UNC CHARLOTTE

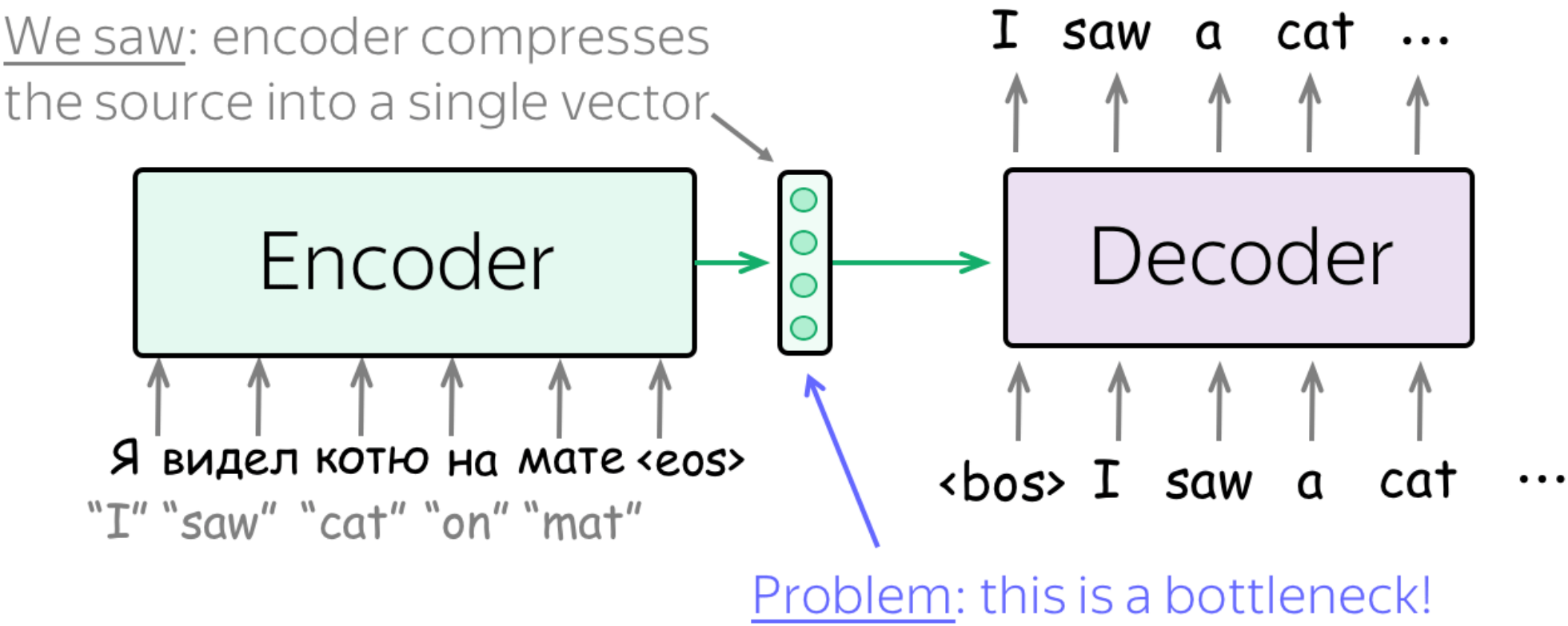
The WILLIAM STATES LEE COLLEGE *of* ENGINEERING

Introduction to ML
Lecture 10: Attention

Hamed Tabkhi
Department of Electrical and Computer Engineering,
University of North Carolina Charlotte (UNCC)
htabkhiv@uncc.edu

Problem

- Problem: Fixed source representation is suboptimal:
  (i) for the encoder, it is hard to compress the sentence
  (ii) for the decoder, different steps are seeing the same information (context), while they may need different information at different steps.



UNC CHARLOTTE

Problem

- In the models we looked at so far, the encoder compressed the whole source sentence into a single vector.
- This can be very hard - the number of possible source sentences (hence, their meanings) is infinite.
- When the encoder is forced to put all information into a single vector, it is likely to forget something.
- Not only it is hard for the encoder to put all information into a single vector - this is also hard for the decoder.
- The decoder sees only one representation of source. However, at each generation step, different parts of source can be more useful than others.
- But in the current setting, the decoder has to extract relevant information from the same fixed representation - hardly an easy thing to do.

Attention: A High-Level View

- Attention: At different steps, let a model "focus" on different parts of the input.

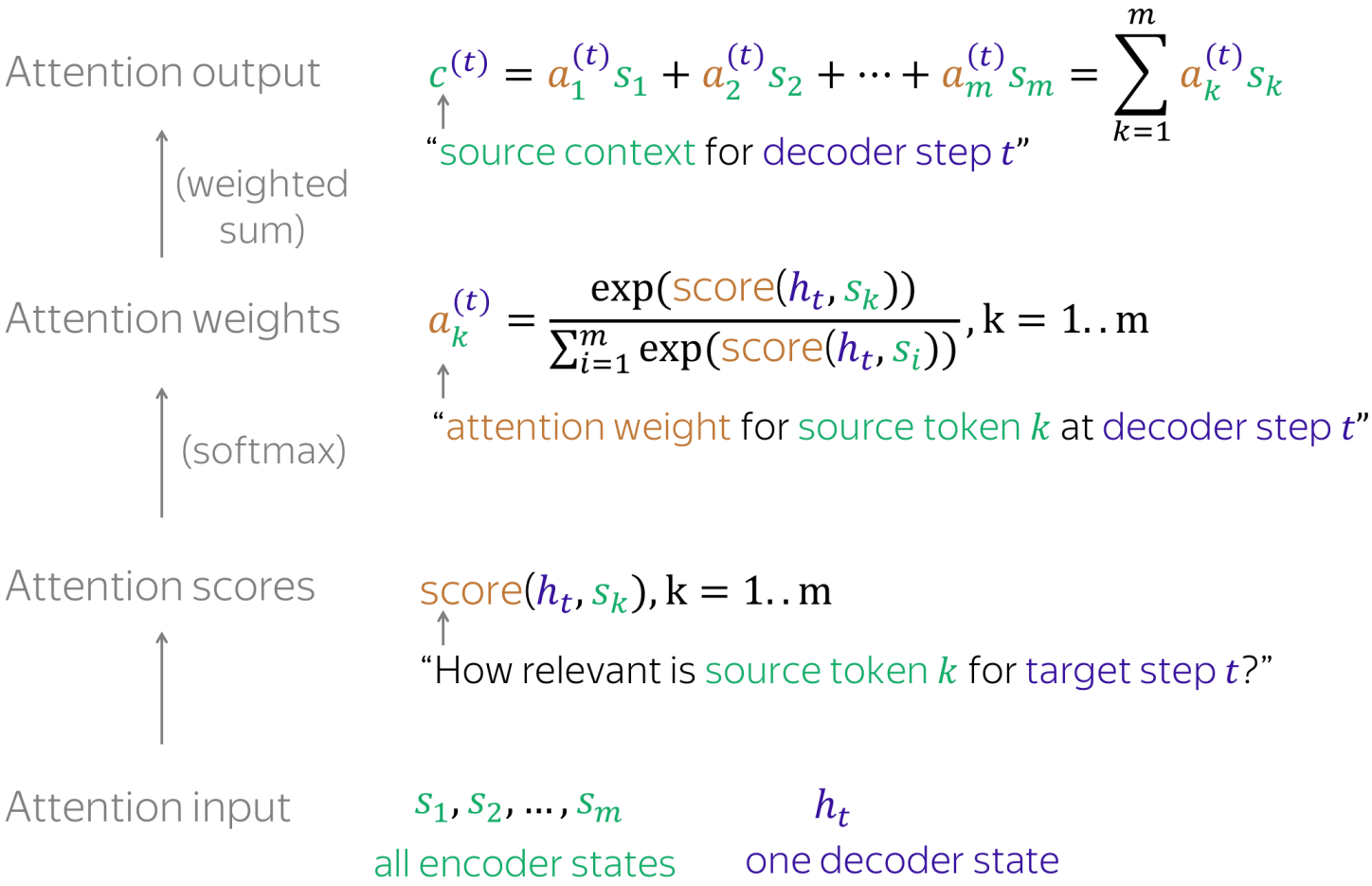
- Attention was introduced in the paper Neural Machine Translation by Jointly Learning to Align and Translate to address the fixed representation problem.

- An attention mechanism is a part of a neural network. At each decoder step, it decides which source parts are more important.

- **In this setting, the encoder does not have to compress the whole hidden states over time steps into a single vector - it gives representations for all source hidden states (for example, all RNN states instead of the last one).**

Attention: A High-Level View

The general computation scheme is shown below.



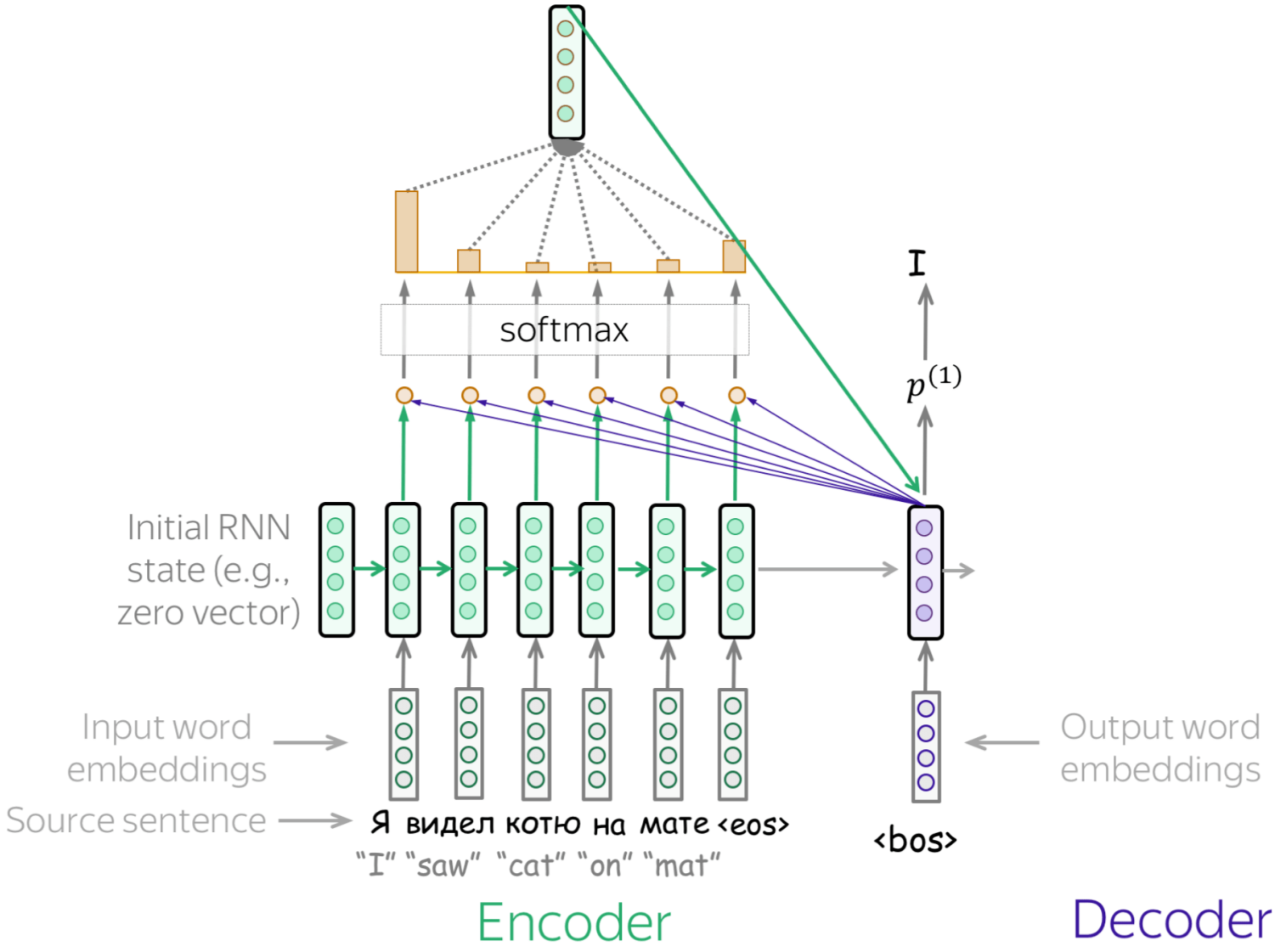
Each decoder step t will have its own set of attention input, defining what is the importance of encoder steps for the decoder step

UNC CHARLOTTE

Everything is differentiable - learned end-to-end!

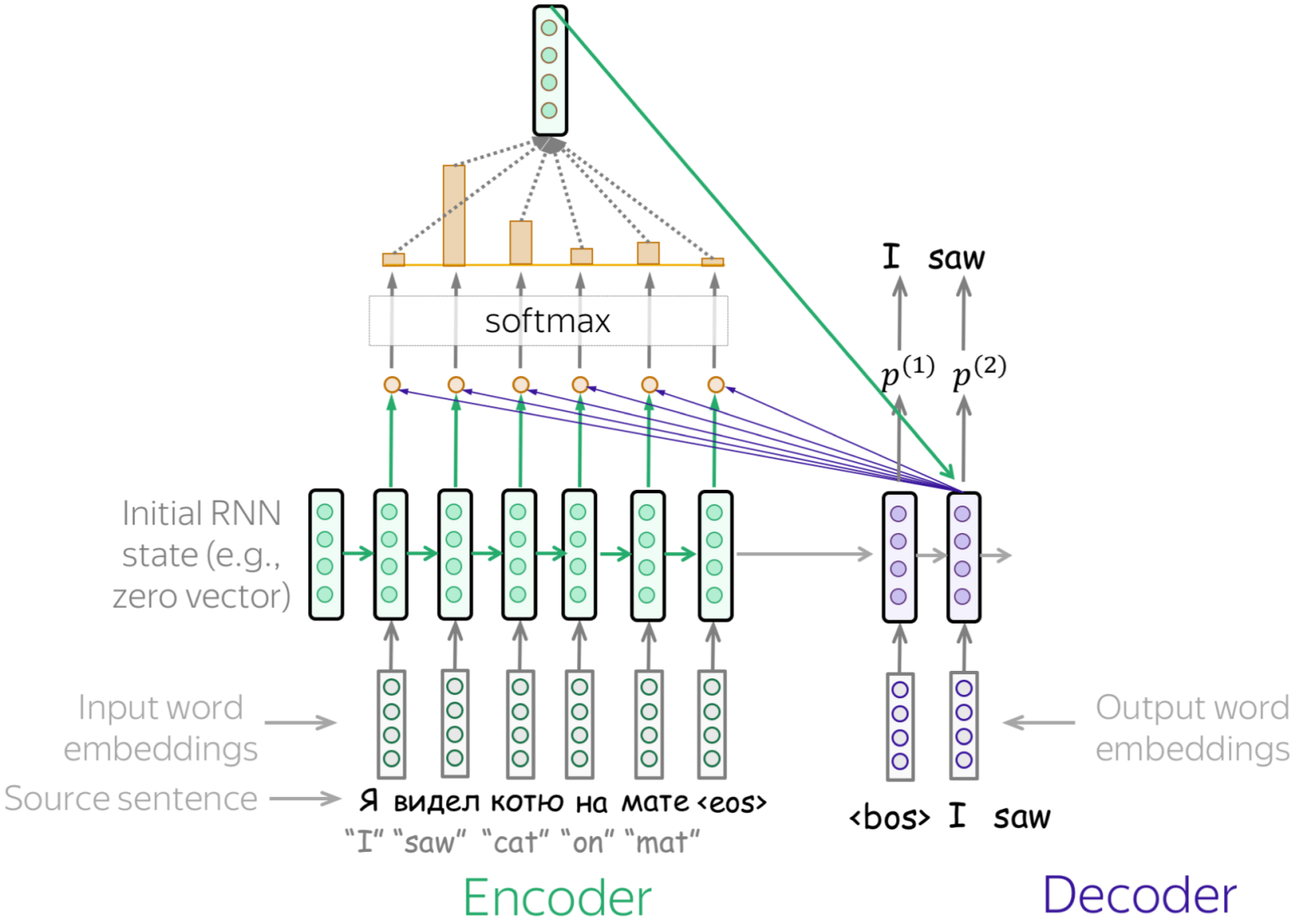
- **The main idea that a network can learn which input parts are more important at each step.**
- Since everything here is differentiable (attention function, softmax, and all the rest), a model with attention can be trained end-to-end.
- Differentiability means that gradient decent would work!
- **You don't need to specifically teach the model to pick the words you want - the model itself will learn to pick important information.**

Everything is differentiable - learned end-to-end!

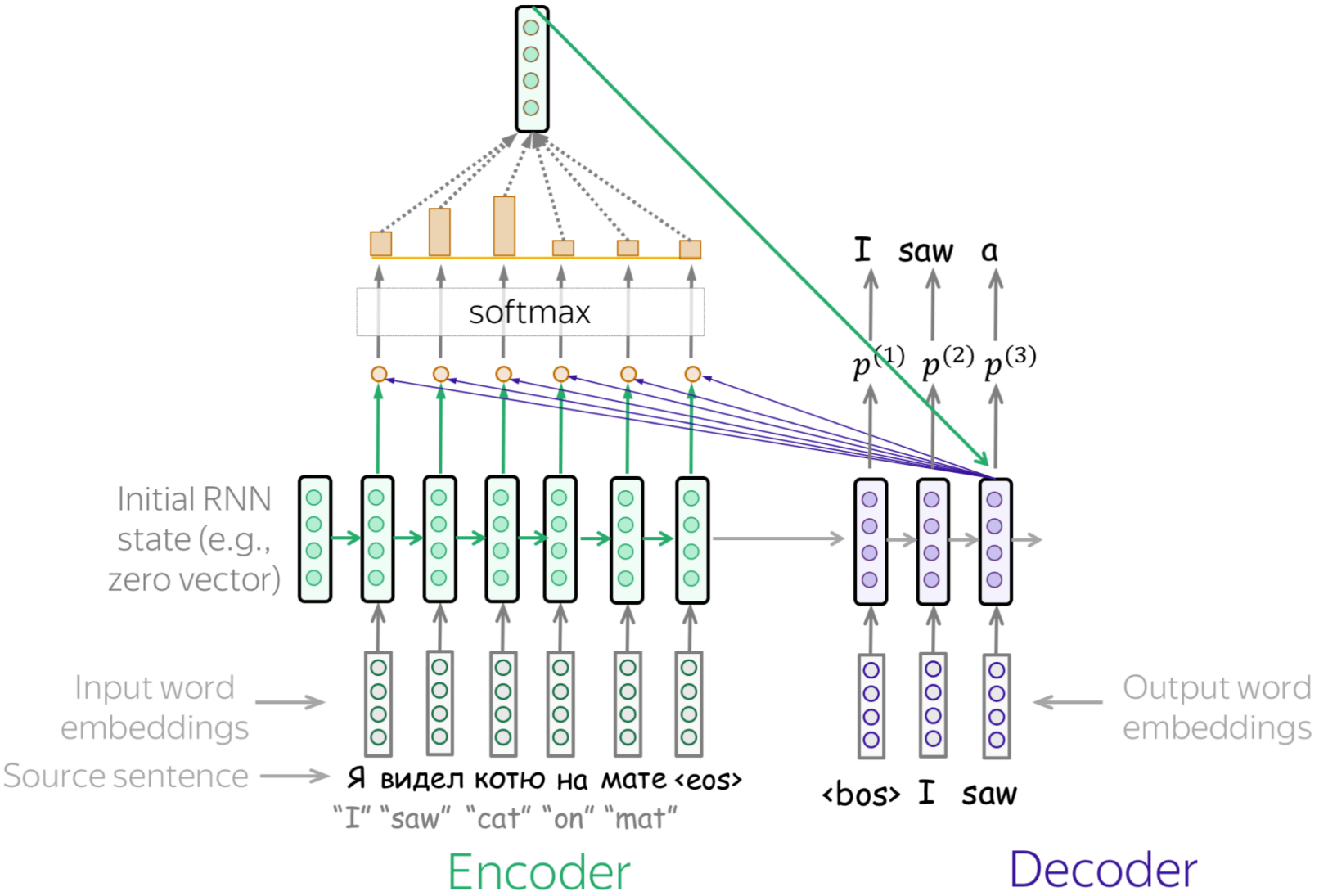


Basically, we are training simultaneously a classifier to pick the most important (context) from encoder, per each decoder stage!

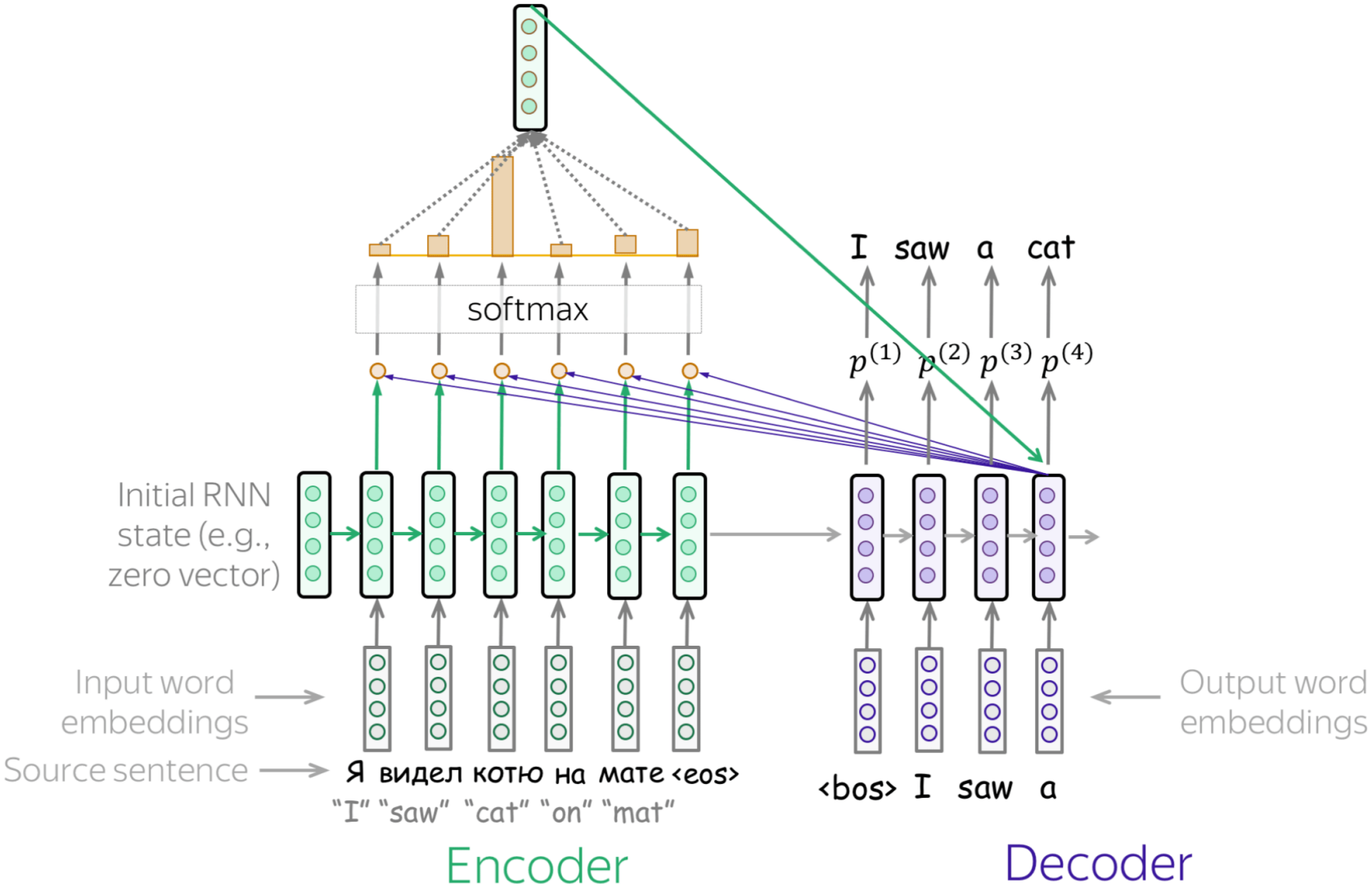
UNC CHARLOTTE



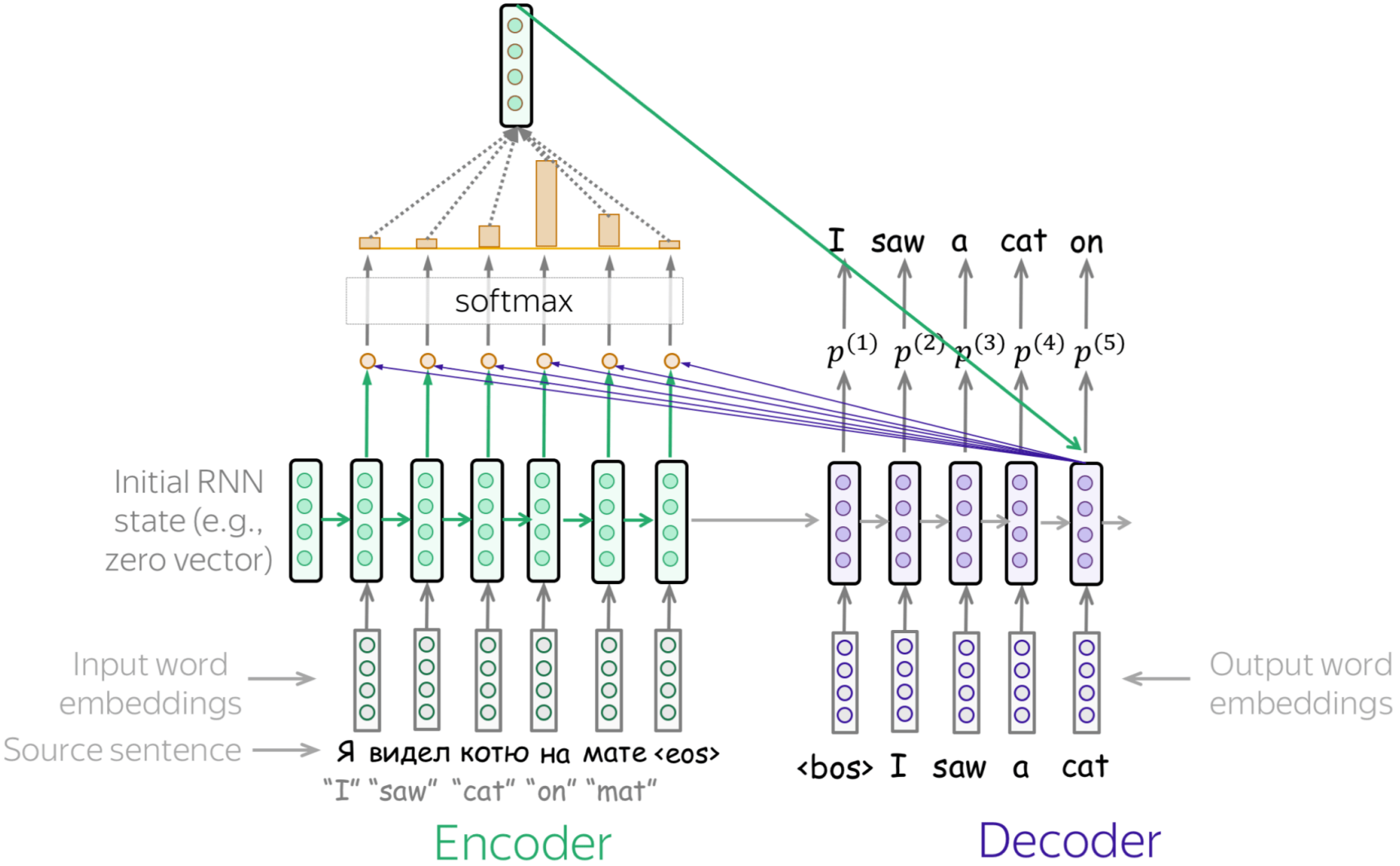
UNC CHARLOTTE



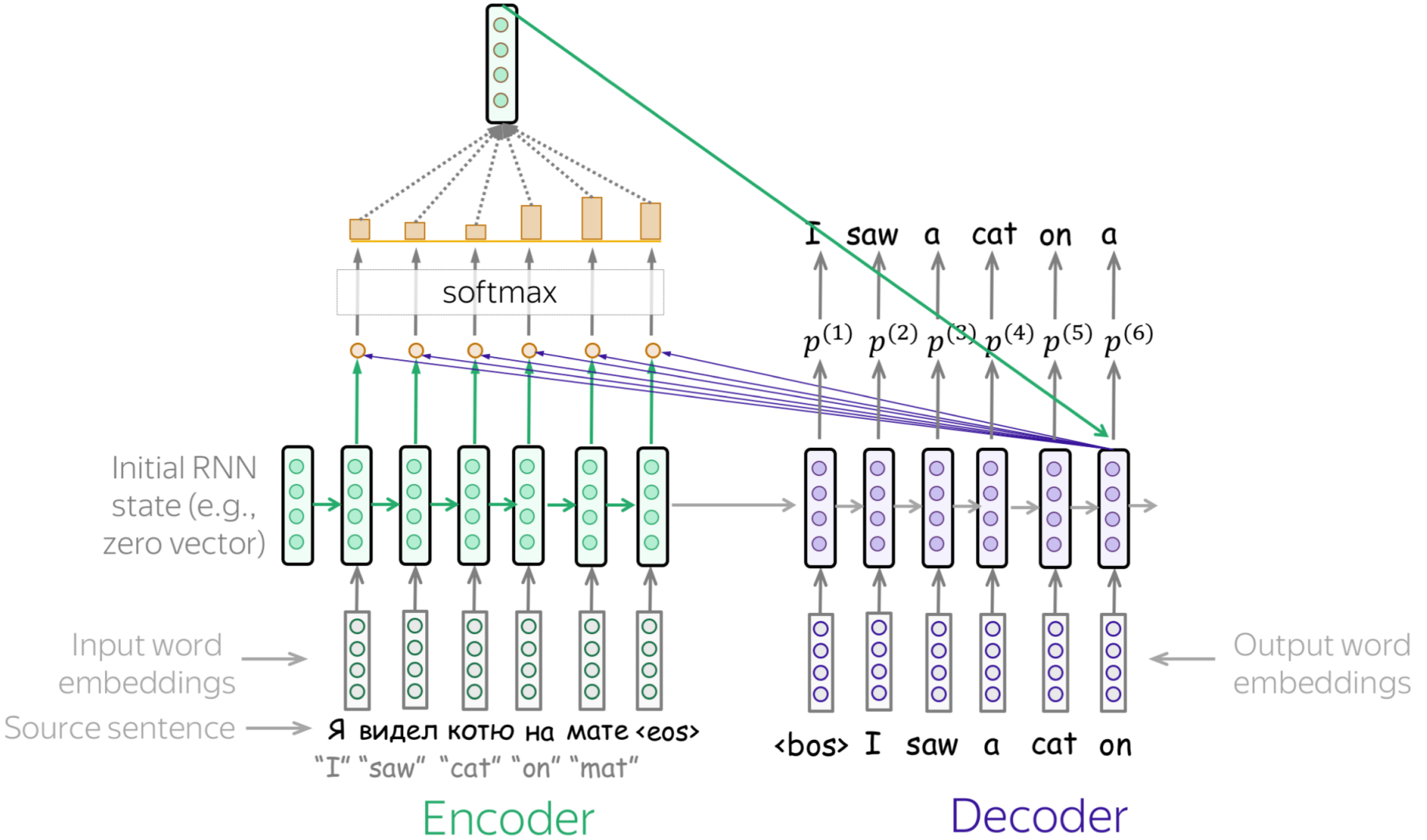
UNC CHARLOTTE



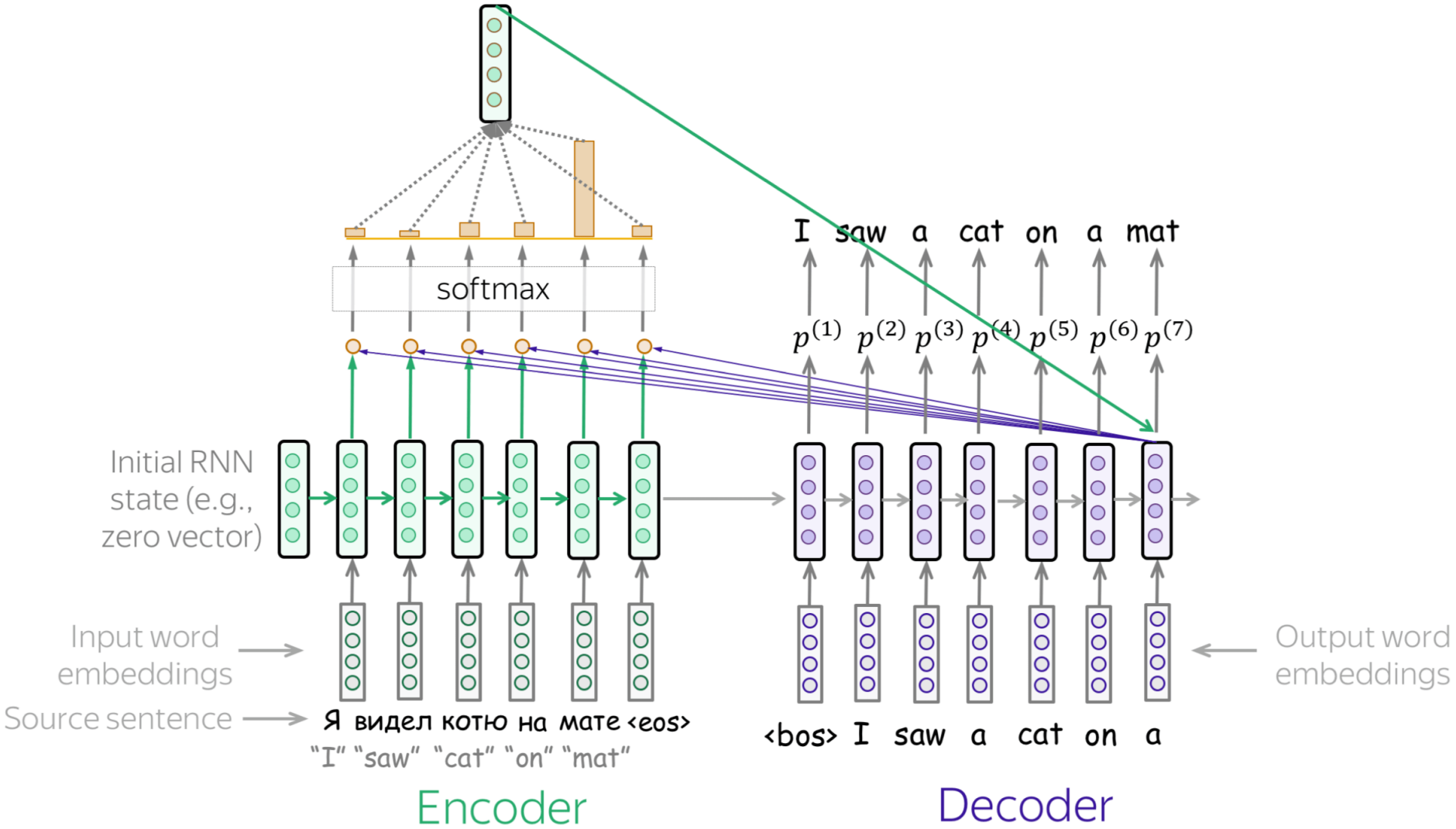
UNC CHARLOTTE



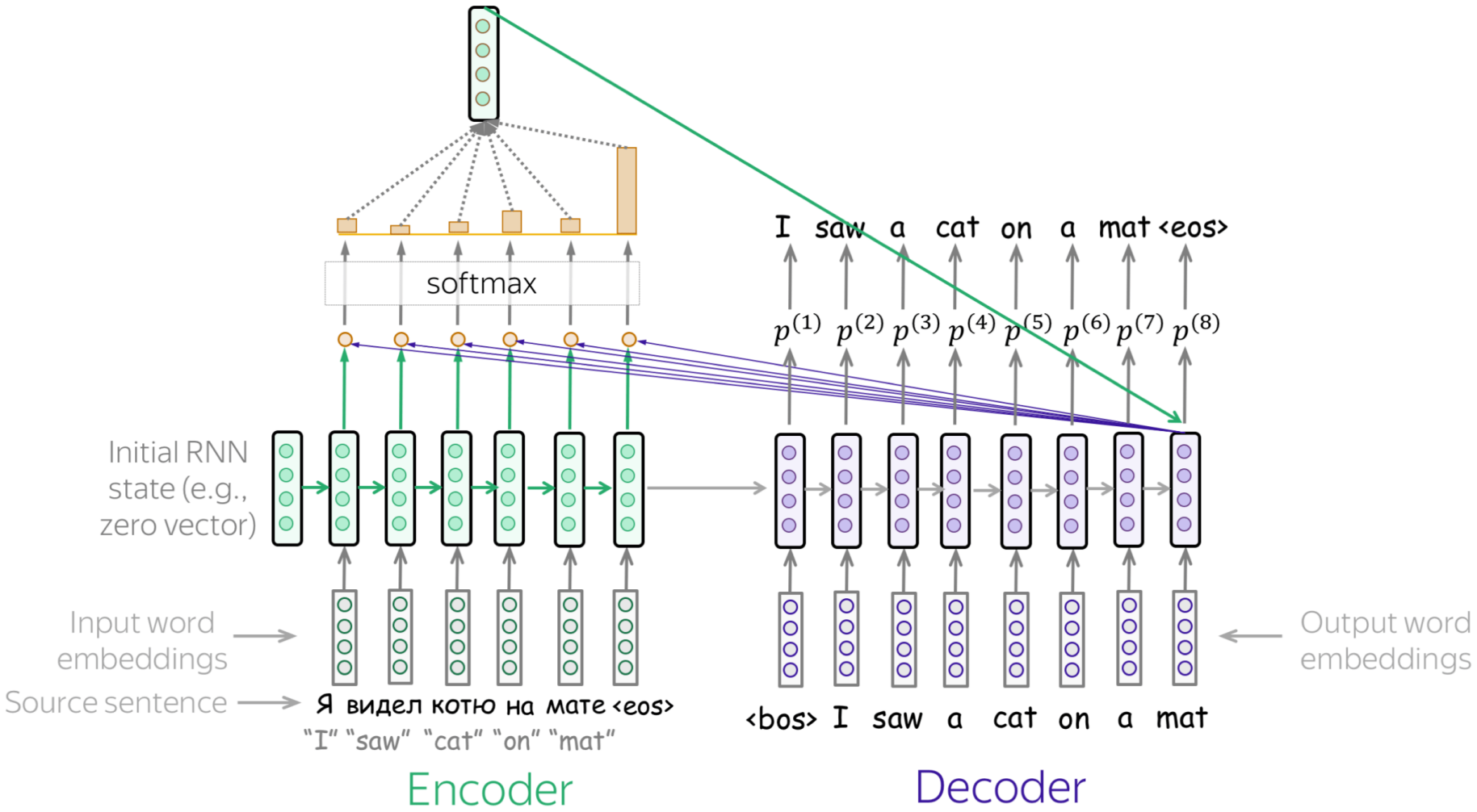
UNC CHARLOTTE



UNC CHARLOTTE



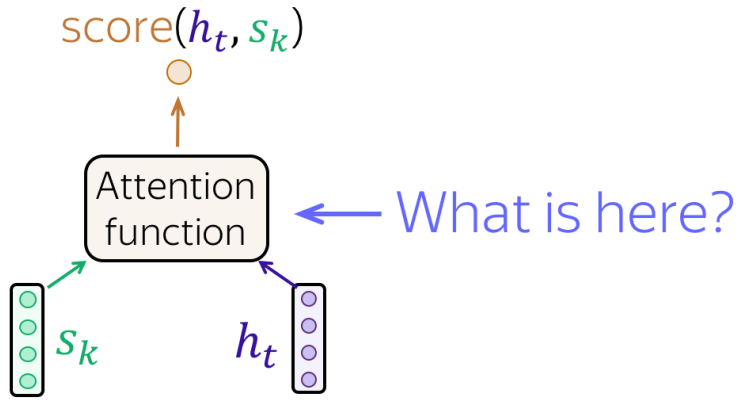
UNC CHARLOTTE



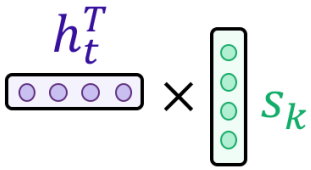
UNC CHARLOTTE

How to Compute Attention Score?

- In the general pipeline above, we haven't specified how exactly we compute attention scores.
- You can apply any function you want - even a very complicated one.
- There are several popular and simple variants which work quite well

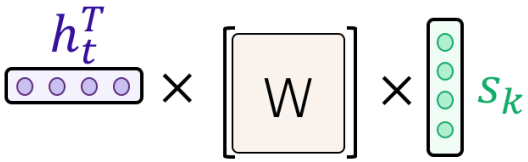


Dot-product



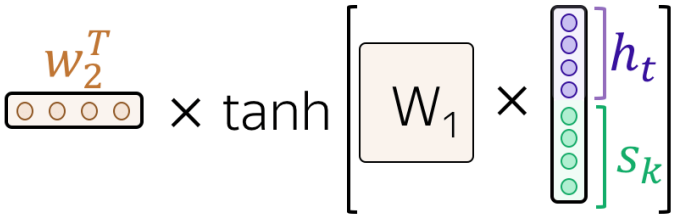
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W s_k$$

Multi-Layer Perceptron



$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

UNC CHARLOTTE

How to Compute Attention Score?

- The most popular ways to compute attention scores are:

1. dot-product - the simplest method;

2. bilinear function (aka "Luong attention") - used in the paper Effective Approaches to Attention-based Neural Machine Translation;

3. multi-layer perceptron (aka "Bahdanau attention") - the method proposed in the original paper.

Model Variants: Bahdanau and Luong

- When talking about the early attention models, you are most likely to hear these variants:

1. Bahdanau attention - from the paper Neural Machine Translation by Jointly Learning to Align and Translate by Dzmitry Bahdanau, KyungHyun Cho and **Yoshua Bengio** (this is the paper that introduced the attention mechanism for the first time);

2. Luong attention - from the paper Effective Approaches to Attention-based Neural Machine Translation by Minh-Thang Luong, Hieu Pham, **Christopher D. Manning**.

- These may refer to either score functions of the whole models used in these papers. In this part, we will look more closely at these two model variants.

Bahdanau attention

- The encoder: bidirectional
  - To better encode each source word, the encoder has two RNNs, forward and backward, which read input in the opposite directions.
  - For each token, states of the two RNNs are concatenated.
- The attention score: multi-layer perceptron
  - To get an attention score, apply a multi-layer perceptron (MLP) to an encoder state and a decoder state.
- The attention computation is applied: between decoder steps

Attention is used between decoder steps: state h_{t-1} is used to compute attention and its output $c^{(t)}$, and both h_{t-1} and $c^{(t)}$ are passed to the decoder at step t.

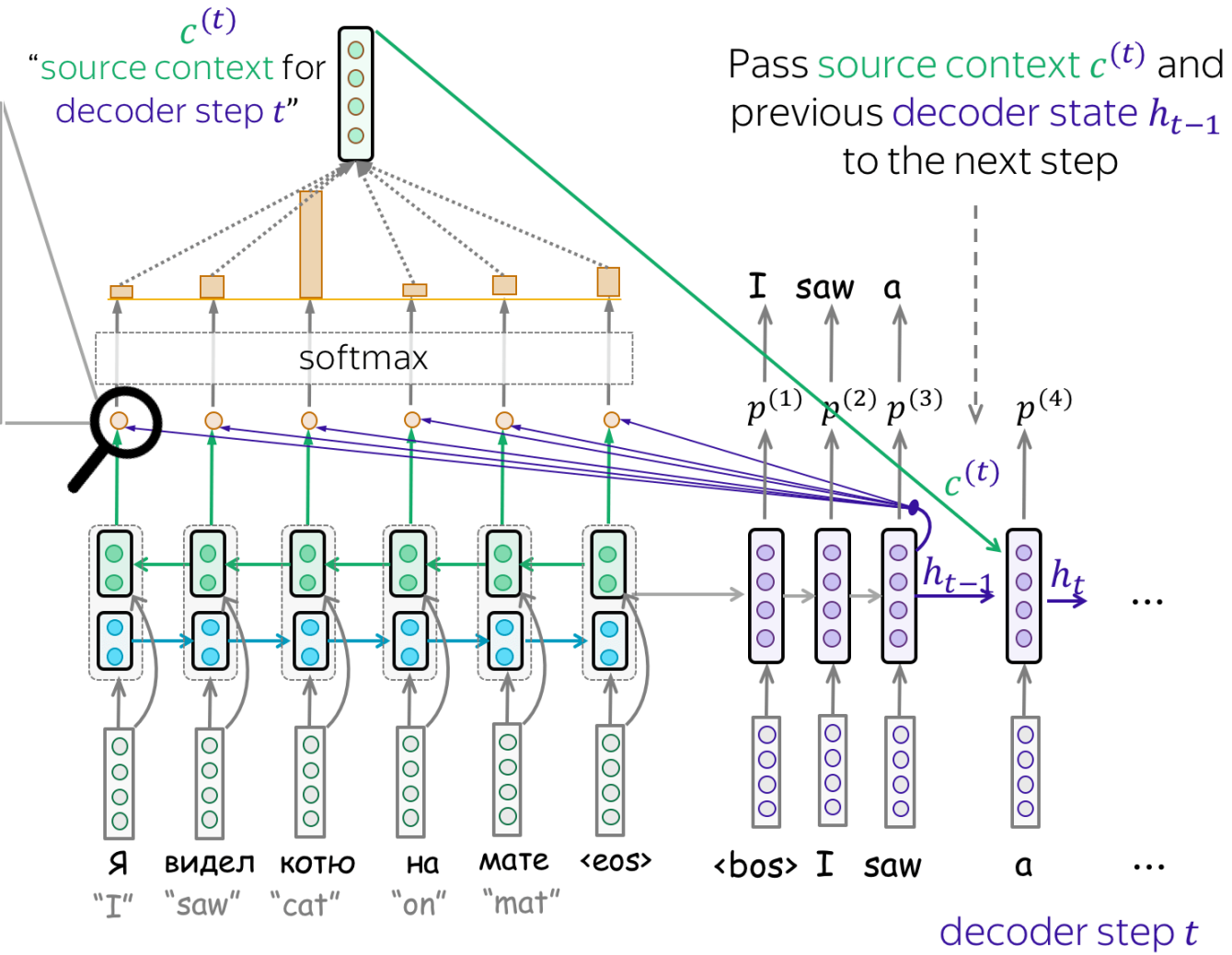
Bahdanau attention

Multi-Layer Perceptron

$$w_2^T \times \tanh\left[W_1 \times \begin{bmatrix} h \\ s_k \end{bmatrix} \right]$$

$$\text{score}(h, s_k) = w_2^T \cdot \tanh(W_1[h, s_k])$$

Bidirectional encoder
Concatenate states from forward and backward RNNs



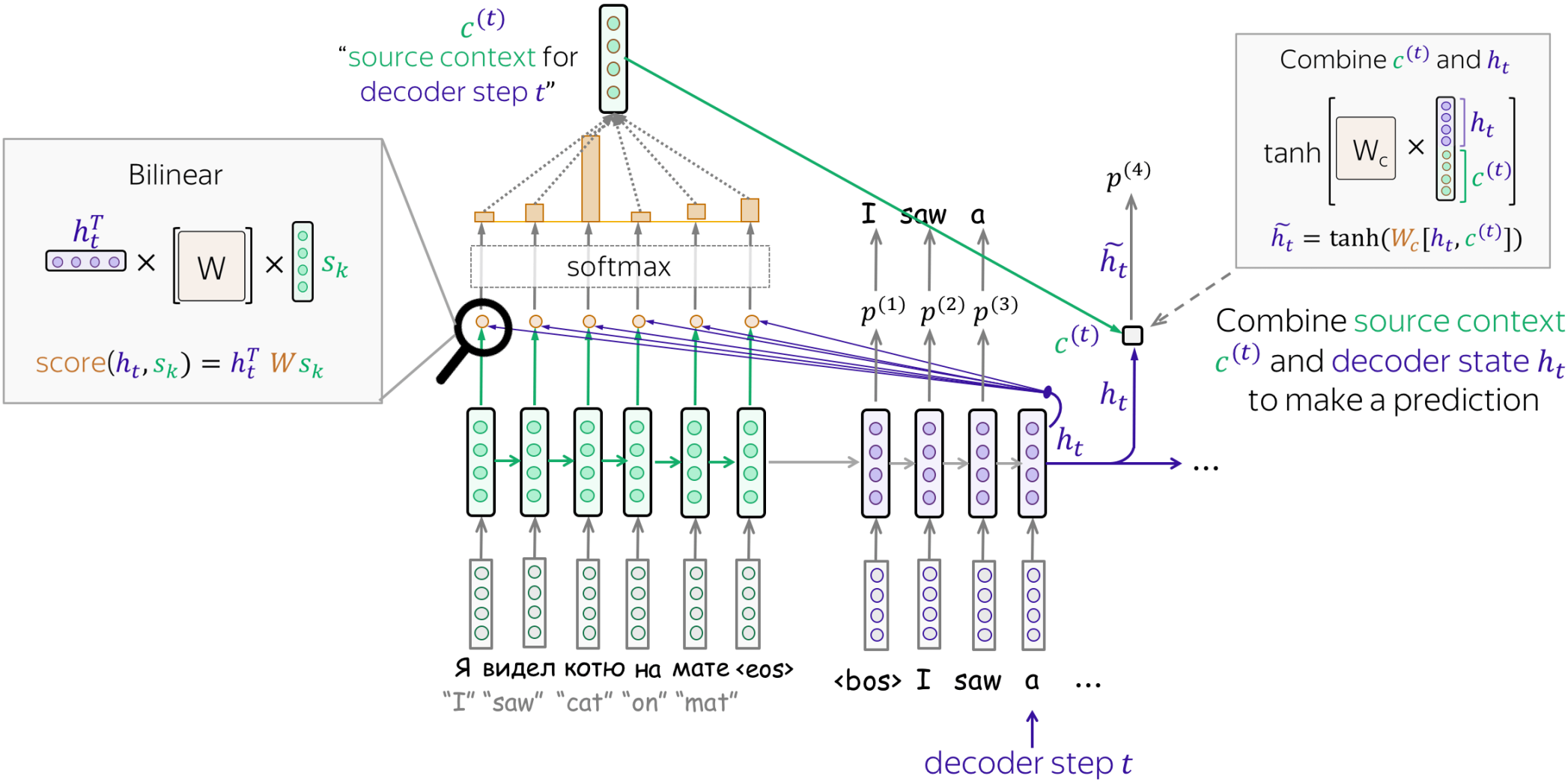
UNC CHARLOTTE

Luong Model

- While the paper considers several model variants, the one which is usually called "Luong attention" is the following:
  - encoder: unidirectional (simple)
  - attention score: bilinear function
  - attention applied: between decoder RNN state and prediction for this step

Attention is used after RNN decoder step t before making a prediction. State h_t used to compute attention and its output $c^{(t)}$. Then h_t is combined with $c^{(t)}$ to get an updated representation $\tilde{h}_t$, which is used to get a prediction.

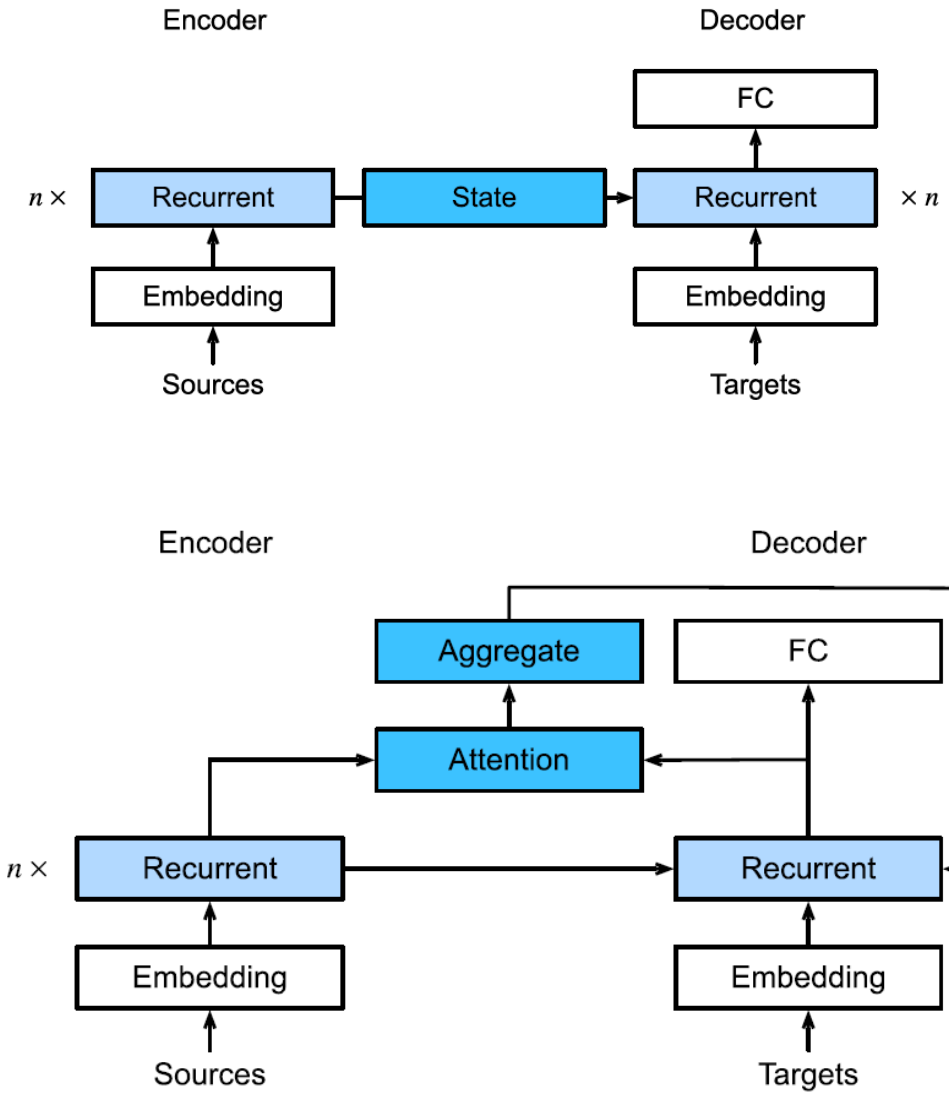
Luong Model



UNC CHARLOTTE

The Bahdanau Attention Mechanism

- Inspired by the idea of learning to align, Bahdanau *et al.* (2014) proposed a differentiable attention model *without* the unidirectional alignment limitation.
- When predicting a token, if not all the input tokens are relevant, the model aligns (or attends) only to parts of the input sequence that are deemed relevant to the current prediction.
- This is then used to update the current state before generating the next token.
- *Bahdanau attention mechanism* has arguably turned into one of the most influential ideas of the past decade in deep learning, giving rise to Transformers (Vaswani *et al.*, 2017)



UNC CHARLOTTE

The Bahdanau Attention Mechanism

- The key idea is that instead of keeping the state, i.e., the context variable **c** summarizing the source sentence as fixed, we dynamically update it, as a function of both the original text (encoder hidden states H) and the text that was already generated (decoder hidden states S).
- This yields **C**, which is updated after any decoding time step *t*.
- Suppose that the input sequence is of length *T*. In this case the context variable is the output of attention pooling:

$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t.$$

Defining the Decoder with Attention

- To implement the RNN encoder-decoder with attention, we only need to redefine the decoder (omitting the generated symbols from the attention function simplifies the design).
- Let's begin with the base interface for decoders with attention by defining the quite unsurprisingly named AttentionDecoder class.

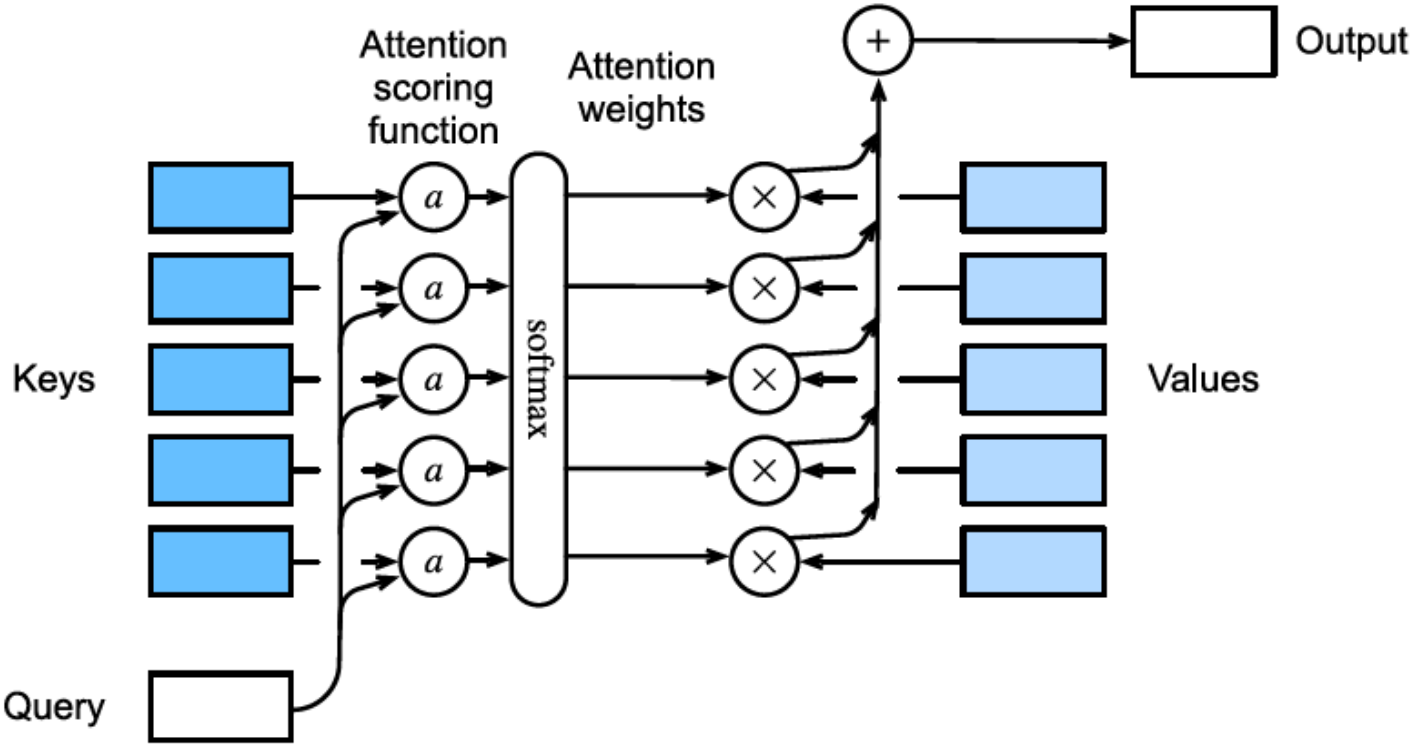
Defining the Decoder with Attention

- We need to implement the RNN decoder in the Seq2SeqAttentionDecoder class.

- The state of the decoder is initialized with:
  (i) the hidden states of the last layer of the encoder at all time steps, used as keys and values for attention;
  (ii) the hidden state of the encoder at all layers at the final time step. This serves to initialize the hidden state of the decoder
  (iii) the valid length of the encoder, to exclude the padding tokens in attention pooling.

- At each decoding time step, the hidden state of the last layer of the decoder, obtained at the previous time step, is used as the query of the attention mechanism.
- Remember similar to before, at the decoder stage, both the output of the attention mechanism and the input embedding are concatenated to serve as the input of the RNN decoder.

Attention Scoring Functions



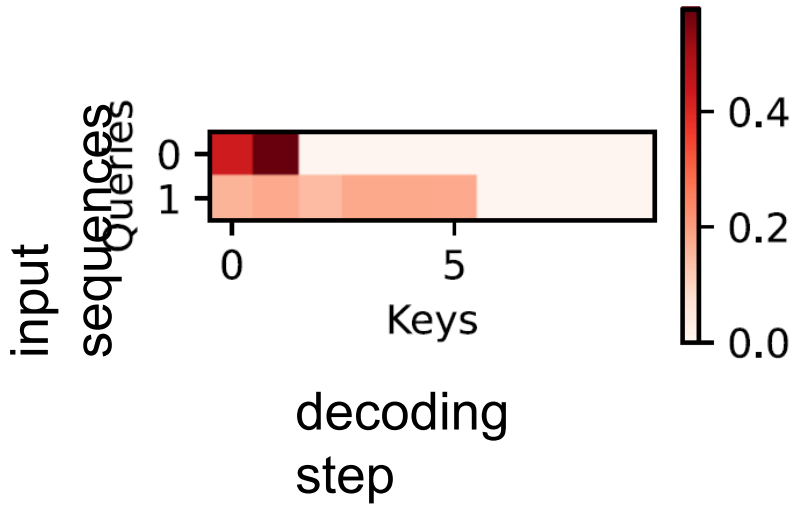
Computing the output of attention pooling as a weighted average of values, where weights are computed with the attention scoring function a and the softmax operation.

Additive Attention

When queries $\mathbf{q}$ and keys $\mathbf{k}$ are vectors of different dimensionalities, we can either use a matrix to address the mismatch via $\mathbf{q}^\top \mathbf{M}\mathbf{k}$, or we can use additive attention as the scoring function. Another benefit is that, as its name indicates, the attention is additive. This can lead to some minor computational savings. Given a query $\mathbf{q} \in \mathbb{R}^q$ and a key $\mathbf{k} \in \mathbb{R}^k$, the *additive attention* scoring function (Bahdanau *et al.*, 2014) is given by

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^\top \tanh(\mathbf{W}_q\mathbf{q} + \mathbf{W}_k\mathbf{k}) \in \mathbb{R},$$

where $\mathbf{W}_q \in \mathbb{R}^{h \times q}$, $\mathbf{W}_k \in \mathbb{R}^{h \times k}$, and $\mathbf{w}_v \in \mathbb{R}^h$ are the learnable parameters.



UNC CHARLOTTE