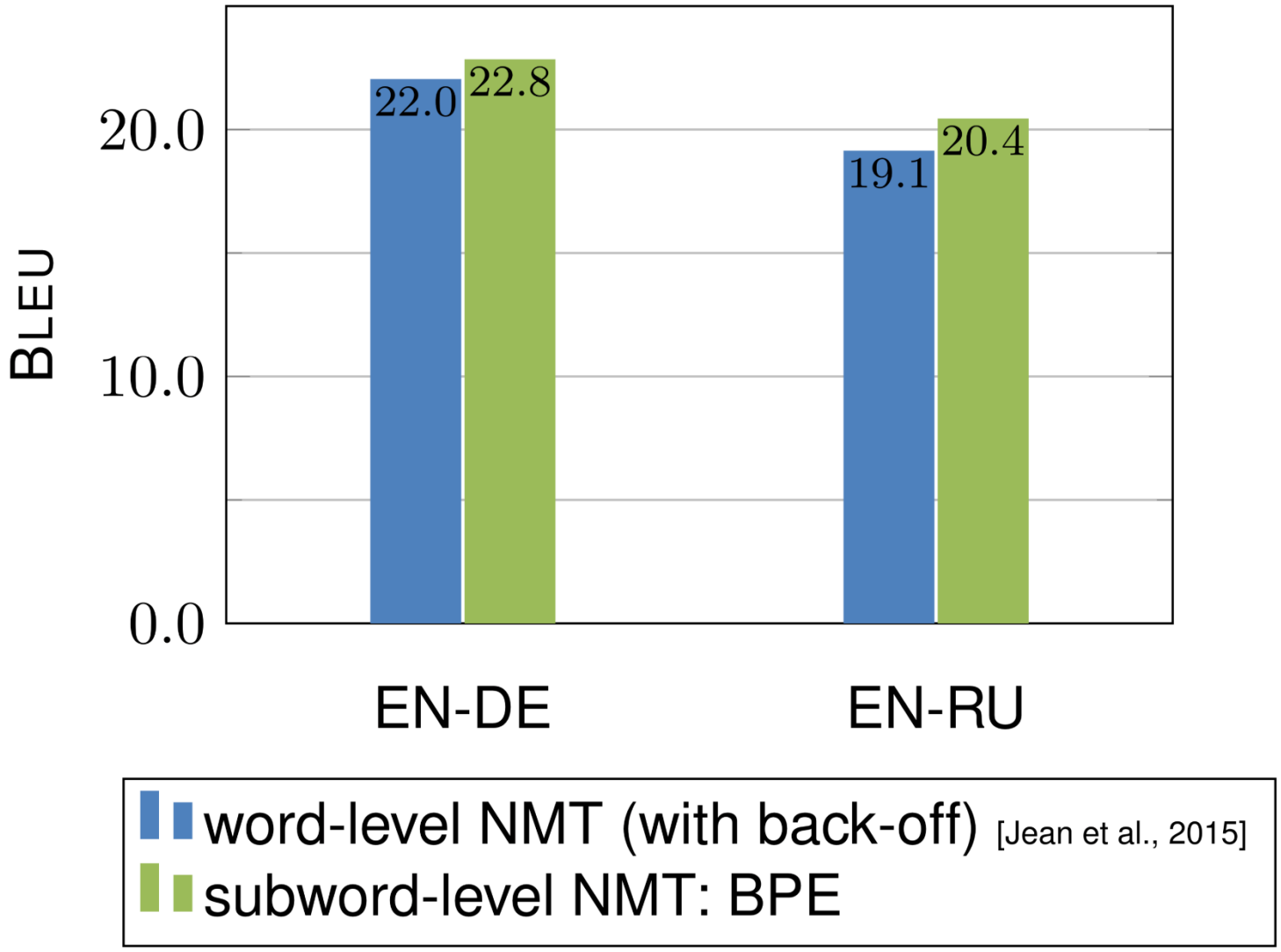
- One hyperparameter: number of operations
  - Defines vocabulary size, e.g., 50k operations limit the vocab to 50k (+ number of characters)
- Can handle unseen forms that happen due to morphological variation
  - drive
  - driving
  - driven

Byte-Pair Encoding

Downside

- Segmentation is not morphologically consistent
  - drive, driving, driven may have different segmentations
- Cannot benefit from same-root forms
  - Segmentation is learned independent of the task

Subword NMT: Translation Quality



Examples

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
word-level (with back-off)	Forschungsinstitute
character bigrams	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE	Gesundheits|forsch|ungsin|stitute
source	rakfisk
reference	ракфиска (rakfiska)
word-level (with back-off)	rakfisk → UNK → rakfisk
character bigrams	ra|kf|is|k → ра|кф|ис|к (ra|kf|is|k)
BPE	rak|f|isk → рак|ф|иска (rak|f|iska)

Byte-Pair Encoding

Super simple to train:

https://github.com/rsennrich/subword-nmt

```
./learn_bpe.py -s {num_operations} < {train_file} > {codes_file}
./apply_bpe.py -c {codes_file} < {test_file}
```

Byte-Pair Encoding

- BPE-level subword segmentation is currently the most widely used technique for open-vocabulary NMT
- character-level models are theoretically attractive, but currently require specialized architectures and more computational resources

References

References (1)

Bahdanau, D., Cho, K., and Bengio, Y. (2015).

Neural Machine Translation by Jointly Learning to Align and Translate.

In Proceedings of the International Conference on Learning Representations (ICLR).

Chung, J., Cho, K., and Bengio, Y. (2016).

A Character-level Decoder without Explicit Segmentation for Neural Machine Translation.

CoRR, abs/1603.06147.

Gage, P. (1994).

A New Algorithm for Data Compression.

C Users J., 12(2):23–38.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015).

On Using Very Large Target Vocabulary for Neural Machine Translation.

In

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o

pages 1–10, Beijing, China. Association for Computational Linguistics.

Lee, J., Cho, K., and Hofmann, T. (2016).

Fully Character-Level Neural Machine Translation without Explicit Segmentation.

ArXiv e-prints.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015).

Character-based Neural Machine Translation.

ArXiv e-prints.

References (2)

Luong, M.-T. and Manning, D. C. (2016).

Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models.

In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1054–1063. Association for Computational Linguistics.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015).

Addressing the Rare Word Problem in Neural Machine Translation.

In

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o pages 11–19, Beijing, China. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016).

Neural Machine Translation of Rare Words with Subword Units.

In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany.