

Sequence to Sequence Models, Attention and the Transformer with applications to Neural Machine Translation

Dr Cass Zhixue Zhao

zhixue.zhao@sheffield.ac.uk

Twitter @casszzx



The
University
Of
Sheffield.

Previously Covered

- **Task:** Neural Language Modelling
 - Estimate the probability of a text (for example, a sentence)
- **Model:** Recurrent Neural Networks (RNNs)
 - Given a sequence of tokens (for example, words in a sentence), RNNs “read” them one by one and process the information
- **Improvement:** Attention
 - At different steps, let a model “focus” on different parts of the input



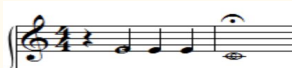
In this Lecture

- **Sequence to Sequence Models**
- **Demonstrating Task:** Neural Machine Translation (NMT)
- **Model:** Encoder-Decoder with RNNs
- **Improvement:** Attention and The Transformer
- **Evaluation:** Automatic Metrics and Manual Judgements

Sequence to Sequence Models

- Sequence to Sequence models have revolutionised NLP applications and are now commonly used in end user applications.
- What are Sequence to Sequence models?
 - Input is a Sequence (any sequence)
 - Output is a Sequence (again, any sequence, but for our purpose: text)

Sequence to Sequence Models: Examples

Machine Translation	A pesquisa em PNL é muito excitante.	NLP research is very exciting.
Speech to Text		Hello there!
Image Captioning		A woman playing tennis
Named entity recognition	It is our choices, Harry, that show what we truly are, far more than our abilities.	Harry
Neural Music Generation	No input (or some note)	

Sequence to Sequence Models: More Examples

Who is wearing glasses?



→ The Man



→ The Woman

Sequence to Sequence Models: More Examples*

Logical Inference

Input: Shelley is from Virginia, but is visiting that city with that famous market where they throw the fish. Going home next Tuesday!

Question: Is it likely that Shelley will be near the Pacific Ocean this weekend?

Model Output: The city with the famous market where they throw the fish is Seattle, Washington. Seattle is on the Pacific Ocean. Shelley is visiting Seattle, so she will be near the Pacific Ocean this weekend. The answer is "yes", it is likely that Shelley will be near the Pacific Ocean this weekend.

*PaLM: MoE
Decoder-only



Demonstrating Task

Neural Machine Translation

What is Machine Translation?

Machine Translation (MT) consists of translating a sentence \mathbf{x} from one language (the *source*) to a sentence \mathbf{y} in another language (the *target*)



What is Neural Machine Translation Model?

- **Neural Machine Translation** (NMT) is Machine Translation using neural networks (as opposed to alignment and phrase based translation).
- Typically we use **sequence-to-sequence models** (seq2seq).
- These models are **end-to-end differentiable**

Task Formulation

- We want to translate from English to Spanish
- Formally, what this means is: we want to find best Spanish sentence **y**, given English sentence **x**

Hello, how are you → **Hola como estas**

→ **Hola como estamos**

→ **Dónde estás**

Task Formulation

- We want to translate from English to Spanish
- Formally, what this means is: we want to find best / the most probable Spanish sentence y , given English sentence x
- Problems:
 - How do we come up with candidate sentences?
 - How do we search through this large list of possible sentences?

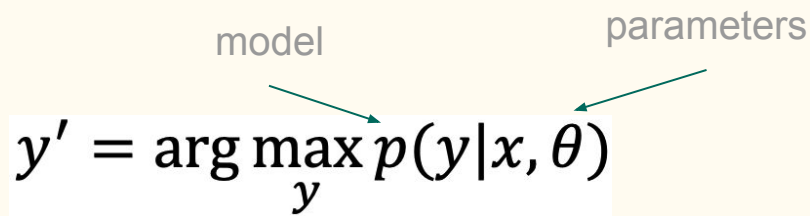
Task Formulation

- Problems:
 - How do we come up with candidate sentences?
 - Generate sentences using a neural network (seq2seq)
 - We are going to teach the neural network to generate candidate sentences
 - How do we search through this large list of possible sentences?
 - Beam Search

Task Formulation

More Formally:

- We want to find best Spanish sentence \mathbf{y} , given English sentence \mathbf{x}


$$y' = \arg \max_y p(y|x, \theta)$$

modelling

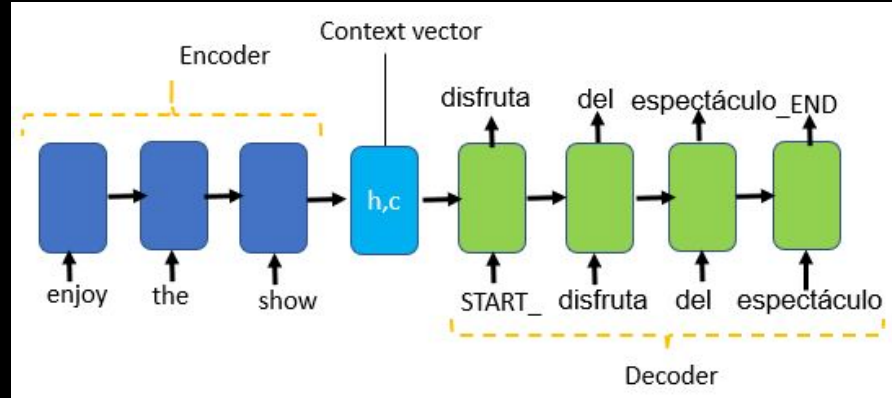
What is the model
 $p(y|x, \theta)$?

learning

How do we find to
find θ

search

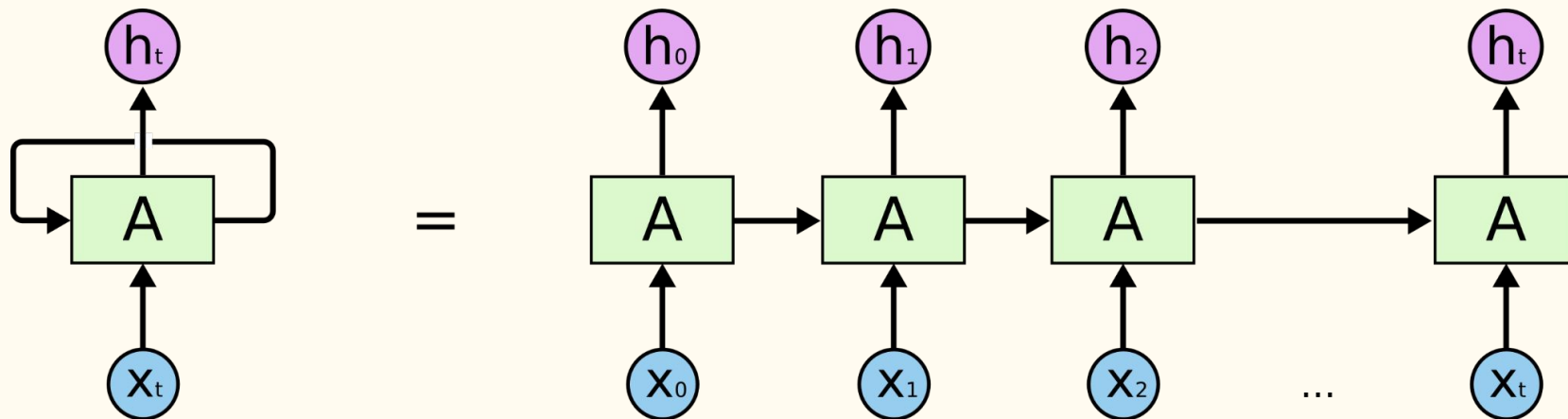
How do we find the
best?



The Model

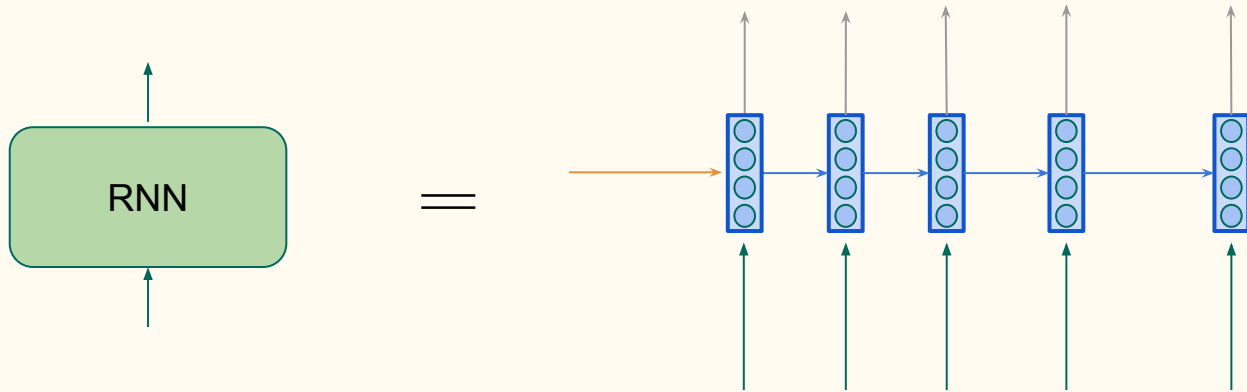
Encoder-Decoder Architecture (with RNNs)

Recap: RNN representations

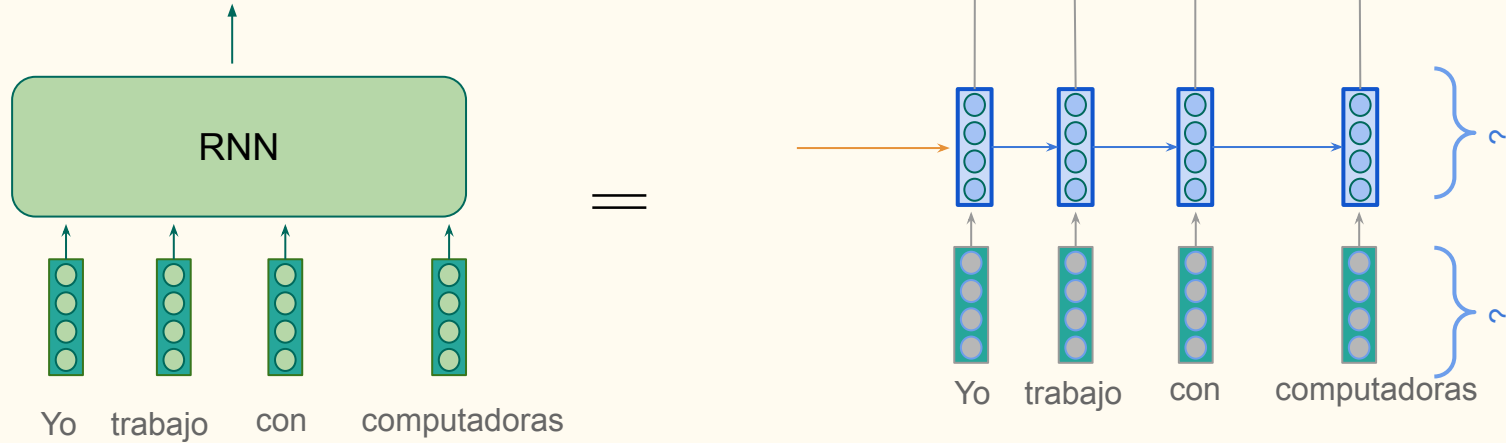


$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})$$

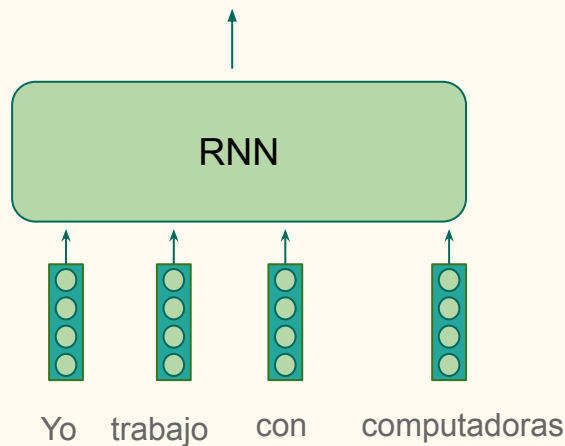
Quick Note on RNN representations



Quick Note on RNN representations

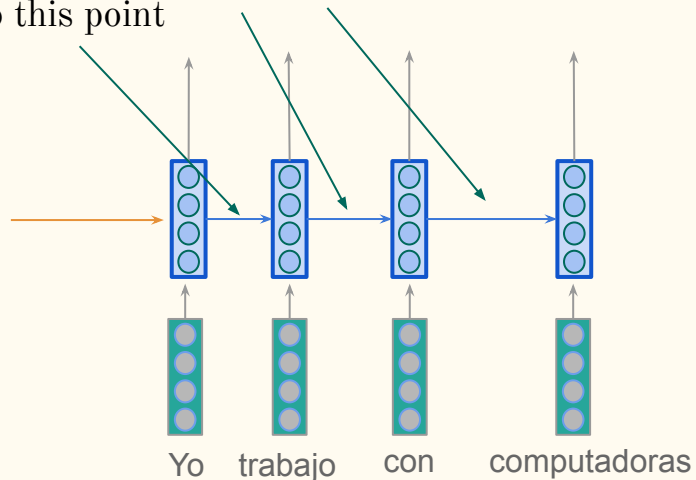


Quick Note on RNN representations



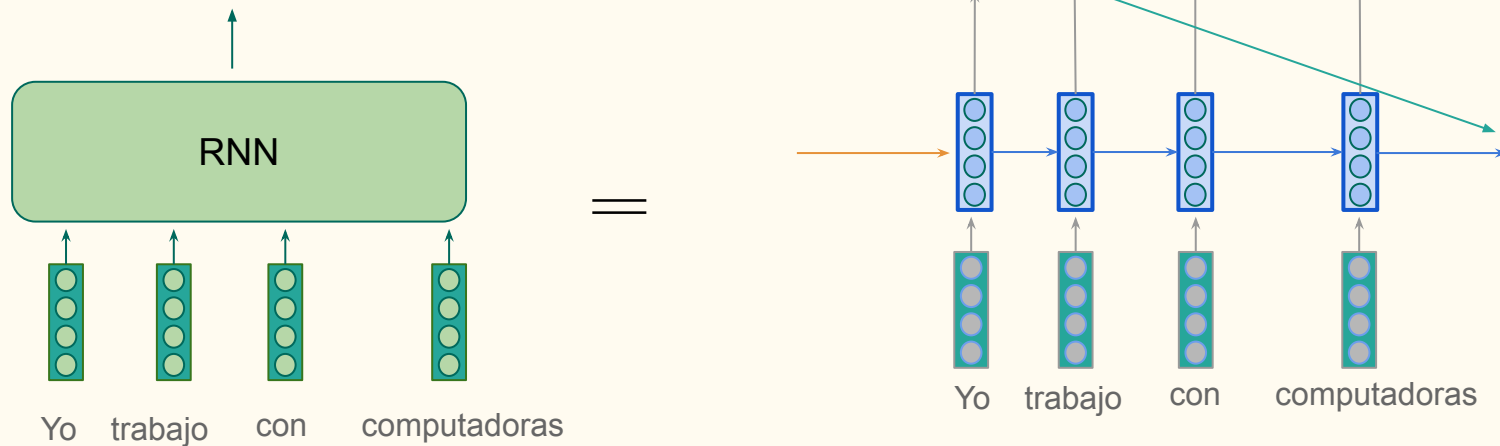
=

“Representation” of sentence
up to this point

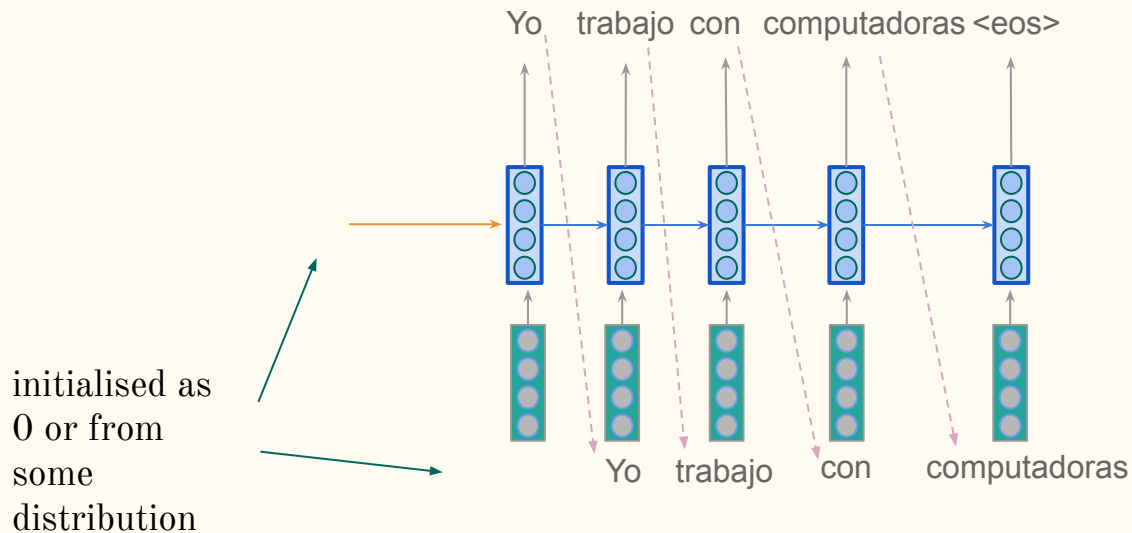


Quick Note on RNN representations

“Representation” of
entire sentence.



Recall: Language Modelling with RNNs



Notice that RNN based Language Models provide contextual representations

Peters et al 2017

Peters et al 2018 (ELMo)

* ELMo includes CNNs

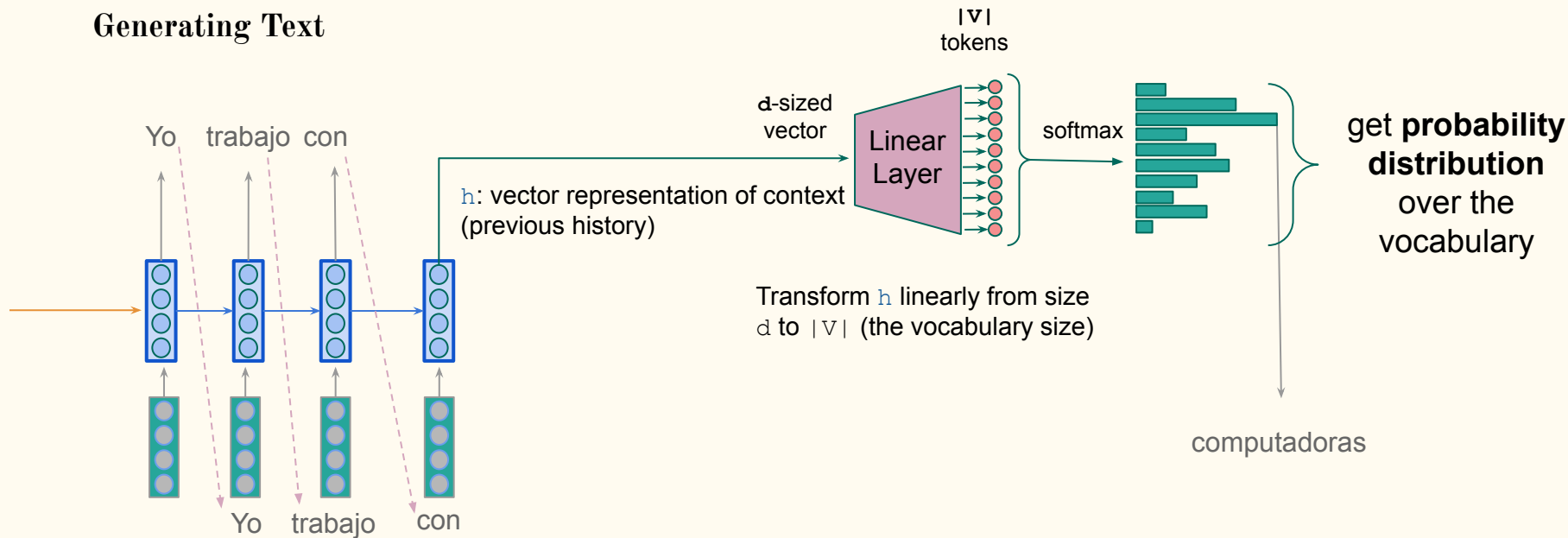
They also capture grammatical information

Colorless green recurrent networks dream hierarchically

“the girl[you met yesterday through her friends] thinks.
..”.

Recall: Language Modelling with RNNs

Generating Text

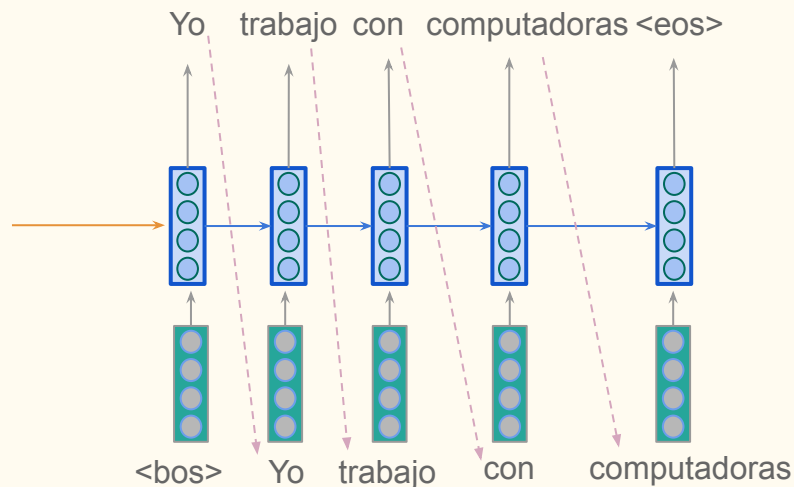


Recall: Language Models

Language Models:

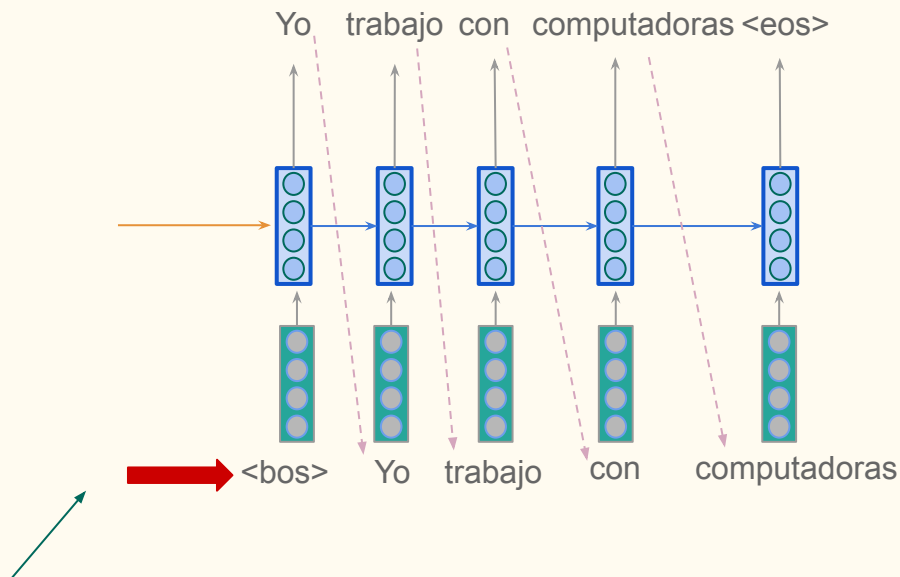
$$P(y_1, y_2, \dots, y_n) = \prod_{t=1}^n p(y_t | y_{<t})$$

Language Modelling to Sequence Generation



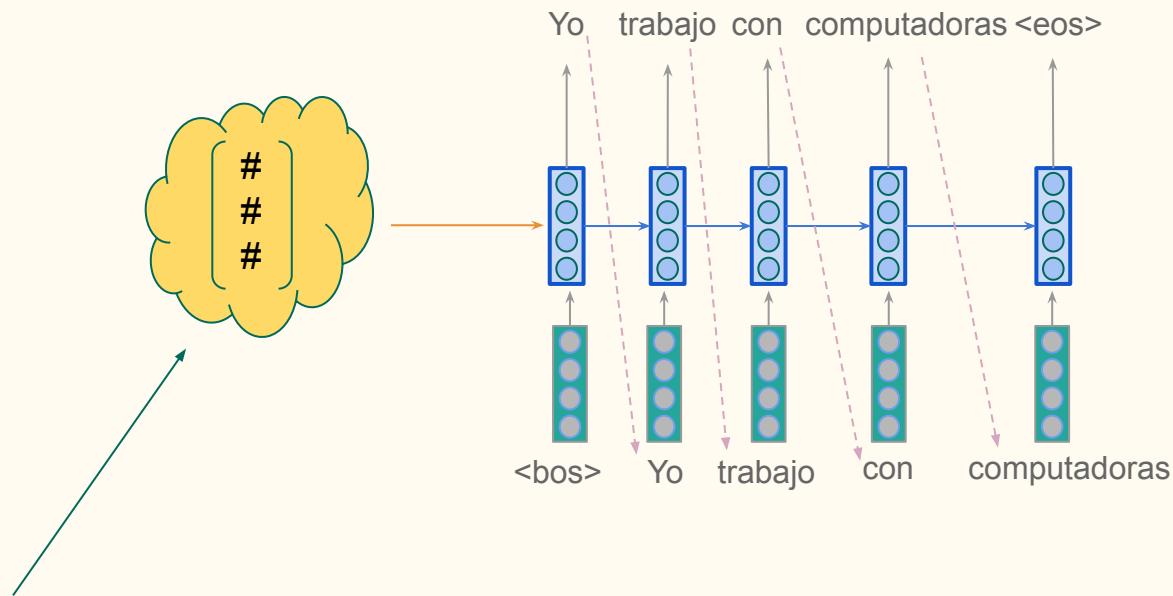
We can initialise “this” with a “prompt” that acts as the start of an idea or a thought that will now be fledged out by the generated text

Language Modelling to Sequence Generation



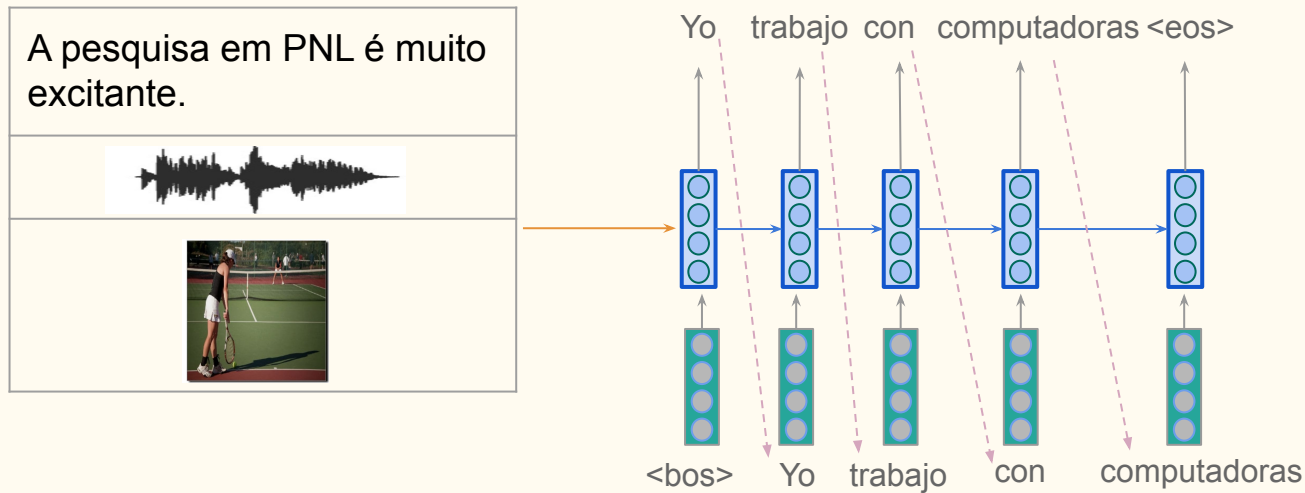
We can initialise “this” with a “prompt” that acts as the start of an idea or a thought that will now be fledged out by the generated text

Language Modelling to Sequence Generation



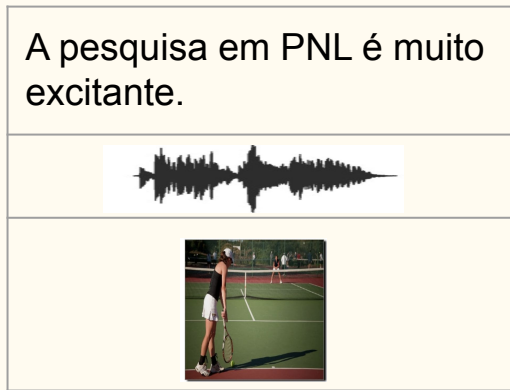
We can initialise this ***with a representation of an idea or a thought*** that will now be fledged out by the generated text

Language Modelling to Sequence Generation



↑
We can initialise this ***with a representation of an idea or a thought*** that will now be fledged out by the generated text

Encoding Information

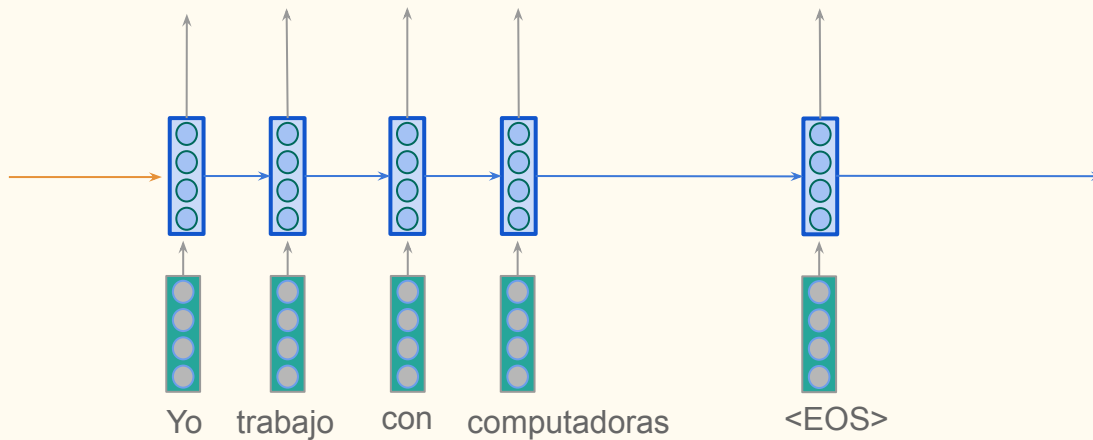


*How can we convert this into
an “encoding” or a vector?*

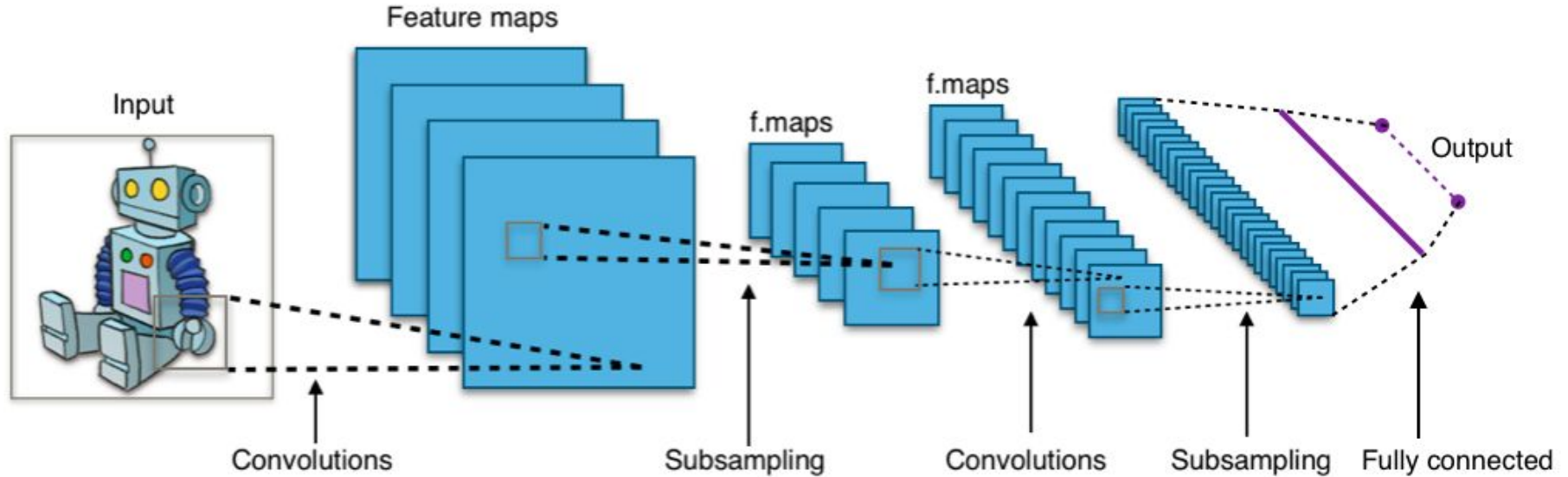
Let's start with the text?

Encoding Information (text)

We already
know how to
do this!

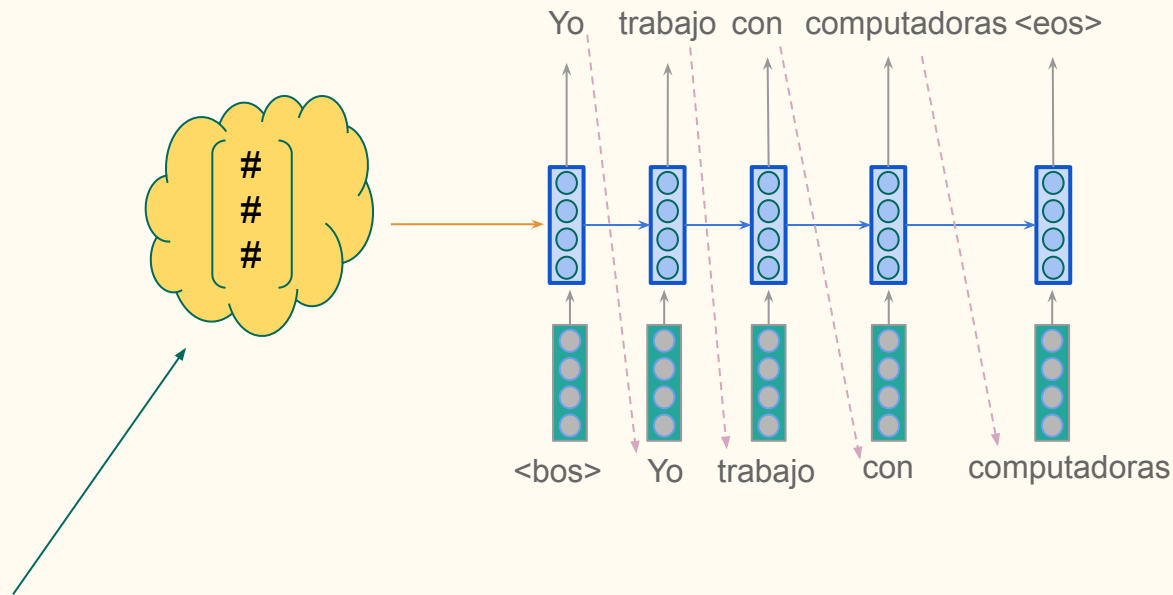


Encoding Information (images)



“Thought” vector to Encoder-Decoder

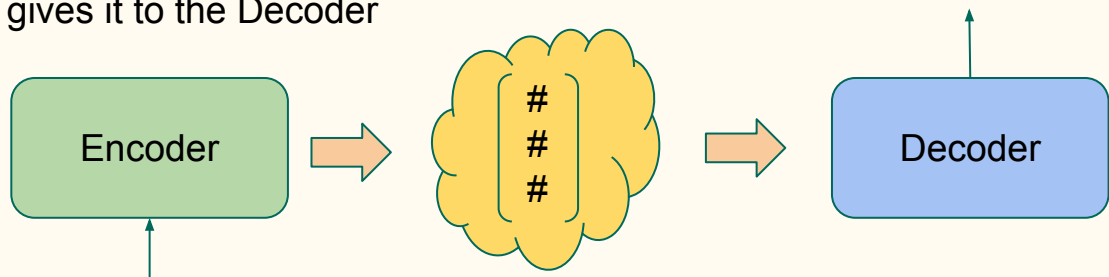
Use an
Encoder!



We can initialise this ***with a representation of an idea or a thought*** that will now be fledged out by the generated text

Encoder-Decoder

Encoder reads the *source*,
produces its representation,
and gives it to the Decoder



Decoder uses the *source*
representation to generate
the *output*

Conditional Language Models

Language Models:

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

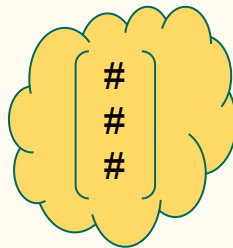
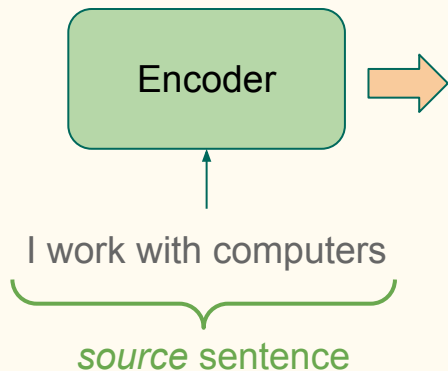
Conditional
Language Models:

$$P(y_1, y_2, \dots, y_n, | \underset{\substack{\uparrow \\ \text{condition on source } x}}{x}) = \prod_{t=1}^n p(y_t | y_{<t}, x)$$

\mathcal{X} does not always need to be a sequence of tokens.
(e.g. image captioning)

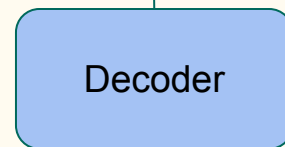
Encoder-Decoder for Translation

Encoder reads the *source*, produces its representation, and gives it to the Decoder



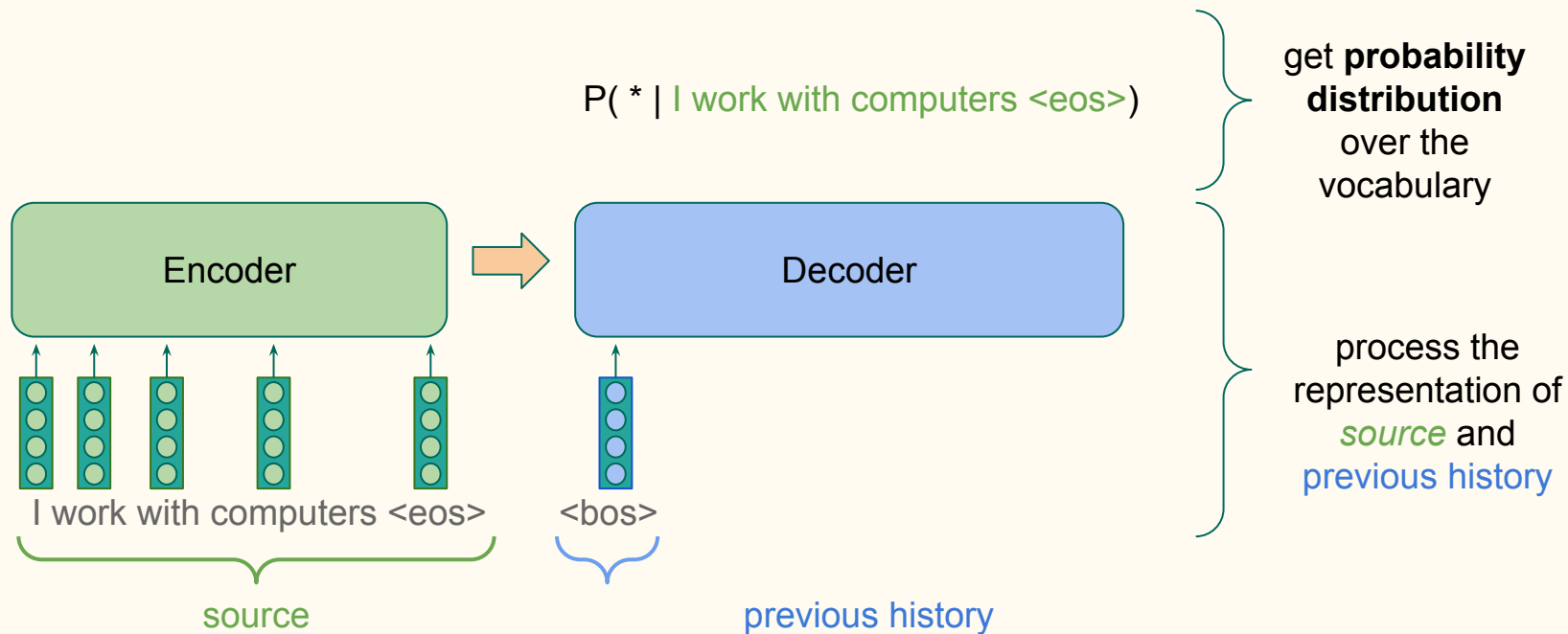
target sentence

Yo trabajo con computadoras

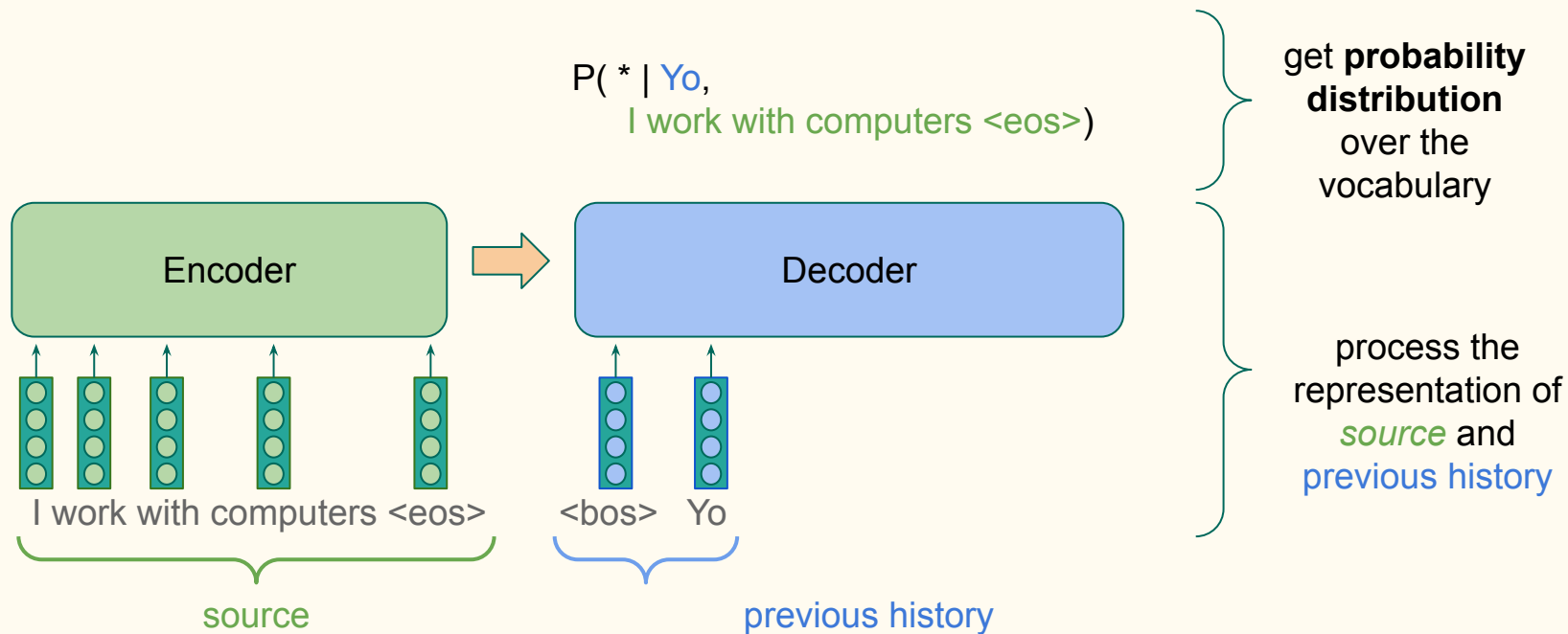


Decoder uses the *source* representation to generate the *target* sentence

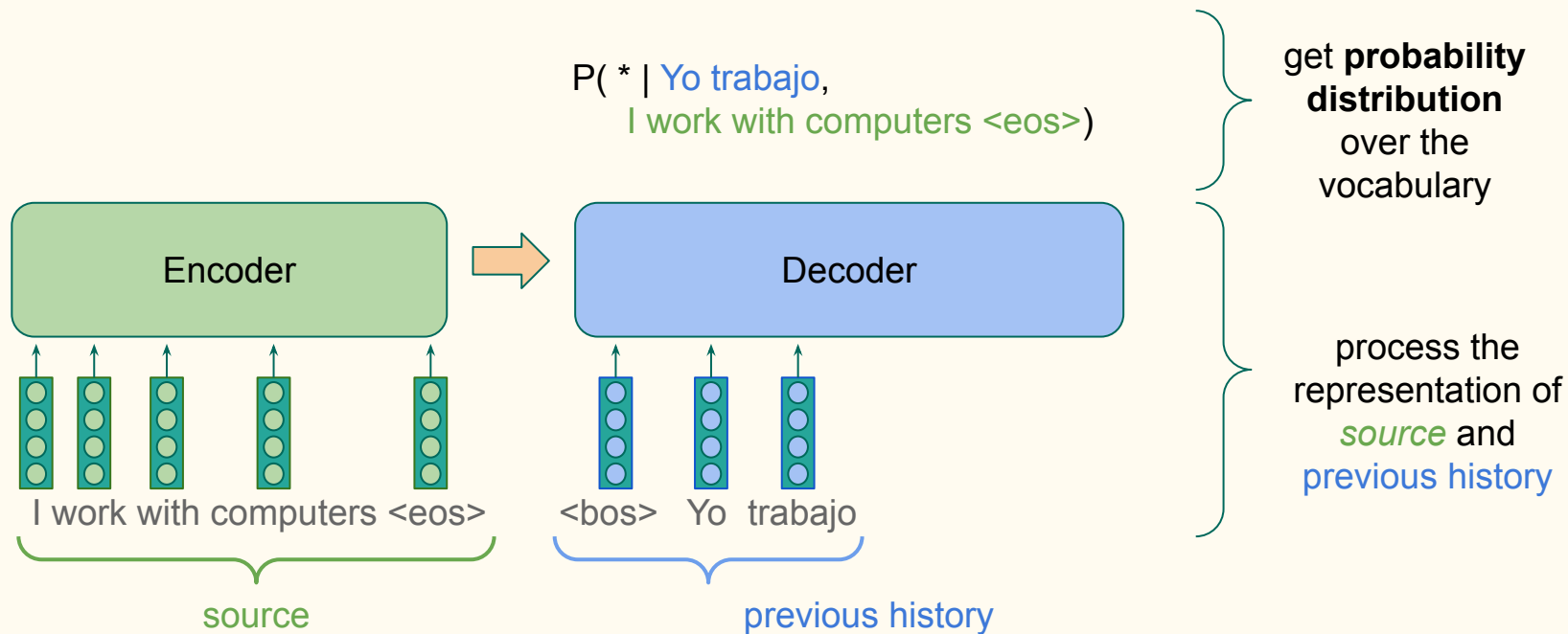
Sequence Generation



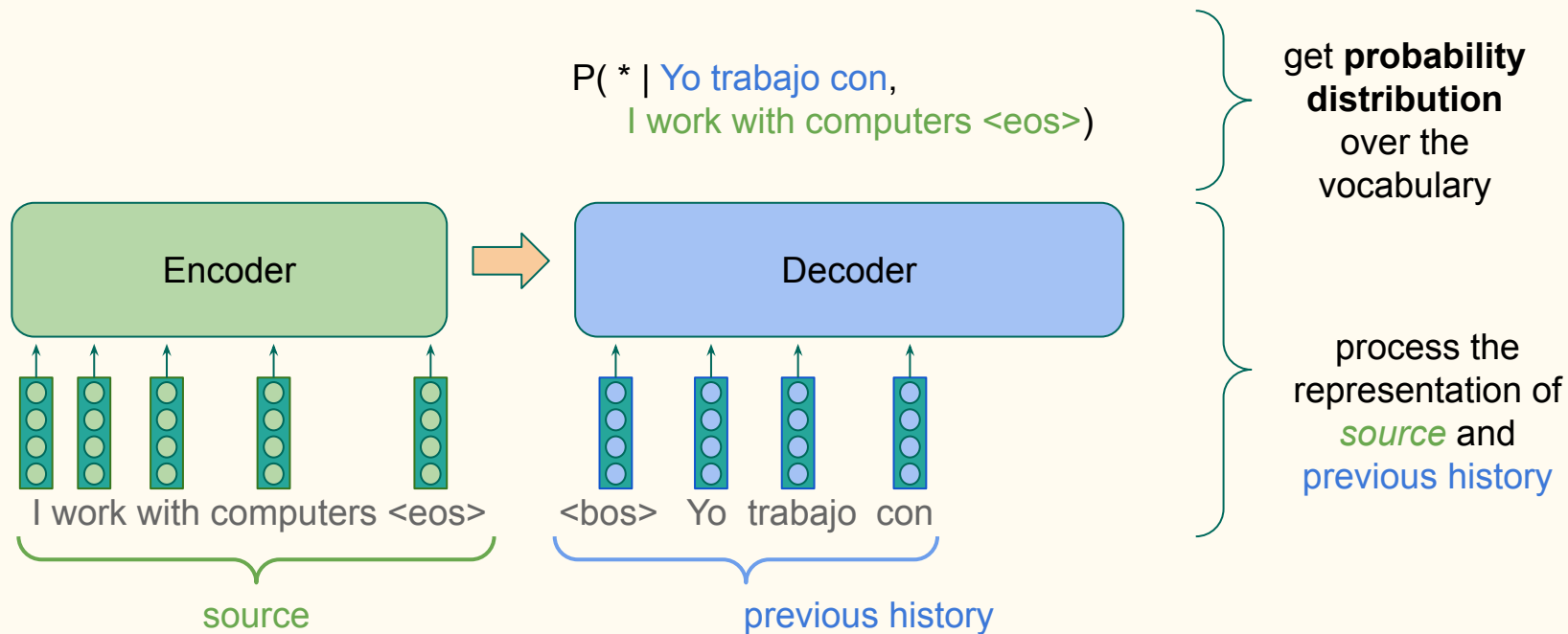
Sequence Generation



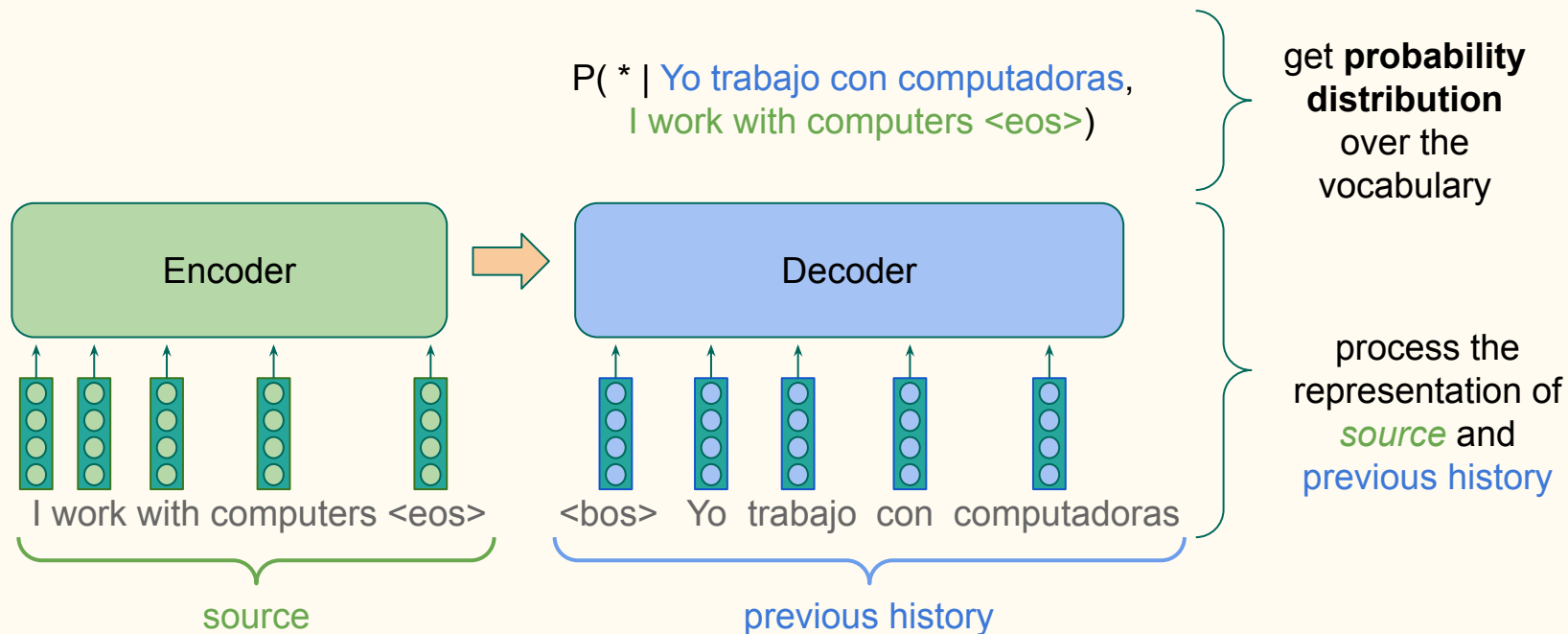
Sequence Generation



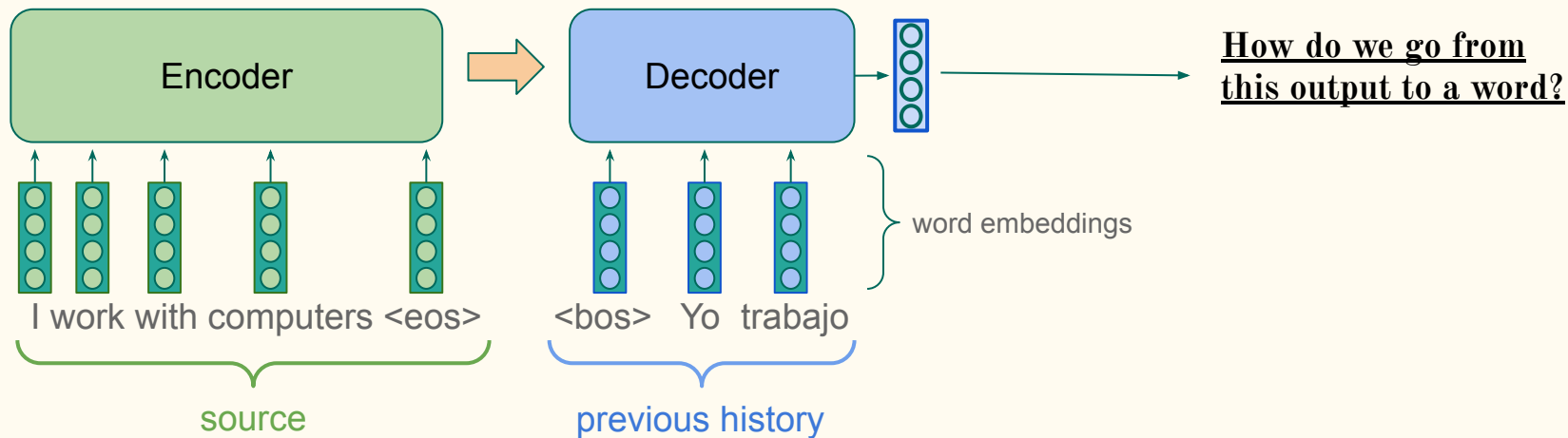
Sequence Generation



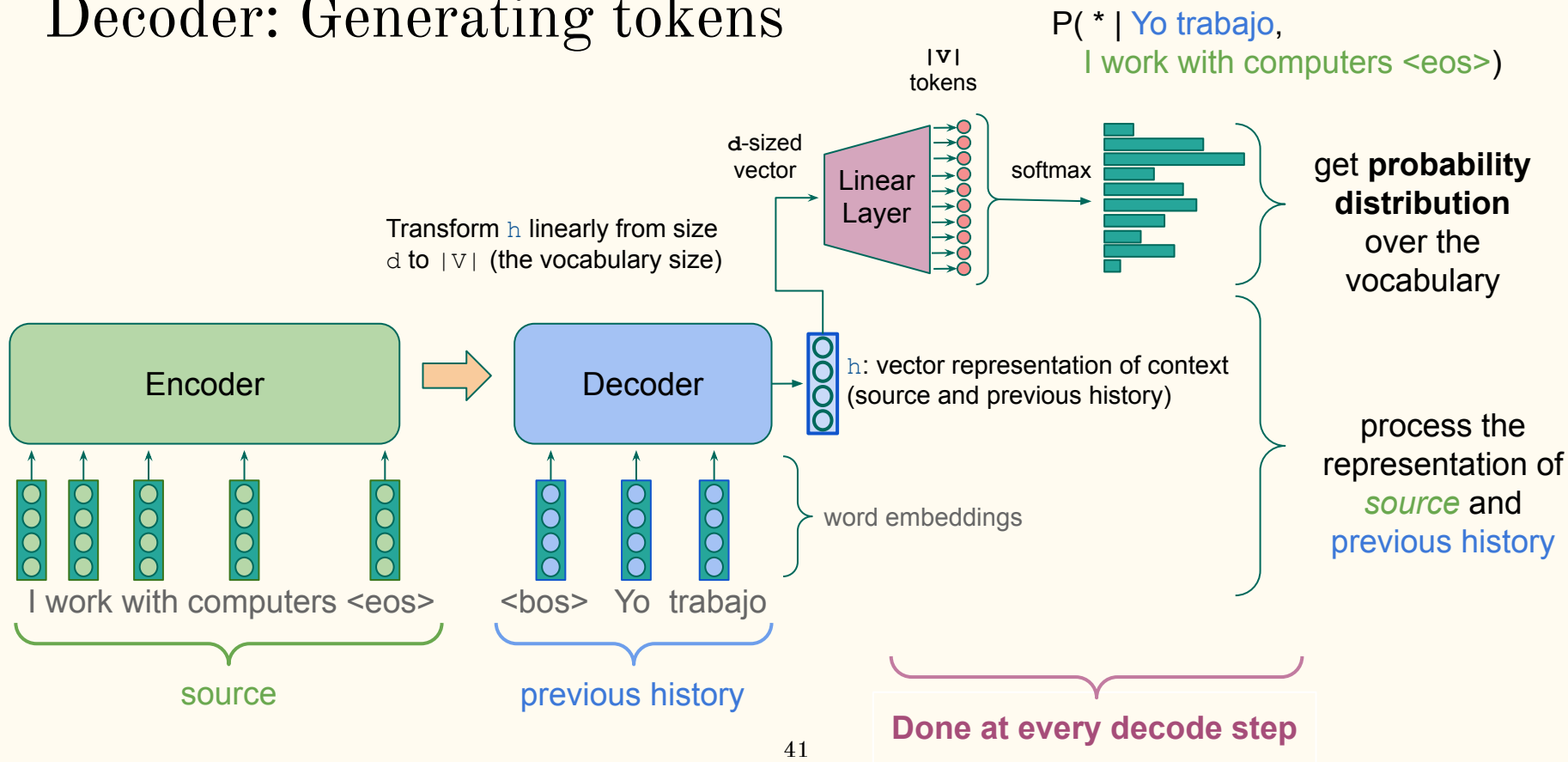
Sequence Generation



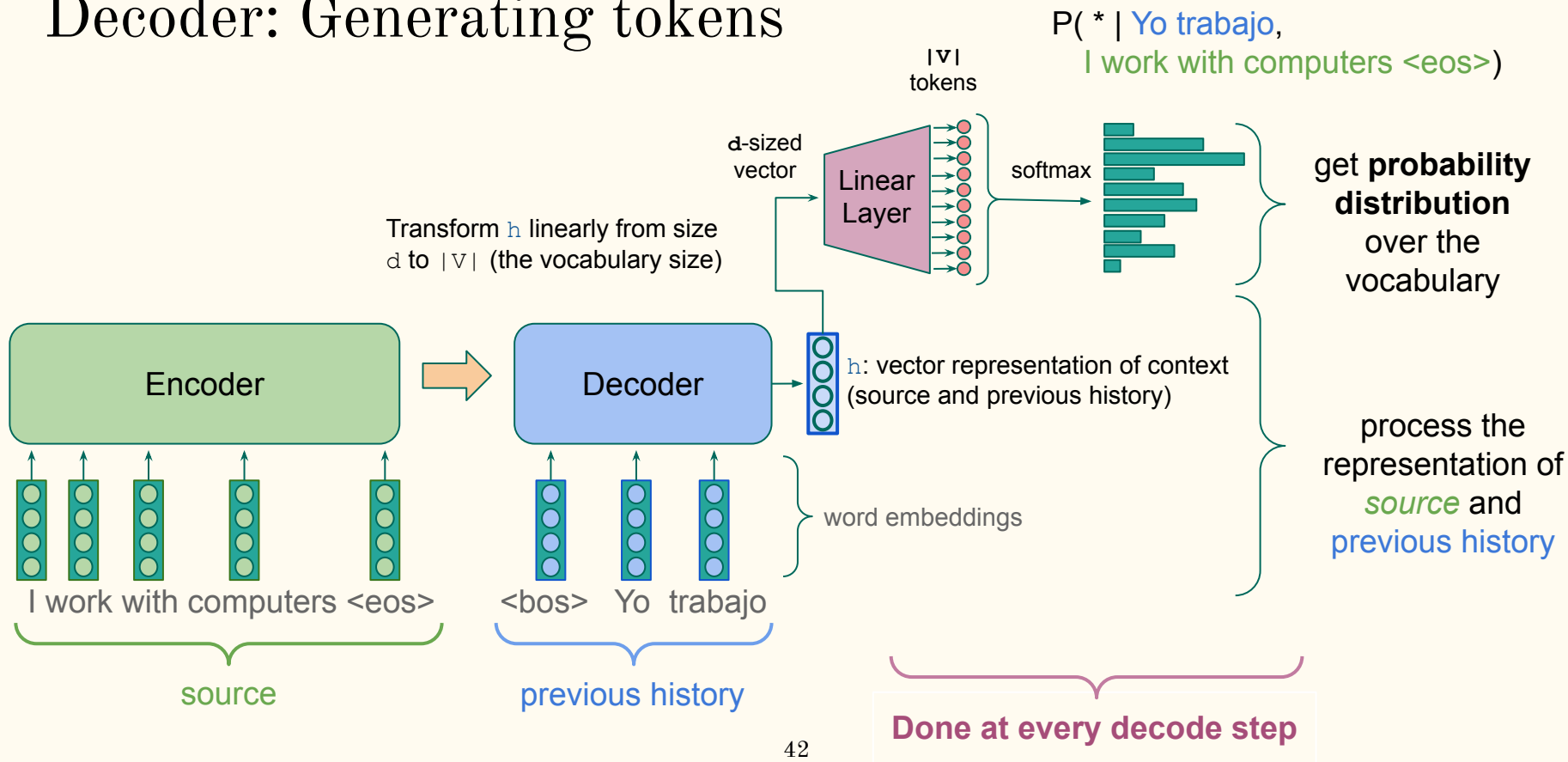
Decoder: Generating tokens



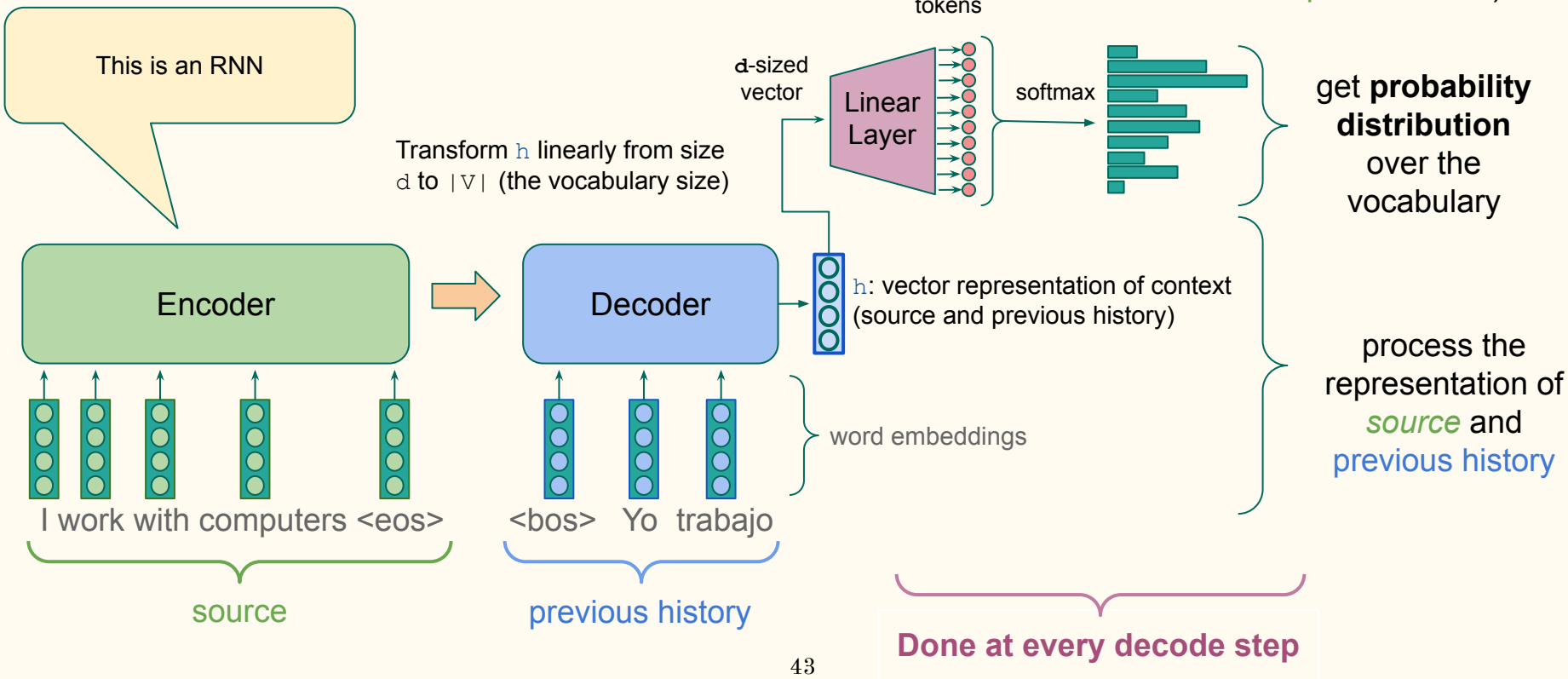
Decoder: Generating tokens



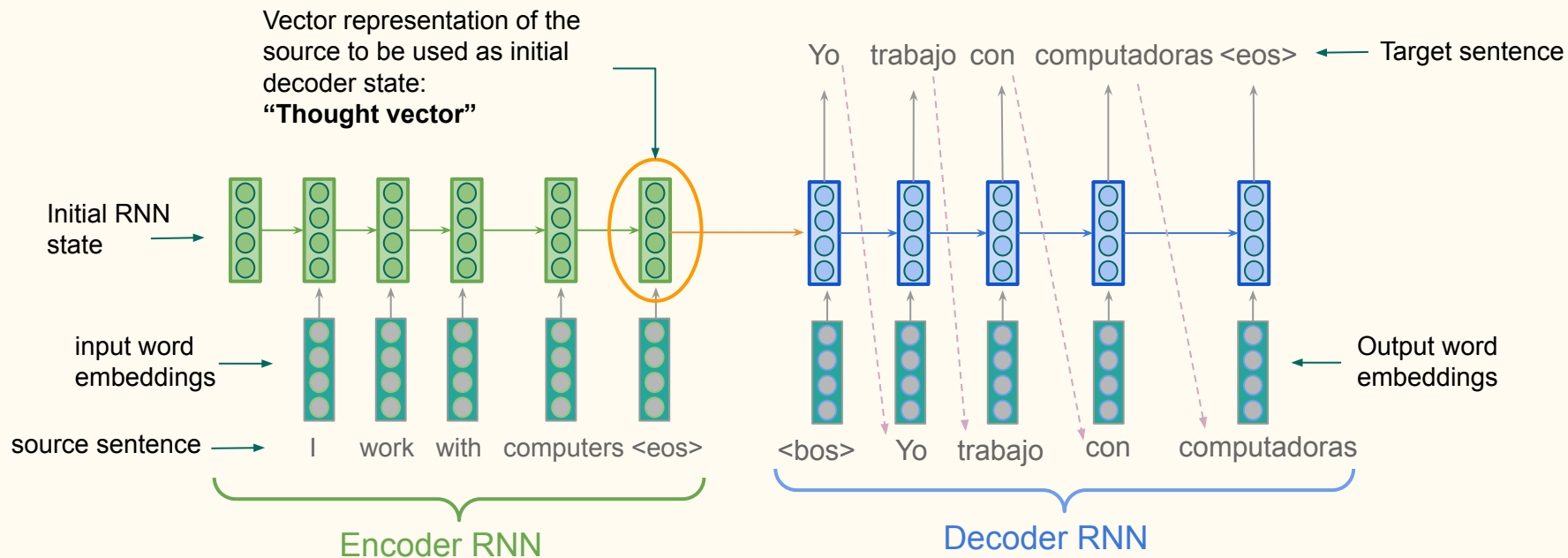
Decoder: Generating tokens



Decoder: Generating tokens



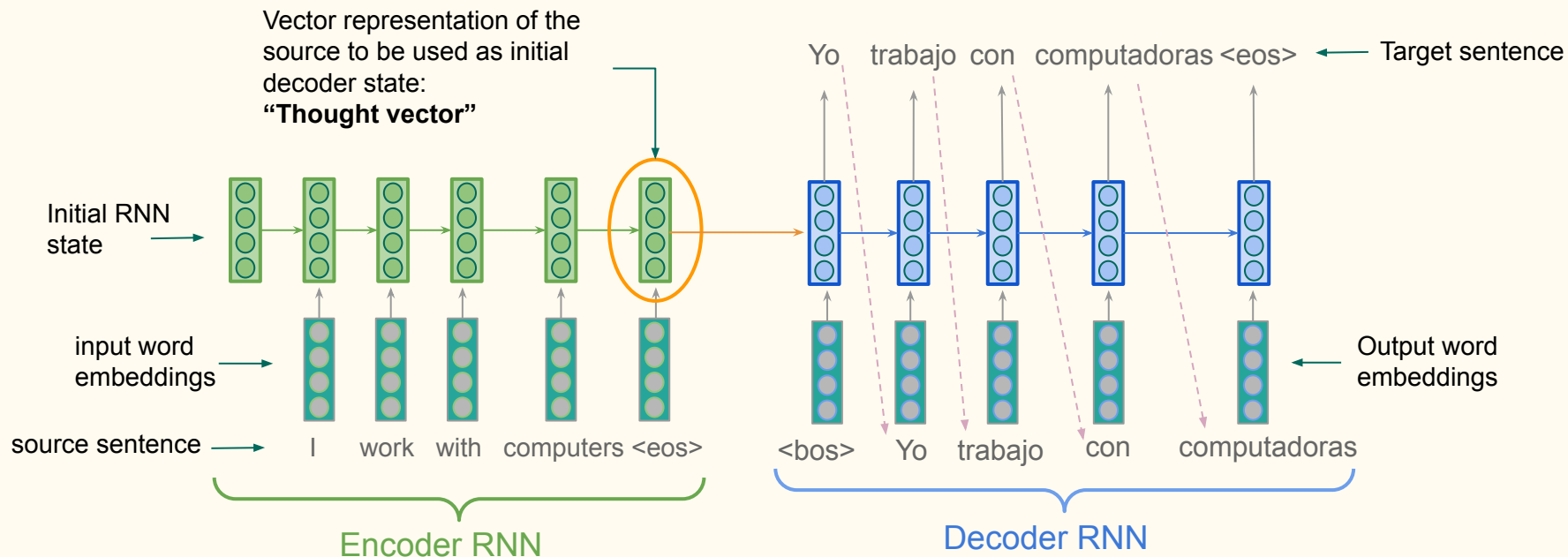
Encoder-Decoder with RNNs



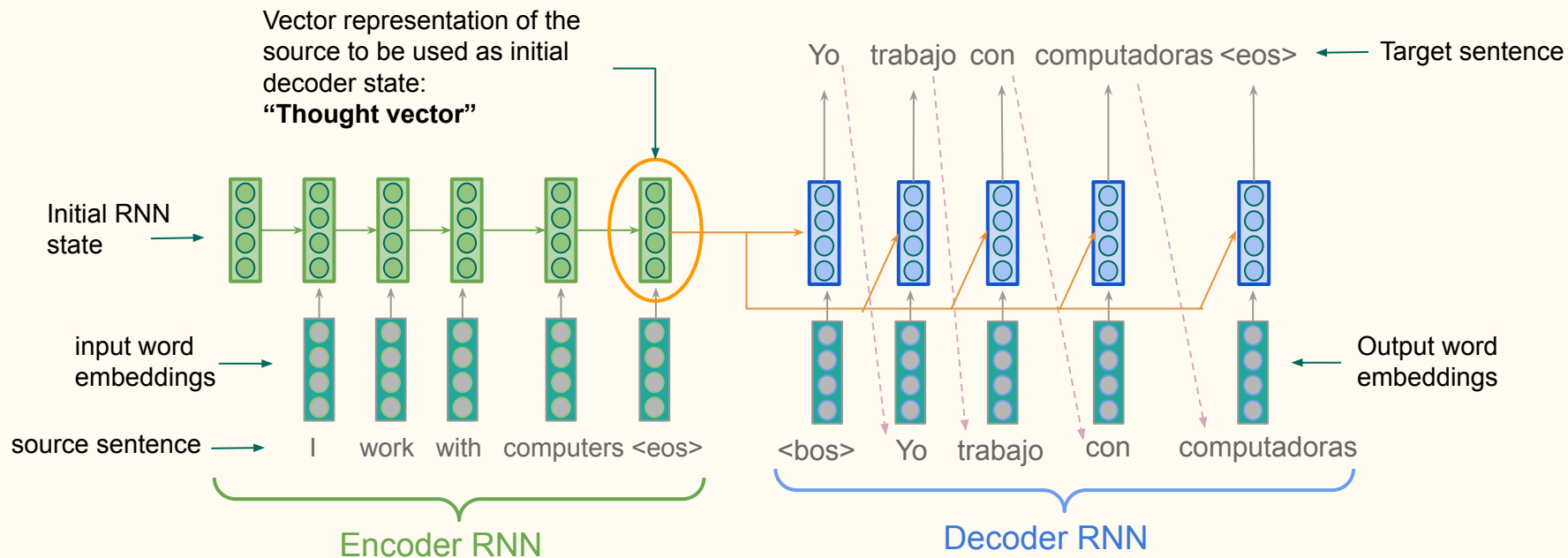
Seq2Seq Architectures

- This architecture works (after training) surprisingly good and is one of the more incredible NLP results.
- This is not the only possible architecture (although this has been shown to be the best)
- Also, in practice, these units are rarely RNNs. They are either GRU or LSTMs or BiLSTMs
- Models where information from previous time-steps is used to predict the output at the current time step are called autoregressive

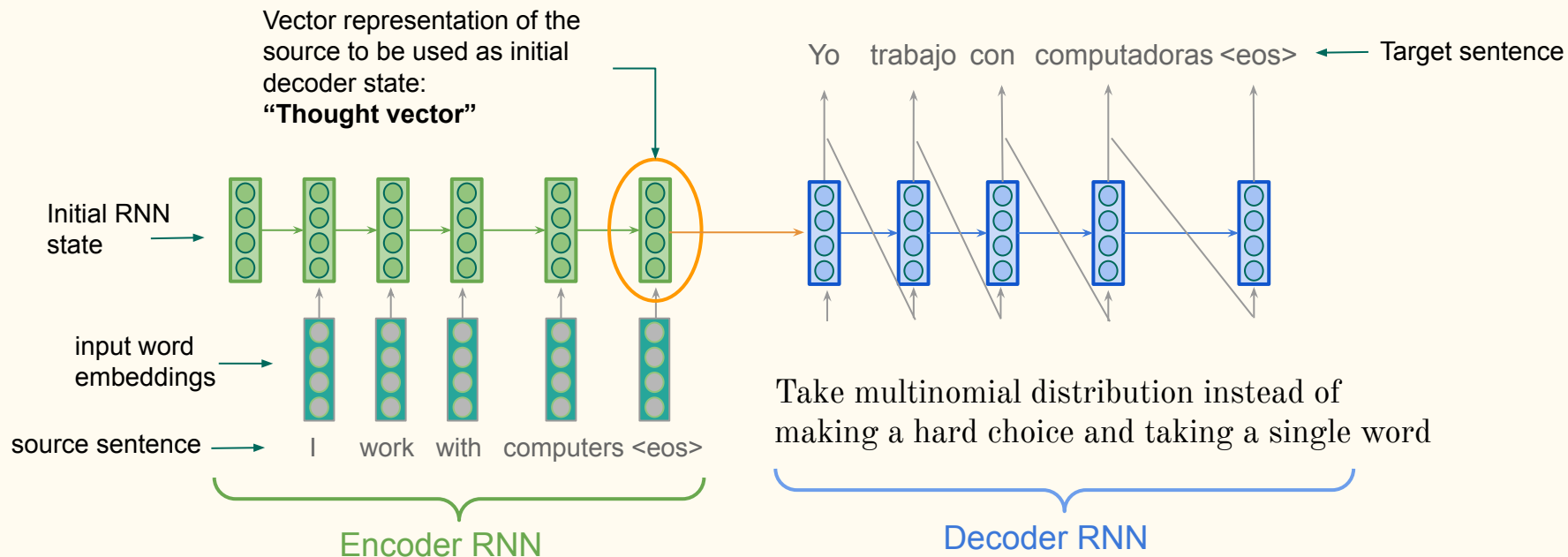
Can you think of alternatives?



Encoder-Decoder with RNNs: Alternative 1



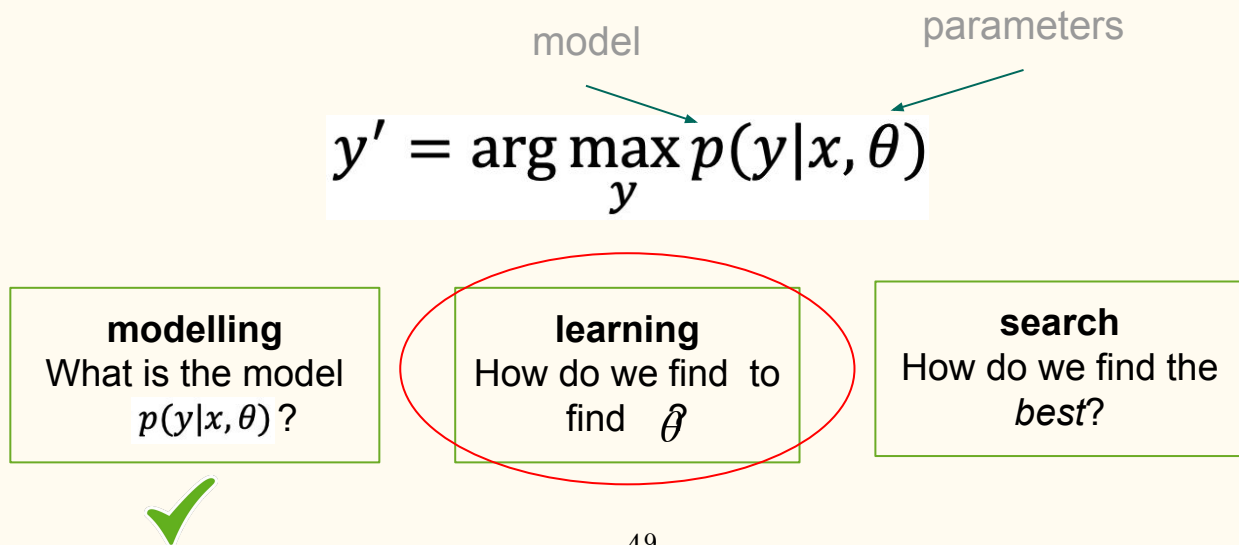
Encoder-Decoder with RNNs: Alternative 2



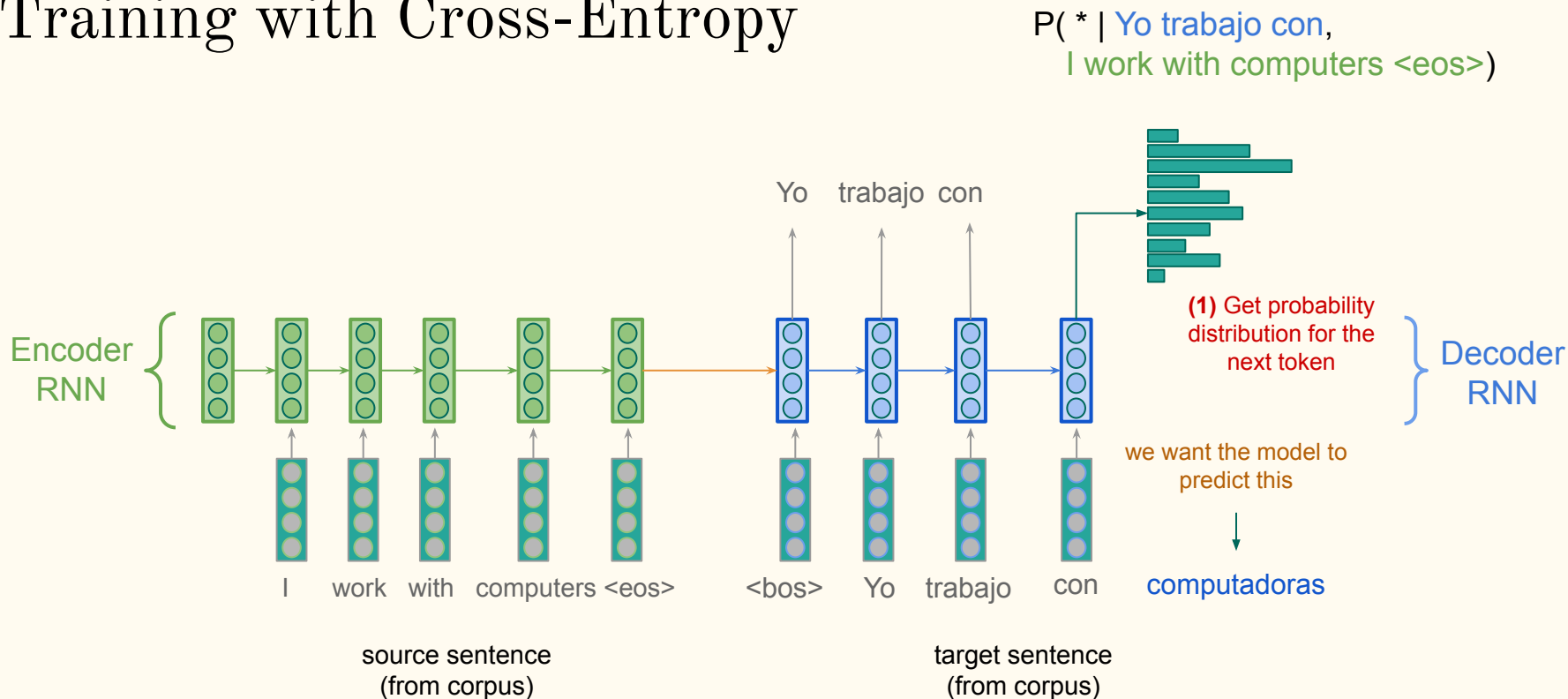
Task Formulation

More Formally:

- We want to find best Spanish sentence \mathbf{y} , given English sentence \mathbf{x}

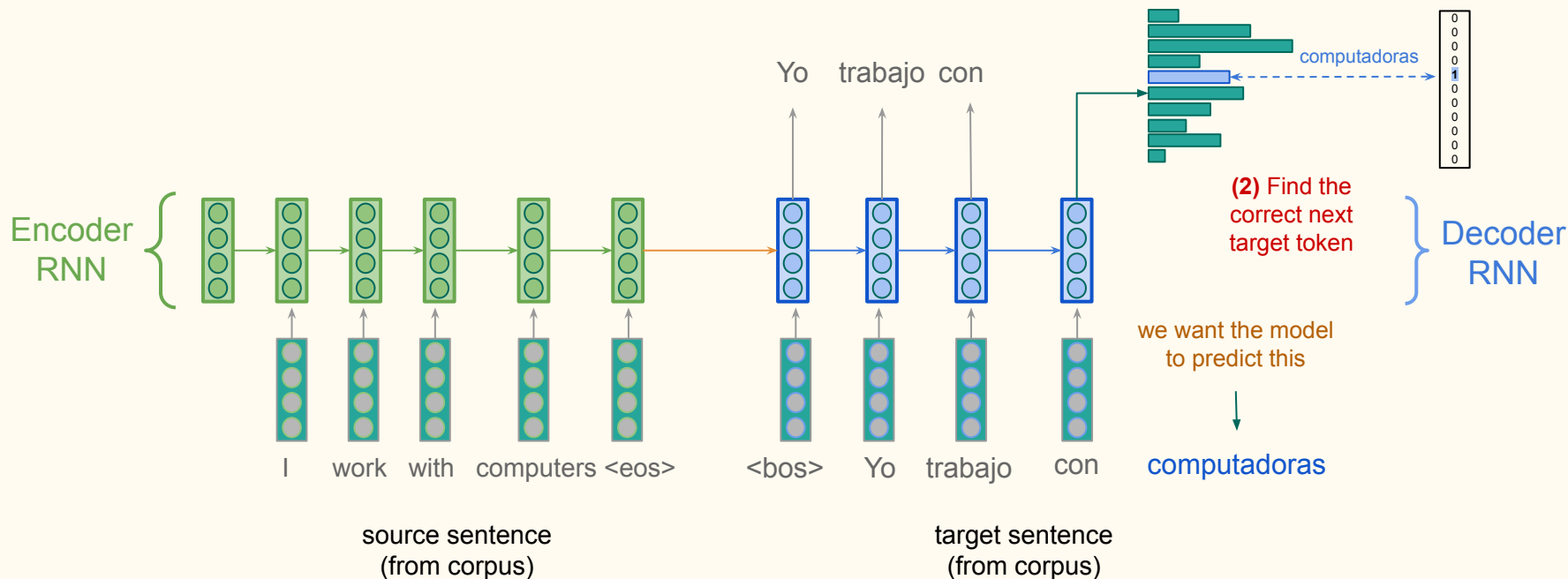


Training with Cross-Entropy



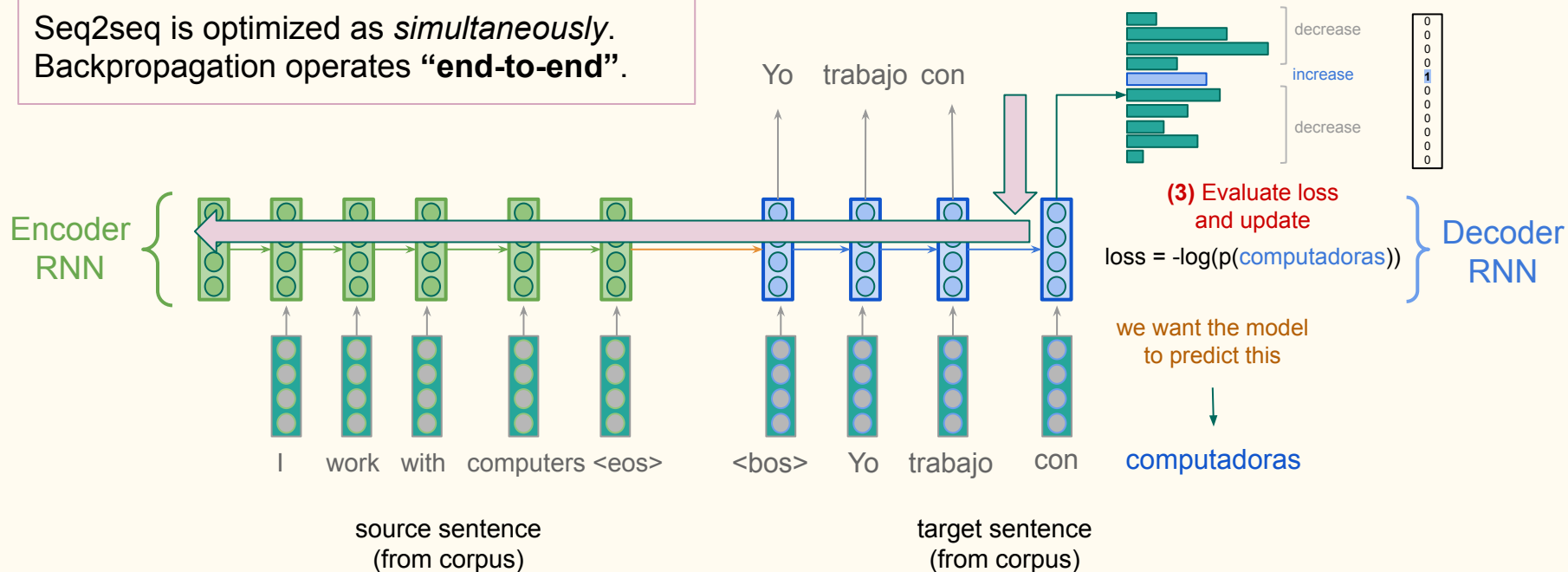
Training with Cross-Entropy

$P(* \mid \text{Yo trabajo con,}$
 $\text{I work with computers } \langle \text{eos} \rangle)$



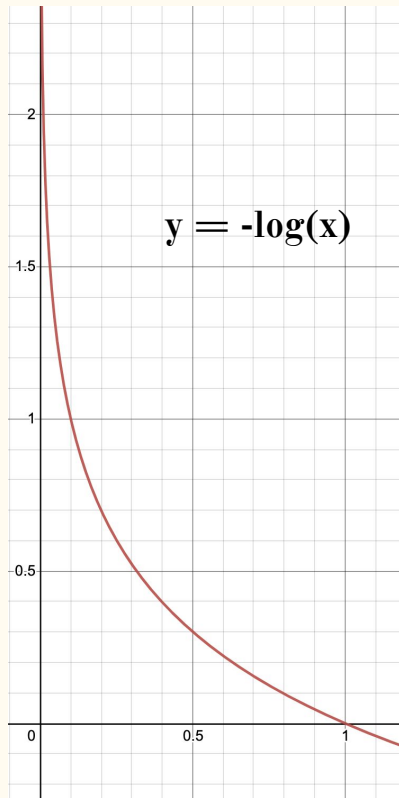
Training with Cross-Entropy

Seq2seq is optimized as *simultaneously*.
Backpropagation operates “**end-to-end**”.



Recall: Cross-Entropy Loss

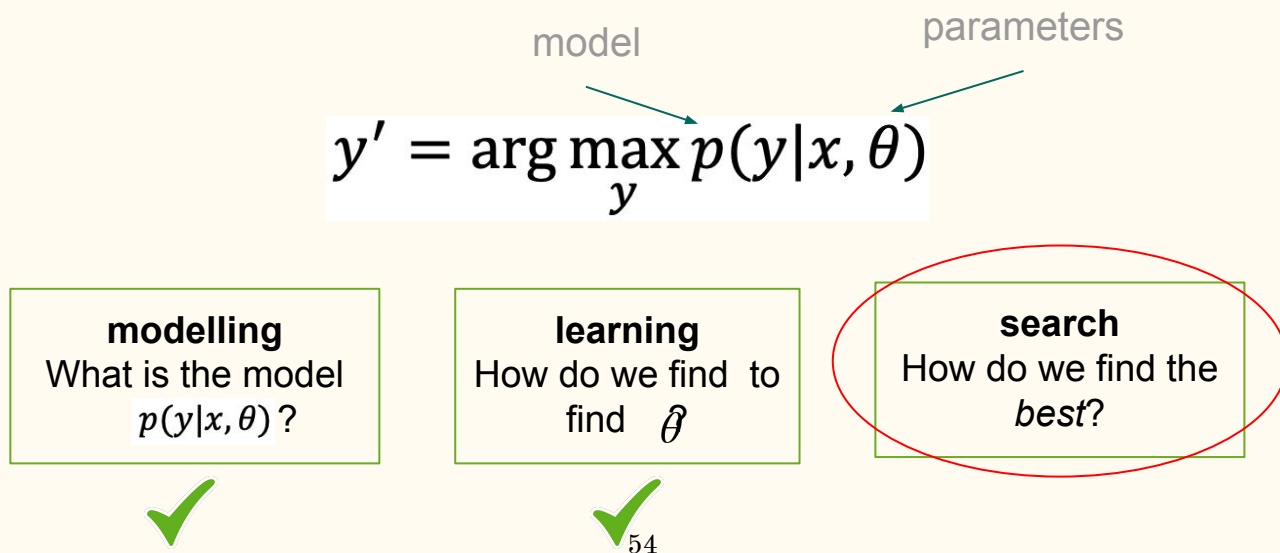
$$\text{loss} = -\log(p(\text{correct}))$$



Task Formulation

More Formally:

- We want to find best Spanish sentence \mathbf{y} , given English sentence \mathbf{x}



Goal: Search for the best candidate

$$y' = \arg \max_y p(y|x) = \arg \max_y \prod_{t=1}^n p(y_t | y_{<t}, x)$$

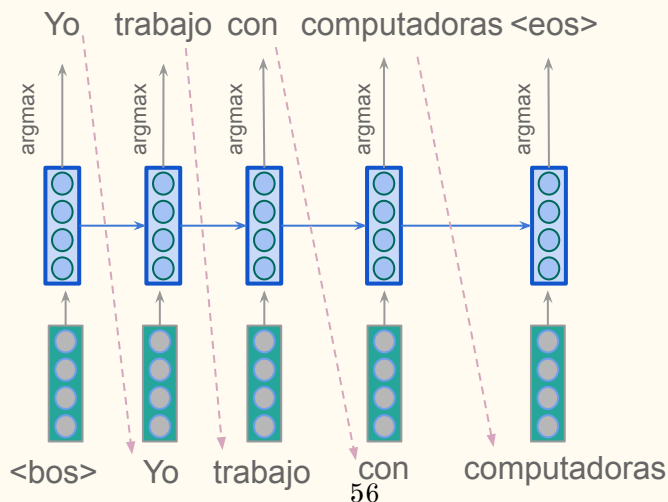
How do we find `argmax`?

A first attempt: Greedy Decoding

$$y' = \arg \max_y p(y|x) = \arg \max_y \prod_{t=1}^n p(y_t | y_{<t}, x)$$

How do we find $\arg\max$?

- **Greedy:** Take the $\arg\max$ (most probable word) on each step of the decoder



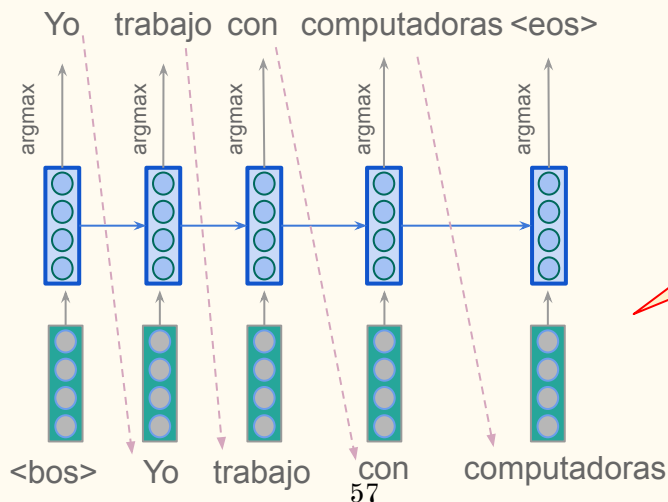
What is wrong with this solution?

A first attempt: Greedy Decoding

$$y' = \arg \max_y p(y|x) = \arg \max_y \prod_{t=1}^n p(y_t | y_{<t}, x)$$

How do we find $\arg\max$?

- **Greedy:** Take the $\arg\max$ (most probable word) on each step of the decoder



PROBLEM

The best token at the current step does not necessarily lead to the best sequence

Beam Search: Core Idea

- On each step of decoder, keep track of the **k** most probable partial translations (called **hypotheses**)
 - k is the **beam size** (in practice around 5 to 10)

- A hypothesis has a **score**:

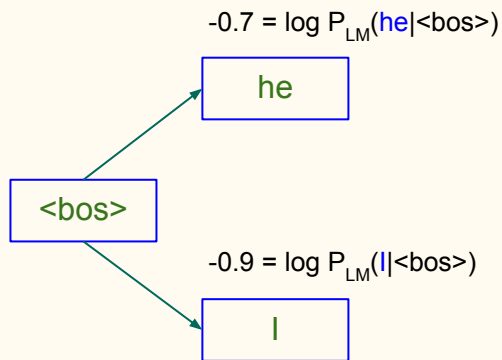
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Scores are all negative, and **higher score is better**
 - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is **not guaranteed** to find optimal solution
 - But it's efficient!

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

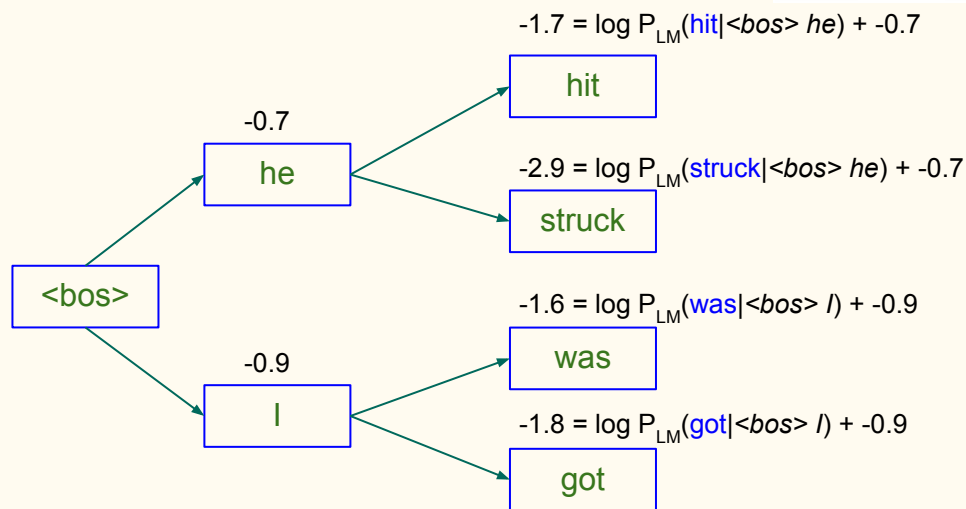


Find top k next words
and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

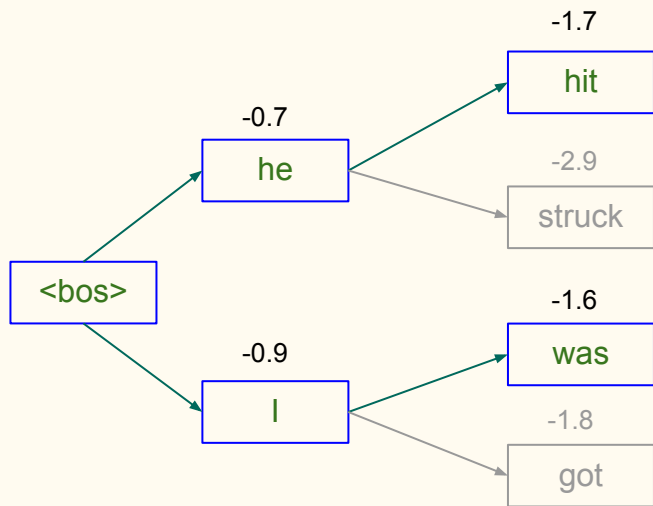


For each of the k hypotheses, find top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

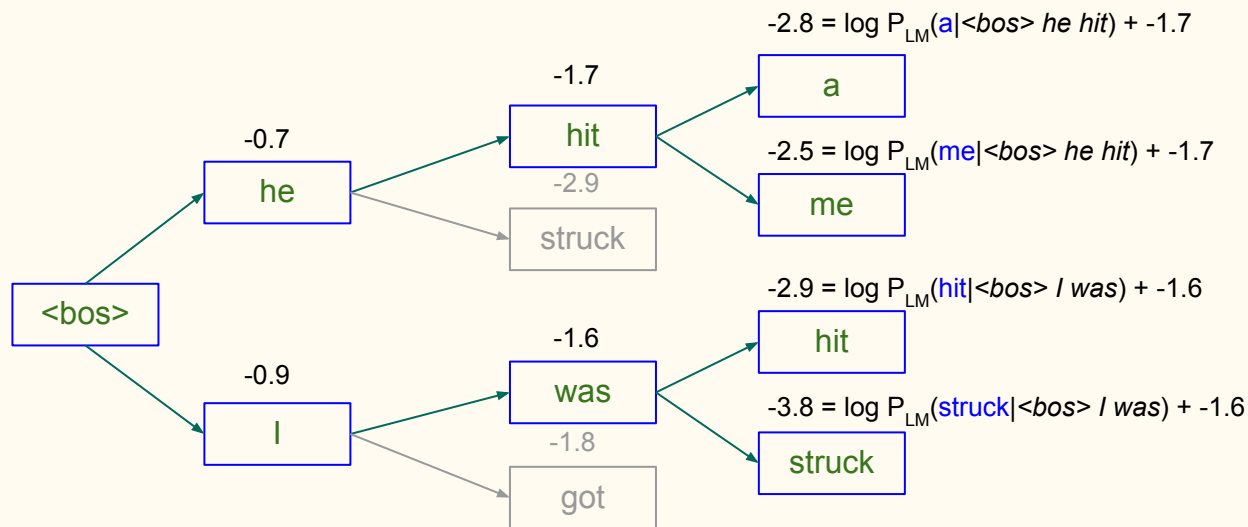


Of these k^2 hypotheses,
just keep k with highest scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

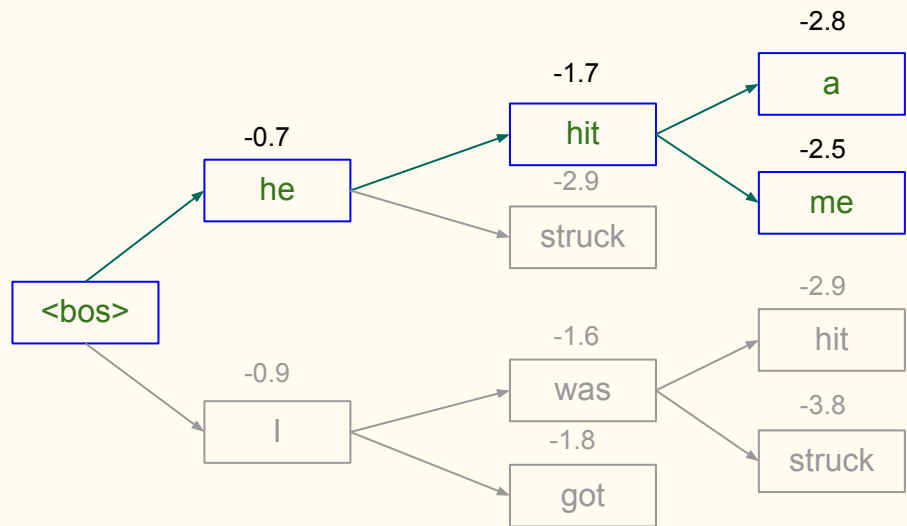


For each of the k hypotheses, find
top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

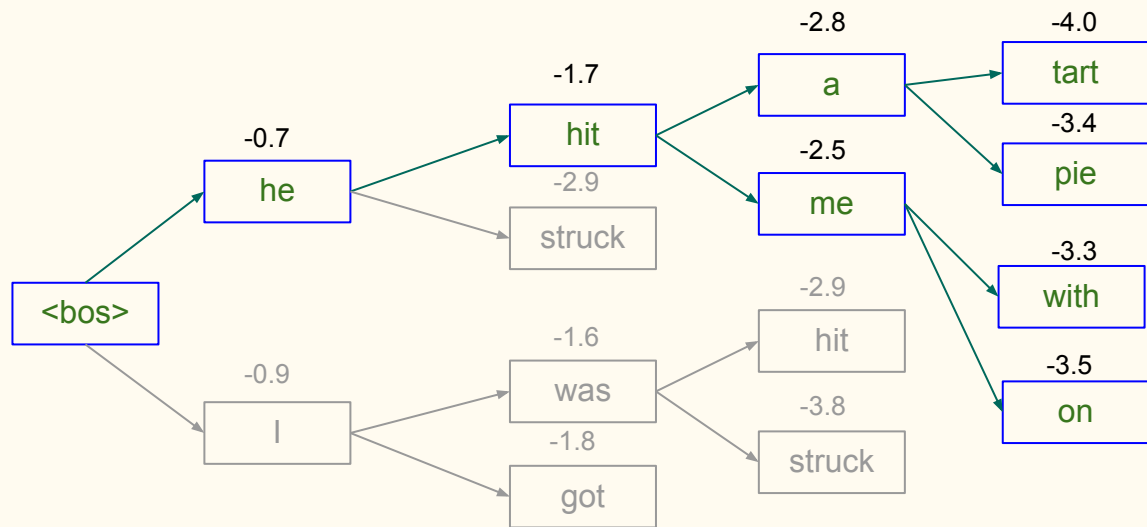


Of these k^2 hypotheses,
just keep k with highest scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

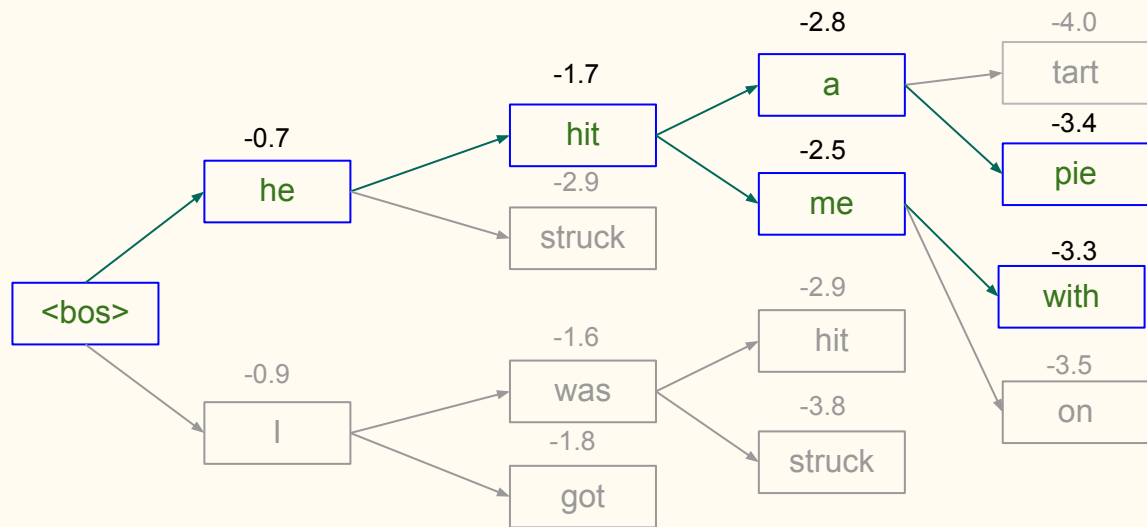


For each of the k hypotheses, find top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

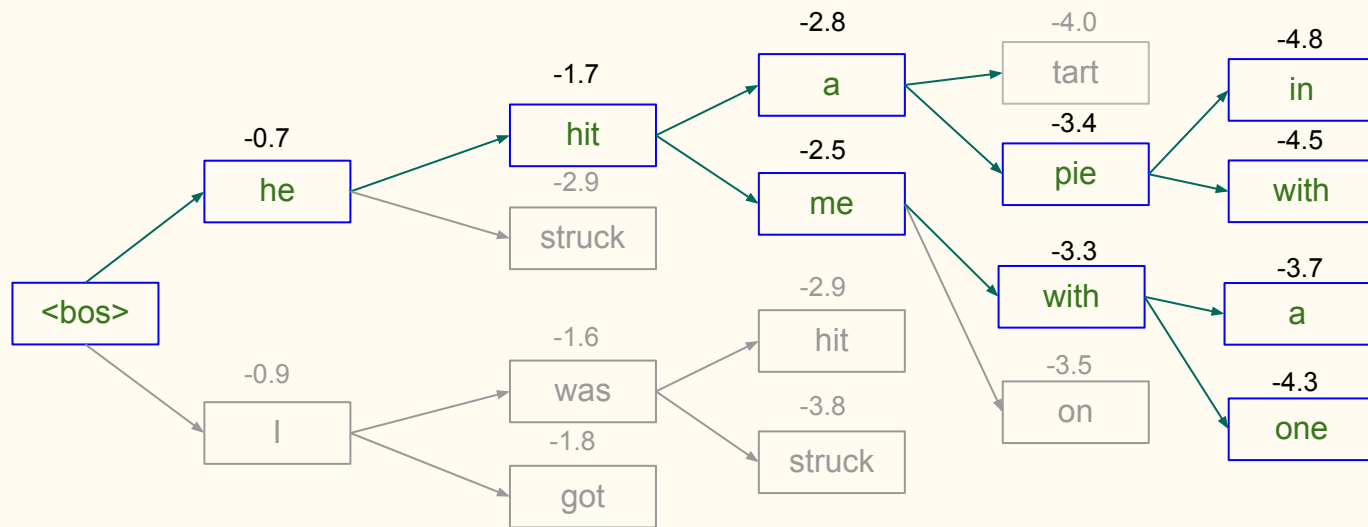


Of these k^2 hypotheses,
just keep k with highest scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

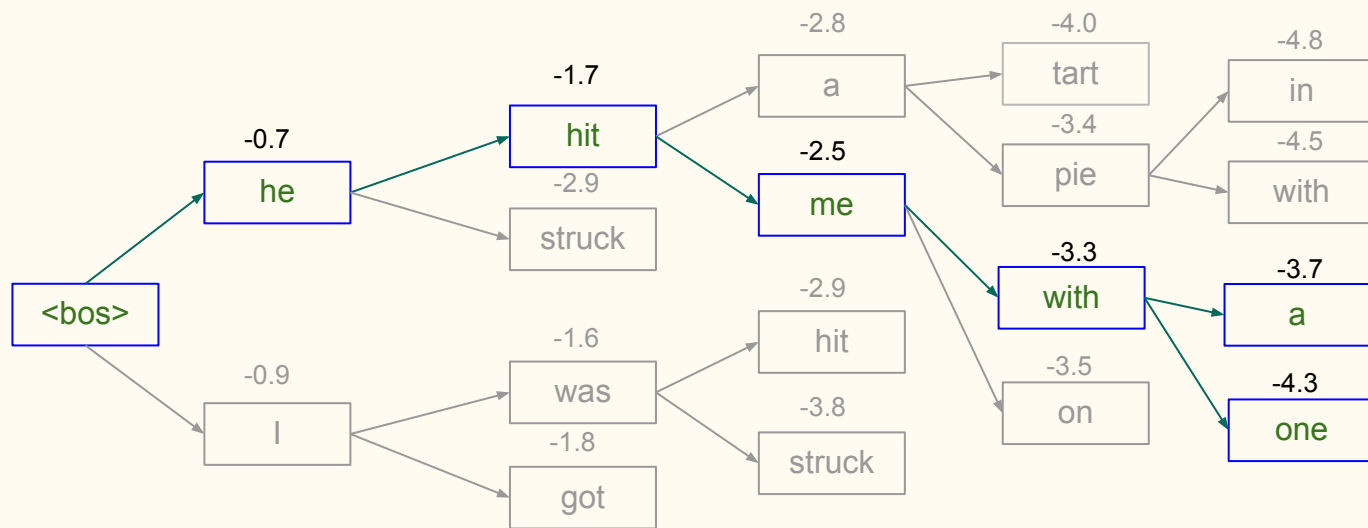


For each of the k hypotheses, find top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

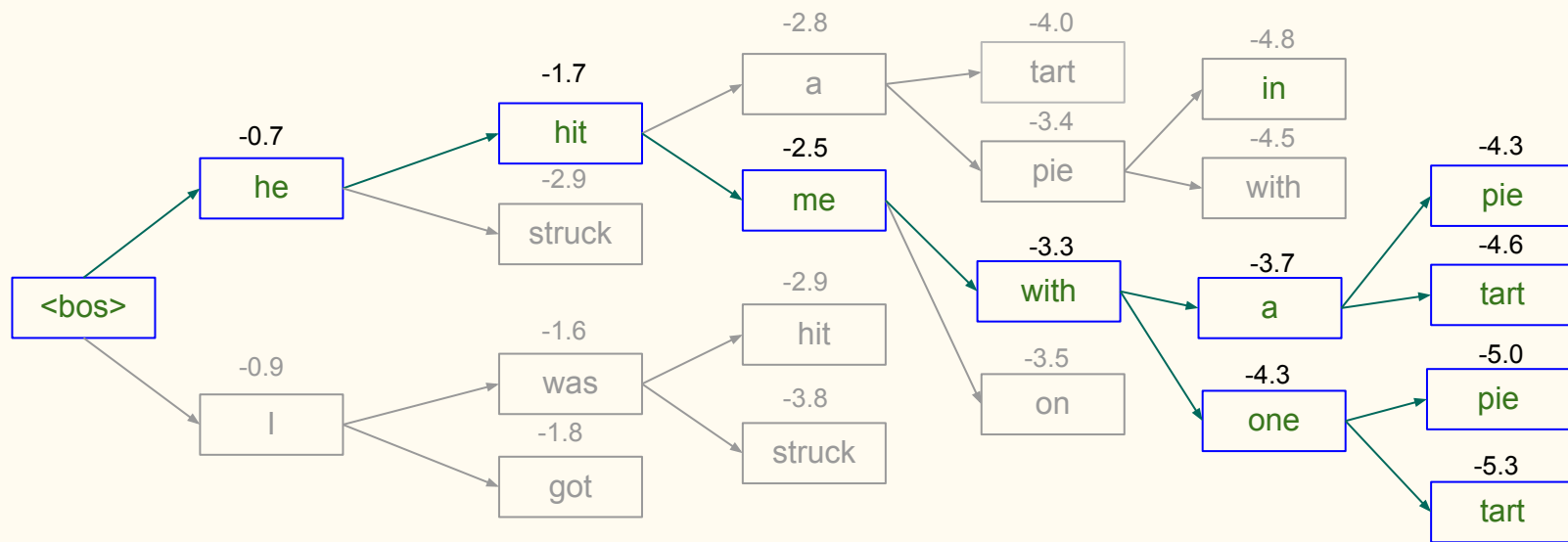


Of these k^2 hypotheses,
just keep k with highest scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

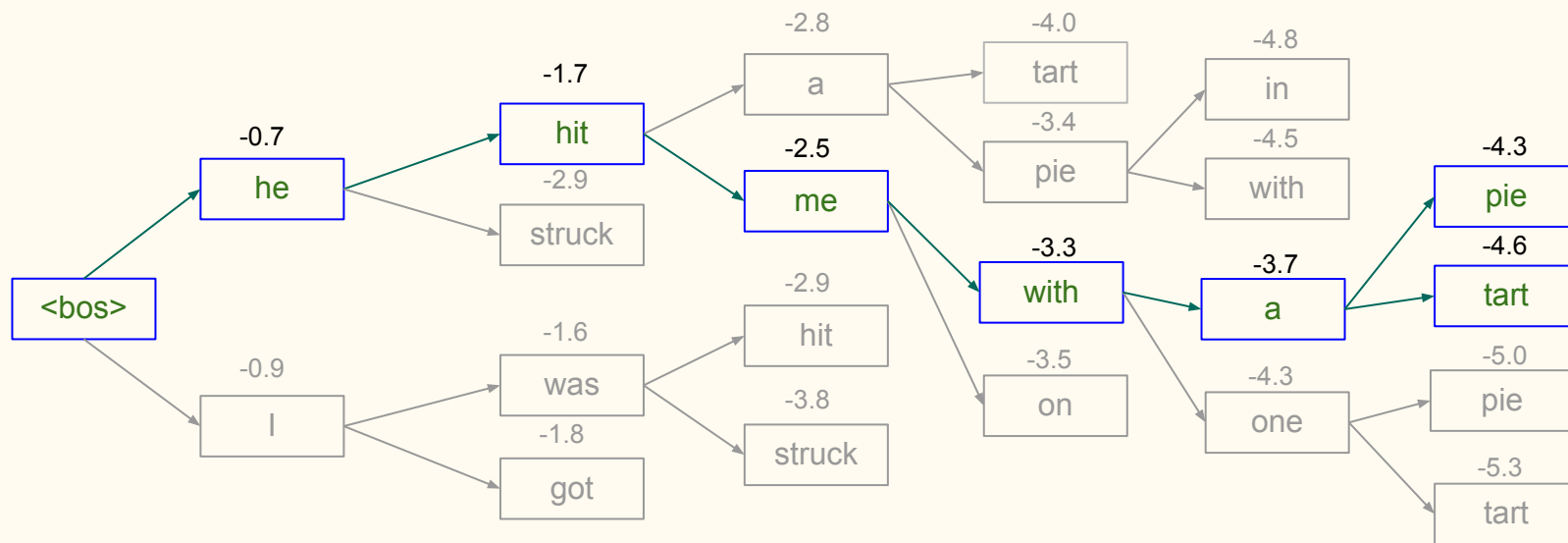


For each of the k hypotheses, find top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

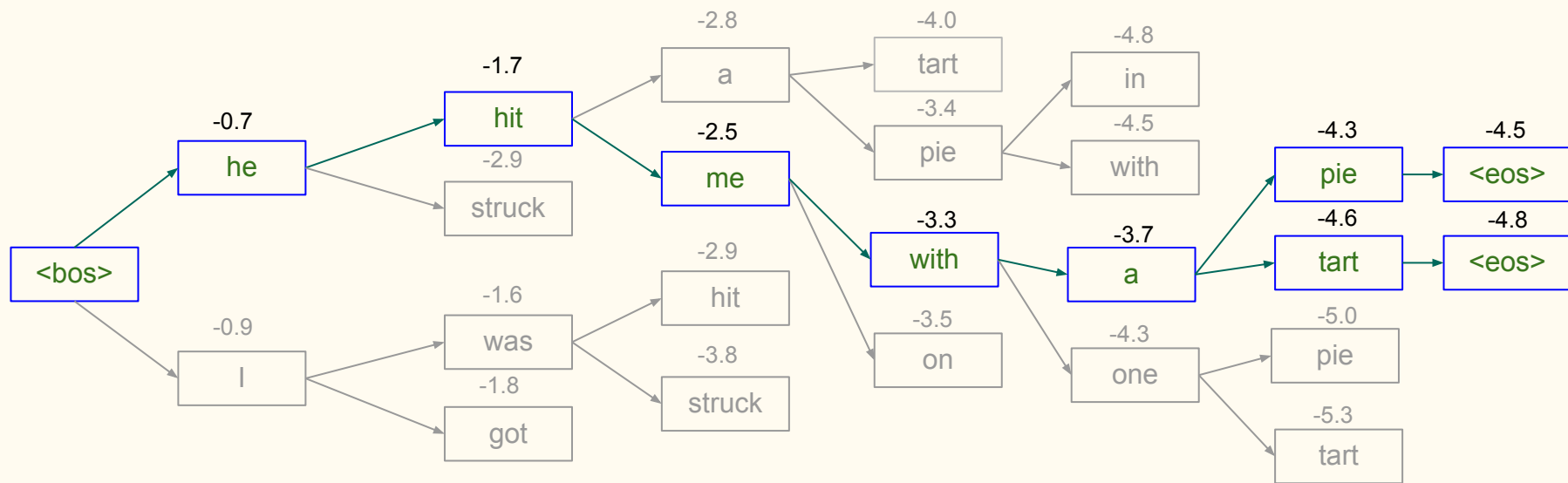


Of these k^2 hypotheses,
just keep k with highest scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

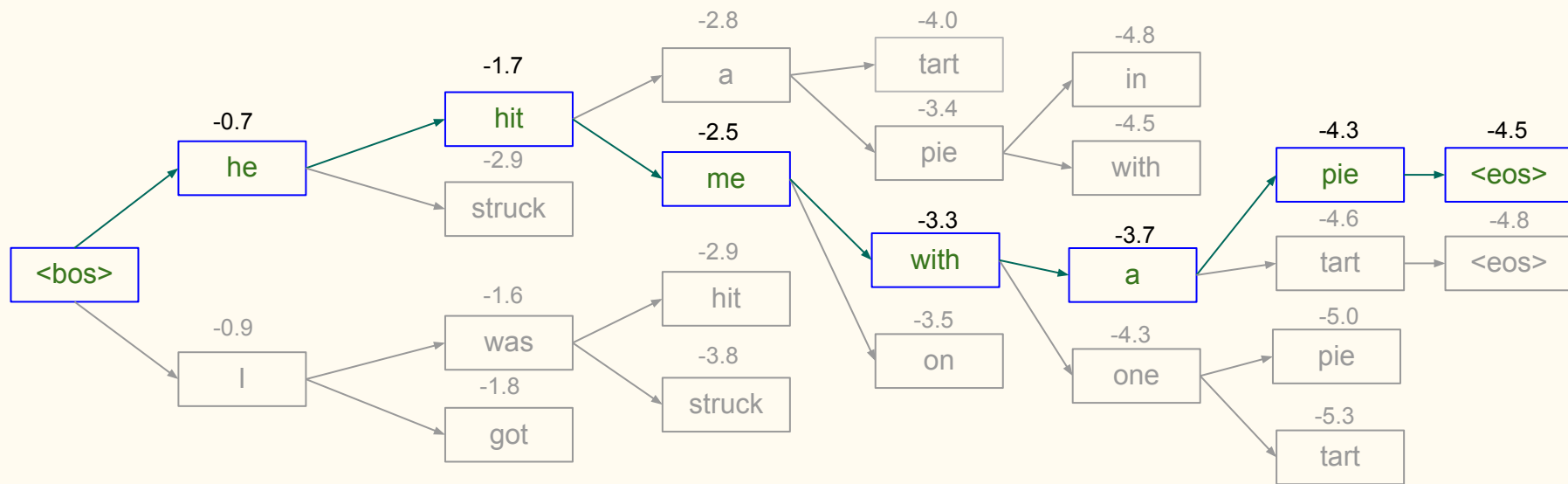


For each of the k hypotheses, find top k next words and calculate scores

Beam Search: Example

Beam Size = $k = 2$

$$\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$



We found the top-scoring hypothesis!

Beam Search: Stopping Criteria

- Different hypotheses may produce $\langle \text{eos} \rangle$ tokens on different timesteps
 - When a hypothesis produces $\langle \text{eos} \rangle$, that hypothesis is complete
 - Place it aside and continue exploring other hypotheses via beam search
- Usually we continue beam search until:
 - We reach timestep T (where T is some pre-defined cutoff), or
 - We have at least n completed hypotheses (where n is pre-defined cutoff)

Task Formulation

- Suppose we are translating French \rightarrow English
- We want to find best English sentence y , given French sentence x

$$y' = \arg \max_y p(y|x, \theta)$$

model

parameters

modelling

What is the model
for $p(y|x, \theta)$?



learning

How to find θ ?



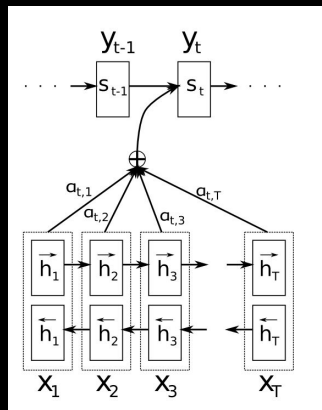
search

How to find the
argmax?



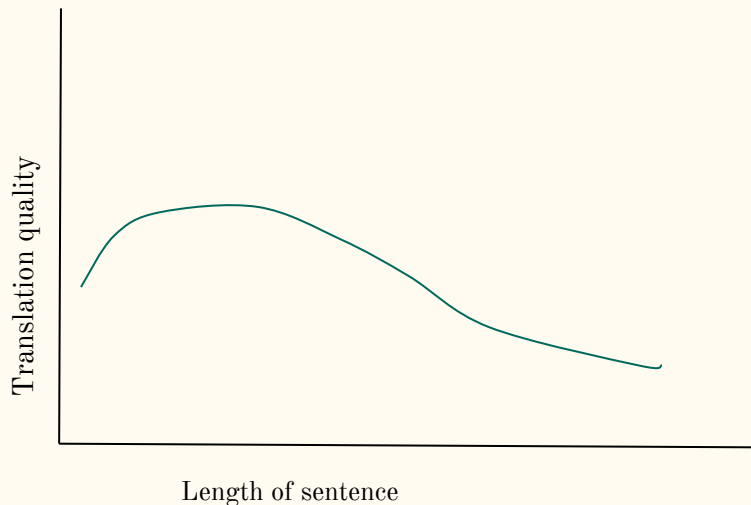
In this Section...

- Seq2Seq models are implemented with an **Encoder-Decoder** architecture
 - The Encoder creates a representation of the **source** sentence (thought vector)
 - The Decoder uses that representation to generate the **target** sentence
- Encoder and Decoder can be implemented using RNNs
 - These are trained end-to-end using **cross-entropy loss**
 - For efficient inference (test time), we can use **beam search**



Improvement Attention

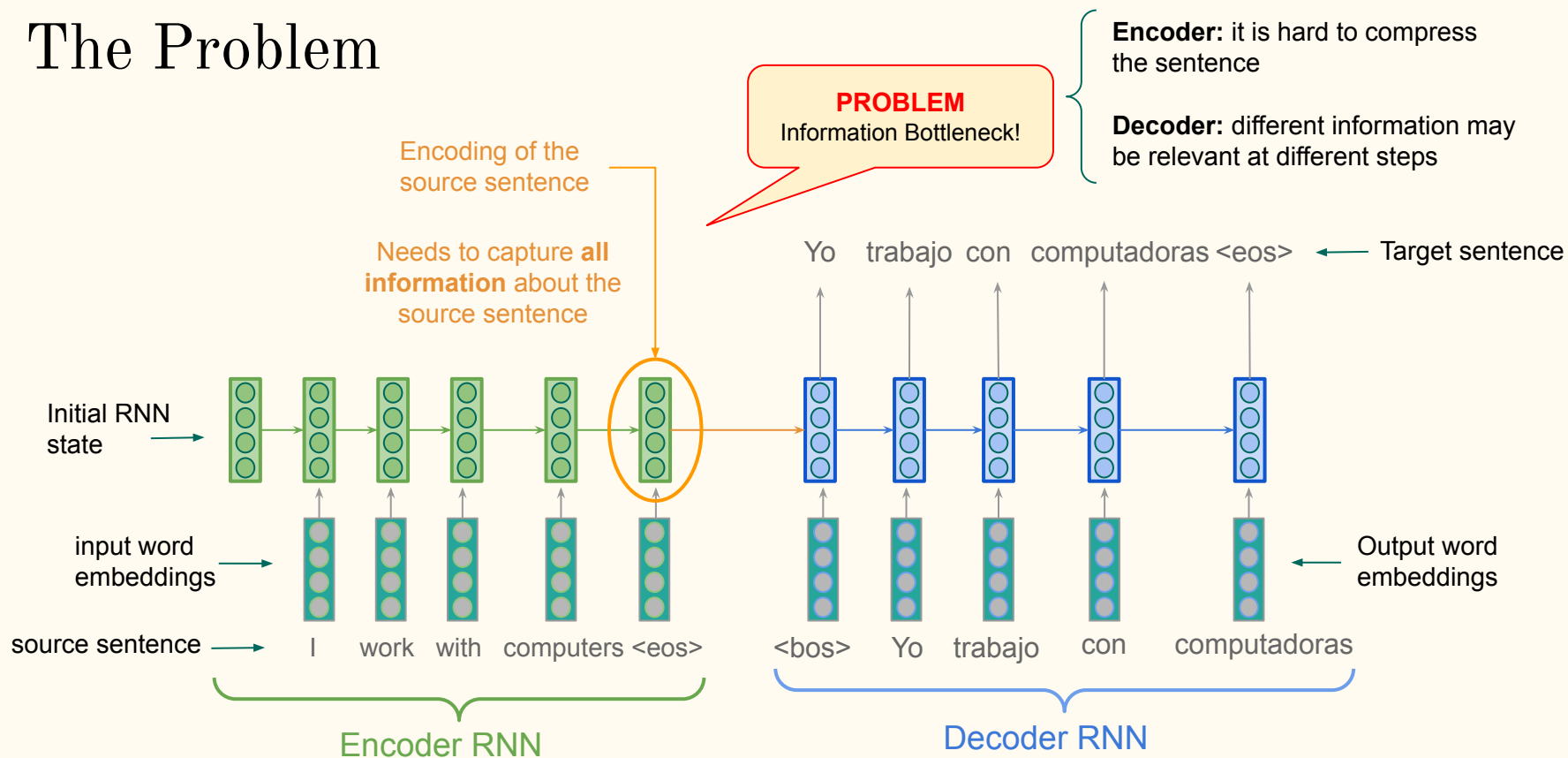
The Problem



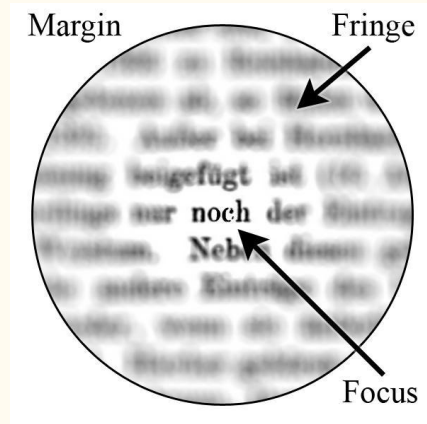
Translation quality drops with increase in the length of sentences.

(Notice that the encoder has to “pack” more information into the same vector representation or thought vector)

The Problem



Attention: A Biological Motivation



Focusing on specific information is the only way to avoid information overload.

Critical to biological systems. (Only way to avoid information overload, remember information processing is expensive)

Attention

We need a way to allow models to **attend to** (focus on) different parts of the input at different timesteps. (Attention was previously used in vision Larochelle & Hinton, 2010)

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau

Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*

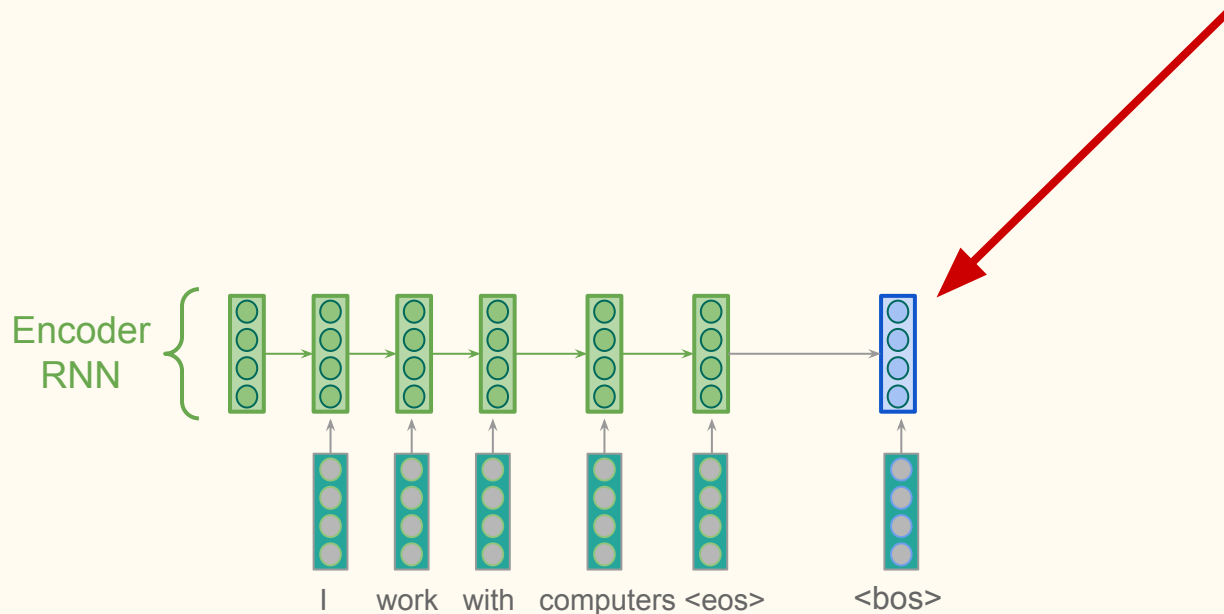
Université de Montréal

Attention

- On each step of the decoder, use **direct connection to the encoder** to **focus on a particular part** of the source sequence
 - Introduced in “*Neural Machine Translation by Jointly Learning to Align and Translate*” (Bahdanau et al., 2015)
- The encoder does not have to compress the whole source into a single vector
 - It gives **representations for all source tokens** (e.g., all RNN states instead of the last one)

Sequence-to-Sequence with Attention: The Goal

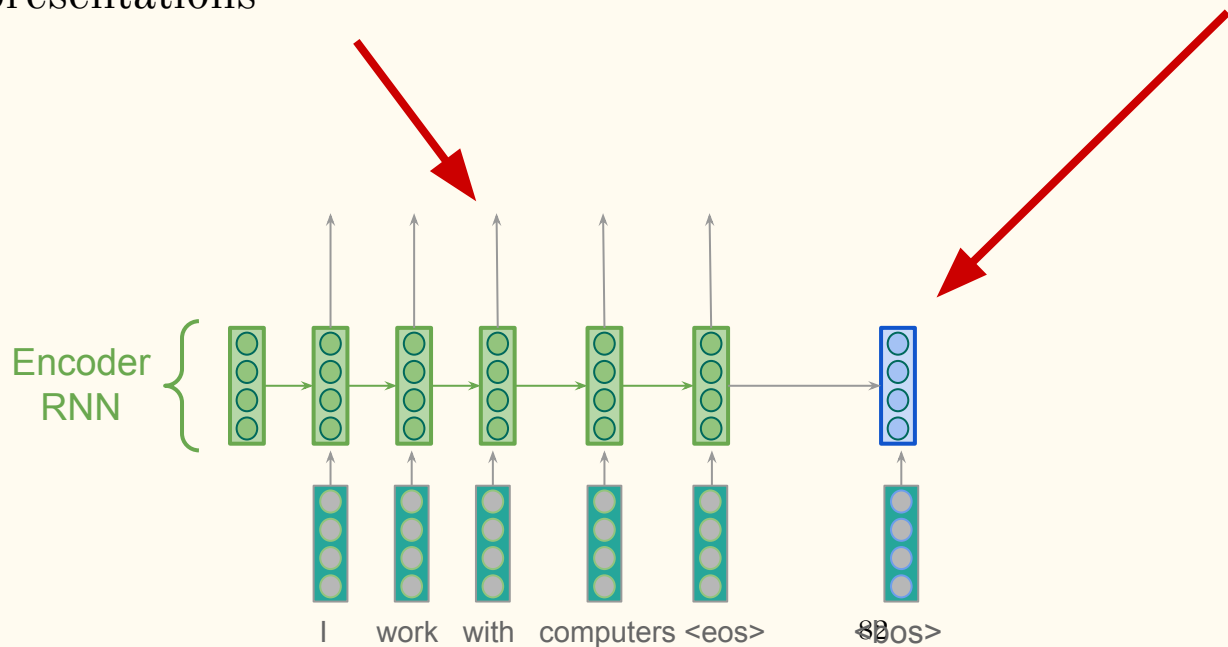
In generating the output at this step



Sequence-to-Sequence with Attention: The Goal

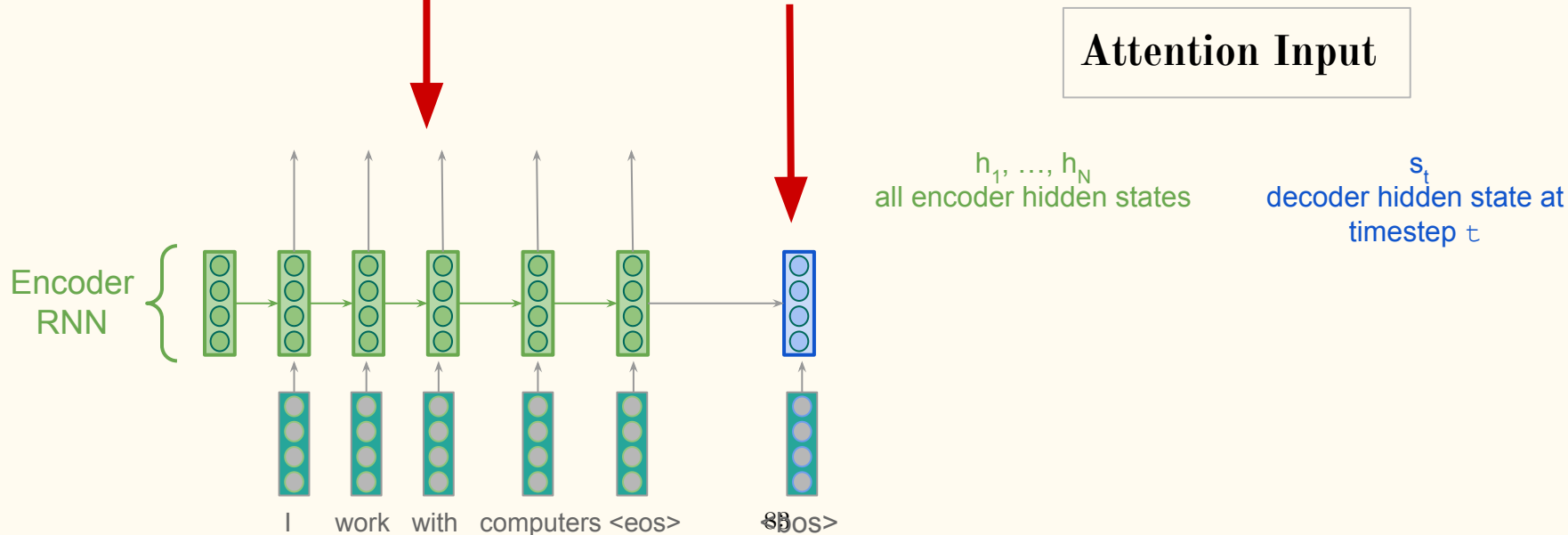
We want to make use of some of these representations

when generating the output at this step



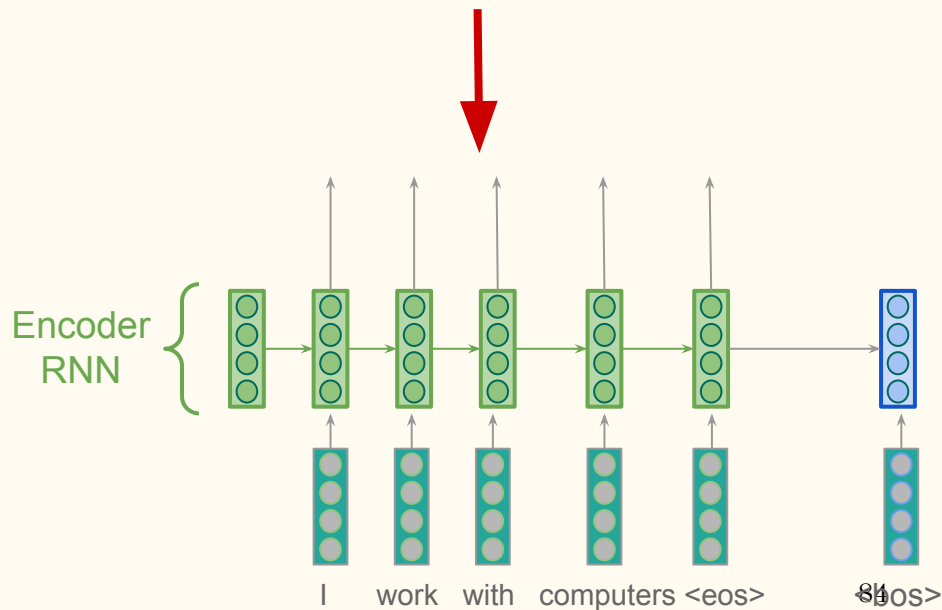
Sequence-to-Sequence with Attention

We use a weighted sum of these based on how “important” each is to generate this output



Sequence-to-Sequence with Attention

Step 1: Find out how “important”
each of these are.



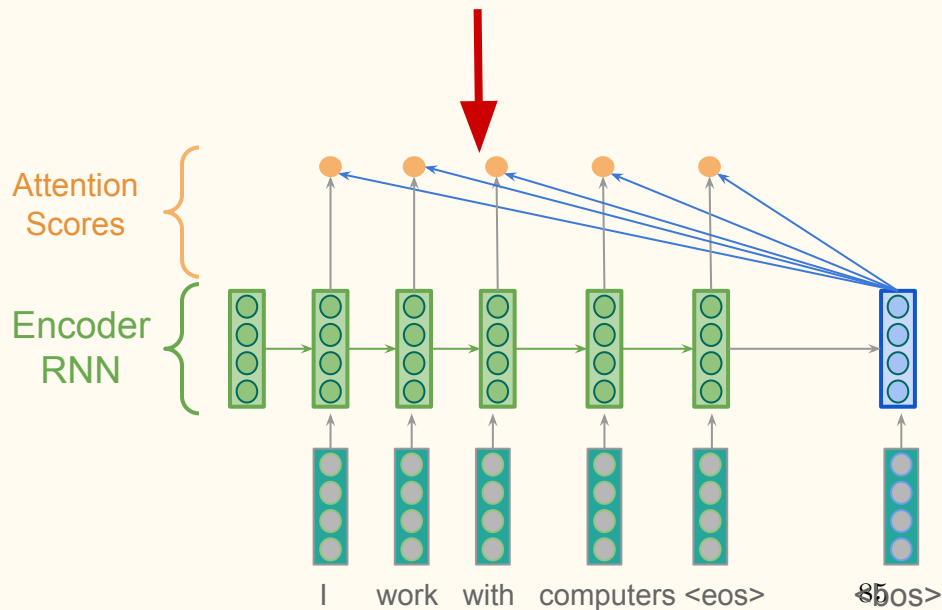
Sequence-to-Sequence with Attention

Step 1: Find out how “important”
each of these are.

Attention Scores

$$\text{score}(s_t, h_k), k = 1..N$$

“How relevant is source token k for target step t ?”

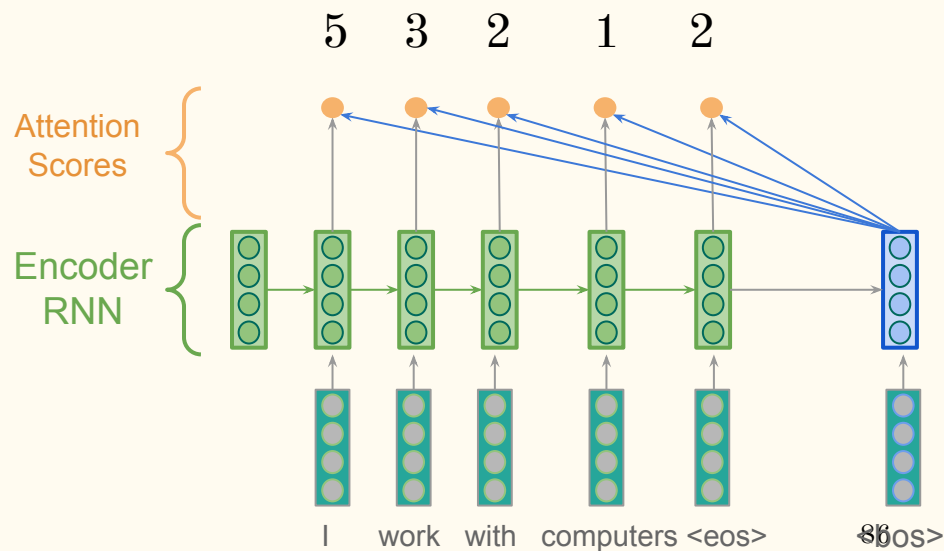


Sequence-to-Sequence with Attention

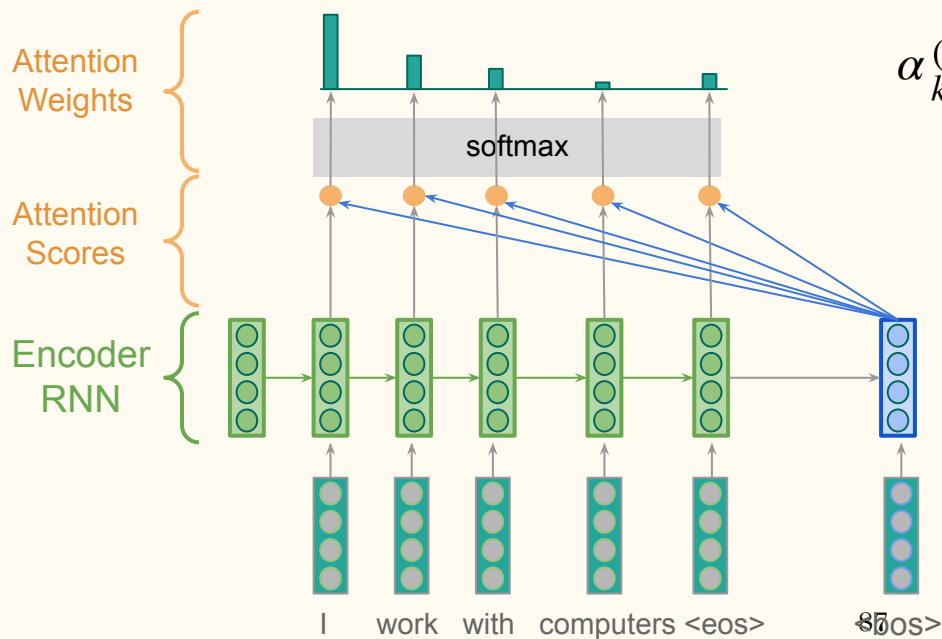
Attention Scores

$$\text{score}(s_t, h_k), k = 1..N$$

“How relevant is source token k for target step t ?”



Sequence-to-Sequence with Attention

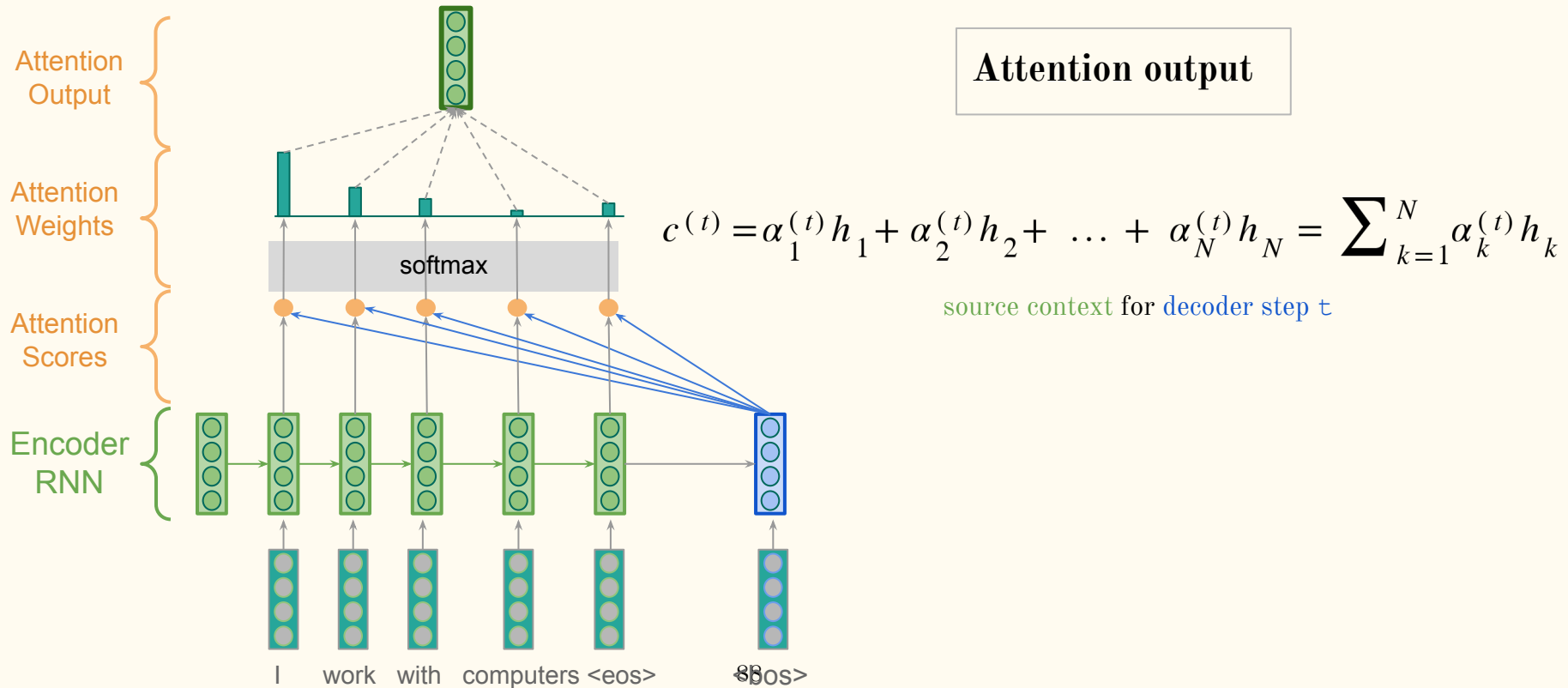


Attention Weights

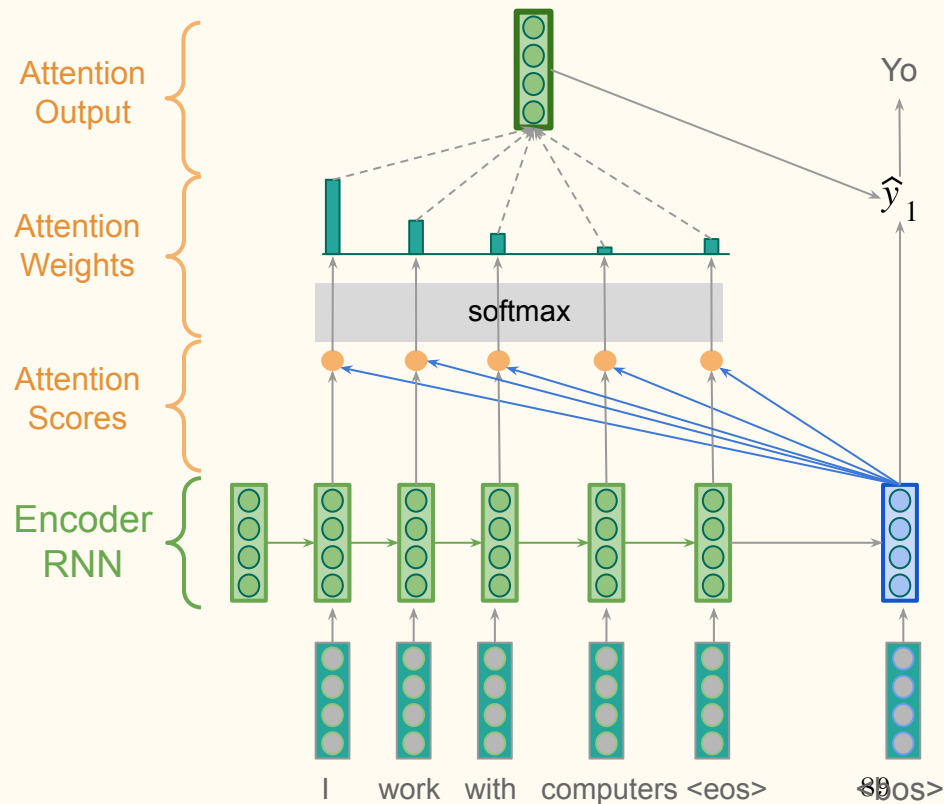
$$\alpha_k^{(t)} = \frac{\exp(\text{score}(s_t, h_k))}{\sum_{i=1}^N \exp(\text{score}(s_t, h_i))}, k = 1..N$$

WARNING: Do not confuse attention weights with learnable weights.

Sequence-to-Sequence with Attention

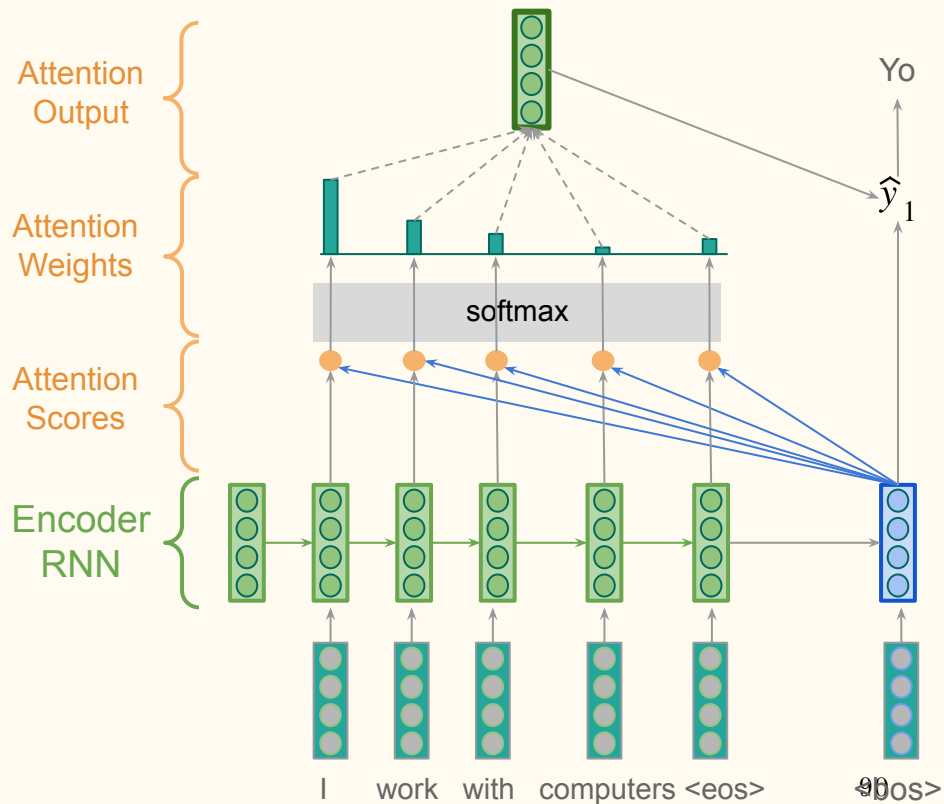


Sequence-to-Sequence with Attention



Now combine the **attention output** with the **internal representation** of the decoder

Sequence-to-Sequence with Attention



(1) **Attention Scores:** For each encoder state, compute its "relevance" for the current decoder state

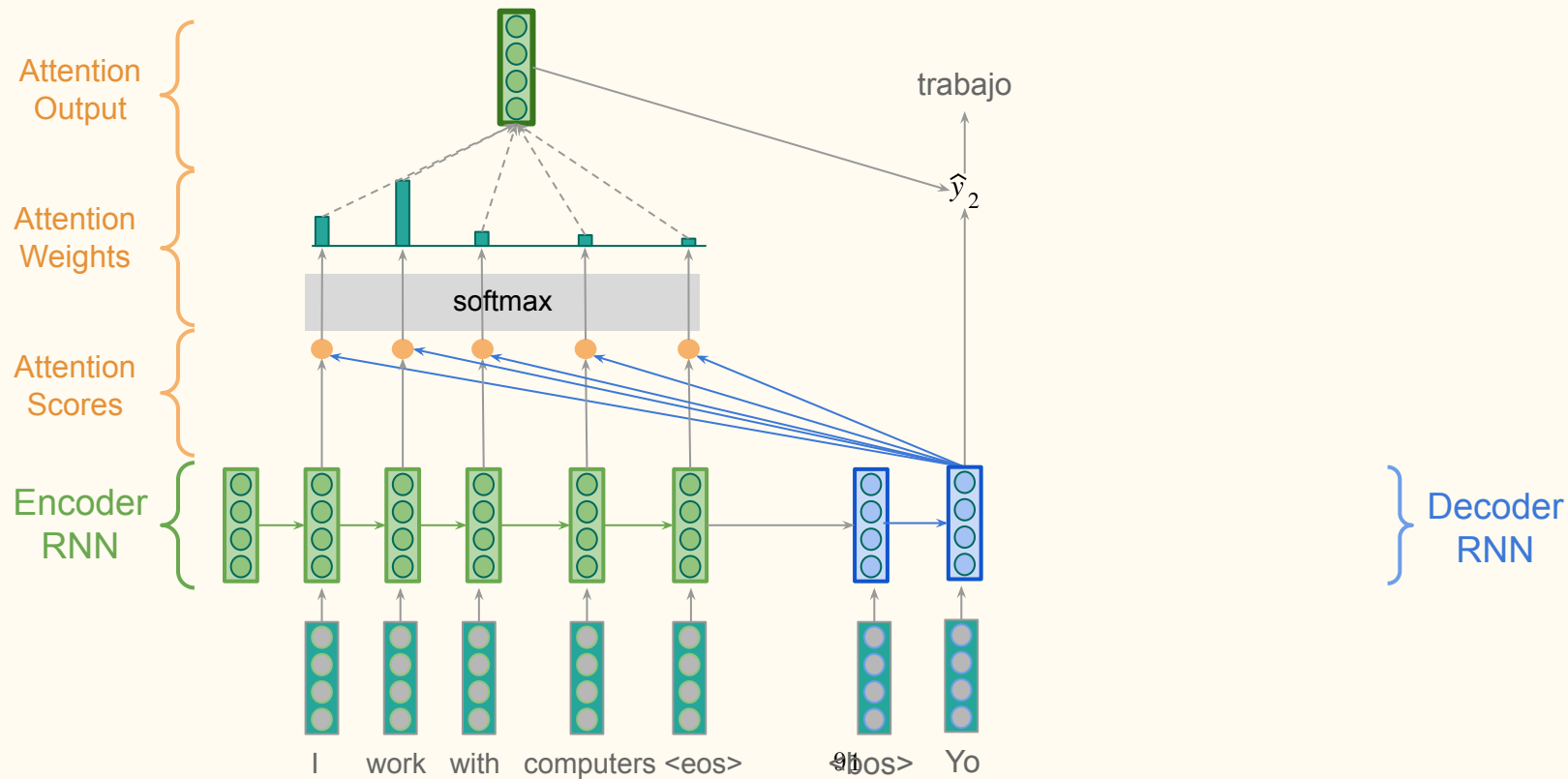
(2) **Attention Weights:** Apply `softmax` to the attention scores to obtain a probability distribution

(3) **Attention Output:** Compute a weighted sum of the encoder hidden states with attention weights

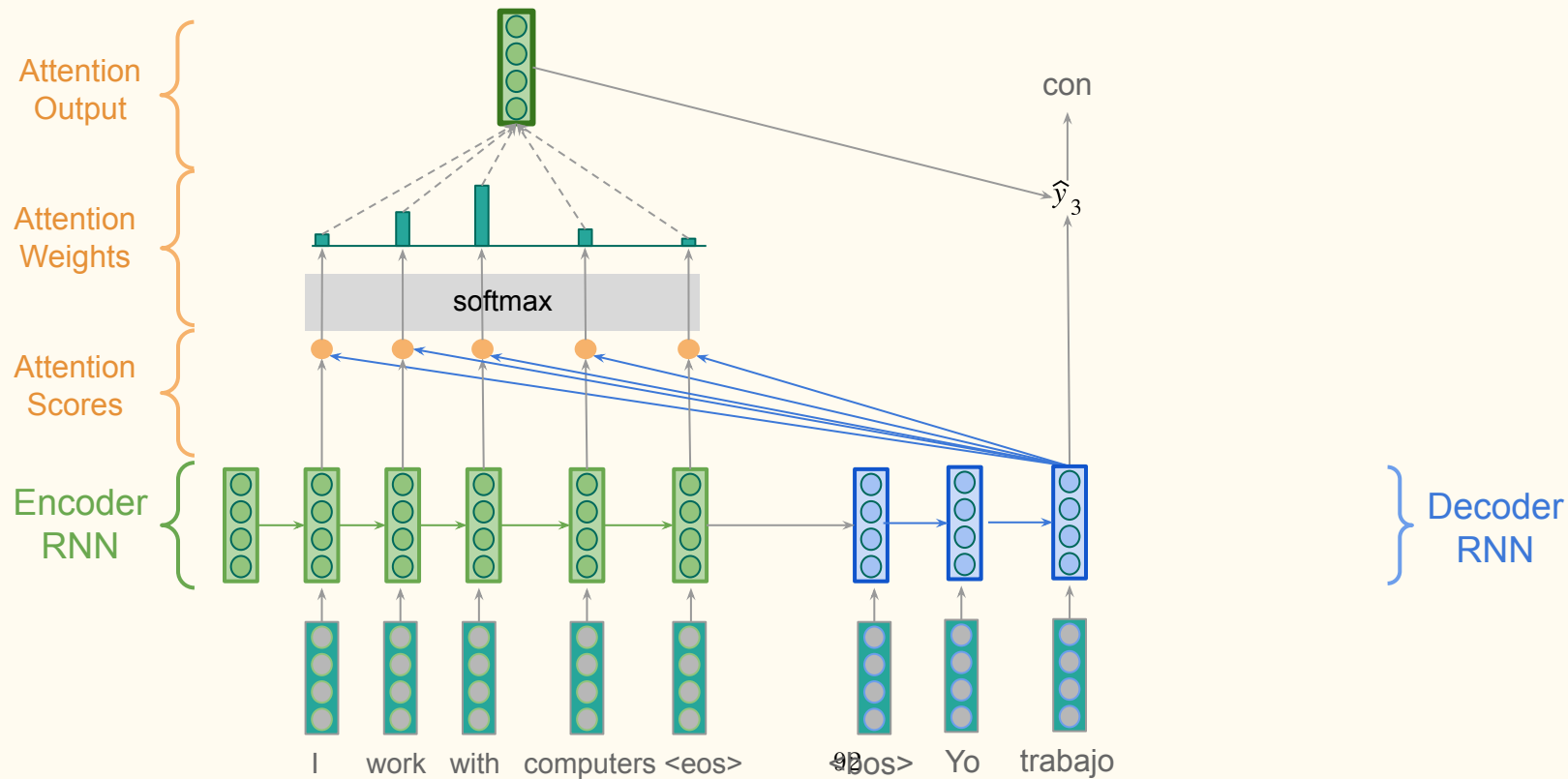
(4) Concatenate attention output with decoder hidden state, then use to compute \hat{y}_1

Decoder RNN

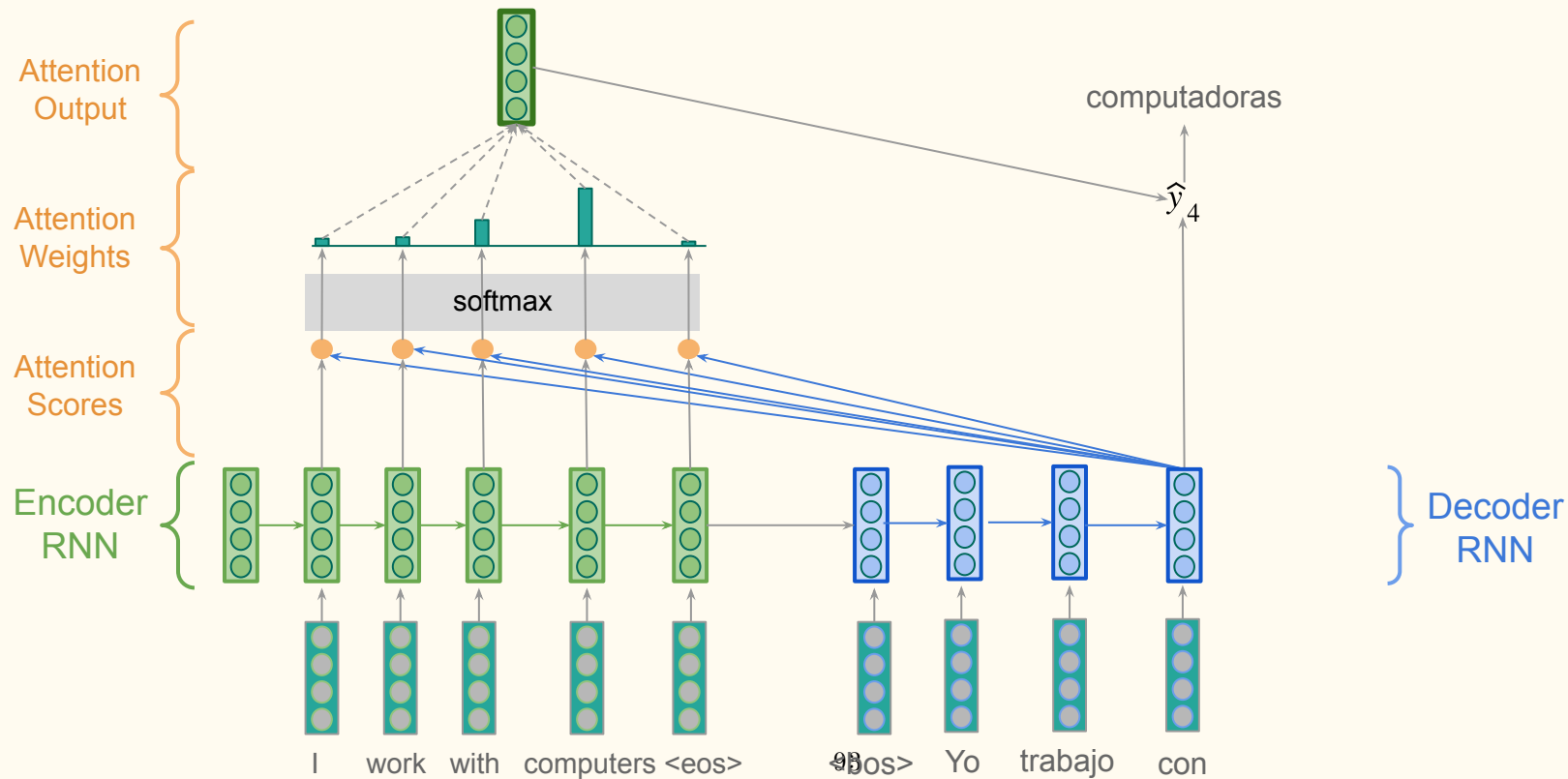
Sequence-to-Sequence with Attention



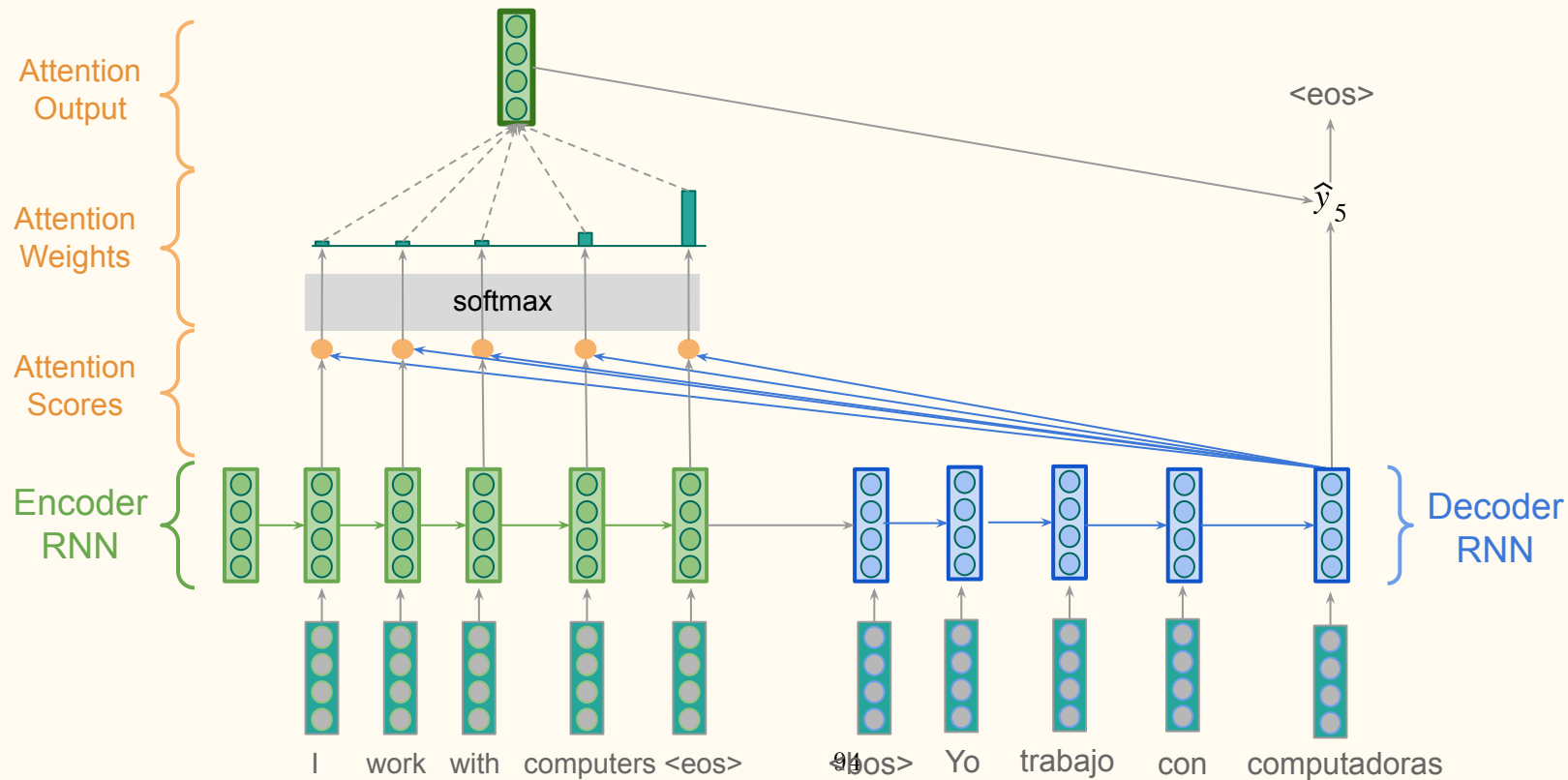
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Computation Pipeline

- **Attention Input**

h_1, \dots, h_N s_t
all encoder hidden states decoder hidden state at timestep t

- **Attention Scores**

$\text{score}(s_t, h_k), k = 1..N$ How do we compute score?

“How relevant is source token k for target step t ?”

- **Attention Weights**

$$\alpha_k^{(t)} = \frac{\exp(\text{score}(s_t, h_k))}{\sum_{i=1}^N \exp(\text{score}(s_t, h_i))}, k = 1..N$$

- **Attention Output**

$$c^{(t)} = \alpha_1^{(t)} h_1 + \alpha_2^{(t)} h_2 + \dots + \alpha_N^{(t)} h_N = \sum_{k=1}^N \alpha_k^{(t)} h_k$$

“source context for decoder step t ”

Attention Score Functions

Several ways to compute $score \in \mathbb{R}^N$ from $h_1, \dots, h_N \in \mathbb{R}^{d_1}$ and $s \in \mathbb{R}^{d_2}$:

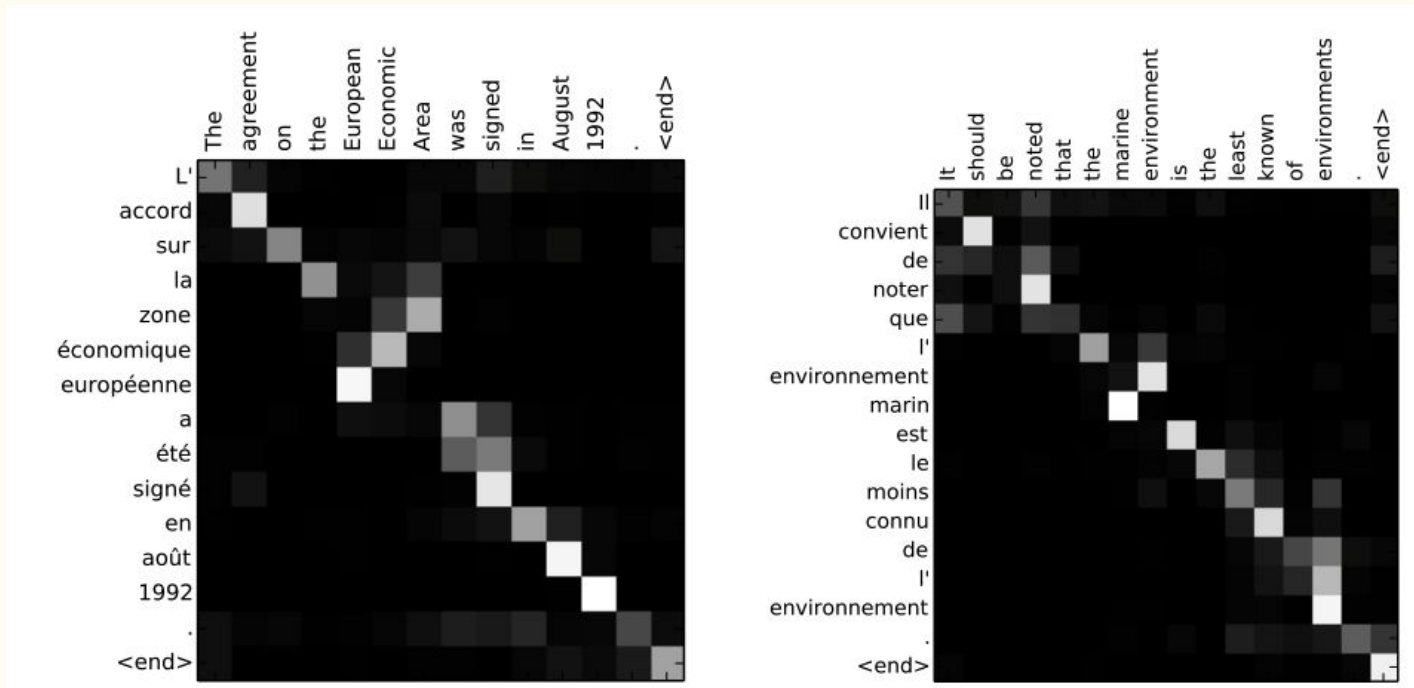
- **Dot-product Attention:** $score_i = s^T h_i \in \mathbb{R}$
- **Multiplicative Attention:** $score_i = s^T W h_i \in \mathbb{R}$
- **Additive Attention:** $score_i = v^T \tanh(W_1 h_i + W_2 s) \in \mathbb{R}$

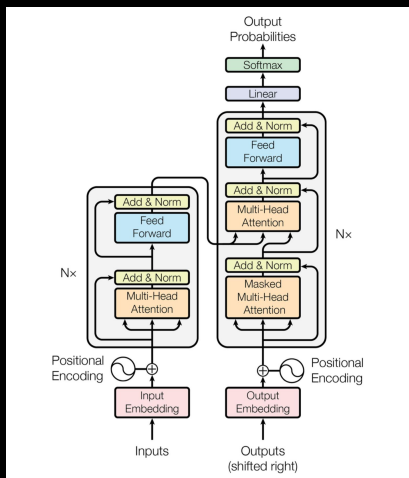
Attention Score Functions

Several ways to compute $score \in \mathbb{R}^N$ from $h_1, \dots, h_N \in \mathbb{R}^{d_1}$ and $s \in \mathbb{R}^{d_2}$:

- **Dot-product Attention:** $score_i = s^T h_i \in \mathbb{R}$
 - “Similarity”
- **Multiplicative Attention:** $score_i = s^T W h_i \in \mathbb{R}$
 - Learn how much importance to give to different parts when calculating “similarity”
- **Additive Attention:** $score_i = v^T \tanh(W_1 h_i + W_2 s) \in \mathbb{R}$
 - Simple single layer neural network to learn the scoring function

Attention provides some interpretability





Innovation

The Transformer

(an overview)

Motivation for Transformer

- Standard Neural Networks require fixed length input (so we need to pad sentences of different length)
 - Also input embeddings are not contextual
- RNNs allow us to get around this: We input one word at a time
 - Embeddings are contextual

Motivation for Transformer

It was found that more layers and more pre-training helped language models

- RNNs need $O(n)$ steps to process a sentence
 - As a consequence, we cannot pre-train on Very Large corpora
- The computational complexity per layer in RNNs is higher (when the input length is $<$ the representation dimensionality)
 - As a consequence, we cannot have too many layers.

Motivation for Transformer

What we need is an architecture that

- provides contextual embeddings
- captures semantic and syntactic information like RNNs do
- can process a sentence in parallel
- has relatively low computational complexity per layer

→ That's what the **Transformer** provides

“Attention is All You Need”

	Seq2Seq without attention	Seq2Seq with attention	Transformer
processing within encoder	RNN	RNN	attention
processing within decoder	RNN	RNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

Self-Attention

The self-attention for each word consists of a:

“Query” → Think of it as the question that tries to establish
this word’s context.

“Key” → (Query * Key) gives the answer to how good
this word is at providing context to the querying
word

“value” → The value of each word

Self-Attention

The self-attention for each word consists of a:

S_i (hidden vector for i th output word)

“Query” → Think of it as the question that tries to establish this word’s context.

h_j (hidden vector for j th input word)

“Key” → ($\text{Query} * \text{Key}$) gives the answer to how good *this* word is at providing context to the querying word (notice, this can be a different word)

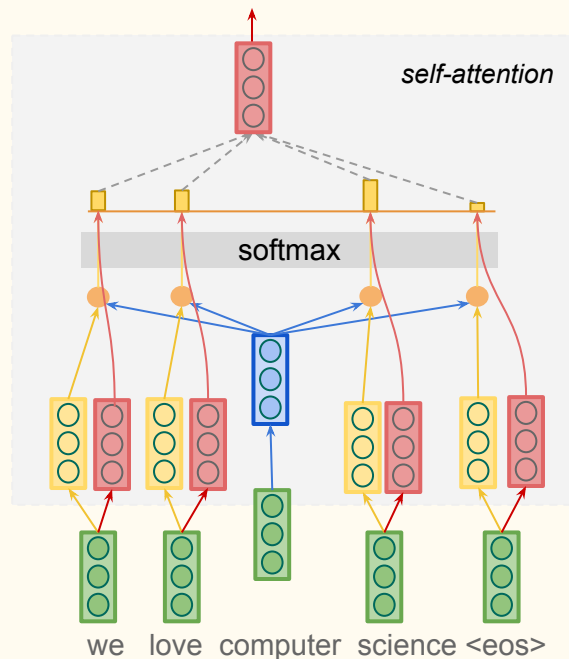
“value” → The value of each word

h_j (hidden vector for j th input word)

Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

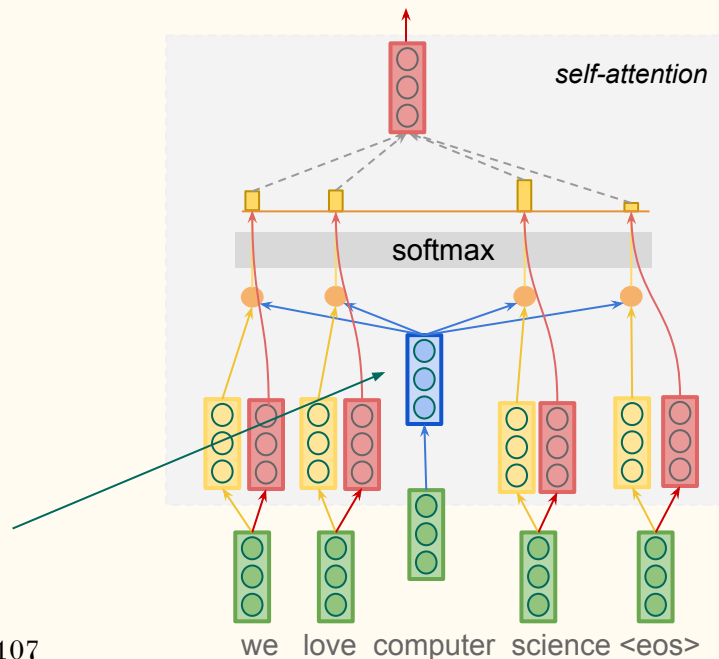


Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

The **Query** aims to establish the context, in this example, of the word “computer” (**from**)



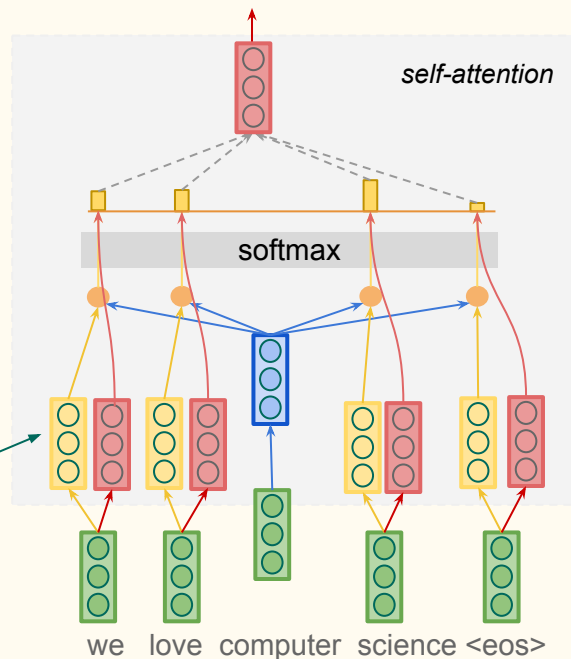
Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

The **Query** aims to establish the context, in this example, of the word “computer” (**from**)

Done by querying each **Key** of all words (**to**)

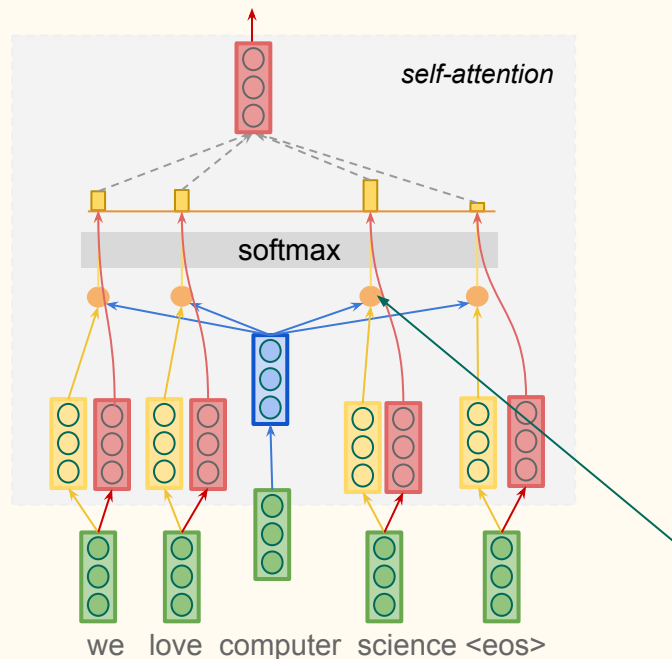


Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

In our example, this might establish “science” as the word the provides the most informative context.

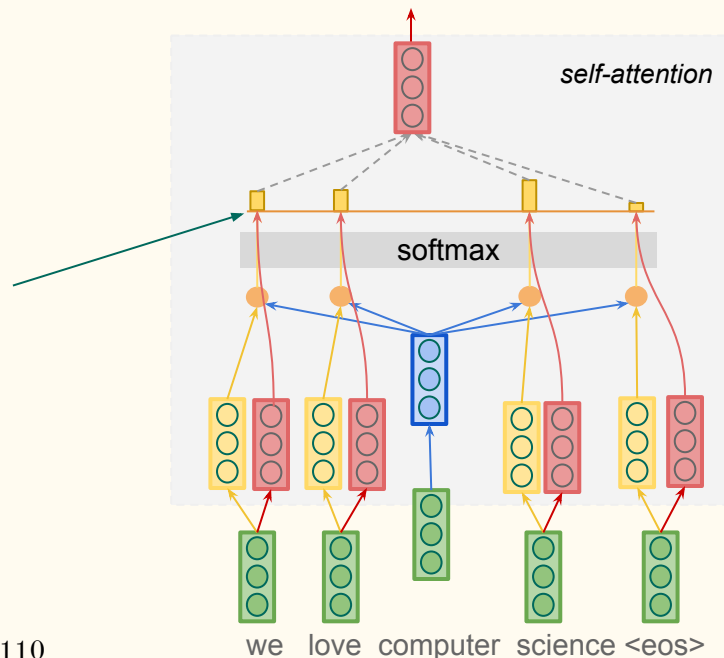


Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

We push the output of this through a softmax and multiply each with the **values** associated with each word (target word, i.e “at”)



Self-Attention: Query, Key, Value

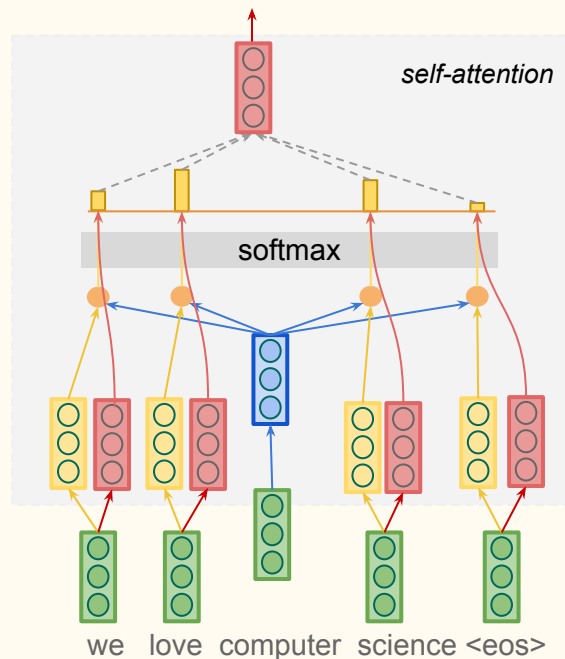
Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

$$\text{Attention}(q, k, v) = \text{softmax} \left(\frac{qk^T}{\sqrt{d_k}} \right) v$$

Vector dimensionality of K, V

Attention Weights



Self-Attention: Query, Key, Value

Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

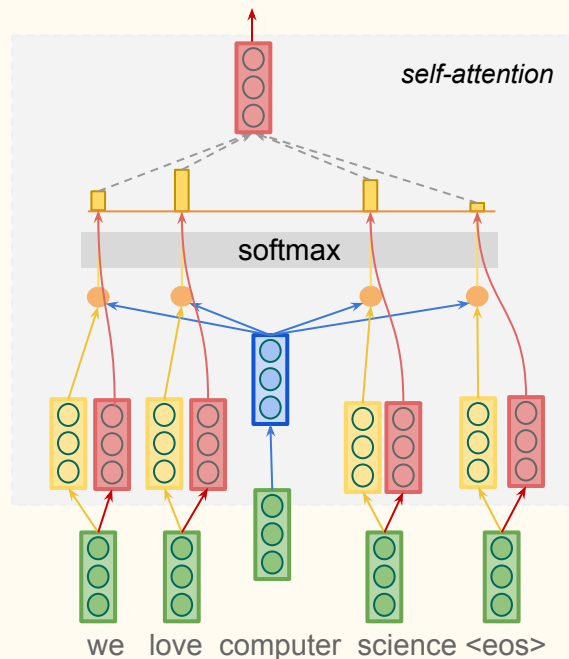
Attention Weights

$$\text{Attention}(q, k, v) = \text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right)v$$

from to

Used to make sure the value of qk^T does not become too large

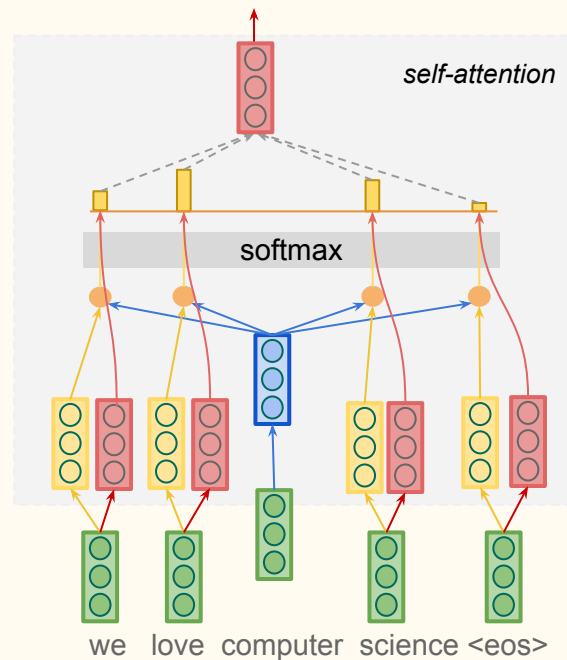
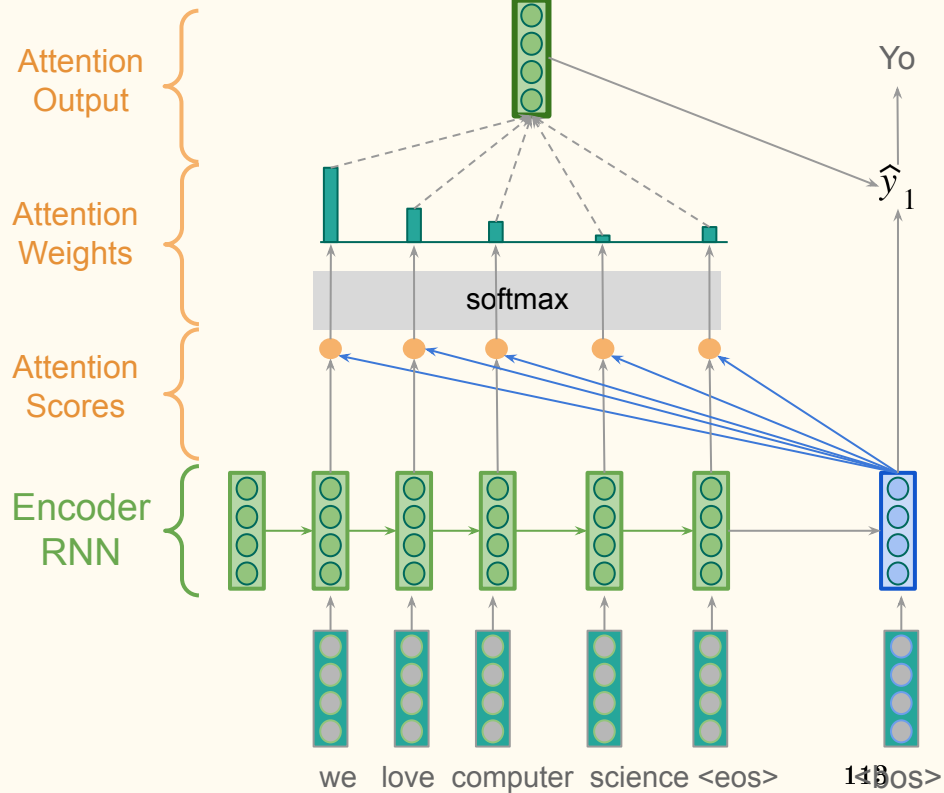
112



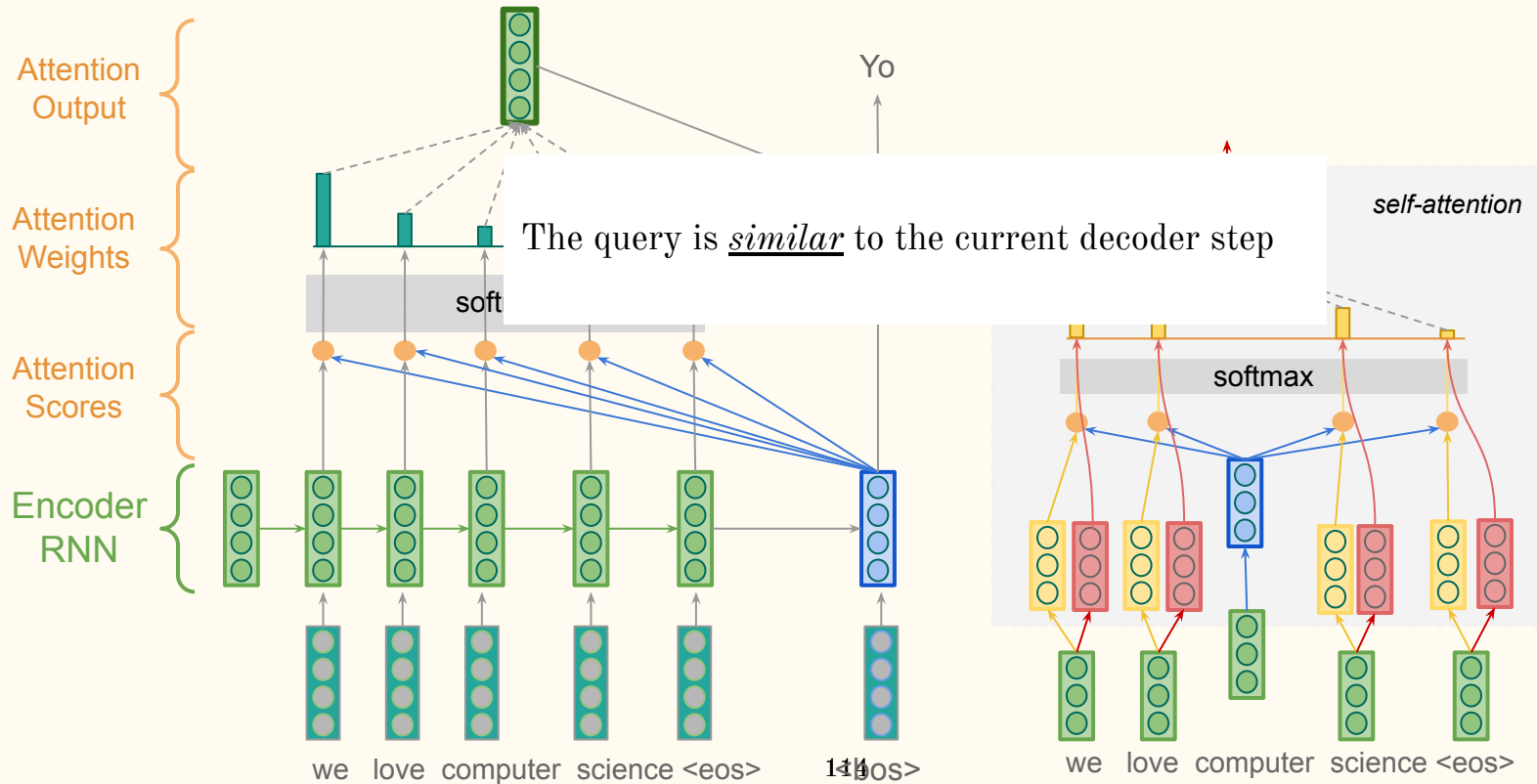
Attention

vs

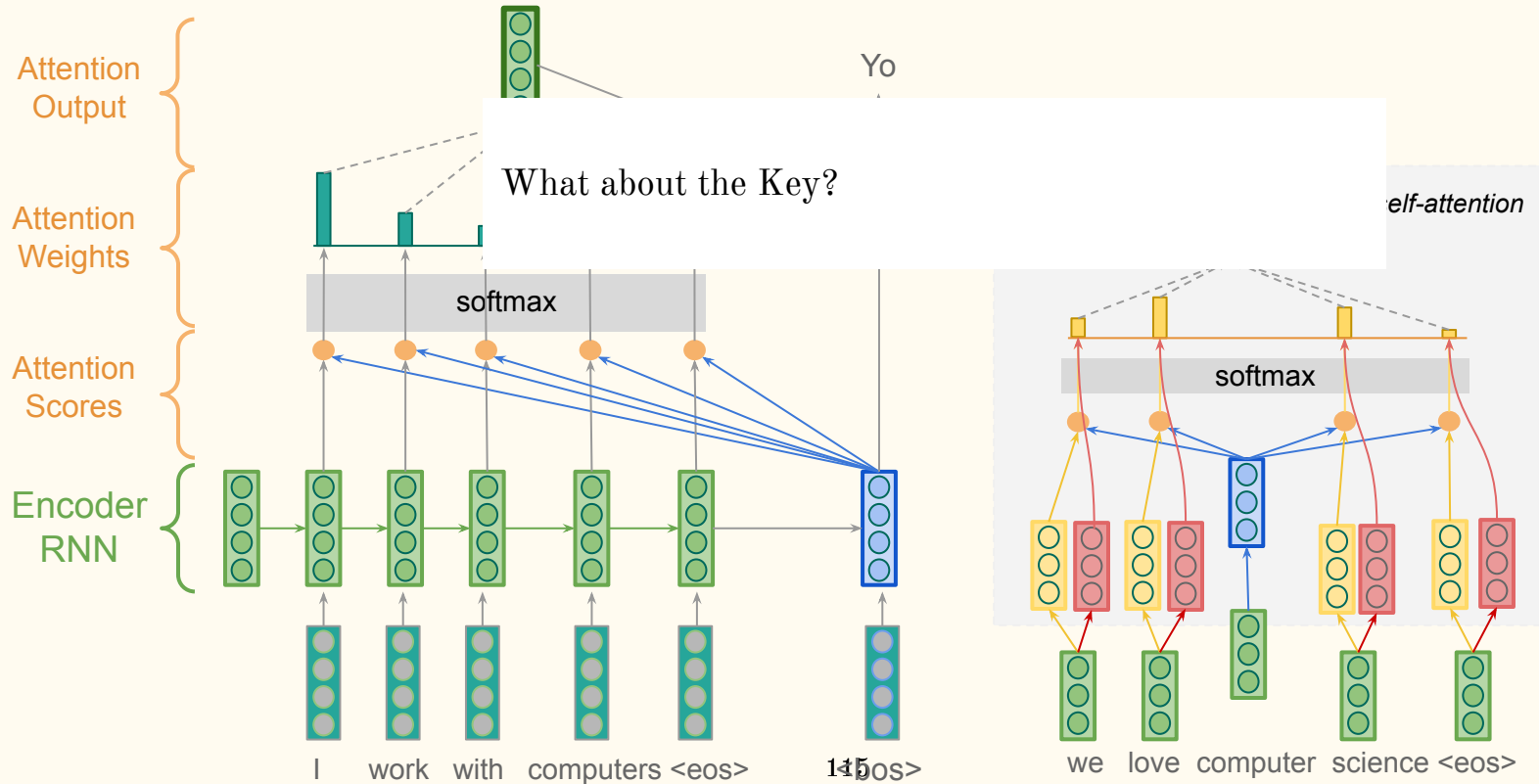
Self-Attention



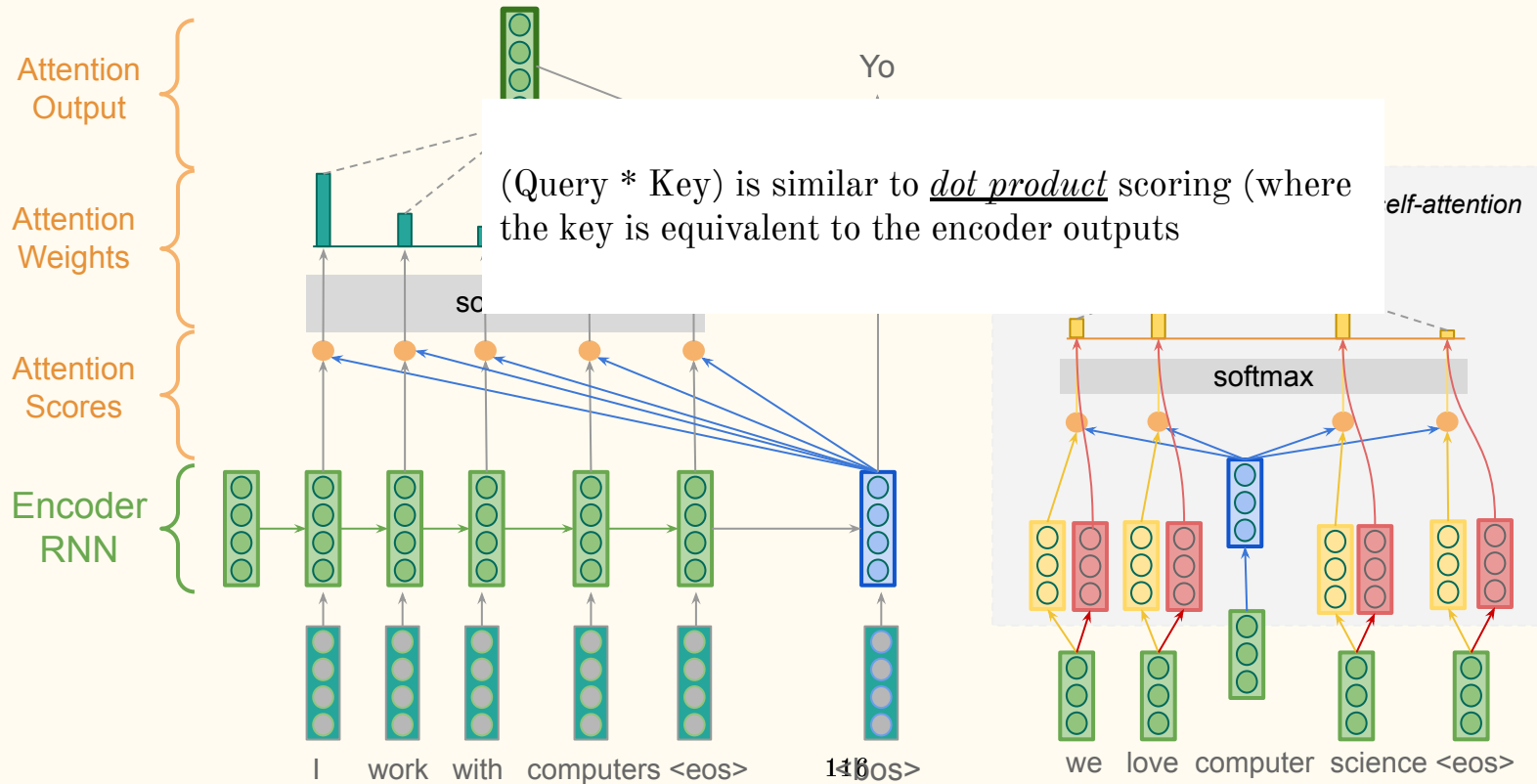
Attention vs Self-Attention



Attention vs Self-Attention



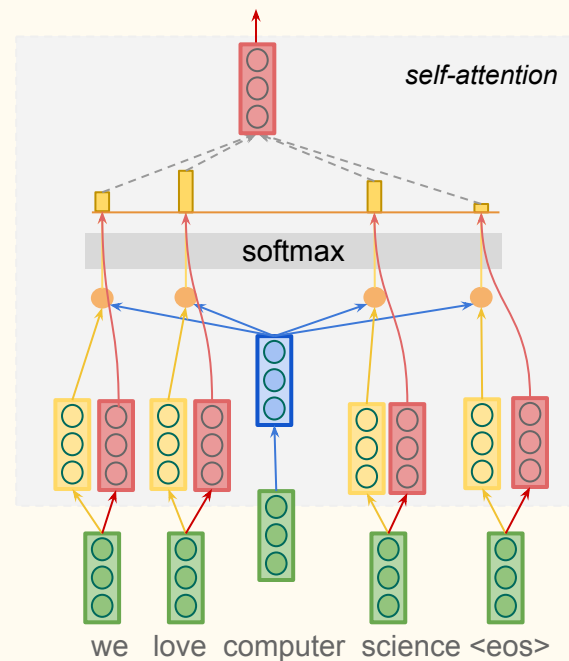
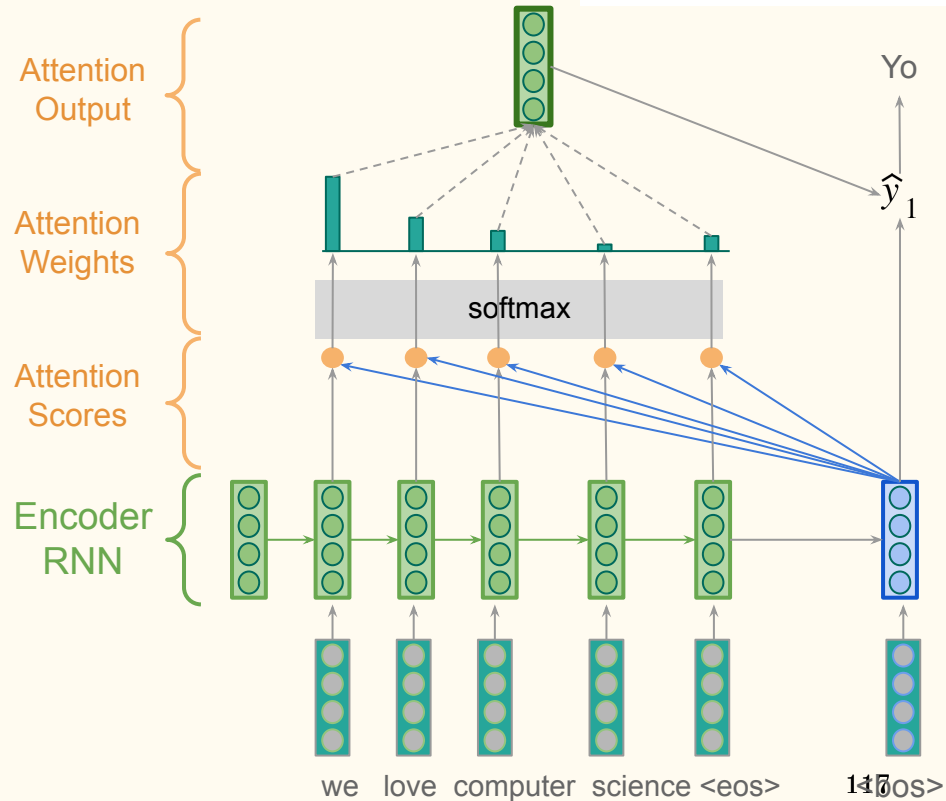
Attention vs Self-Attention



And the Values?

Attention

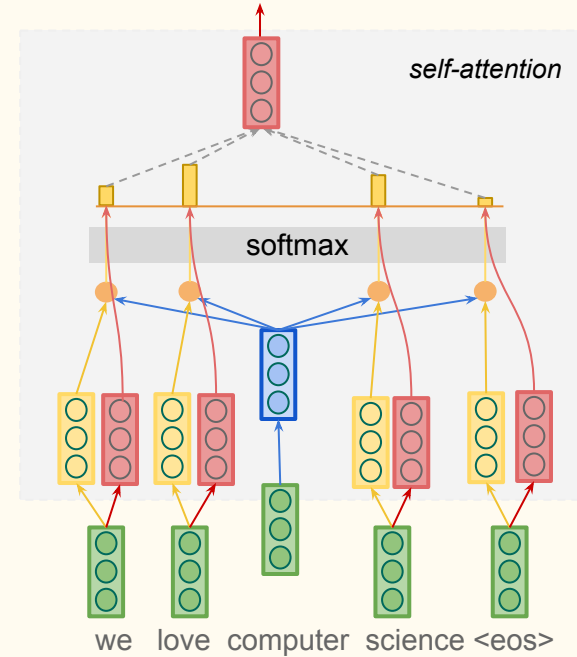
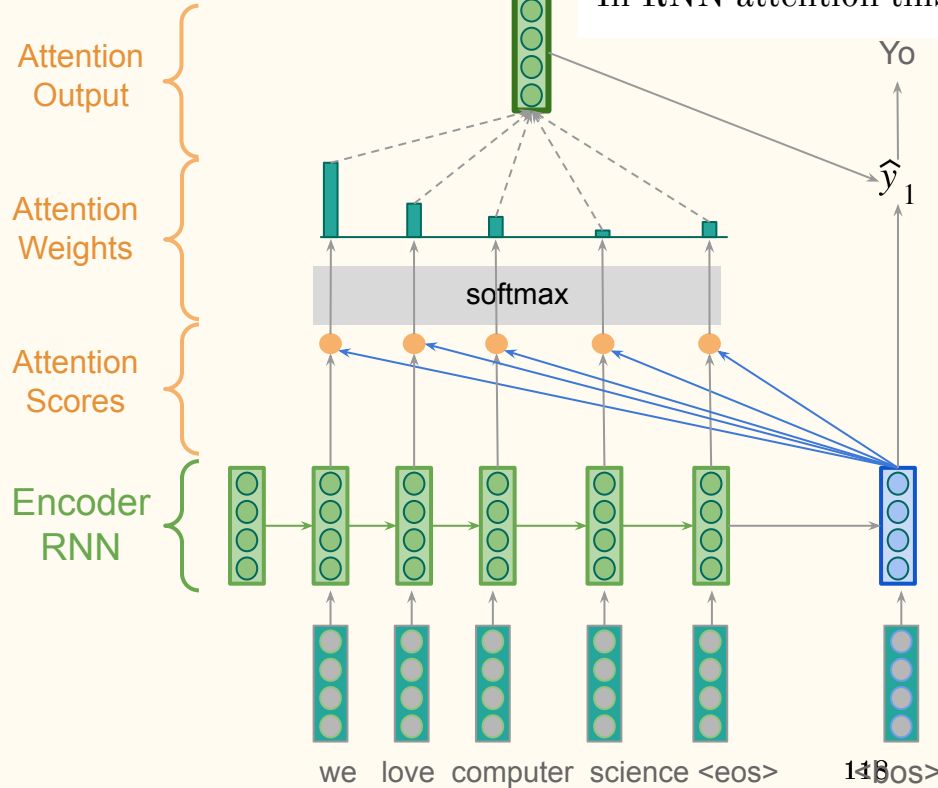
n



Attention

After pushing both through a softmax we multiply the result with the **value** in self-attention before summing it

In RNN attention this is again the encoder states



Attention

vs

Self-Attention

$$\alpha_k^{(t)} = \frac{\exp(\text{score}(s_t, h_k))}{\sum_{i=1}^N \exp(\text{score}(s_t, h_i))}, \quad k = 1..N$$

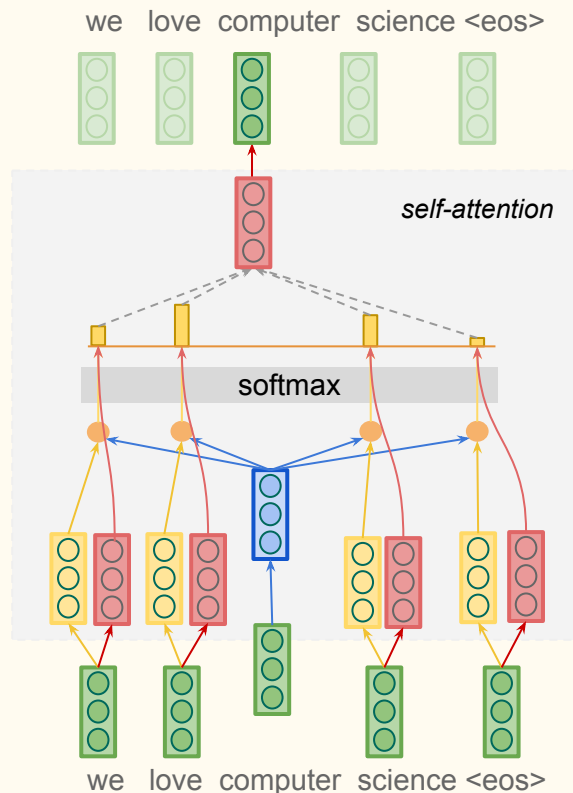
$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k^i)}{\sum_j \exp(q \cdot k^j)} \cdot v^i$$

Self-Attention: Query, Key, Value

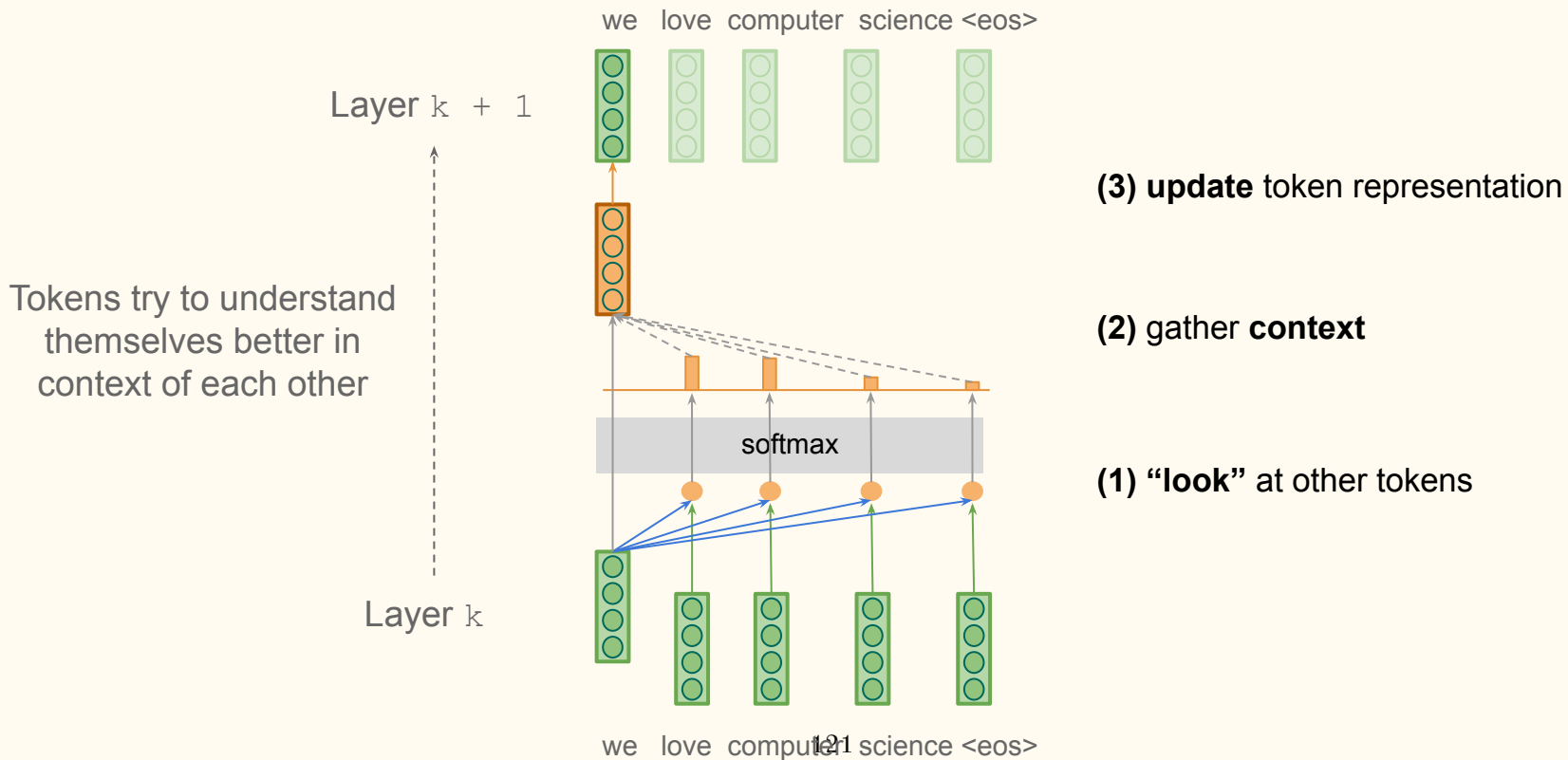
Each vector receives a:

- **Query (q):** vector **from** which the attention is looking
- **Key (k):** vector **at** which the query looks to establish context
- **Value (v):** value of word being looked **at**, weighted based on context

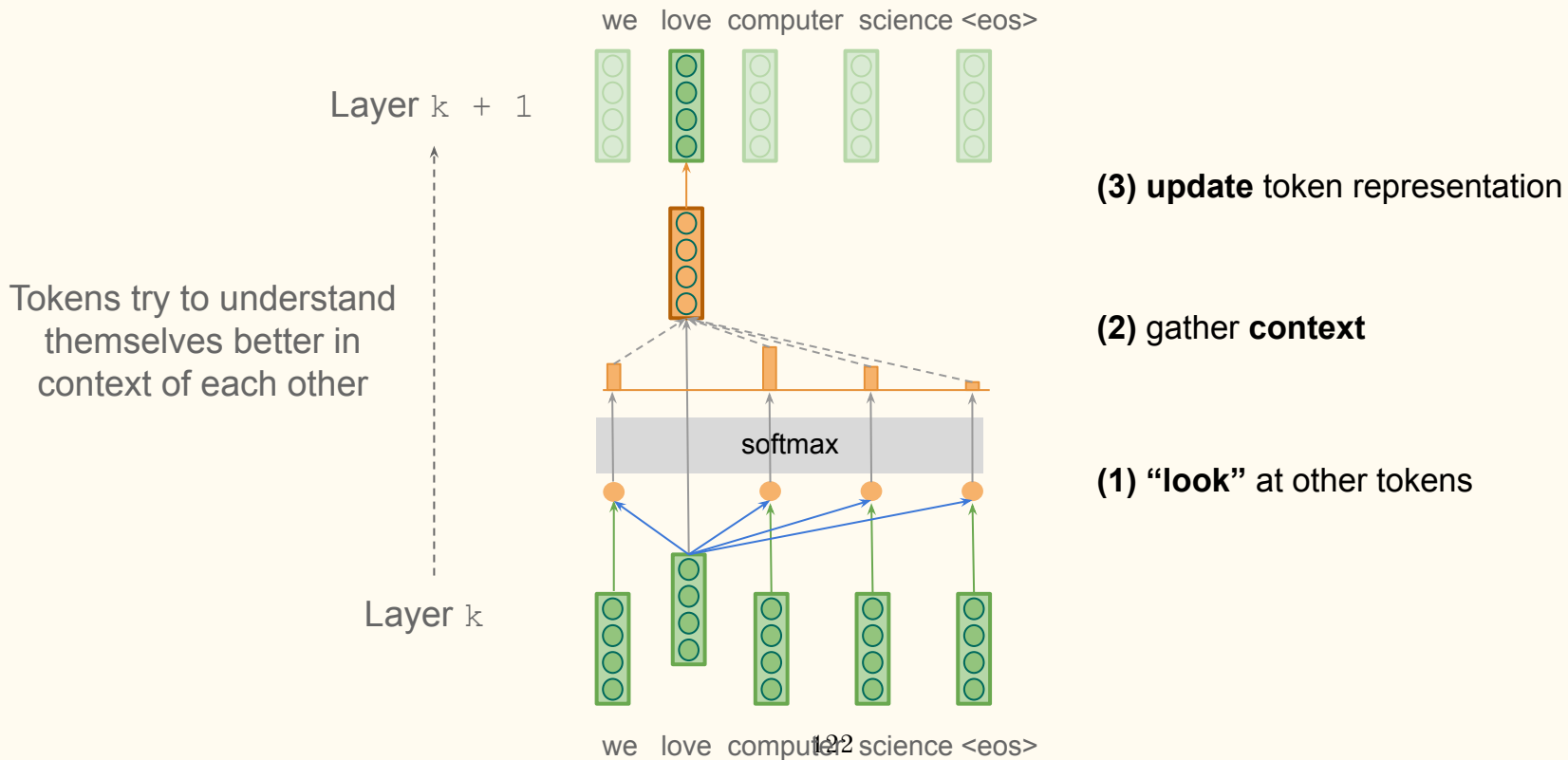
This is then fed to the next layer



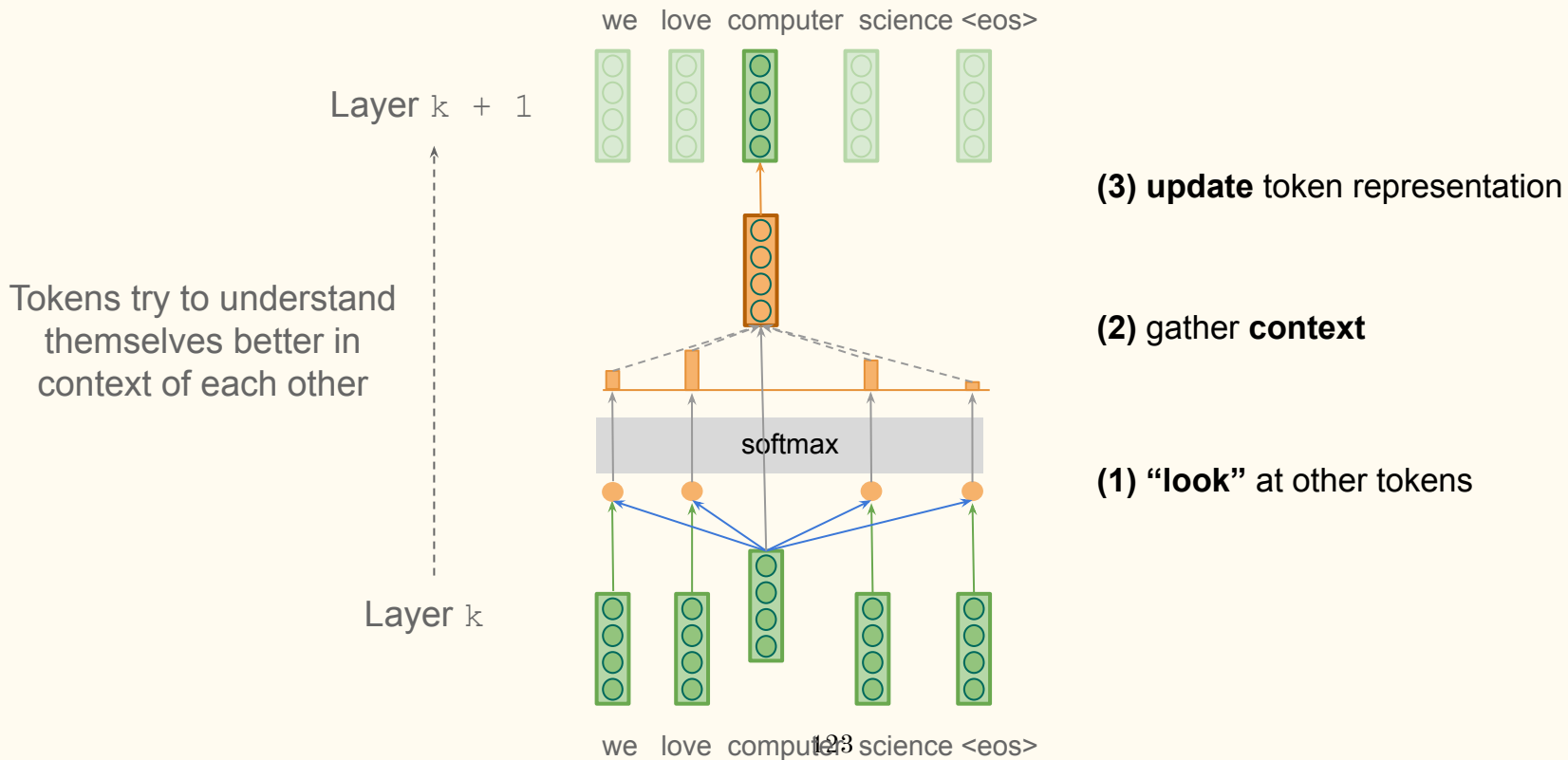
Self-Attention



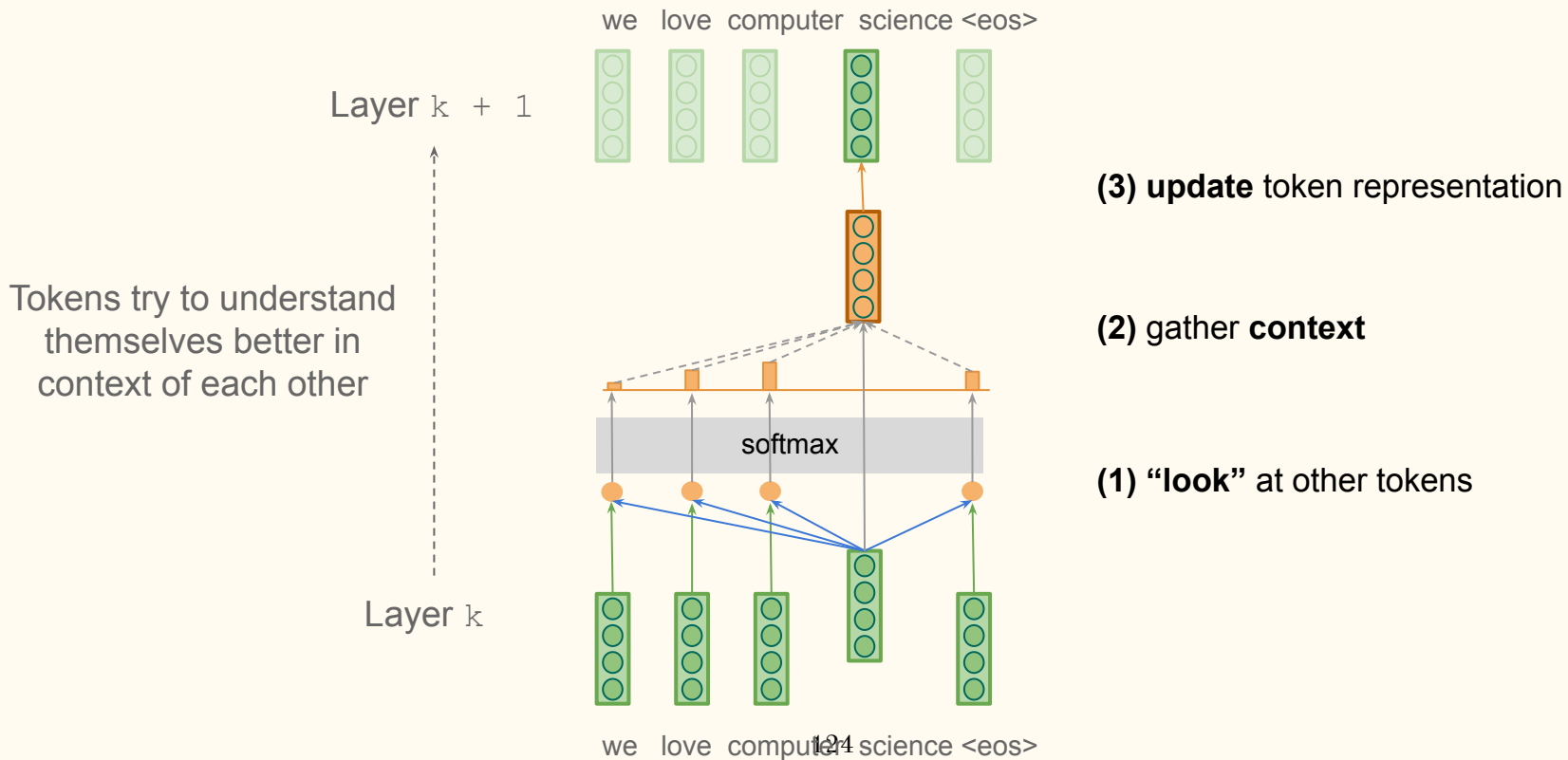
Self-Attention



Self-Attention

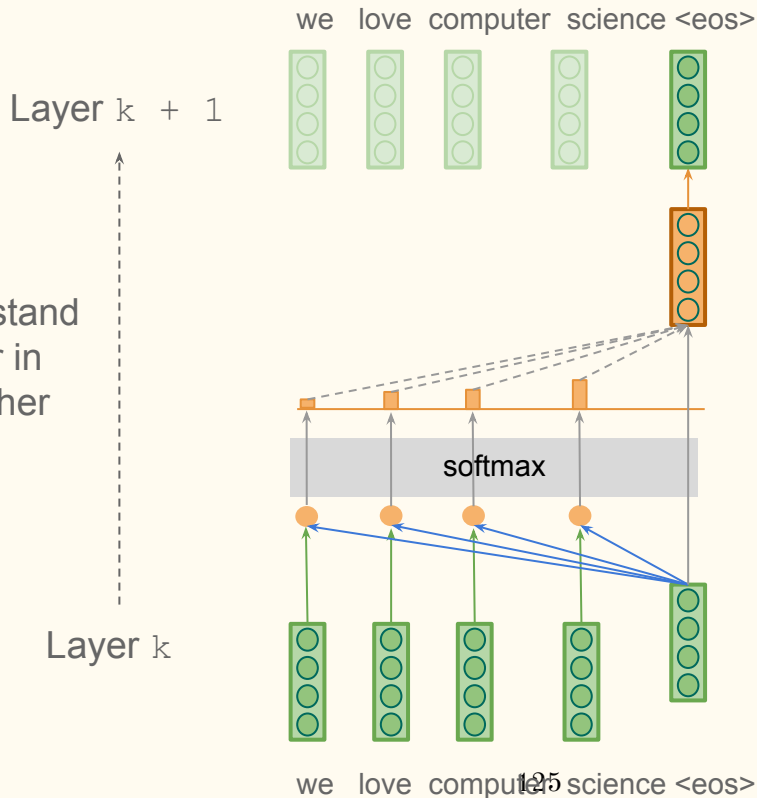


Self-Attention



Self-Attention

Tokens try to understand themselves better in context of each other



In practice, this happens **in parallel**

(3) **update** token representation

(2) gather **context**

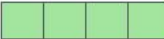
(1) **"look"** at other tokens

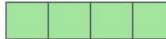
Input

Thinking

Machines

Embedding

x_1 

x_2 

Queries

q_1 

q_2 

Keys

k_1 

k_2 

Values

v_1 

v_2 

Score

$q_1 \cdot k_1 = 112$

$q_2 \cdot k_2 = 96$

Divide by 8 ($\sqrt{d_k}$)

14

12

Softmax

0.88

0.12

Softmax

X

Value

v_1 

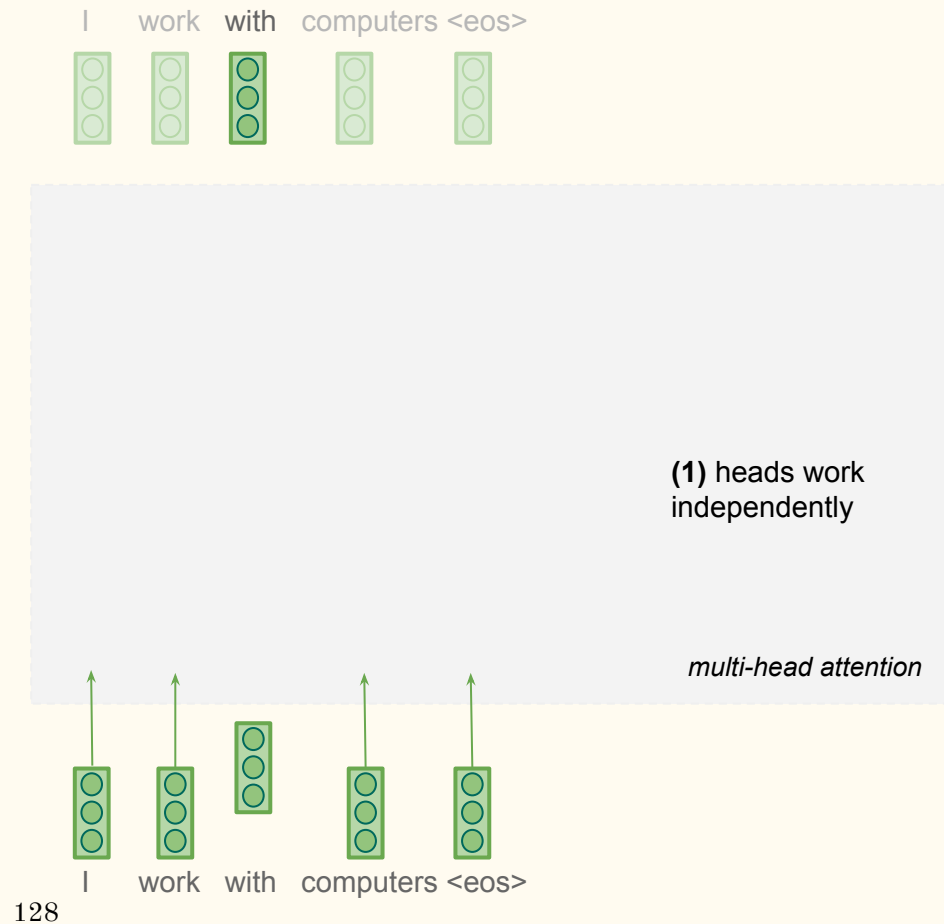
v_2 

Sum

z_1 

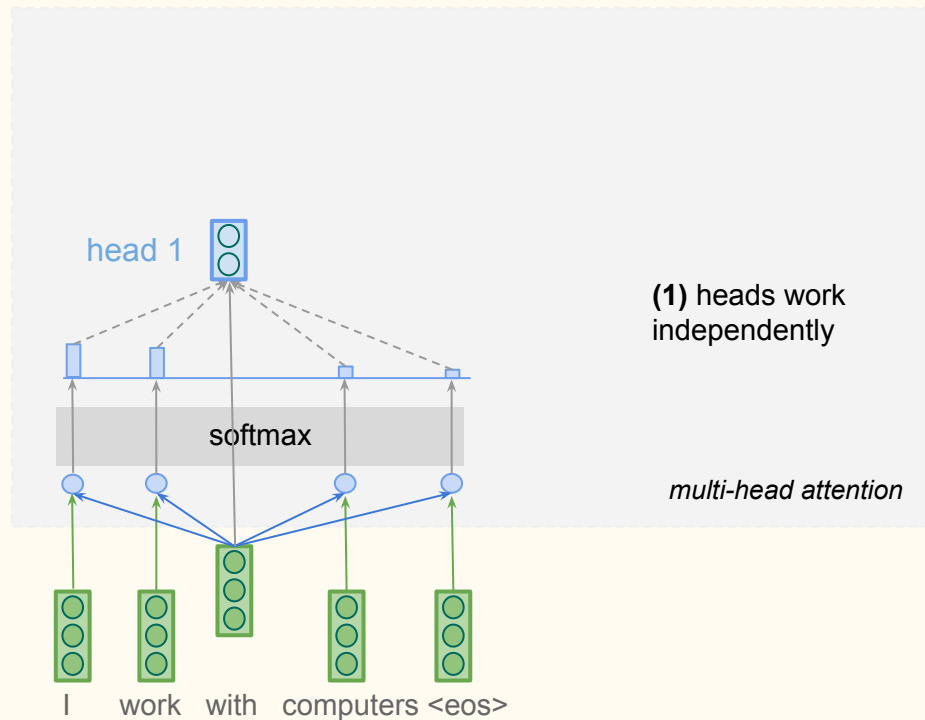
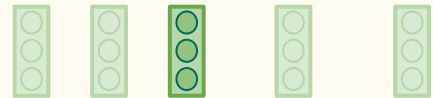
z_2 

Multi-Head Attention



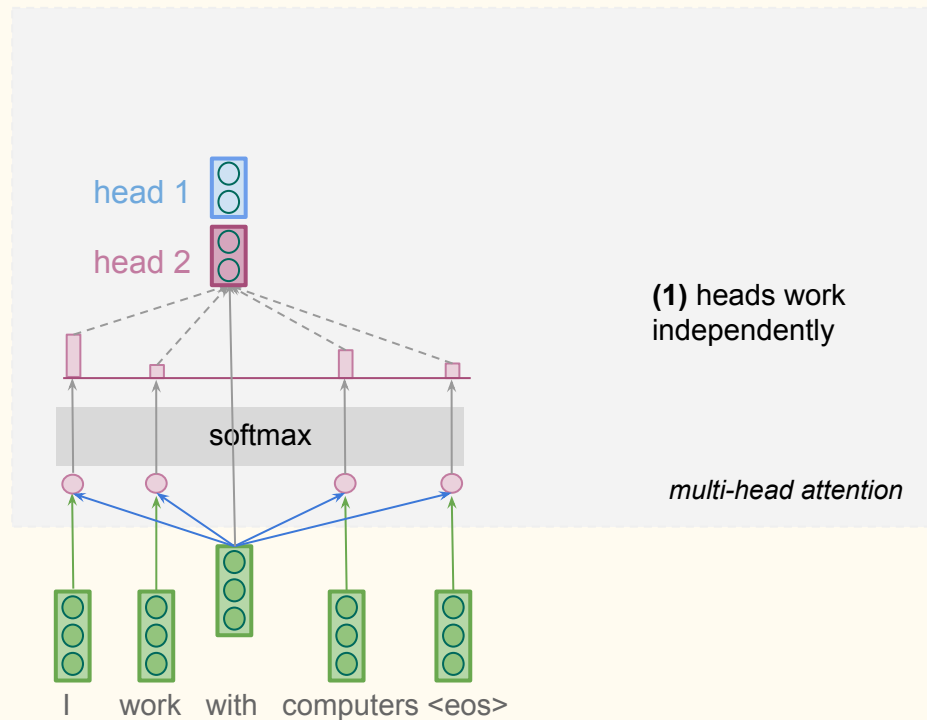

Multi-Head Attention

I work with computers <eos>

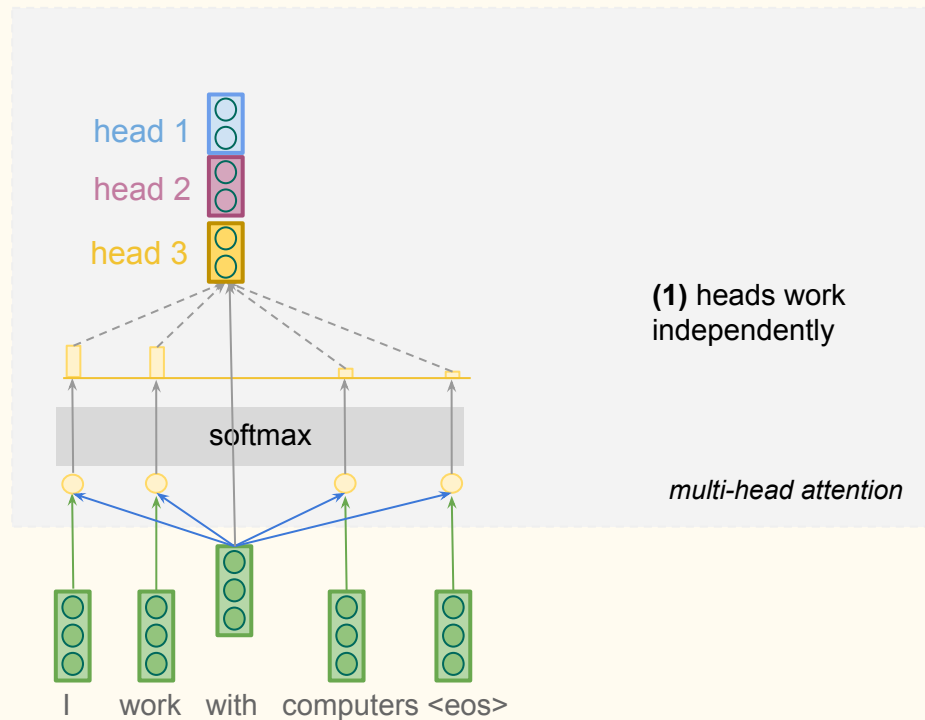
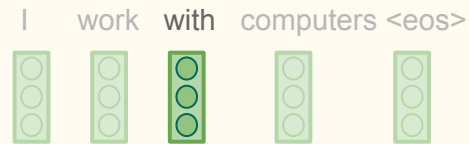


Multi-Head Attention

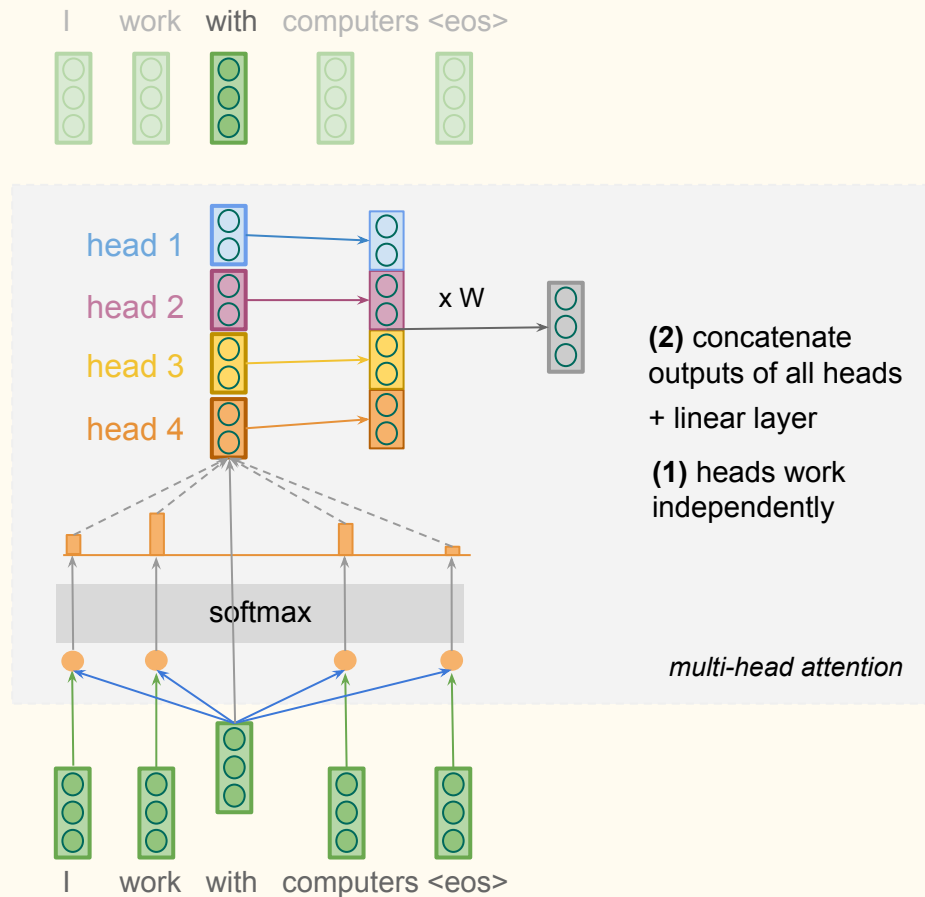
I work with computers <eos>



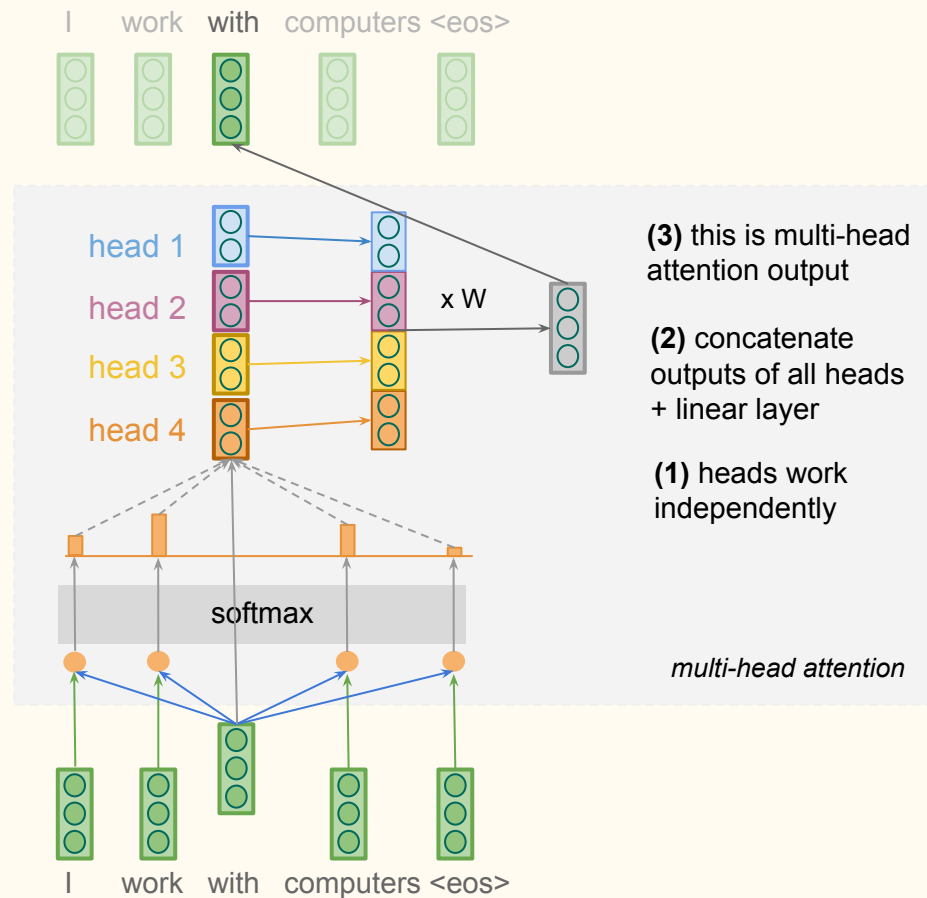
Multi-Head Attention



Multi-Head Attention



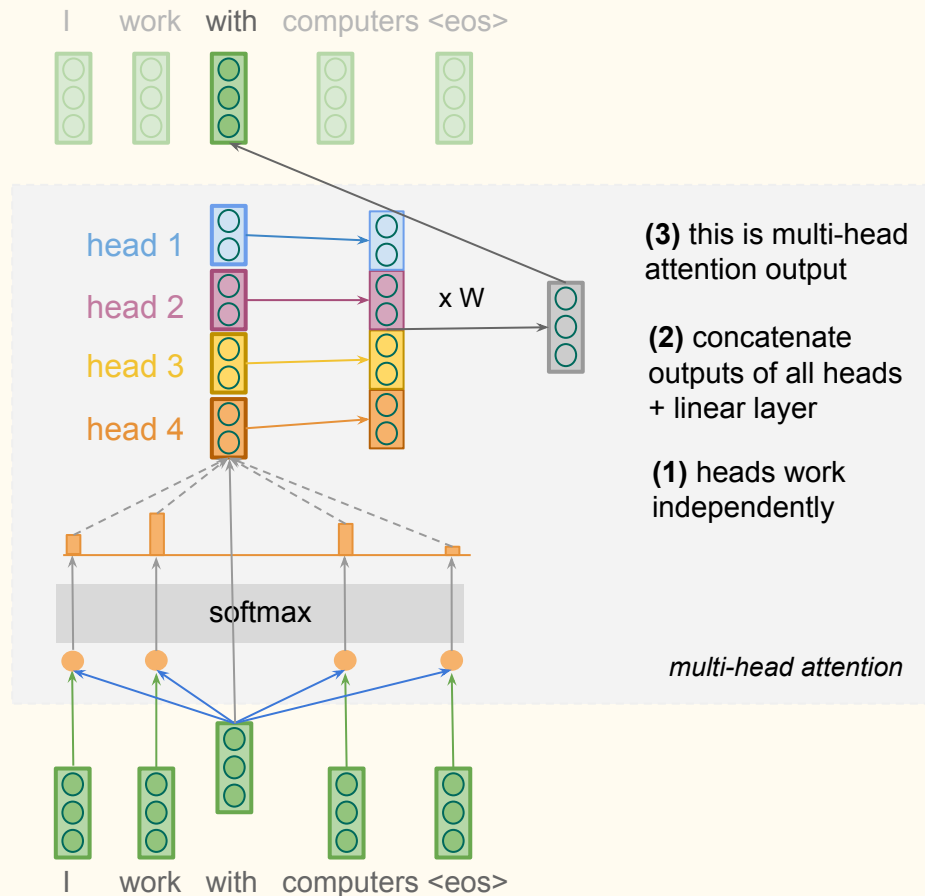
Multi-Head Attention



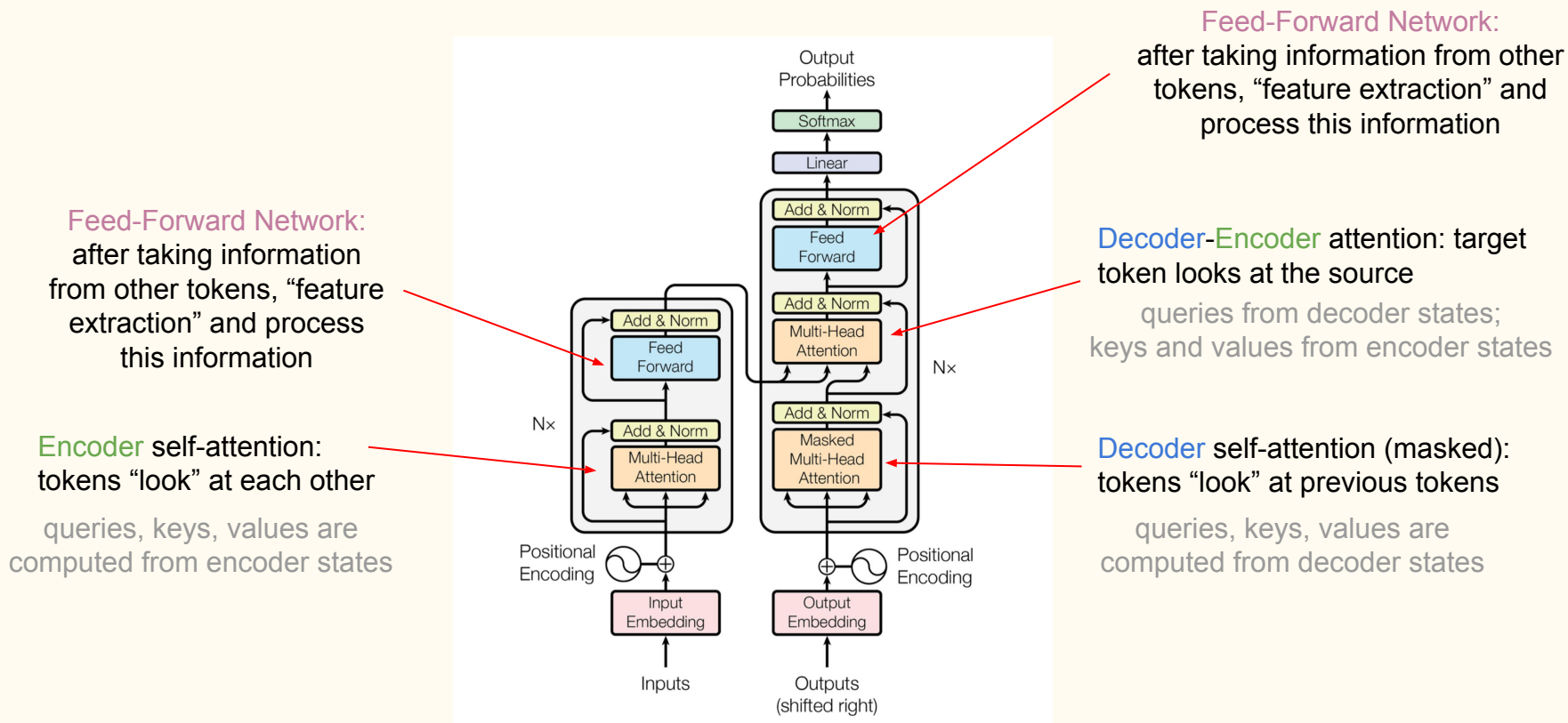
Multi-Head Attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$



Transformer (Vaswani et al., 2017)



Transformer (Vaswani et al., 2017)

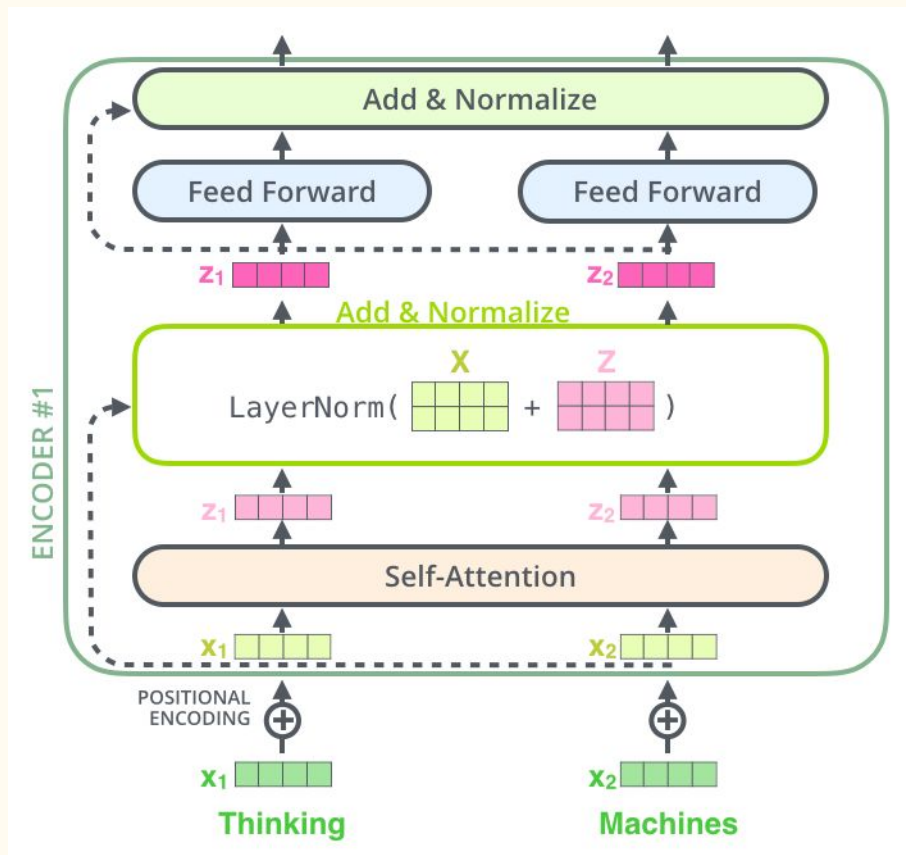
- Positional Encoding
- Masked Self-Attention during training.
- Byte-Pair Encoding for subwords

For details on the other components, read the paper

In this Section...

- Attention in Seq2Seq models with RNNs
 - Allows the model to focus on different parts of the source at different time steps during decoding
- The Transformer
 - “Attention is All You Need”
 - Self-Attention
 - For both Encoder and Decoder, “look around” to get better representations
 - Multi-Head Attention
 - Do not compute attention once, but several times and “combine” the representations (each can focus on different relevant aspects of the sentence)

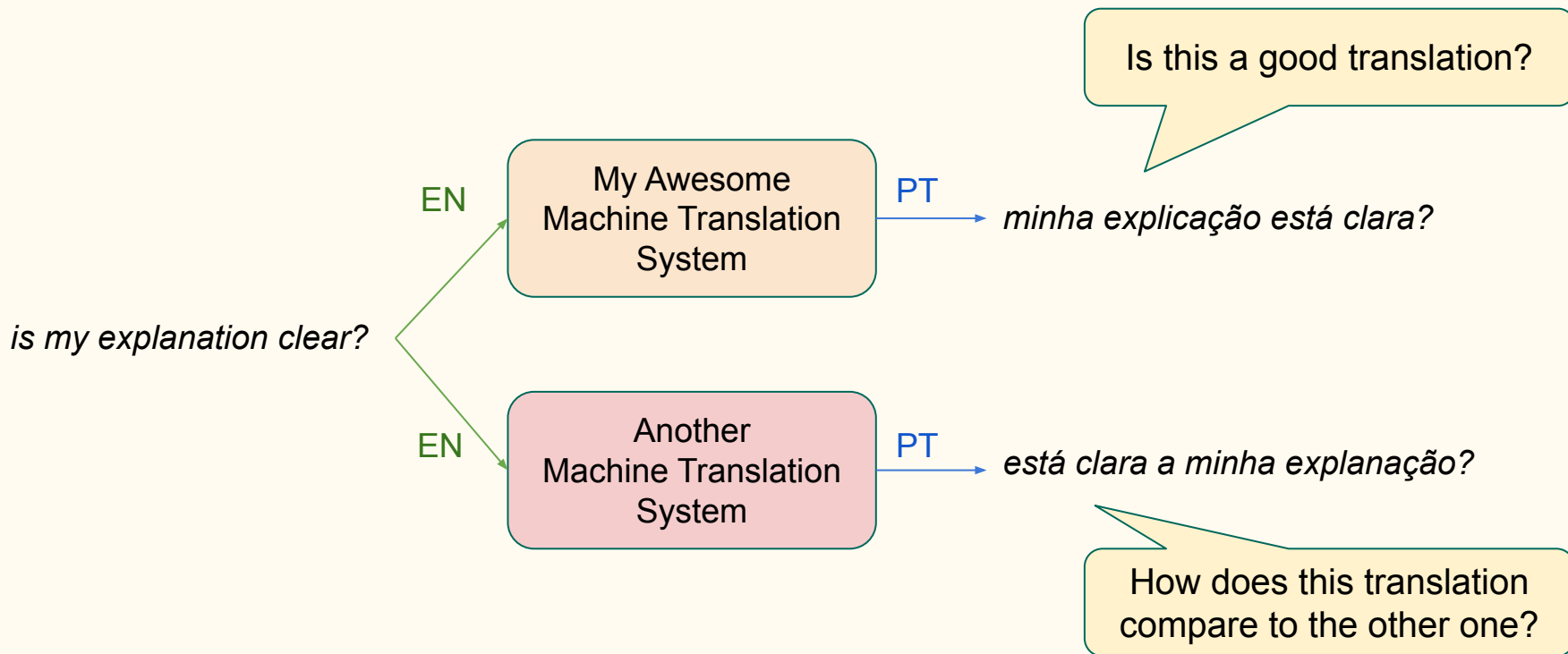
The Residuals



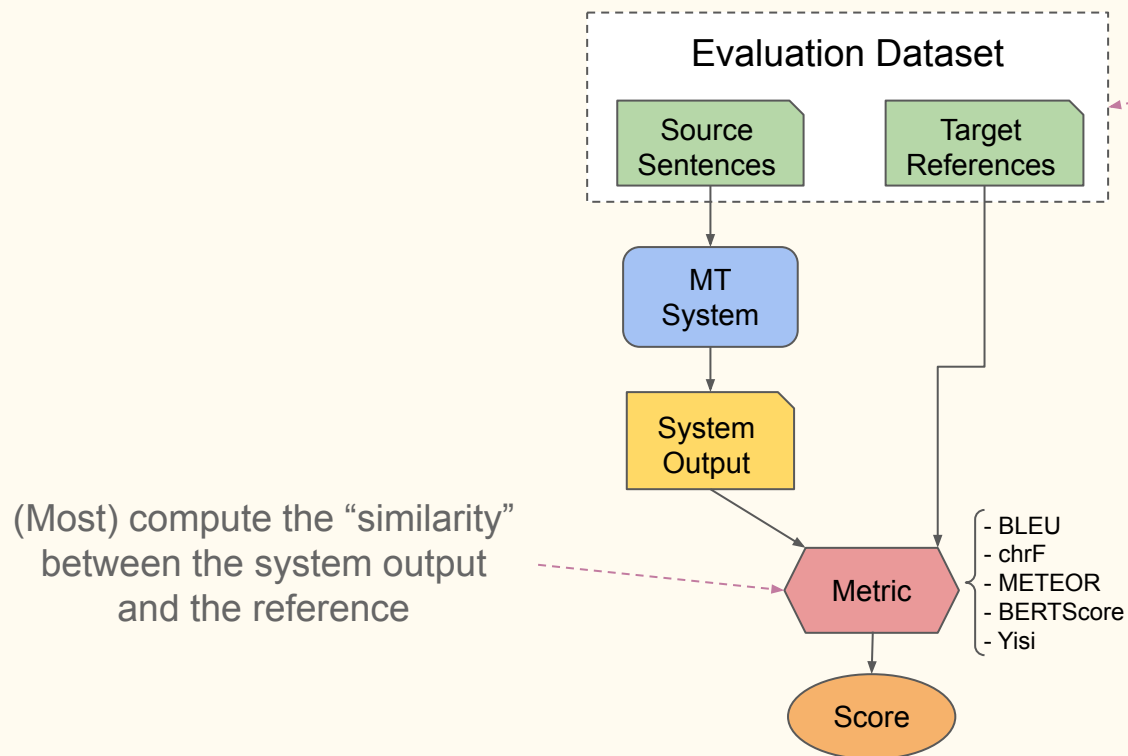
Evaluation

Human Judgements and
Automatic Metrics

How good is an MT output?



Automatic Evaluation



For example, from WMT
<http://www.statmt.org/wmt20/>

Automatic Metrics

Can be used in other Seq2Seq tasks!

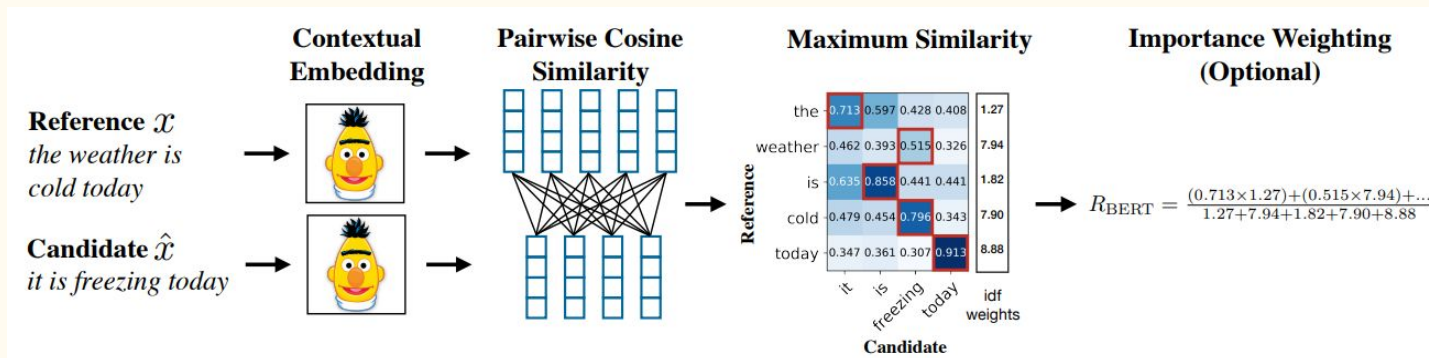
- BLEU (Papineni et al., 2002)

$$p_n = \frac{\sum_{S \in C} \sum_{ngram \in S} Count_{matched}(ngram)}{\sum_{S \in C} \sum_{ngram \in S} Count(ngram)}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1 - \frac{r}{c}} & \text{if } c \leq r \end{cases}$$

$$BLEU = BP \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$

- BERTScore (Zhang et al., 2020)



Human Judgements

- At **development time**, **automatic metrics** are useful because they are **fast to compute**, but they can be imprecise
- At **test time**, **manual evaluation** should also be performed
 - What to evaluate?
 - Fluency
 - Adequacy (a.k.a. meaning preservation)
 - How to evaluate?
 - Likert Scale (discrete score)
 - Direct Assessment (continuous score)
 - Relative Ranking

It's expensive!
But **necessary** to truly
evidence that the model is
translating correctly

Final Remarks

Summary

- **Task:** Neural Machine Translation (NMT)
 - Given a source input sentence generate an output target translation
 - The machine translation model is a single neural network
- **Model:** Encoder-Decoder with RNNs
 - The encoder reads the source (token by token) and creates a representation of the whole sentence that is used by the decoder to generate the target translation
 - The encoder and the decoder are implemented using RNNs
- **Improvement:** Attention
 - Allows the decoder to focus on different parts of the source at different timesteps

Summary

- **Improvement: Transformer**
 - For implementing the encoder, the decoder and their interaction only relies on attention
 - Self-Attention allows to pay attention to all tokens in the same group at the same time
 - Computations can be parallelized, making it more efficient than RNN-based models
- **Evaluation: Automatic Metrics and Manual Judgements**
 - Serve to measure the quality of automatic translations
 - Automatic scores are useful at development time
 - Manual Judgements should be preferred at test time to validate the automatic scores

All discussed in this lecture can be applied to other seq2seq tasks: summarisation, simplification, dialogue generation, etc.

Acknowledgements

- Ref from: [Lecture 9, Lecture 10: Neural Machine Translation and Models with Attention](#)
- Slides adapted from [Fernando Alva Manchego](#)
- Who, in turn, adapted slides from:
 - NLP Course | For You (by Lena Voita)
 - [Sequence to Sequence \(seq2seq\) and Attention](#)
 - Natural Language Processing with Deep Learning (Stanford CS 224N)
 - [Lecture 7: Machine Translation, Sequence-to-Sequence and Attention](#)

Recommended Reading

- Blogs and Web Pages
 - [NLP Course | For You](#) (by Lena Voita)
 - [Attention and Augmented Recurrent Neural Networks](#)
 - [The Illustrated Transformer](#)
- Published Papers
 - [Neural Machine Translation by Jointly Learning to Align and Translate](#)
 - [Attention Is All You Need](#)

Next Lecture

- **Transfer Learning**

- Train a supervised model in one (or many) task(s) and re-use it in another
- BERT → large Transformer-based language model

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. [Neural machine translation by jointly learning to align and translate](#). CoRR, abs/1409.0473.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: A method for automatic evaluation of machine translation](#). In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318, Philadelphia, Pennsylvania. ACL.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#). In International Conference on Learning Representations.