Natural Language Processing with Deep Learning

IFT6289, Winter 2022

Lecture 6: Machine Translation, Sequence to Sequence, and Attention

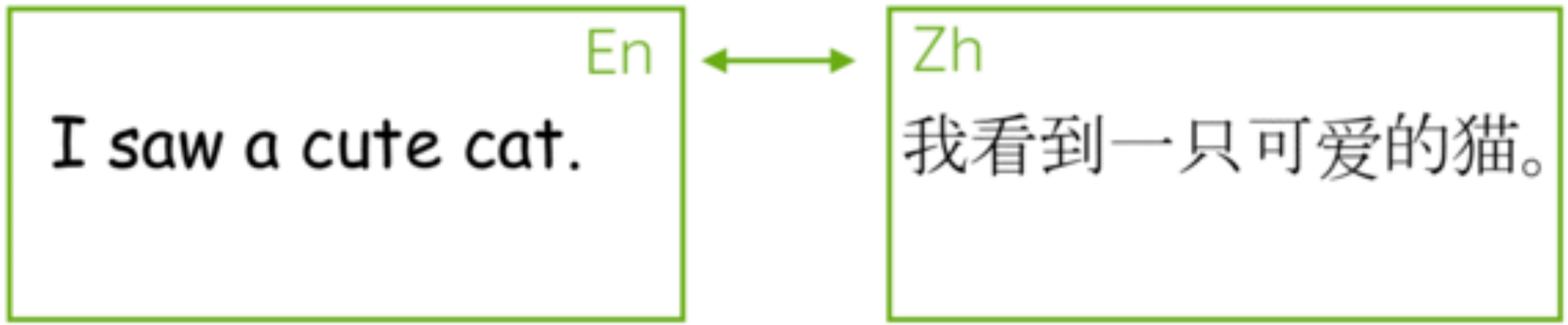
Bang Liu

Lecture outline

1. Machine Translation: History and Evaluation
2. Sequence to Sequence
3. Attention Mechanism
4. Bridging the Gap between Training and Inference

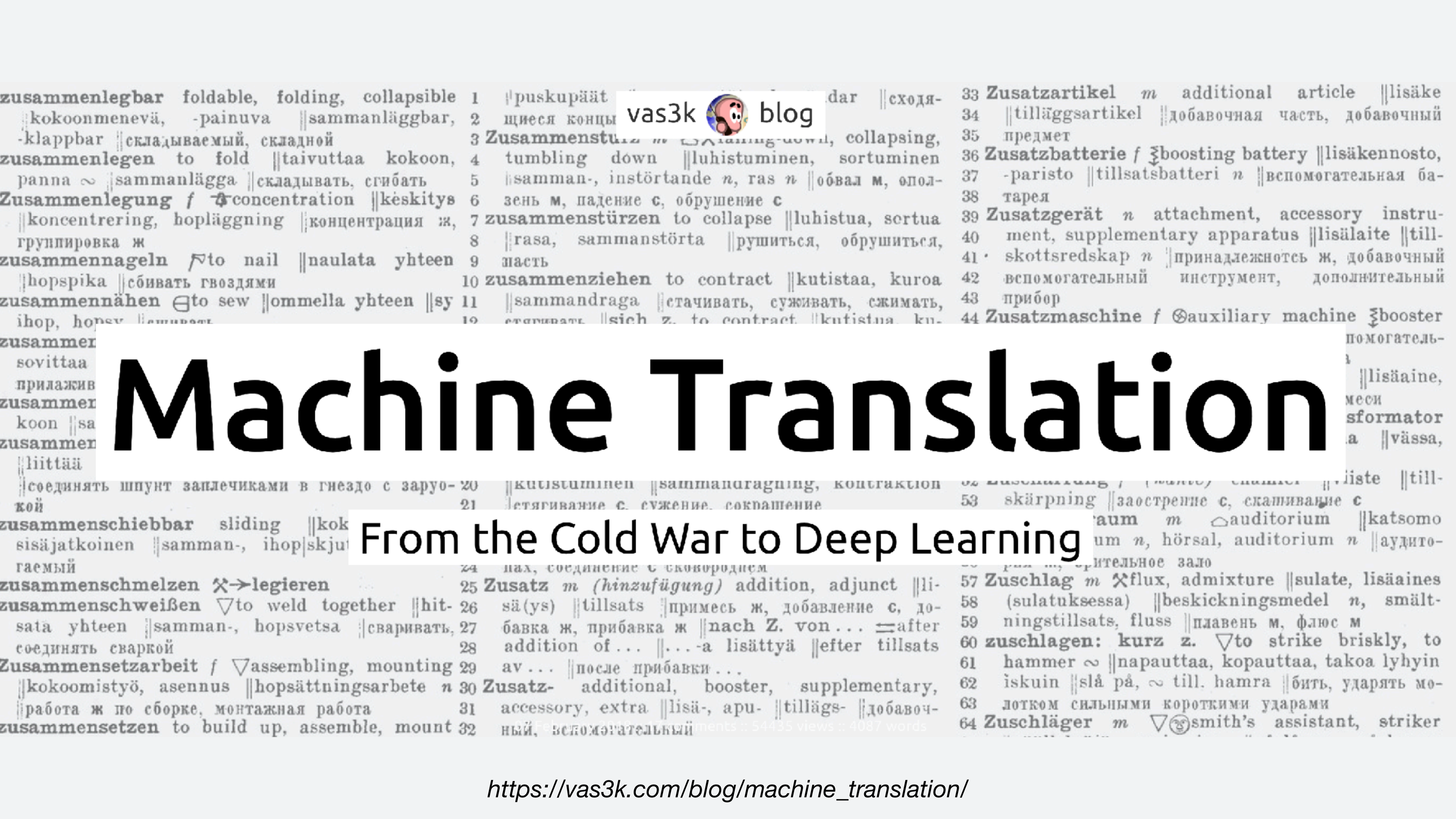
Machine Translation

What is Machine Translation



Commercial Machine Translation Systems

- Google Translate
- Yandex Translate
- Bing Microsoft Translator
- Baidu Fanyi
- DeepL Translator
- Wechat Translate
- ……

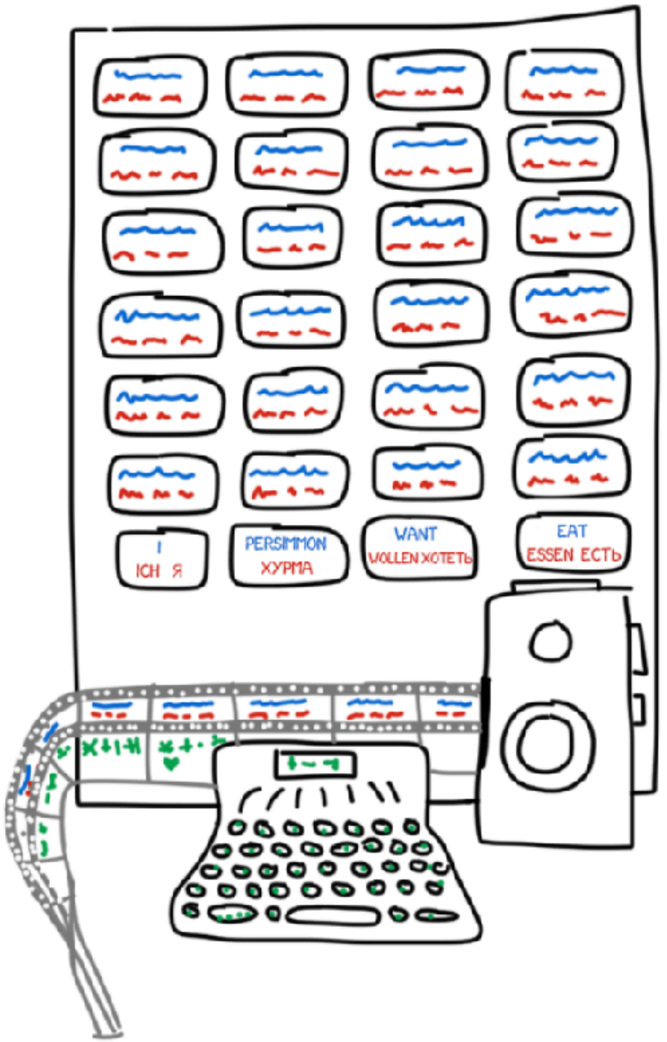

vas3k blog
Machine Translation
From the Cold War to Deep Learning
https://vas3k.com/blog/machine_translation/

1933: The Origins

Story begins in 1933. Soviet scientist Peter Troyanskii presented "the machine for the selection and printing of words when translating from one language to another" to the Academy of Sciences of the USSR. The invention was super simple — cards in four different languages, typewriter, and an old-school film camera.

The operator took the first word from the text, found a corresponding card, took a photo and typed its morphological characteristics (noun, plural, genitive) on the typewriter. The typewriter's keys encoded one of the features. The tape and the camera's film used simultaneously, making a set of frames with words and their morphology.

Despite the breakthrough, the invention was not considered useful and no one would know about it until two Soviet scientists found his patents in 1956.



The translating machine of P. P. Troyanskii (Illustration made from descriptions. No photos left, obviously)

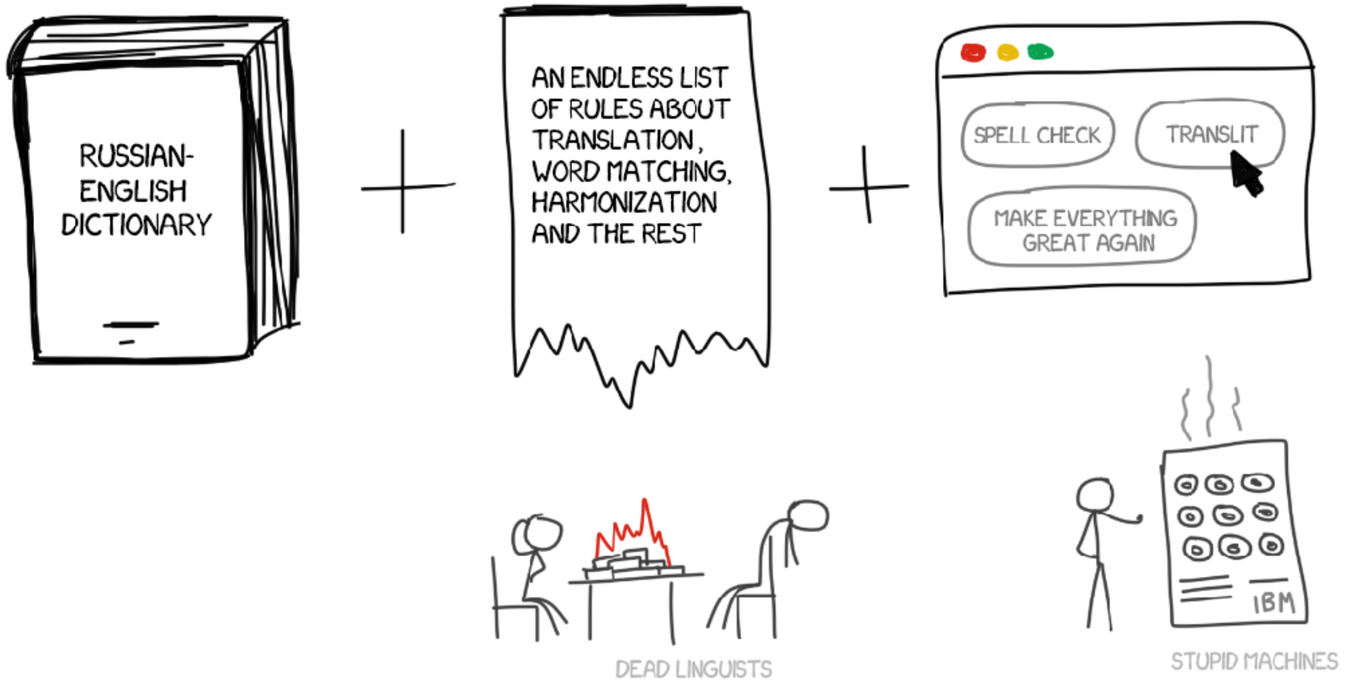
1954: The Georgetown-IBM Experiment

At the beginning of the Cold War, on January 7th 1954, at IBM New York headquarters, the IBM 701 computer automatically translated 600 sentences from Russian into English. A first in history, the computer was able to translate at a pace of 2 and a half lines per second.

However, the translated samples were carefully selected and tested. The system was not more suitable for everyday use than a simple phrasebook. Nevertheless, this started a race for machine translation between countries such as the US, Germany, France and Japan.

From then on, the struggle to improve MT (Machine Translation) lasted for 6 decades resulting in the creation of different models of MT from Rule-Based Machine Translation (RBMT) to Neural Machine Translation (NMT).

The 70″s: Rule-Based MT (RBMT)



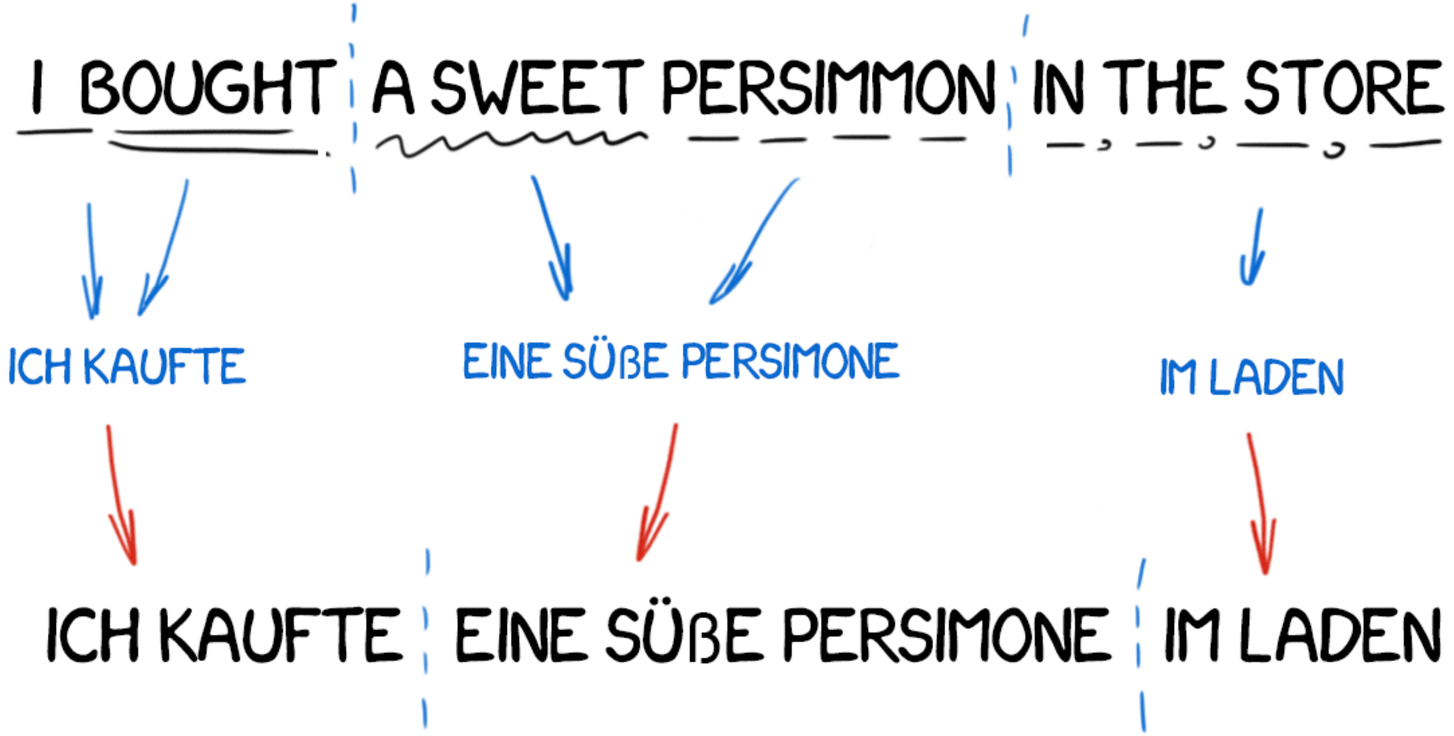
The first rule-based machine translation concepts emerged in the ’70s. The scientist took inspiration in interpreter’s work, trying to program very slow computers to repeat those actions. The system consisted of a bilingual dictionary (RU->EN) and a set of linguistic rules for both languages.

PROMT and Systran are the most famous RBMT systems, but even in the RBMT space, there are some nuances and sub-models.

Direct Machine Translation

The most basic type of machine translation. It separates the text into words, translates them, slightly corrects the morphology and fine tunes the syntax to make the sentence right, more or less. The output returned a translation of quite poor quality and this model is hardly useful.

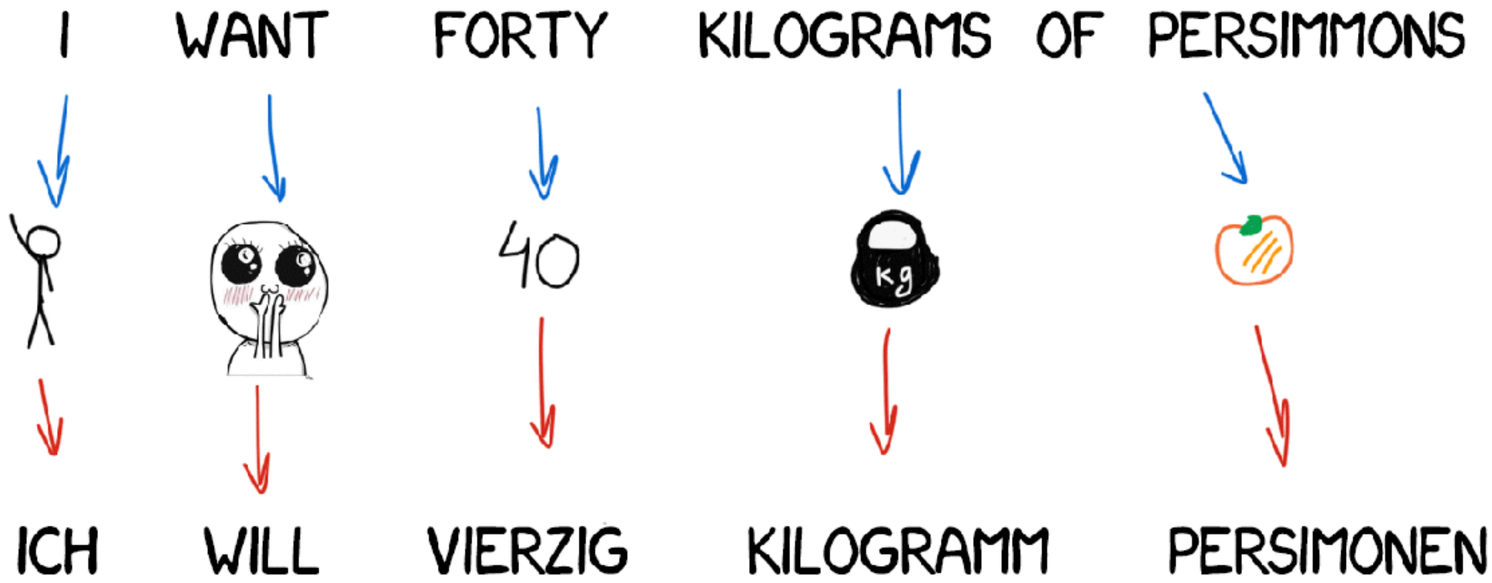
Transfer-based Machine Translation



The main difference from Direct Machine Translation is that there is a preparation process to determine the structure of the sentence to be translated. The system was meant to manipulate whole constructions, not words, in order to generate a better output. In theory.

Translation outputs were still of poor quality and even though it might have simplified grammar rules it became too complex for the system because of the increased number of word constructions compared to single words.

Interlingua Machine Translation



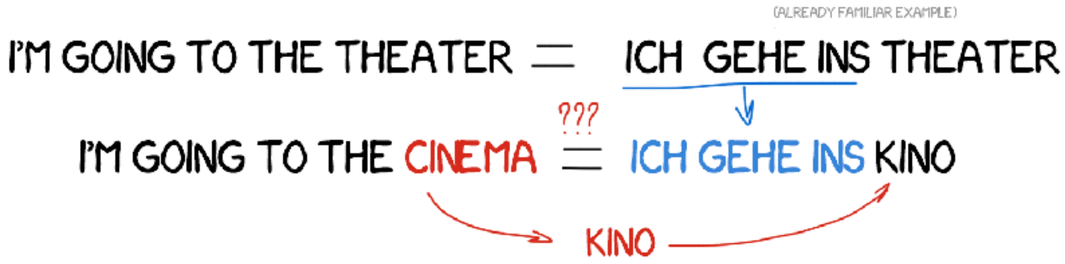
In this model, the source language is first transformed into an intermediate universal representation (Interlingua) which could then be converted into any target language. This was the main singularity of Interlingua Machine Translation, because of the initial conversion into the intermediate representation, it meant also that we could translate one source text into various languages which was not possible in the transfer-based model.

As good as it sounds, it was extremely hard to create such a universal interlingua and scientists dedicating their whole lives to the task did not manage to do so. Although, thanks to them, we now have different levels of language representation such as semantic, syntactic or morphological.

The 80’s: Example-Based MT (EBMT)

Japan was highly interested in Machine Translation. It was identified very soon that the lack of English speakers in the country would be an issue for the upcoming globalisation. Because of a completely different language structure, Rule-based English-Japanese machine translation is pretty much impossible.

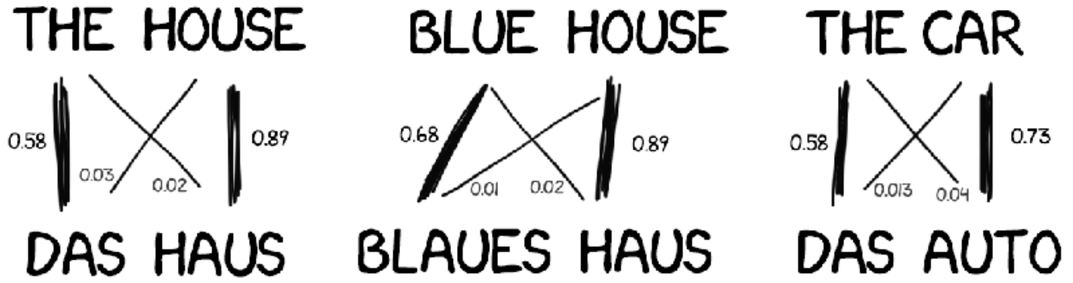
In 1984, Makoto Nagao of Kyoto University had the idea of using ready-made phrases instead of repeated translation. Imagine we have to translate a simple sentence – “I’m going to the cinema” and we have already translated the similar sentence – “I’m going to the theatre” and we can find the word “cinema” in the dictionary.



All the system needs to do is identify the missing word and then translate it. EBMT unveiled a major breakthrough, the machine can be fed with existing translations, without having to spend years in creating rules and exceptions.

The 90s: Statistical Machine Translation (SMT)

In the early 90s, at the IBM Research Center, a machine translation system was tested based not on rules and linguistics but on previous translations analysis and patterns. The model is based on the fact that, based on statistics of millions of previous translations, the machine is able to understand a pattern and choose the most adequate translation. “If people translate that way, I will”, and so Statistical Machine Translation was born. The method was more performant than any previous ones and zero linguists were needed. The more data the system had to calculate its statistics, the better the output was.

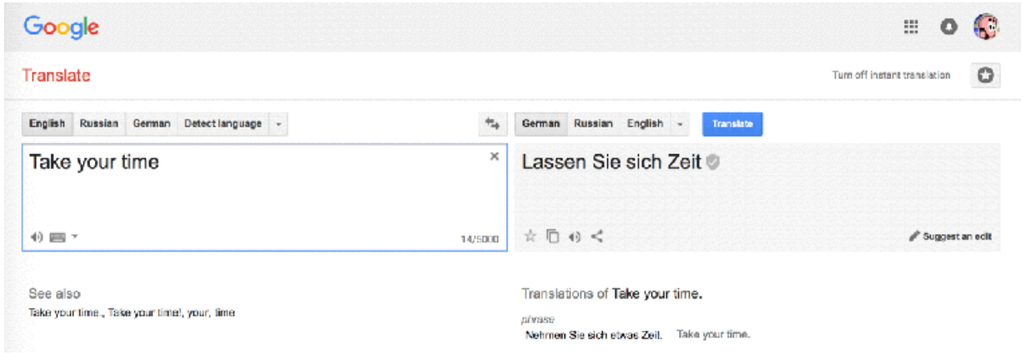


The SMT went through different models over the years as it struggled with different languages dimensions such as word order or when new words need to be added to the output. The system evolved from word-based SMT to phrase-based SMT which was able to handle word re-ordering and came with a few additional lexical hacks.
Phrase-based SMT became the state-of-the-art of Machine Translation and was used by all the high-profile online translators such as Bing, Yandex or Google Translate. The latter would then revolutionize the Machine Translation world a few years later.

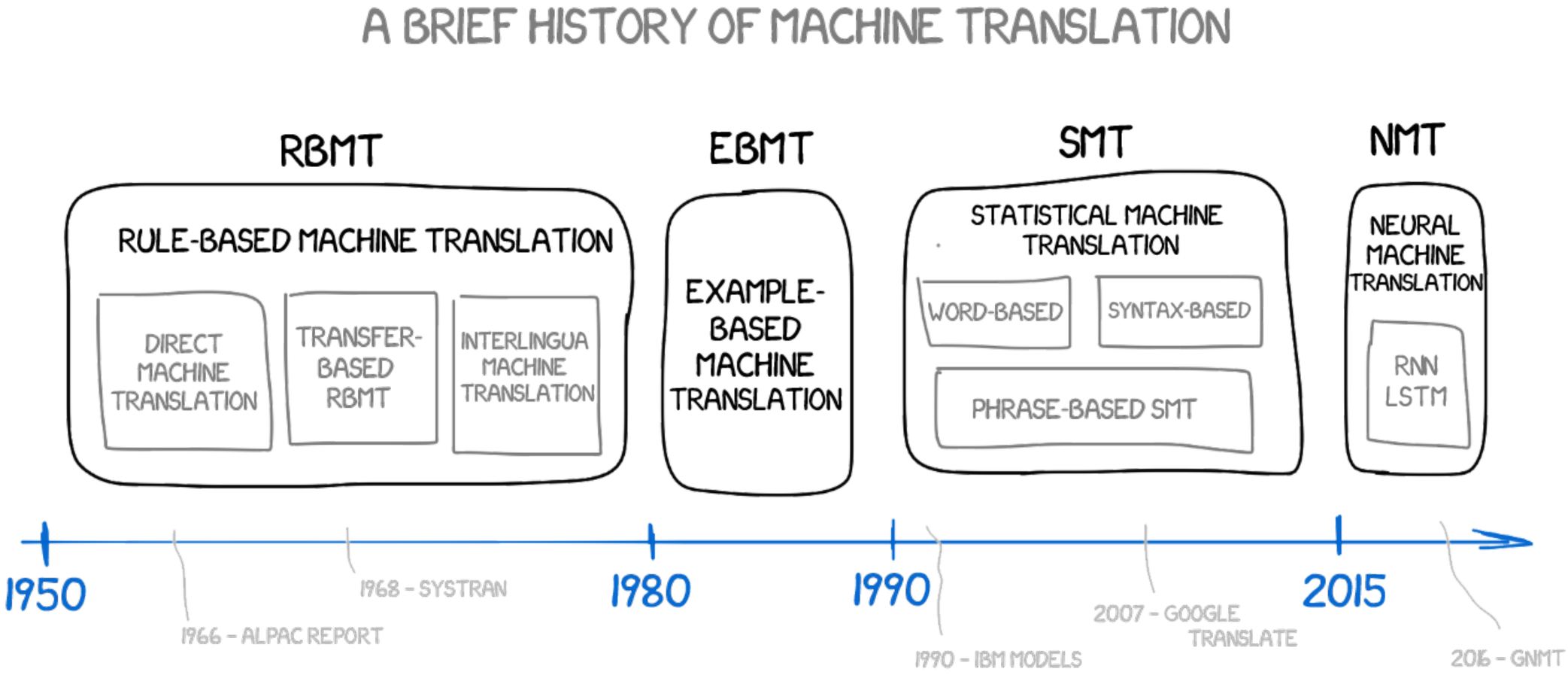
2016: The Neural MT (NMT) Revolution

In November 2016, Google made a game-changing announcement announcing the launch of the Google Neural Machine Translation System (GNMT). The idea was similar to transferring style between photos such as programs like Prisma that can turn a photo into a painting imitating famous artists' style. If we can transfer the style to a photo, how about imposing a language to a source text?

The idea was to be able to translate while keeping the essence of the source text (just like the artist's style). The source text is encoded into a set of specific features by one neural network and then decoded back into text in the target language by another network. Both networks speak a different language and don't know about each other but they both can understand the set of features extracted from the source text. This is quite similar to the idea of Interlingua Machine Translation. In a few years, NMT surpassed every system that developed previously and with the implementation of deep learning, it was able to implement improvements without being taught to do so.



History of Machine Translation



How Good is Machine Translation Today?

March 14 2018:

“ **Microsoft reaches a historic milestone, using AI to match human performance in translating news from Chinese to English**”

https://techcrunch.com/2018/03/14/microsoft-announces -breakthrough -in-chinese -to-english-machine -translation/

But also

https://www.haaretz.com/israel -news/palestinian-arrested-over-mistranslated-good-morning- facebook - post- 1.5459427

Still many problem: low-resource, multi-lingual, knowledge, common sense, etc.
MT is still an active research field

How good is a translation?

Problem: no single right answer

这个 机场 的 安全 工作 由 以色列 方面 负责 .

Israeli officials are responsible for airport security.
Israel is in charge of the security at this airport.
The security work for this airport is the responsibility of the Israel government.
Israeli side was in charge of the security of this airport.
Israel is responsible for the airport's security.
Israel is responsible for safety work at this airport.
Israel presides over the security of the airport.
Israel took charge of the airport security.
The safety of this airport is taken charge of by Israel.
This airport's security is the responsibility of the Israeli security officials.

Evaluation of MT

- How good is a given machine translation system?
- Many different translations acceptable
- Evaluation metrics
  - Subjective judgments by human evaluators
  - Automatic evaluation metrics

Evaluation of MT: Adequacy and Fluency

- Human judgment
  - Given: machine translation output
  - Given: input and/or reference translation
  - Task: assess quality of MT output

- Metrics
  - **Adequacy:** does the output convey the meaning of the input sentence? Is part of the message lost, added, or distorted?
  - **Fluency:** is the output fluent? Involves both grammatical correctness and idiomatic word choices.

Evaluation of MT: Adequacy and Fluency

Adequacy	
5	all meaning
4	most meaning
3	much meaning
2	little meaning
1	none

Fluency	
5	flawless English
4	good English
3	non-native English
2	disfluent English
1	incomprehensible

Evaluation of MT: Challenges

- No single correct answer
- Human evaluators disagree

Automatic Evaluation Metrics

- Goal: computer program that computes quality of translations
- Advantages: low cost, optimizable, consistent
- Basic strategy
  - Given: MT output
  - Given: human reference translation
  - Task: compute similarity between them

Precision and Recall of Words

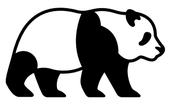
SYSTEM A: Israeli officials ~~responsibility~~ ~~of~~ airport ~~safety~~

REFERENCE: Israeli officials are responsible for airport security

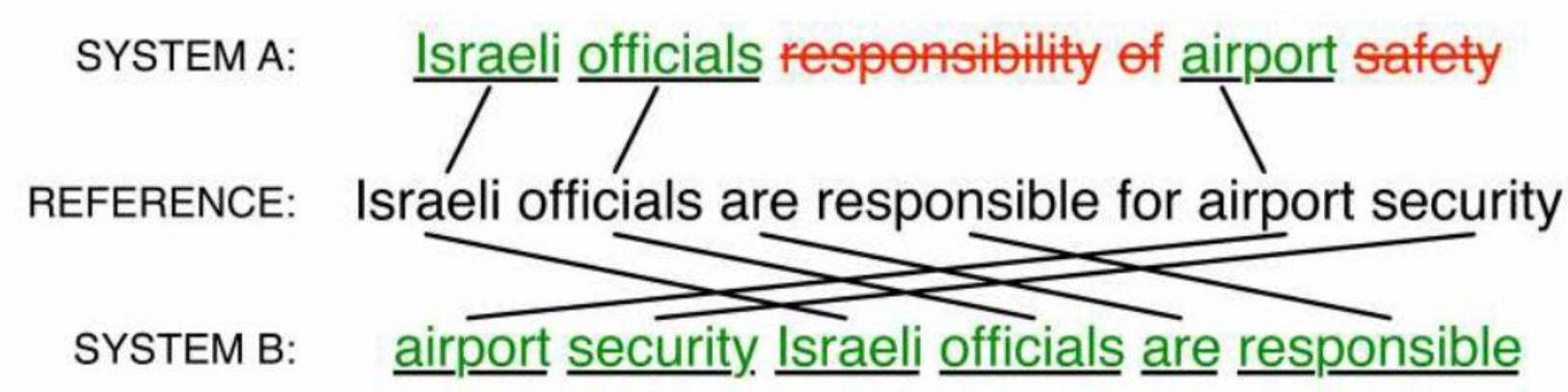
Precision $$\frac{correct}{output\text{-}length} = \frac{3}{6} = 50\%$$

Recall $$\frac{correct}{reference\text{-}length} = \frac{3}{7} = 43\%$$

F-measure $$\frac{precision \times recall}{(precision + recall)/2} = \frac{.5 \times .43}{(.5 + .43)/2} = 46\%$$



Precision and Recall of Words



Metric	System A	System B
precision	50%	100%
recall	43%	100%
f-measure	46%	100%

flaw: no penalty for reordering

BLEU: Bilingual Evaluation Understudy

N-gram overlap between machine translation output and reference translation

Compute precision for n-grams of size 1 to 4

Add brevity penalty (for too short translations)

$$\text{BLEU} = \min\left(1, \frac{\textit{output-length}}{\textit{reference-length}}\right) \left(\prod_{i=1}^{4} \textit{precision}_i\right)^{\frac{1}{4}}$$

Typically computed over the entire corpus, not single sentences

Multiple Reference Translations

To account for variability, use multiple reference translations

- n-grams may match in any of the references
- closest reference length used

Example

SYSTEM: Israeli officials (2-GRAM MATCH) responsibility of (2-GRAM MATCH) airport (1-GRAM) safety

REFERENCES:
Israeli officials are responsible for airport security
Israel is in charge of the security at this airport
The security work for this airport is the responsibility of the Israel government
Israeli side was in charge of the security of this airport

BLEU Examples

SYSTEM A: Israeli officials (2-GRAM MATCH) responsibility of airport (1-GRAM MATCH) safety

REFERENCE: Israeli officials are responsible for airport security

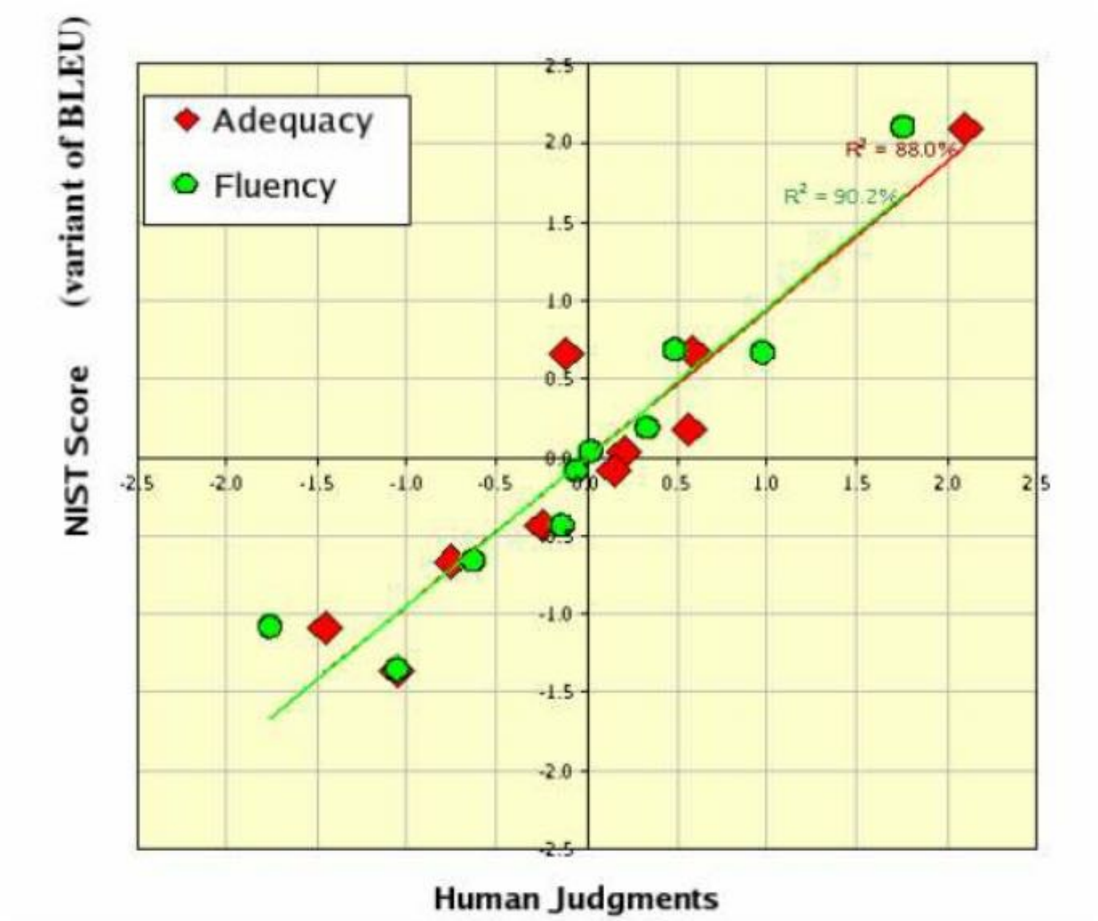
SYSTEM B: airport security (2-GRAM MATCH) Israeli officials are responsible (4-GRAM MATCH)

Metric	System A	System B
precision (1gram)	3/6	6/6
precision (2gram)	1/5	4/5
precision (3gram)	0/4	2/4
precision (4gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

Drawbacks of Automatic Metrics

- All words are treated as equally relevant
- Operate on local level
- Scores are meaningless (absolute value not informative)
- Human translators score low on BLEU

Yet automatic metrics such as BLEU correlate with human judgement



Automatic Metrics

- Essential tool for system development
- Use with caution: not suited to rank systems of different types
- Still an open area of research
  - Connects with semantic analysis

Suggested Readings

- Evaluation of Text Generation: A Survey
- Compression, Transduction, and Creation: A Unified Framework for Evaluating Natural Language Generation
- BERTSCORE: EVALUATING TEXT GENERATION WITH BERT

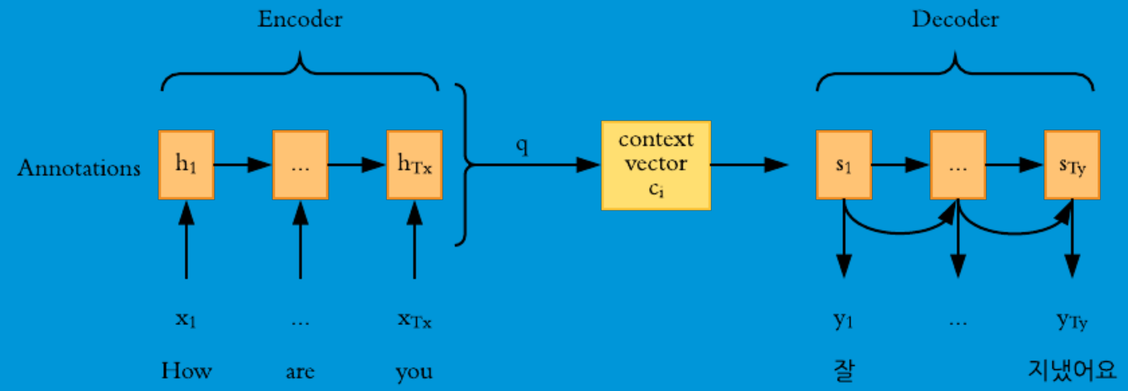
The WMT Evaluation Campaign

The WMT evaluation campaign (http://www.statmt.org) has been run annually since 2006. It is a collection of shared tasks related to machine translation, in which researchers compare their techniques against those of others in the field. The longest running task in the campaign is the translation task, where participants translate a common test set with their MT systems

http://www.statmt.org/

https://www.aclweb.org/anthology/venues/wmt/

Sequence to Sequence



Sequence to Sequence

Formally, in the machine translation task, we have an input sequence $x_1, x_2, \ldots, x_m$ and an output sequence $y_1, y_2, \ldots, y_n$ (note that their lengths can be different). Translation can be thought of as finding the target sequence that is the most probable given the input; formally, the target sequence that maximizes the conditional probability $p(y|x)$: $y^* = \arg\max_y p(y|x)$.

If you are bilingual and can translate between languages easily, you have an intuitive feeling of $p(y|x)$ and can say something like "...well, this translation is kind of more natural for this sentence". But in machine translation, we learn a function $p(y|x, \theta)$ with some parameters θ, and then find its argmax for a given input: $y' = \arg\max_y p(y|x, \theta)$.

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

Sequence to Sequence

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

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

Questions we need to answer

- **modeling**

  How does the model for $p(y|x, \theta)$ look like?

- **learning**

  How to find θ?

- **search**

  How to find the argmax?

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

Sequence to Sequence

To define a machine translation system, we need to answer three questions:

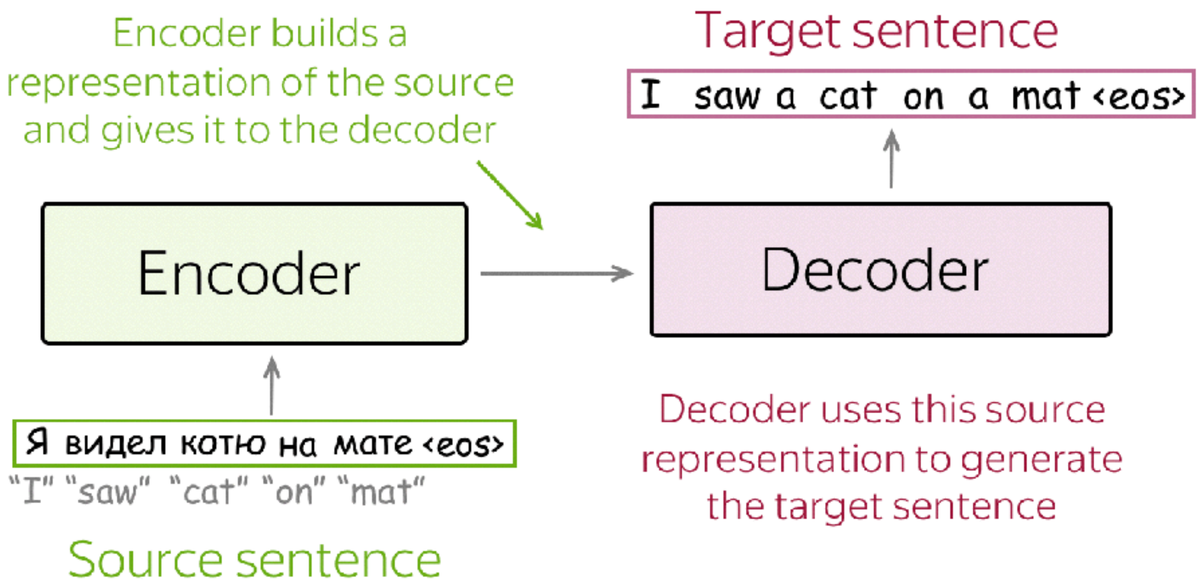
- **modeling** - how does the model for $p(y|x, \theta)$ look like?
- **learning** - how to find the parameters θ?
- **inference** - how to find the best y?

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

Encoder-Decoder Framework

Encoder-decoder is the standard modeling paradigm for sequence-to-sequence tasks. This framework consists of two components:

- **encoder** - reads source sequence and produces its representation;
- **decoder** - uses source representation from the encoder to generate the target sequence.



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

Conditional Language Models

In the Language Modeling lecture, we learned to estimate the probability $p(y)$ of sequences of tokens $y = (y_1, y_2, \ldots, y_n)$. While **language models** estimate the **unconditional** probability $p(y)$ of a sequence y, **sequence-to-sequence models** need to estimate the **conditional** probability p(y|x) of a sequence y given a source x. That's why sequence-to-sequence tasks can be modeled as **Conditional Language Models (CLM)** - they operate similarly to LMs, but additionally receive source information x.

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

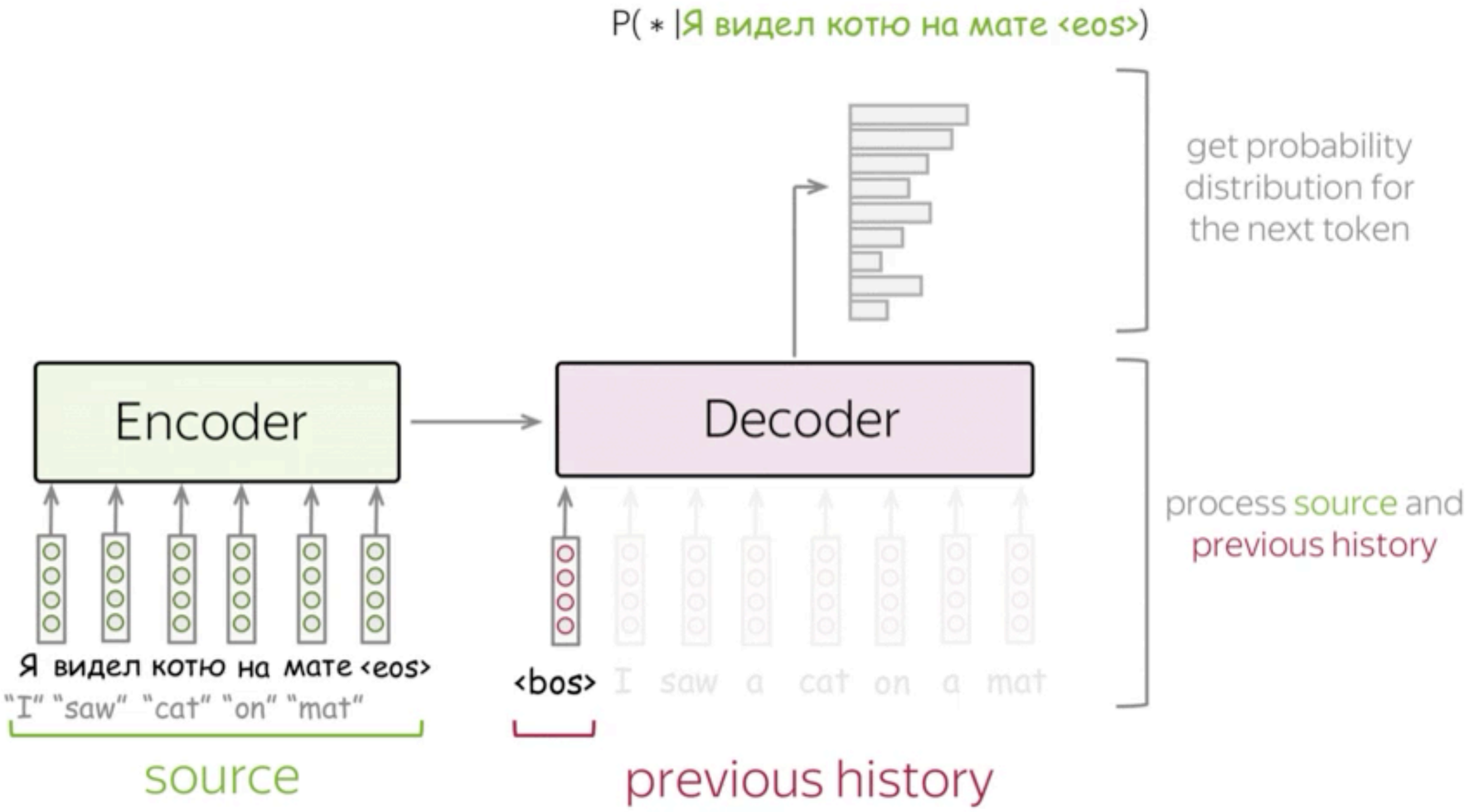
Conditional Language Models: $$P(y_1, y_2, \ldots, y_n, |x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$$

condition on source x

Note that Conditional Language Modeling is something more than just a way to solve sequence-to-sequence tasks. In the most general sense, **x can be something other than a sequence of tokens**. For example, in the Image Captioning task, x is an image and y is a description of this image.

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

Conditional Language Models



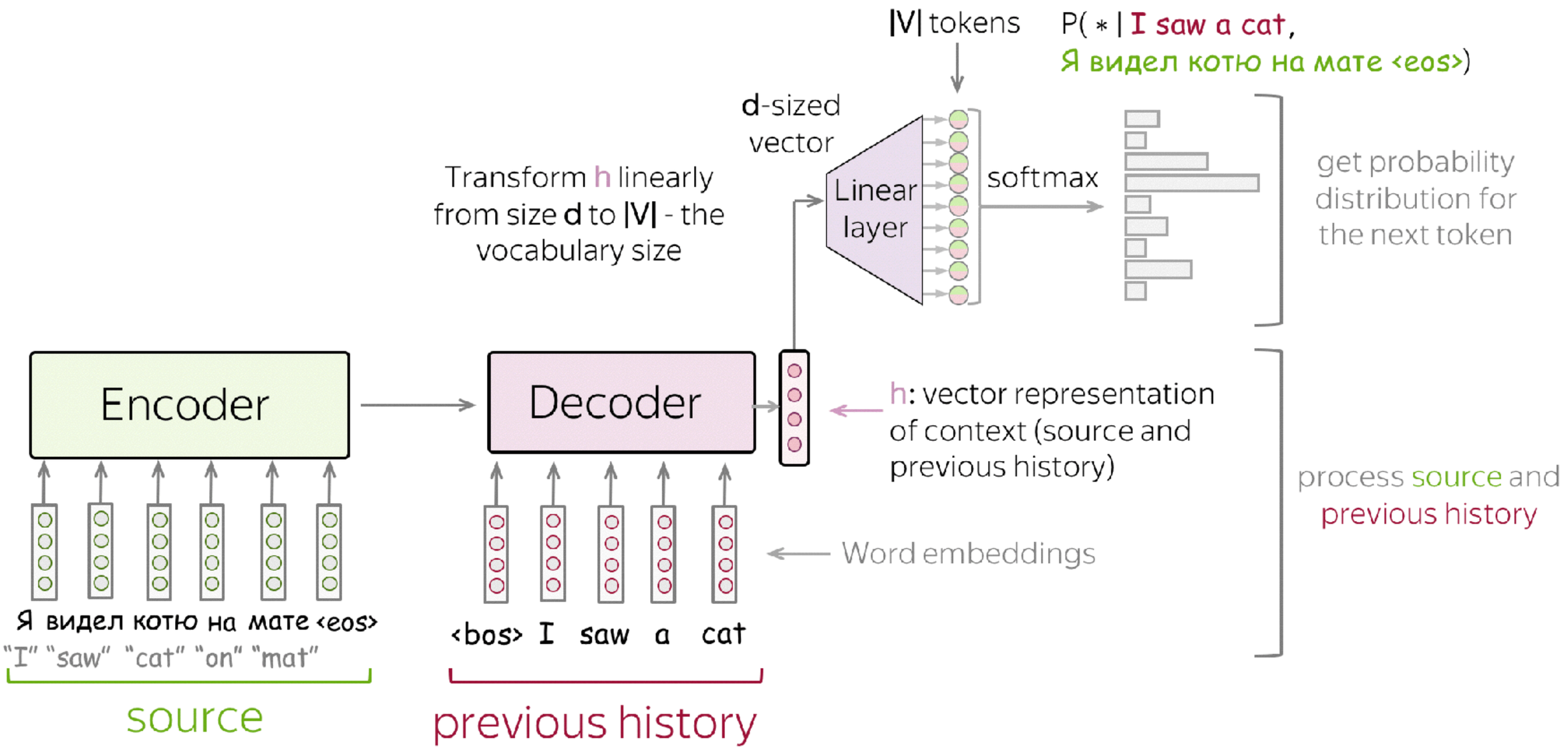
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Conditional Language Models

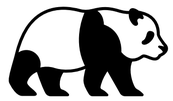
Since the only difference from LMs is the presence of source $\boldsymbol{x}$, the modeling and training is very similar to language models. In particular, the high-level pipeline is as follows:

- 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.

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

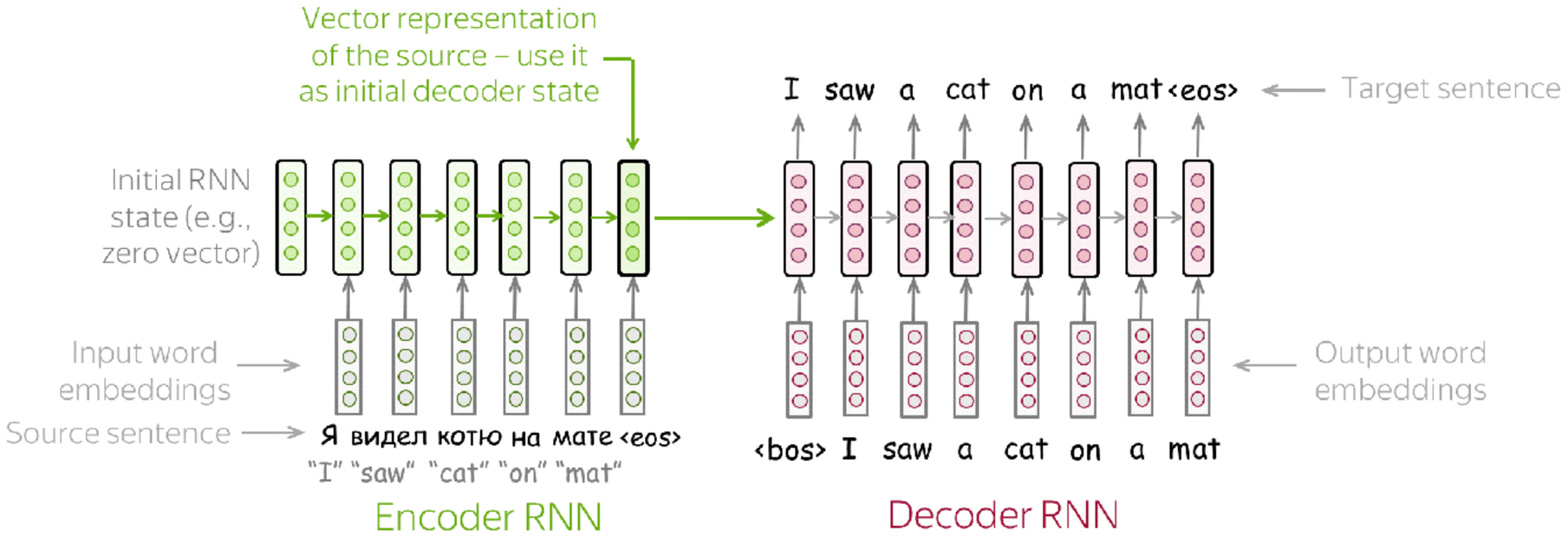


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



Simple Model: Two RNNs for Encoder & Decoder

The simplest encoder-decoder model consists of two RNNs (LSTMs): one for the encoder and another for the decoder. Encoder RNN reads the source sentence, and the final state is used as the initial state of the decoder RNN. The hope is that the final encoder state "encodes" all information about the source, and the decoder can generate the target sentence based on this vector.

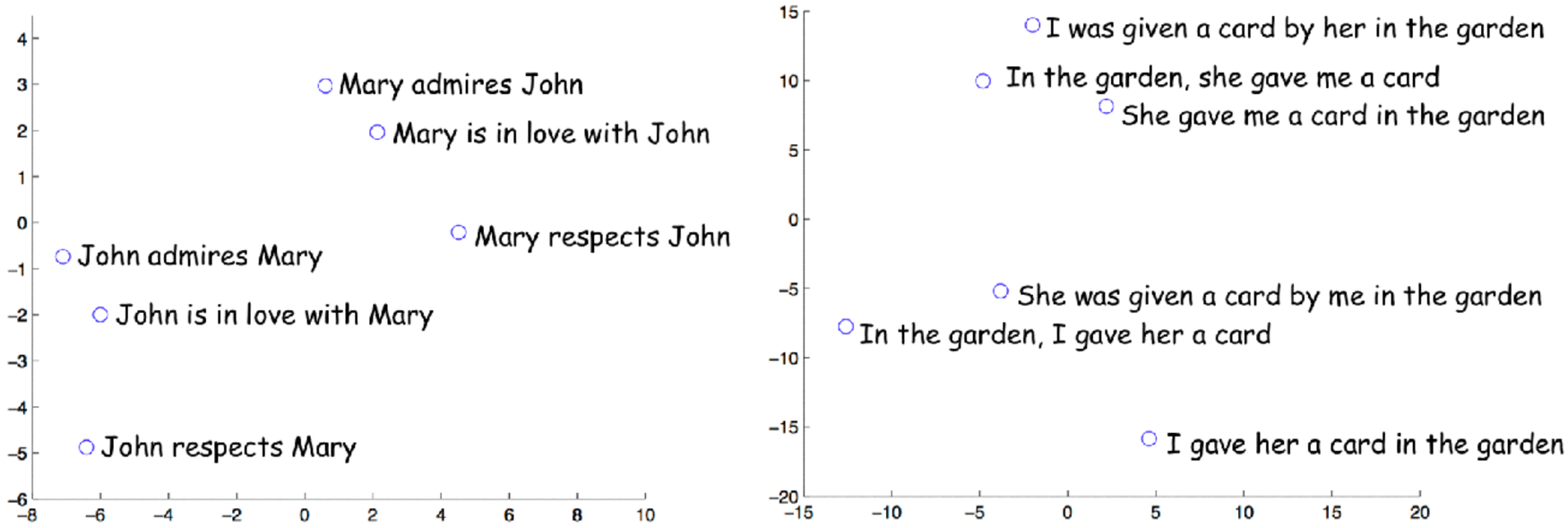


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Simple Model: Two RNNs for Encoder & Decoder

This model can have different modifications: for example, the encoder and decoder can have **several layers**. Such a model with several layers was used, for example, in the paper Sequence to Sequence Learning with Neural Networks - one of the first attempts to solve sequence-to-sequence tasks using neural networks.

In the same paper, the authors looked at the last encoder state and visualized several examples - look below. Interestingly, **representations of sentences with similar meaning but different structure are close**!

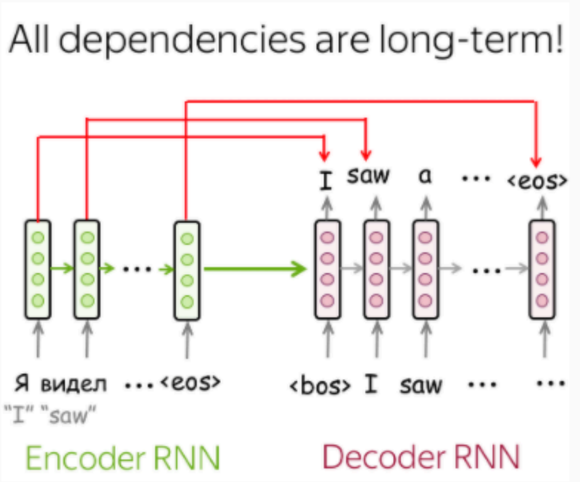


The examples are from the paper Sequence to Sequence Learning with Neural Networks

Training Trick

Training trick to make simple LSTMs w/o attention work

Simple LSTMs without attention does not work very well: all dependencies are long-term, and it is hard for the model. For example, by the time the decoder has to generate the beginning of a translation, it may already forget the most relevant early source tokens.



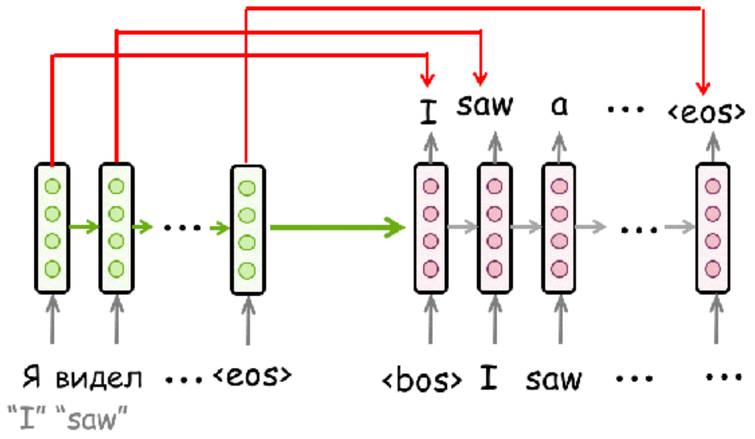
? Can you change the training pipeline (without modifying the model) to make it easier for the model to remember the beginning of the source when it starts to generate the target?

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

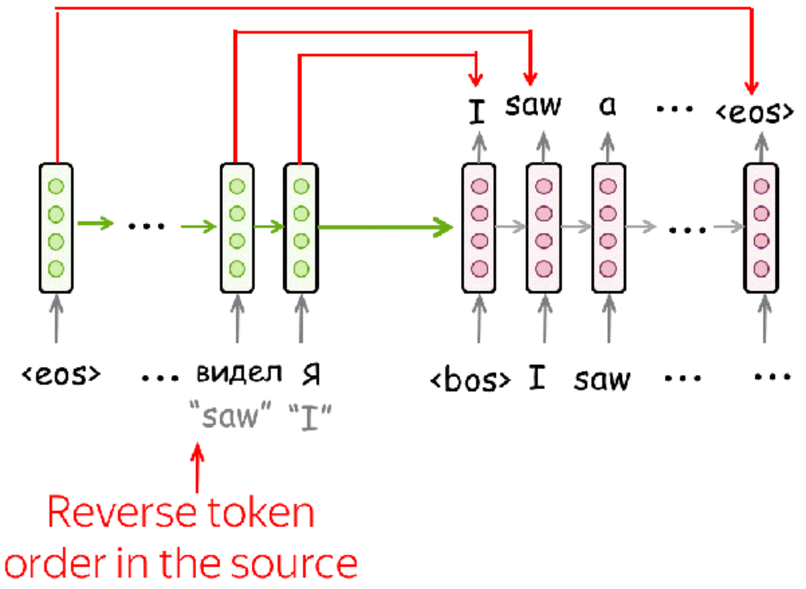
Training Trick: Reverse Order of Source Tokens

The paper Sequence to Sequence Learning with Neural Networks introduced an elegant trick to make simple LSTM seq2seq models work better: reverse the order of the source tokens (but not the target). After that, a model will have many short-term connections: the latest source tokens it sees are the most relevant for the beginning of the target.

Before: all dependencies are long-term



After: many short-term dependencies



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

Training: The Cross-Entropy Loss

Similarly to neural LMs, neural seq2seq models are trained to predict probability distributions of the next token given previous context (source and previous target tokens). Intuitively, at each step we maximize the probability a model assigns to the correct token.

Formally, let's assume we have a training instance with the source $\boldsymbol{x} = (x_1, \ldots, x_m)$ and the target $\boldsymbol{y} = (y_1, \ldots, y_n)$. Then at the timestep $\boldsymbol{t}$, a model predicts a probability distribution $\boldsymbol{p^{(t)}} = \boldsymbol{p}(*|y_1, \ldots, y_{t-1}, x_1, \ldots, x_m)$. The target at this step is $\boldsymbol{p^*} = \textbf{one-hot}(y_t)$, i.e., we want a model to assign probability 1 to the correct token, $\boldsymbol{y_t}$, and zero to the rest.

The standard loss function is the **cross-entropy loss**. Cross-entropy loss for the target distribution $\boldsymbol{p^*}$ and the predicted distribution $\boldsymbol{p}$ is

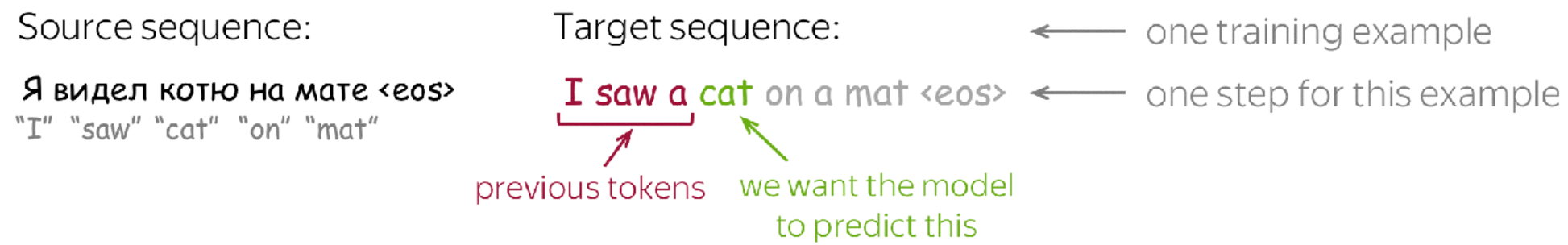
$$\boldsymbol{Loss}(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i).$$

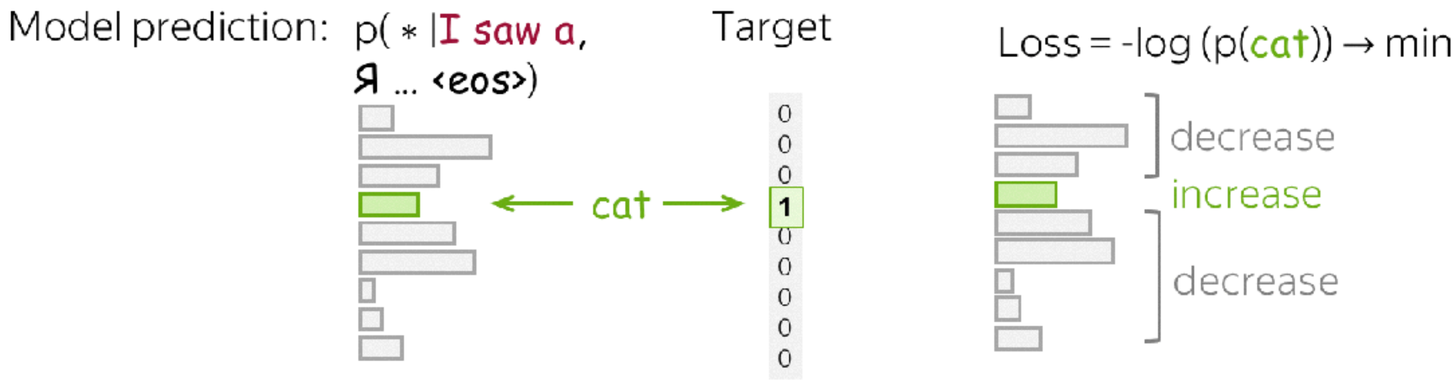
Since only one of $\boldsymbol{p_i^*}$ is non-zero (for the correct token $\boldsymbol{y_t}$), we will get

$$\boldsymbol{Loss}(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t}, x)).$$

Training: The Cross-Entropy Loss

At each step, we maximize the probability a model assigns to the correct token. Look at the illustration for a single timestep.





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

Training: The Cross-Entropy Loss

For the whole example, the loss will be $-\sum_{t=1}^{n} \log(p(y_t|y_{<t}, x))$. Look at the illustration of the training process (the illustration is for the RNN model, but the model can be different).

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

Inference

Now when we understand how a model can look like and how to train this model, let's think how to generate a translation using this model. We model the probability of a sentence as follows:

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

How to find the argmax?

Now the main question is: how to find the argmax?

Note that **we can not find the exact solution.** The total number of hypotheses we need to check is $|V|^n$, which is not feasible in practice. Therefore, we will find an approximate solution.

Inference: Greedy Decoding

- **Greedy Decoding:** At each step, pick the most probable token

The straightforward decoding strategy is greedy - at each step, generate a token with the highest probability. This can be a good baseline, but this method is inherently flawed: the best token at the current step does not necessarily lead to the best sequence.

$$\arg\max_{y} \prod_{t=1}^{n} p(y_t|y_{<t}, x) \neq \prod_{t=1}^{n} \arg\max_{y_t} p(y_t|y_{<t}, x)$$

Inference: Beam Search

- **Beam Search:** Keep track of several most probably hypotheses

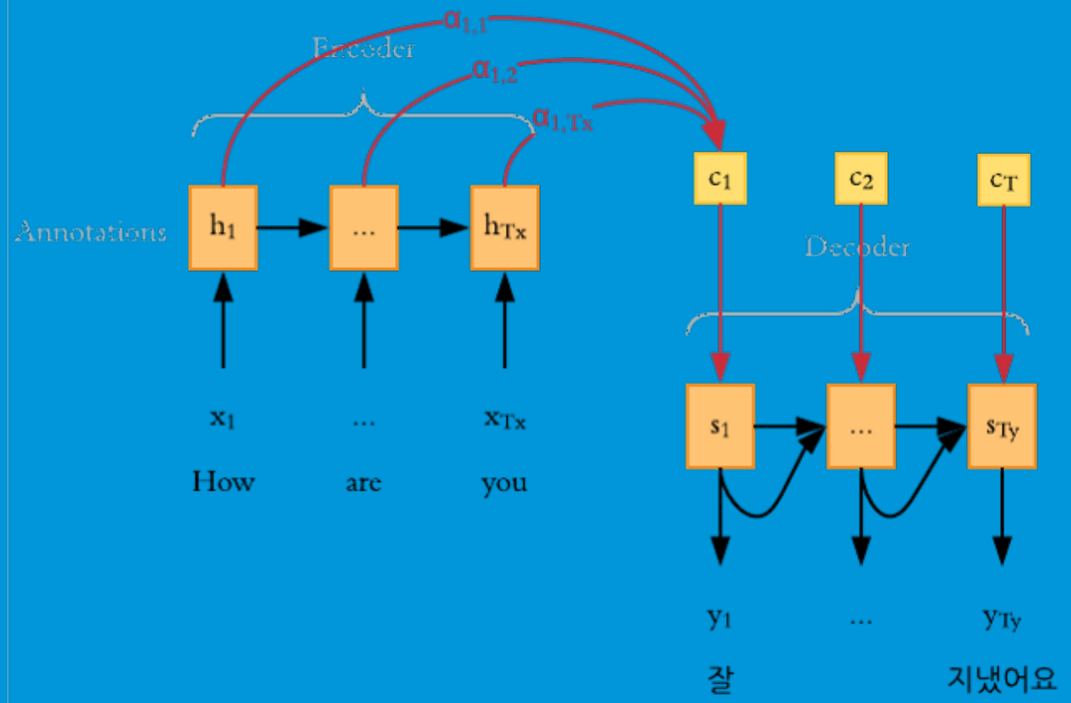
Instead, let's keep several hypotheses. At each step, we will be continuing each of the current hypotheses and pick top-N of them. This is called **beam search**.

<bos>

Start with the begin of sentence token or with an empty sequence

Usually, the beam size is 4-10. Increasing beam size is computationally inefficient and, what is more important, leads to worse quality.

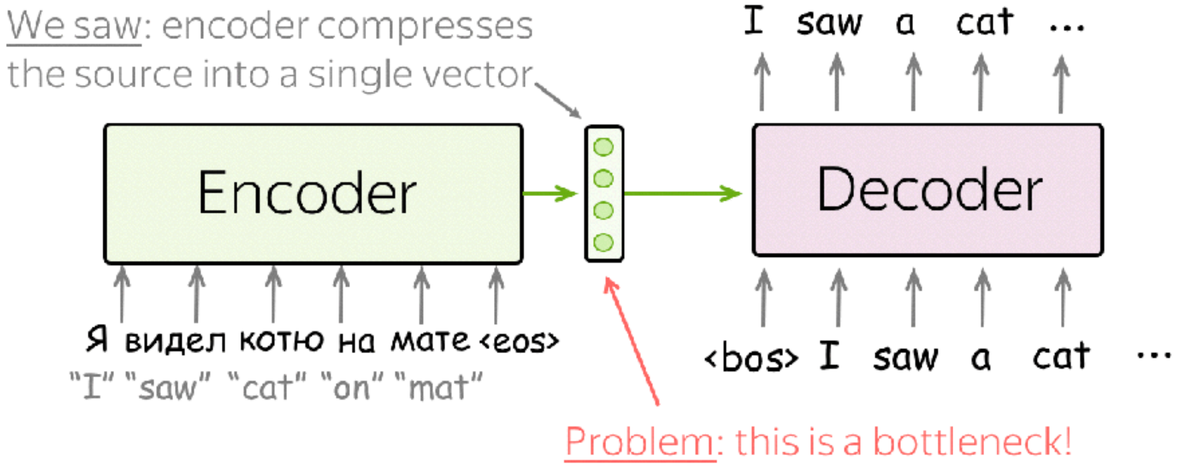
Attention


Encoder
Annotations
h1
...
hTx
x1
...
xTx
How
are
you
α1,1
α1,2
α1,Tx
c1
c2
cT
Decoder
s1
...
sTy
y1
...
yTy
잘
지냈어요

The Problem of Fixed Encoder Representation

Problem: Fixed source representation is suboptimal: (i) for the encoder, it is hard to compress the sentence; (ii) for the decoder, different information may be relevant at different steps.

In the models we looked at so far, the encoder compressed the whole source sentence into a single vector. This can very hard - the number of possible meanings of source is infinite. When the encoder is forced to put all information into a single vector, it is likely to forget something.



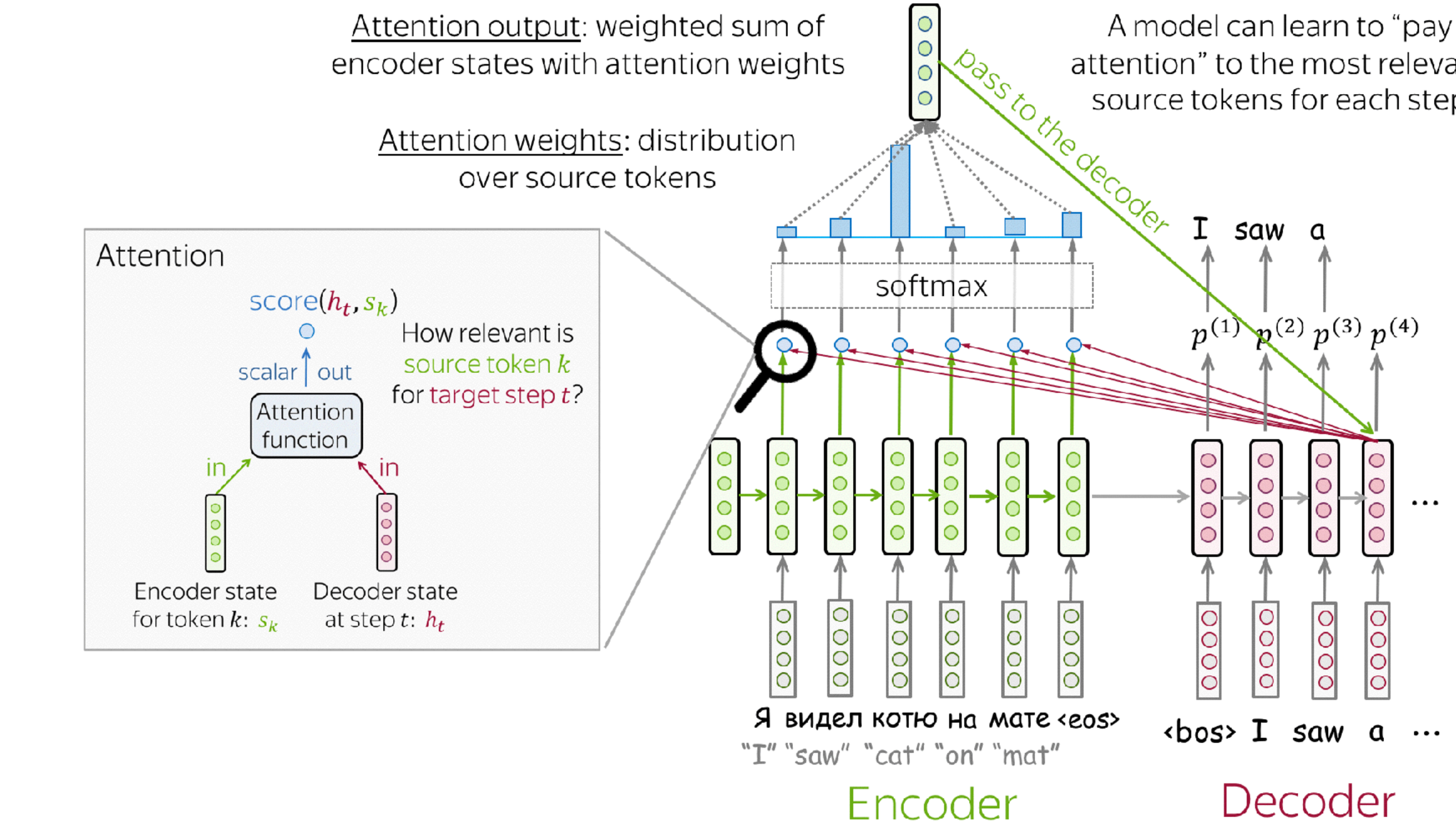
Not only it is hard for the encoder to put all information into a single vector - this is also hard for the decoder. The decoder sees only one representation of source. However, at each generation step, different parts of source can be more useful than others. But in the current setting, the decoder has to extract relevant information from the same fixed representation - hardly an easy thing to do.

Attention: A High-Level View

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

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

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 source into a single vector - it gives representations for all source tokens (for example, all RNN states instead of the last one).


Attention output: weighted sum of encoder states with attention weights
Attention weights: distribution over source tokens
A model can learn to "pay attention" to the most relevant source tokens for each step
pass to the decoder
Attention
score(h_t, s_k)
scalar
out
Attention function
in
in
How relevant is source token k for target step t?
Encoder state for token k: s_k
Decoder state at step t: h_t
softmax
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
I saw a
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

How Attention Works

At each decoder step, attention

- receives **attention input**: a decoder state h_t and all encoder states s_1, s_2, ..., s_m;
- computes **attention scores**
  For each encoder state s_k, attention computes its "relevance" for this decoder state h_t. Formally, it applies an attention function which receives one decoder state and one encoder state and returns a scalar value $score(h_t, s_k)$;
- computes **attention weights**: a probability distribution - softmax applied to attention scores;
- computes **attention output**: the weighted sum of encoder states with attention weights.

How Attention Works

The general computation scheme is shown below.

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\text{score}(h_t, s_k), \text{k} = 1..\text{m}$$

"How relevant is source token k for target step t?"

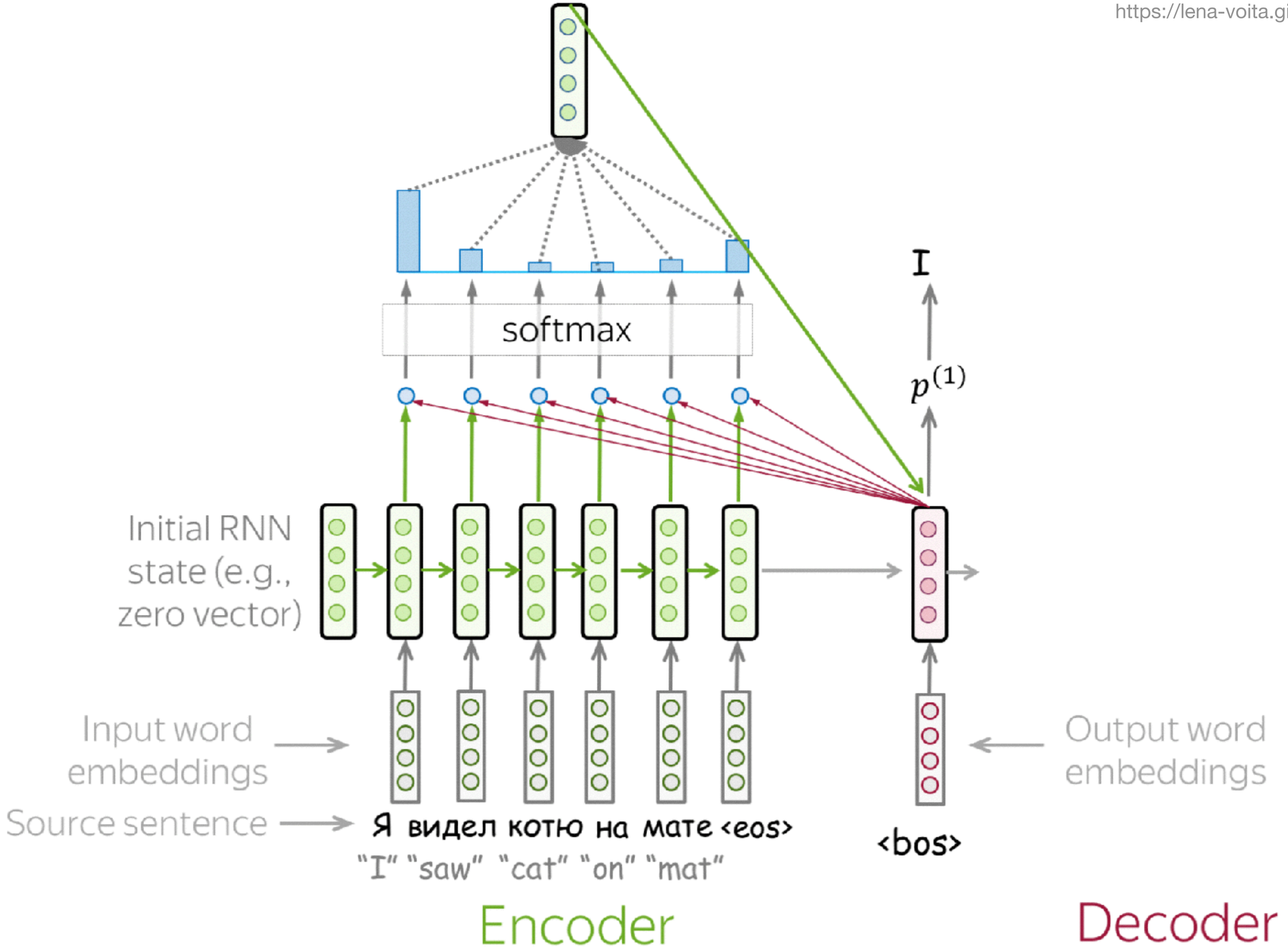
Attention input

$s_1, s_2, \ldots, s_m$ all encoder states

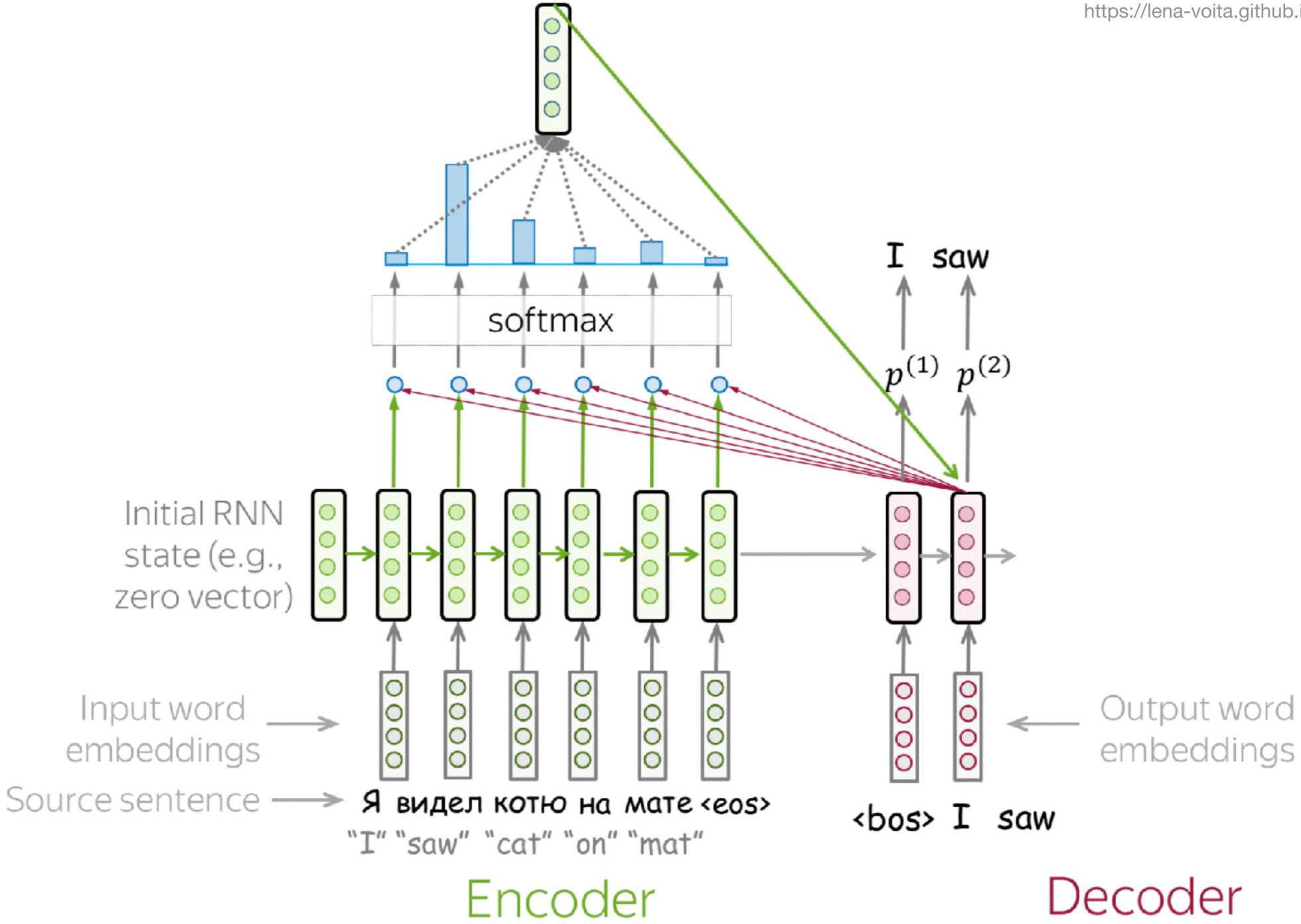
h_t one decoder state

Note: Everything is differentiable - learned end-to-end!

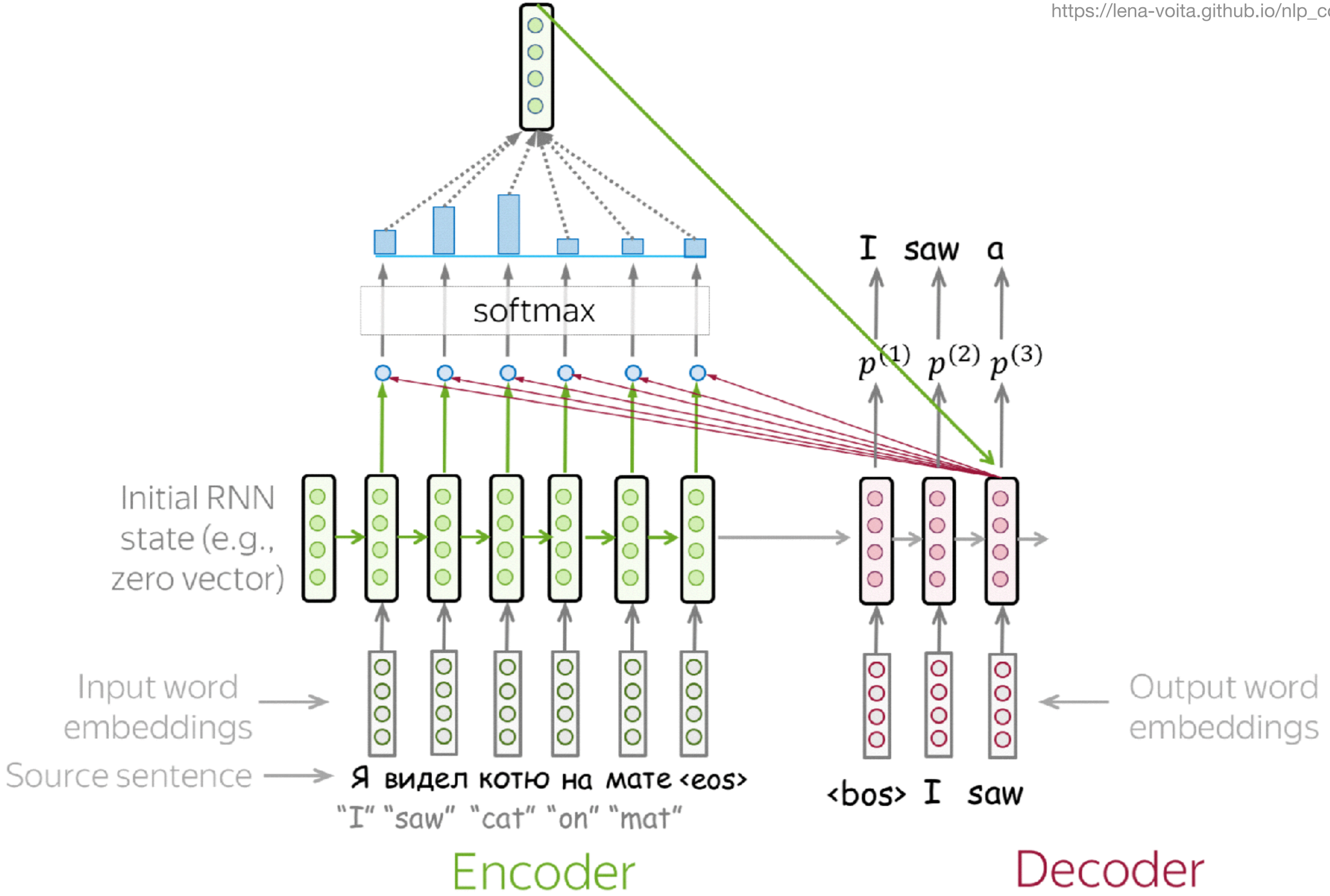
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



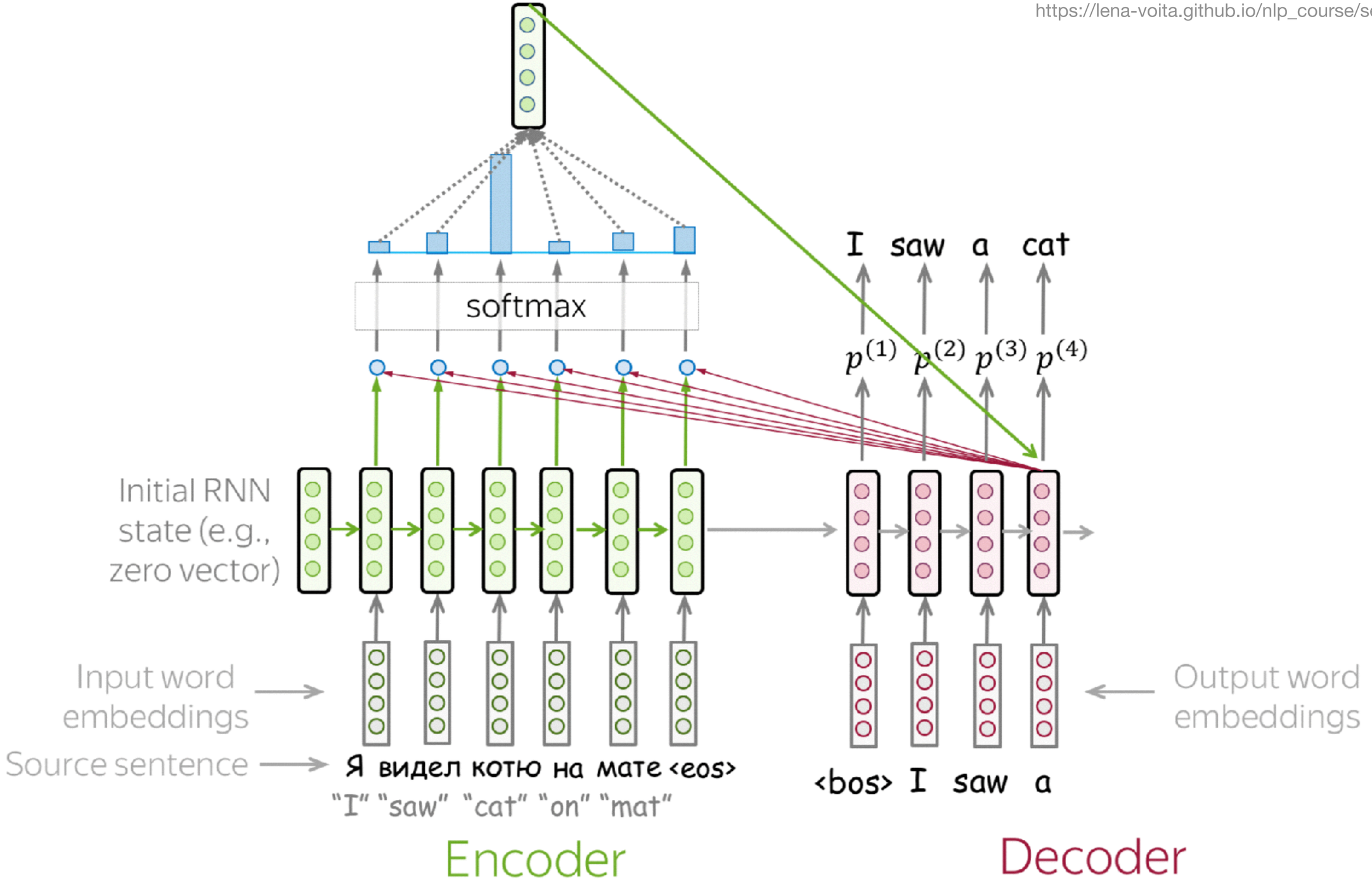
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



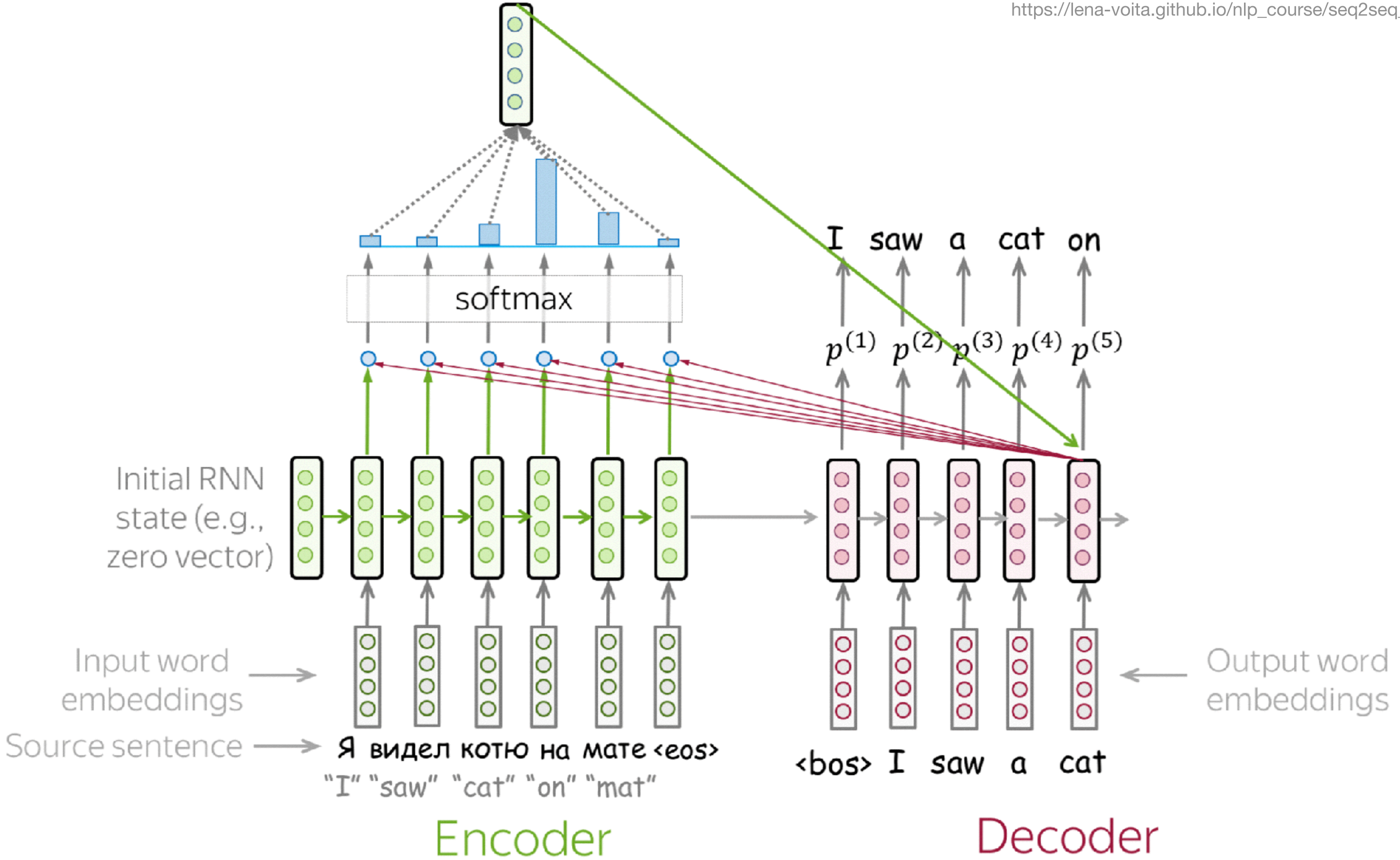
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



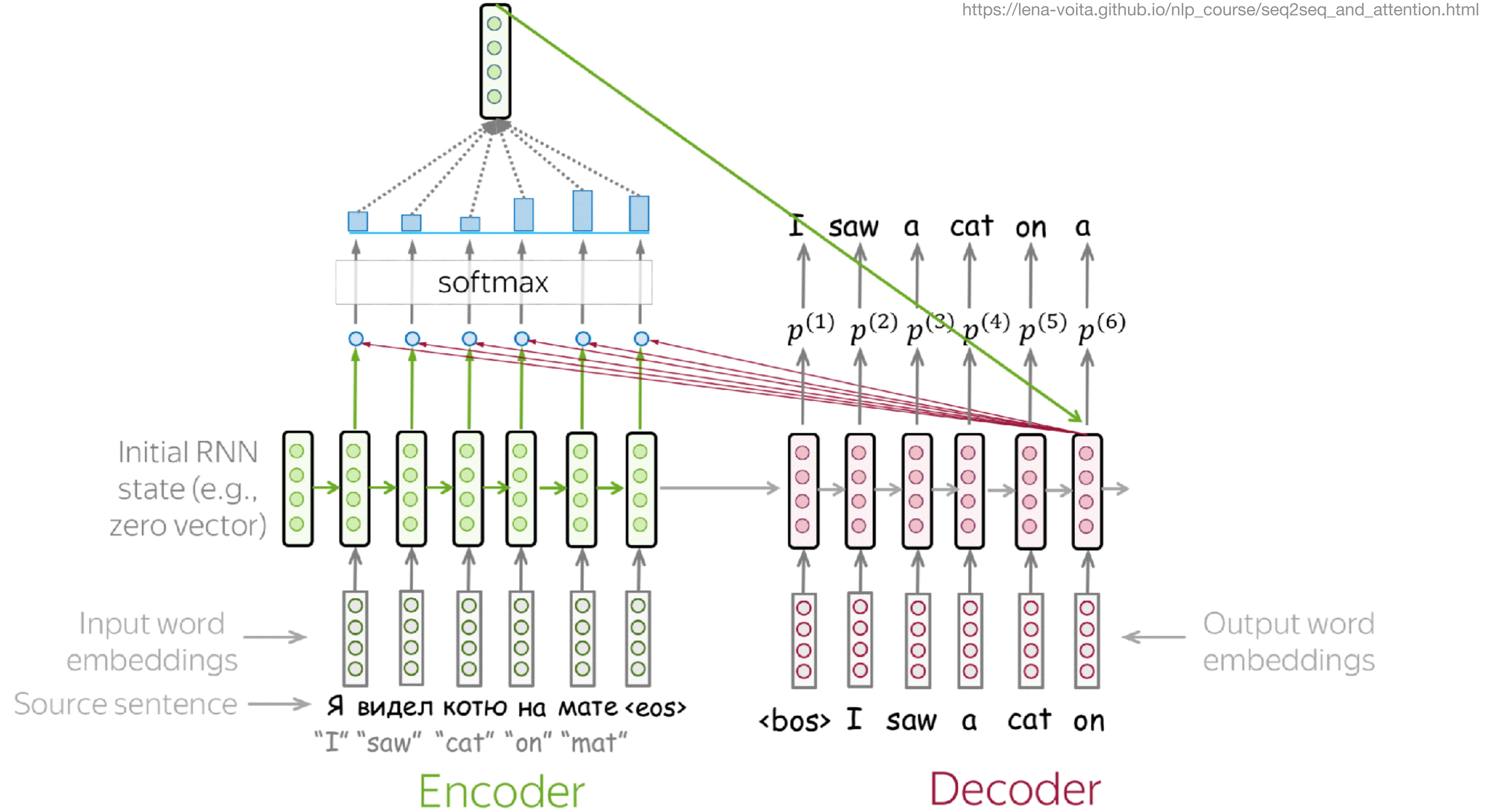
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



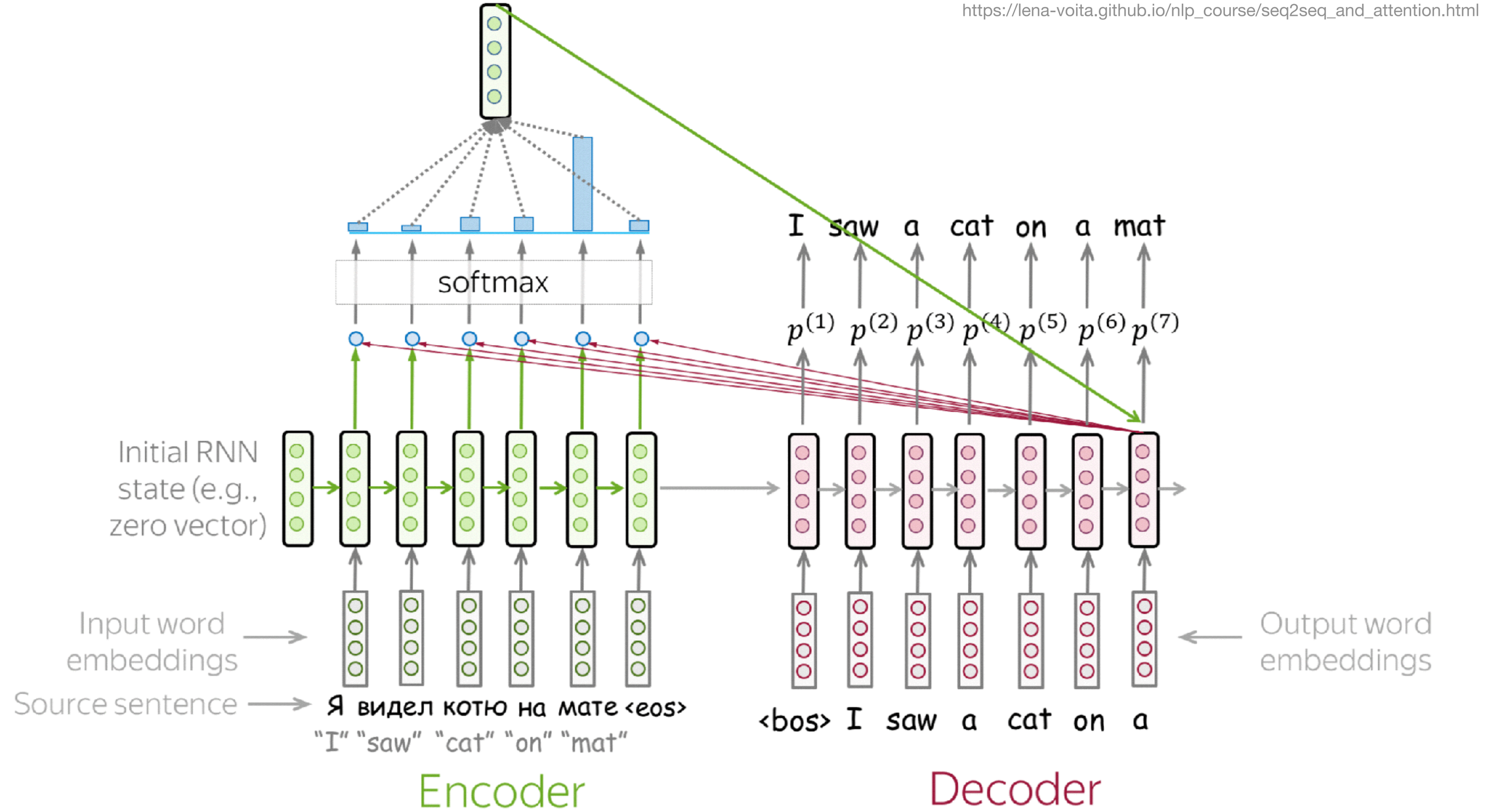
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



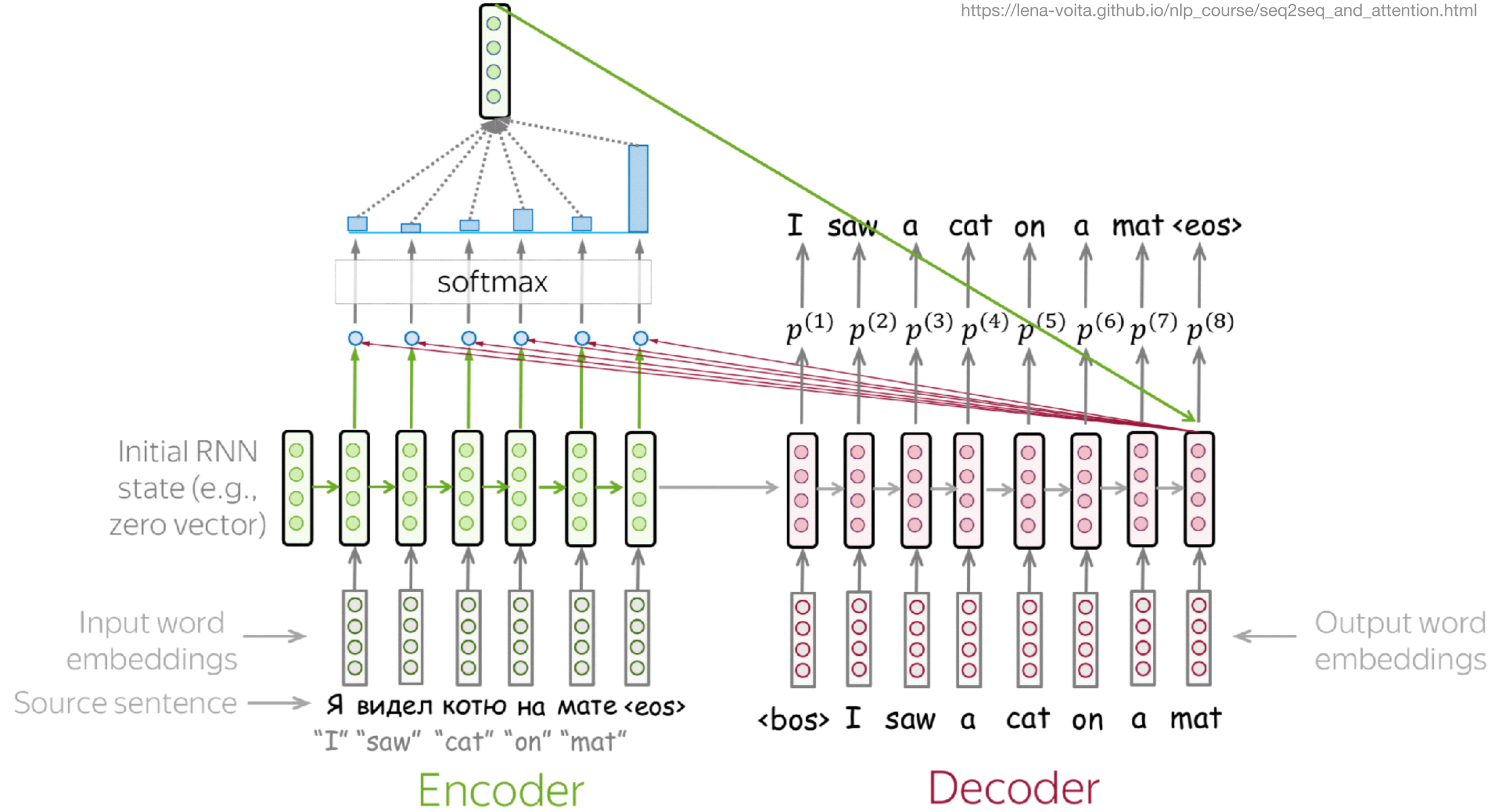
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



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

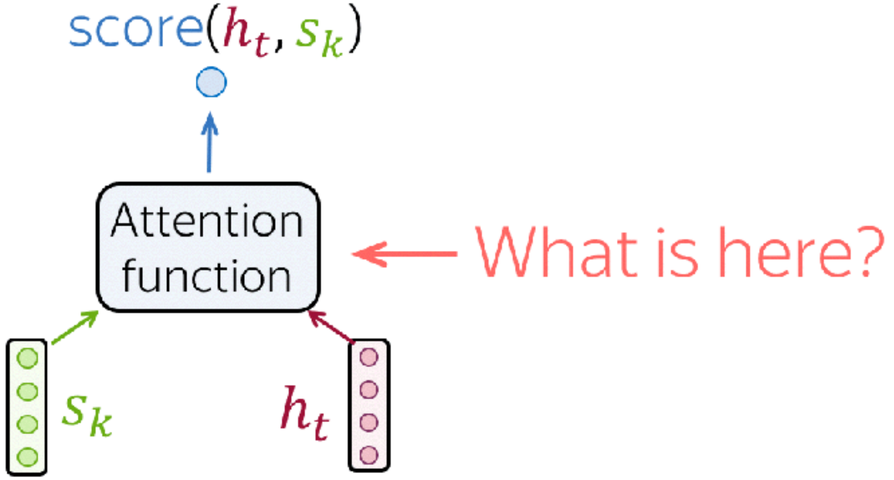


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



How to Compute Attention Scores

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. However, usually you don't need to - there are several popular and simple variants which work quite well.



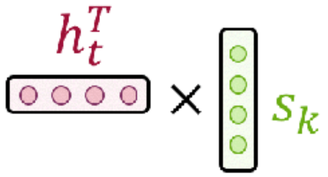
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

How to Compute Attention Scores

The most popular ways to compute attention scores are:

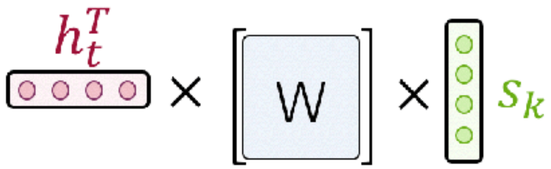
- **dot-product** - the simplest method;
- **bilinear function** (aka "Luong attention") - used in the paper Effective Approaches to Attention-based Neural Machine Translation;
- **multi-layer perceptron** (aka "Bahdanau attention") - the method proposed in the original paper.

Dot-product



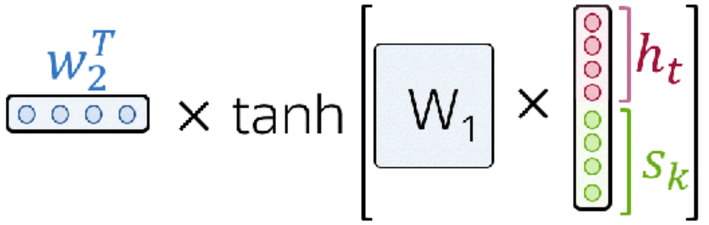
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



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

Multi-Layer Perceptron



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

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

Model Variants: Bahdanau and Luong

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

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

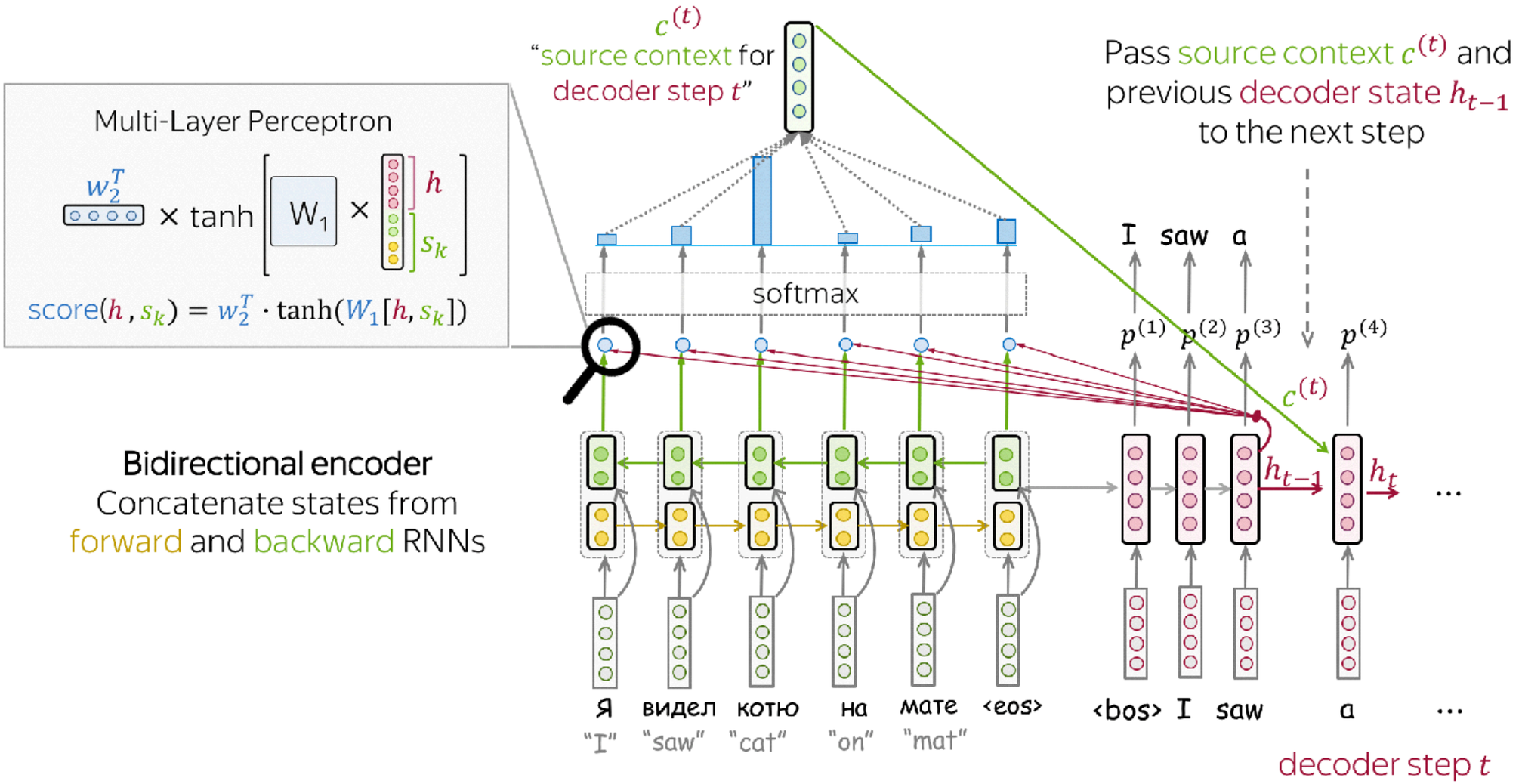
These may refer to either score functions of the whole models used in these papers. In this part, we will look more closely at these two model variants.

Bahdanau Model

- **encoder: bidirectional**
  To better encoder 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.
- **attention score: multi-layer perceptron**
  To get an attention score, apply a multi-layer perceptron (MLP) to an encoder state and a decoder state.
- **attention 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.

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

Bahdanau Model



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

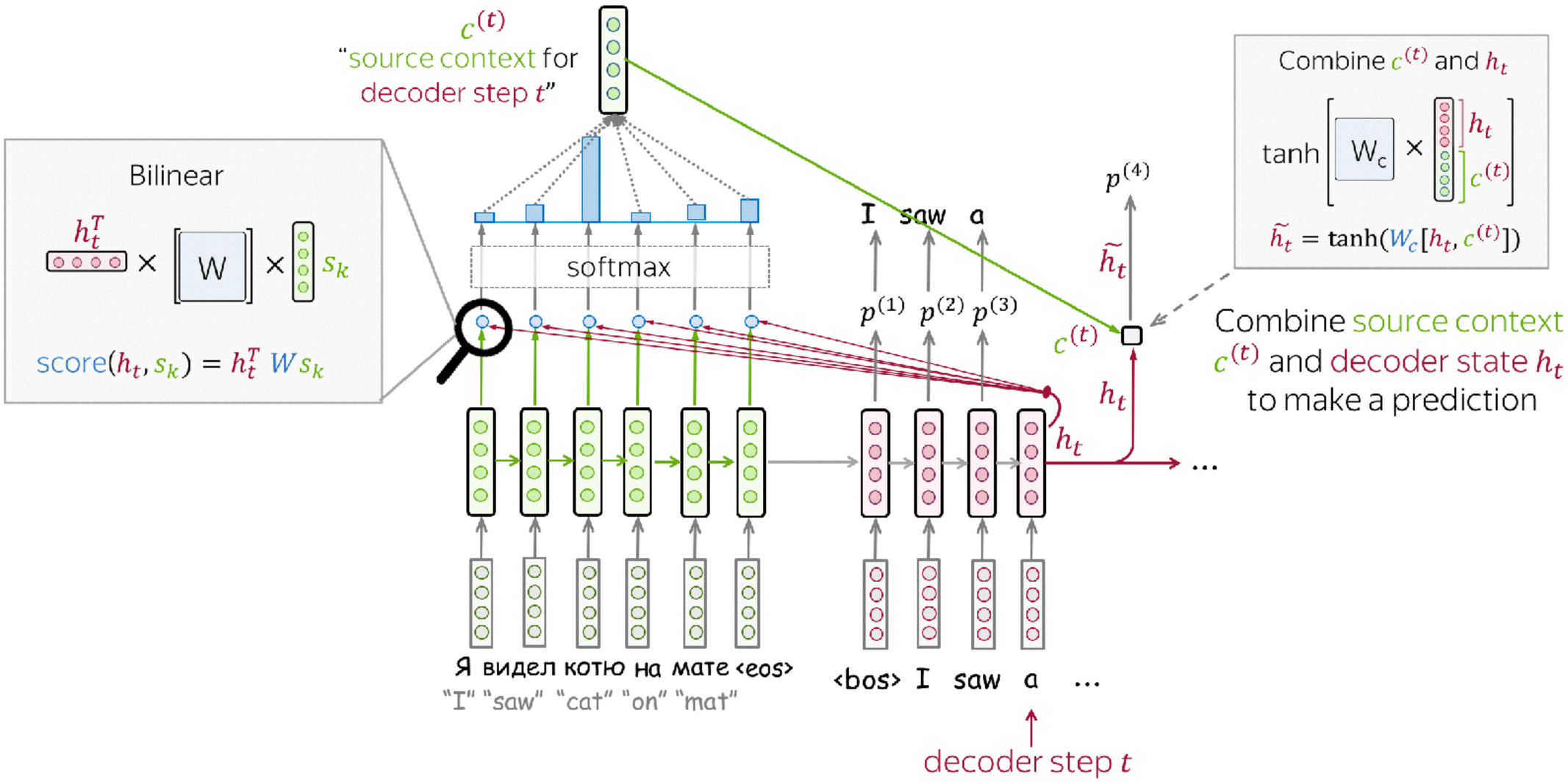
Luong Model

While the paper considers several model variants, the one which is usually called "Luong attention" is the following:

- **encoder: unirectional** (simple)
- **attention score: bilinear function**
- **attention applied: between decoder RNN state t 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.

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

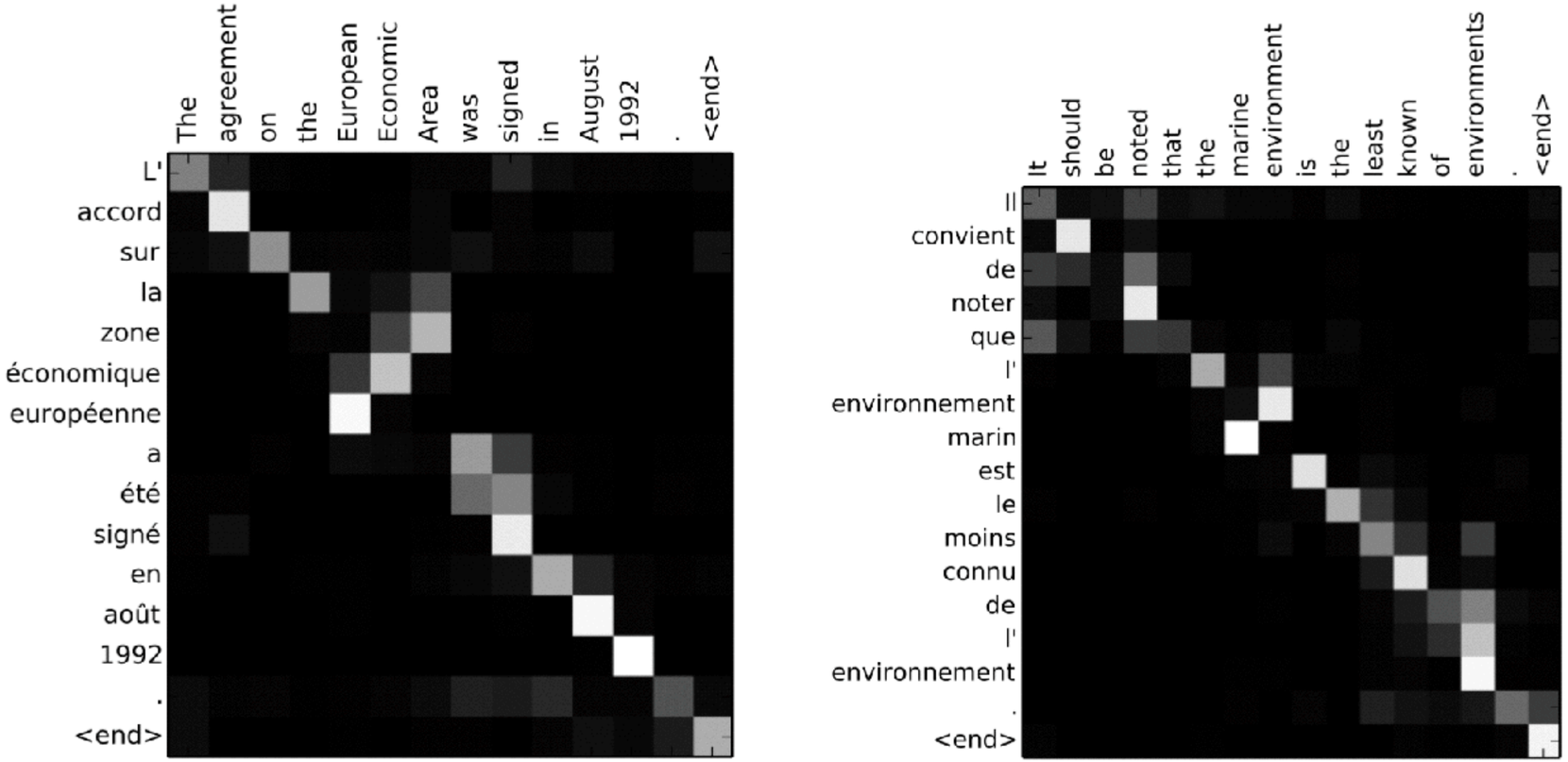
Luong Model



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

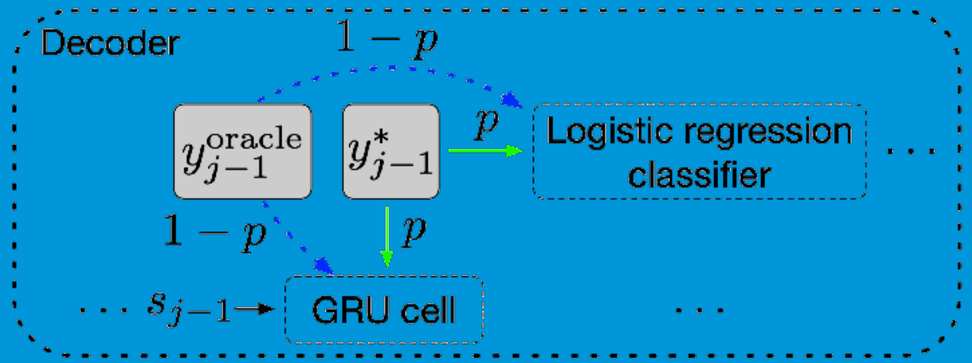
Attention Learns (Nearly) Alignments

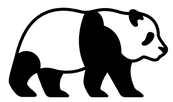
Remember the motivation for attention? At different steps, the decoder may need to focus different source tokens, the ones which are more relevant at this step. Let's look at attention weights - which source words does the decoder use?



The examples are from the paper Neural Machine Translation by Jointly Learning to Align and Translate.

Bridging the Gap between Training and Inference for NMT

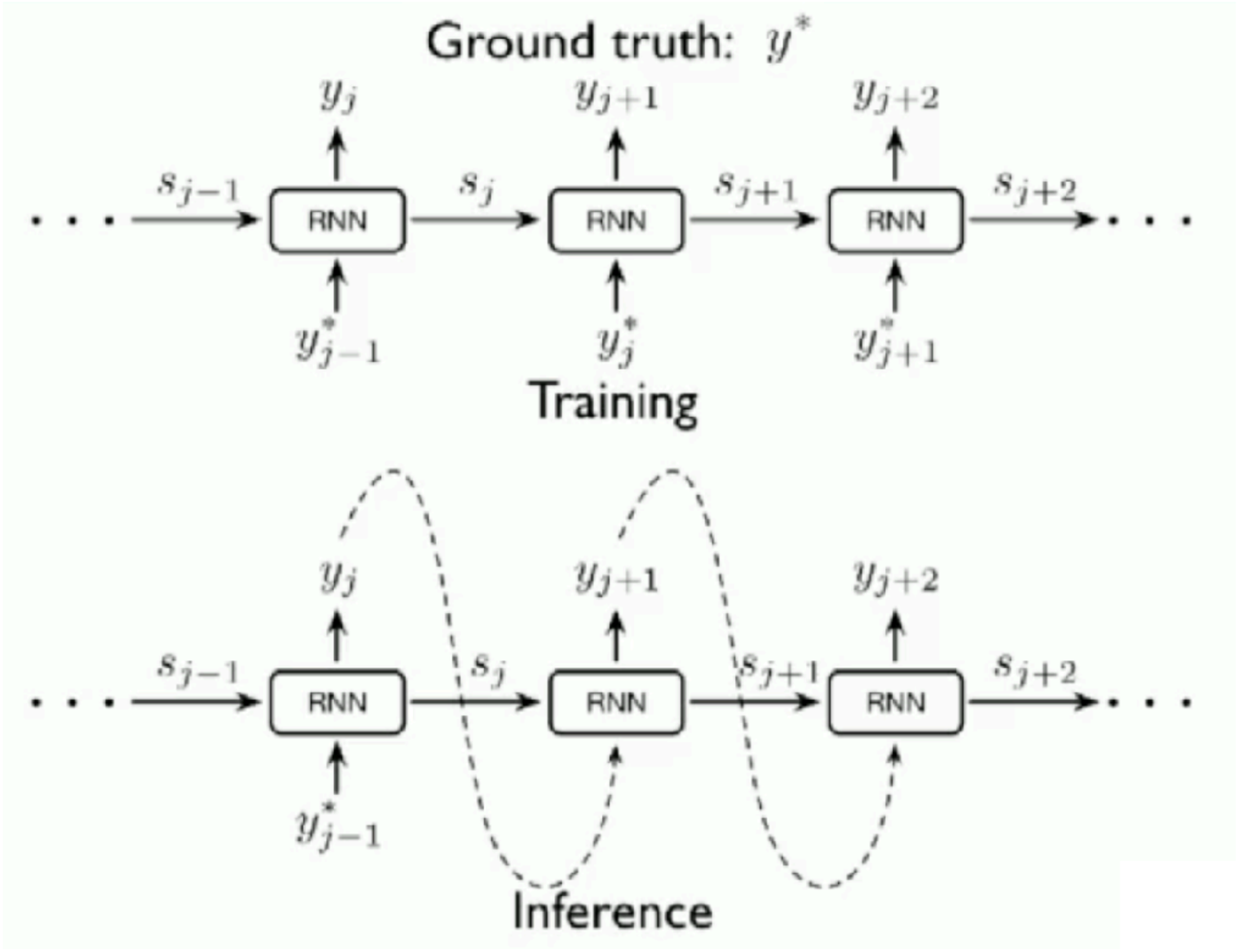




Problem: Exposure Bias

Exposure Bias

- During Training, found truth words as context
- At inference, self-generated words as context



Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Problem: Overcorrection

reference:	We should comply with the rule.
cand1:	We should abide with the rule.
cand2:	We should abide by the law.
cand3:	We should abide by the rule.

Overcorrection

- If only use self-generated words as context

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Problem: Overcorrection

reference:	We should comply with the rule.
cand1:	We should abide with the rule.
cand2:	We should abide by the law.
cand3:	We should abide by the rule.

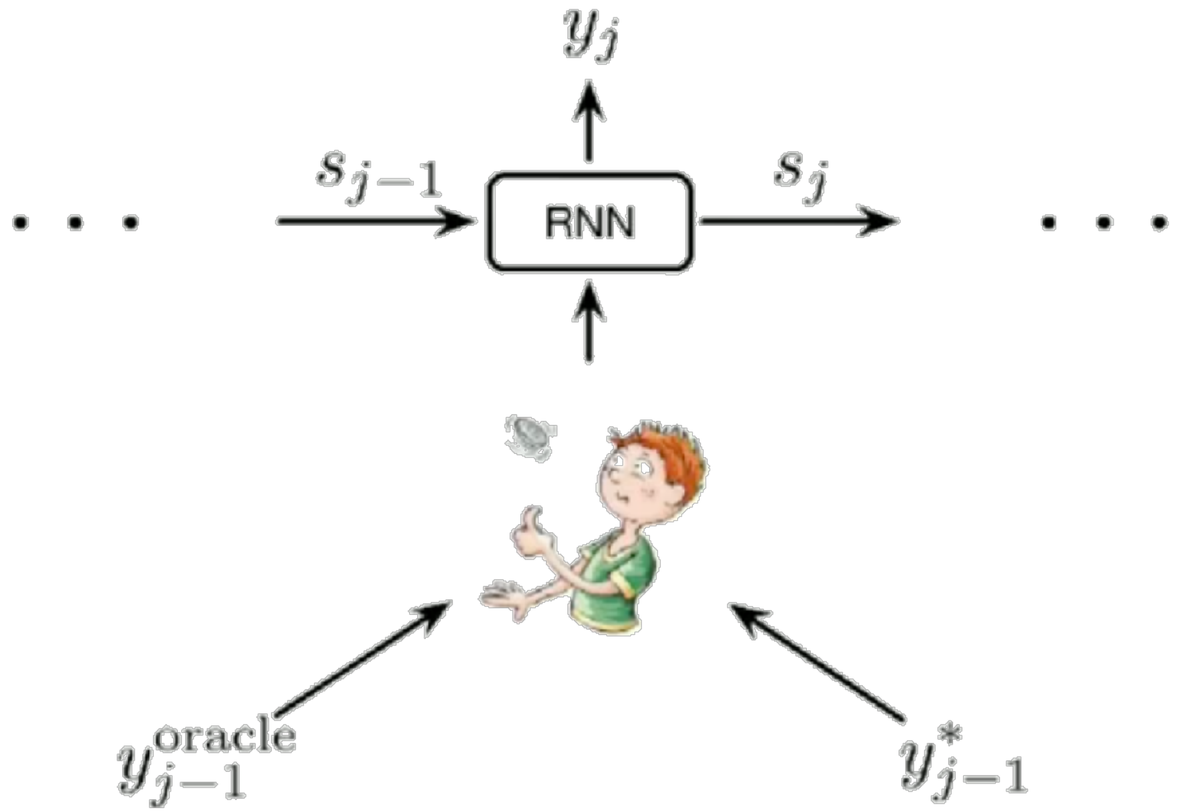
Overcorrection

- If only use self-generated words as context

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Solution

- Feed as context either the ground truth words **or the previous predicted words**, i.e. oracle words, with a certain probability.
- This potentially can **reduce the gap between training and inference** by training the model to handle the situation which will appear during test time.



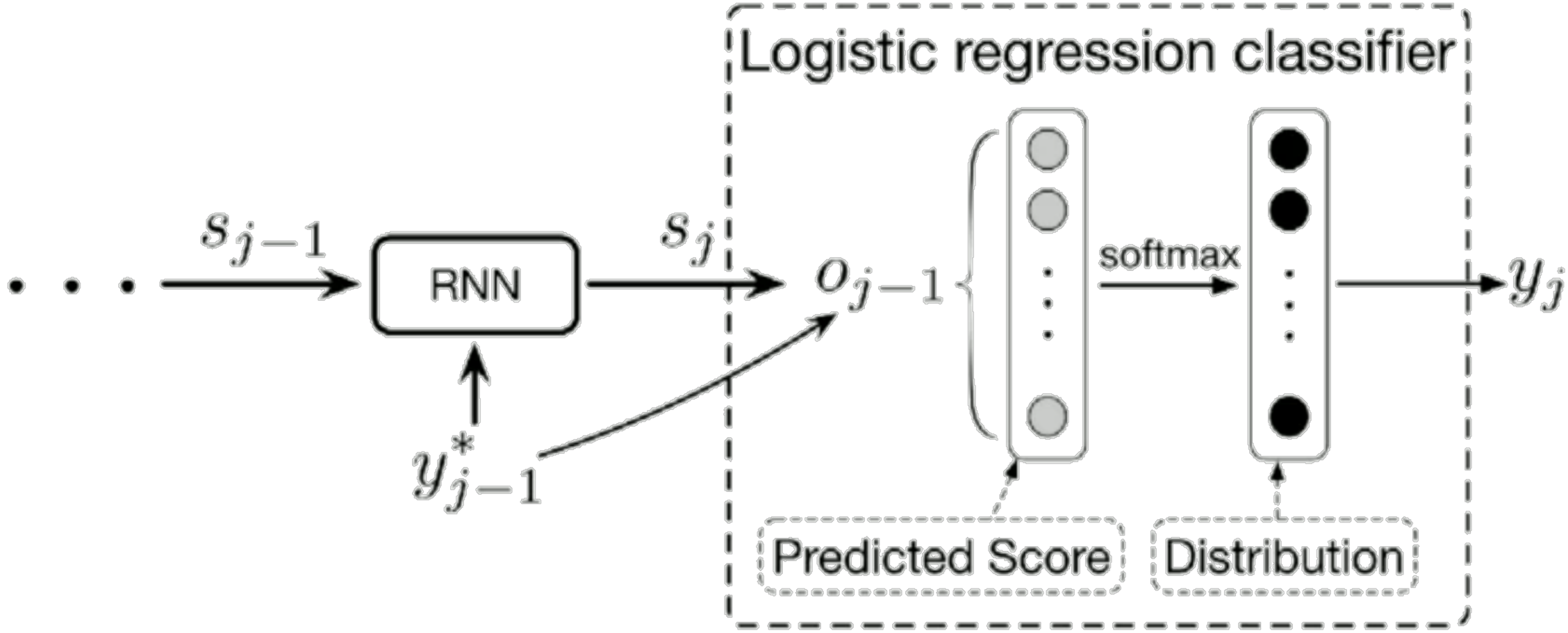
Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Key Parts

- How to generate oracle translation?
- How to sample context?
- How to train the model?

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

The RNNSearch Model



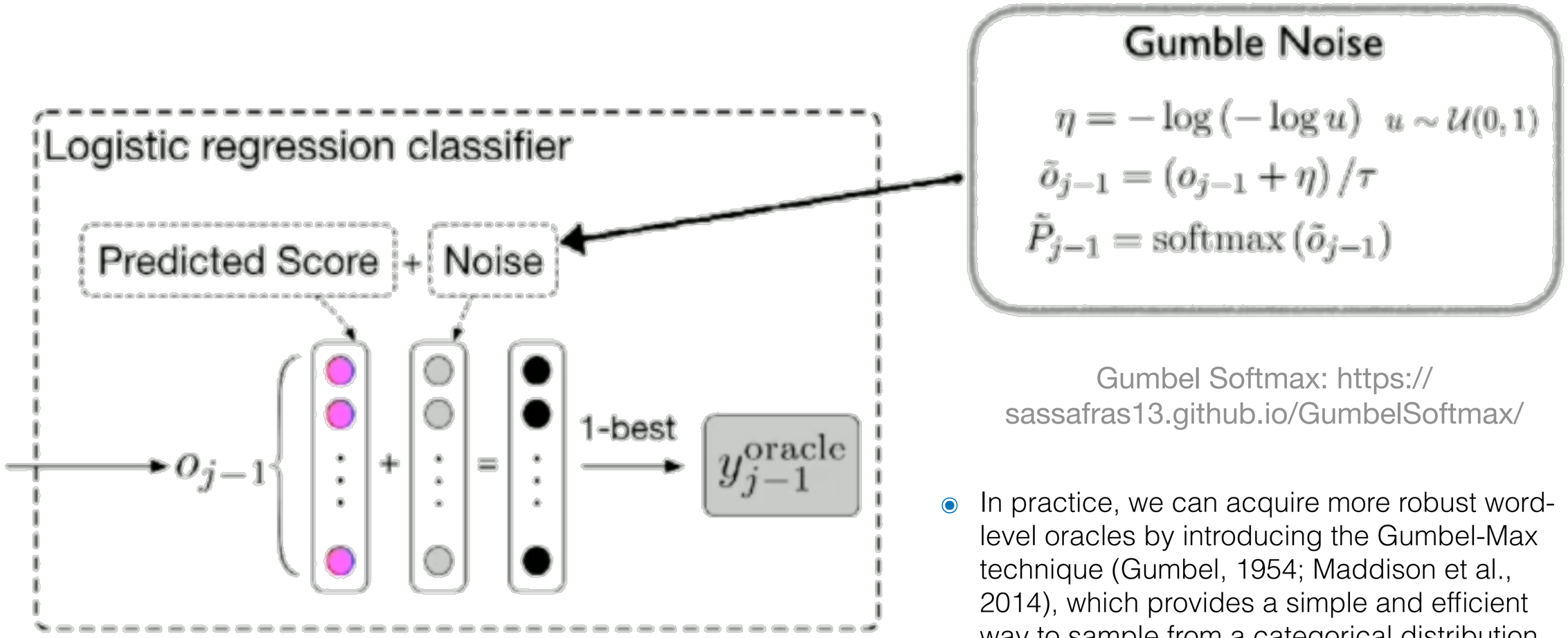
Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Oracle Translation Generation

- Word-level Oracle (WO)
- Sentence-level Oracle (SO)

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Word-level Oracle



Gumbel Softmax: https://sassafras13.github.io/GumbelSoftmax/

- In practice, we can acquire more robust word-level oracles by introducing the Gumbel-Max technique (Gumbel, 1954; Maddison et al., 2014), which provides a simple and efficient way to sample from a categorical distribution.

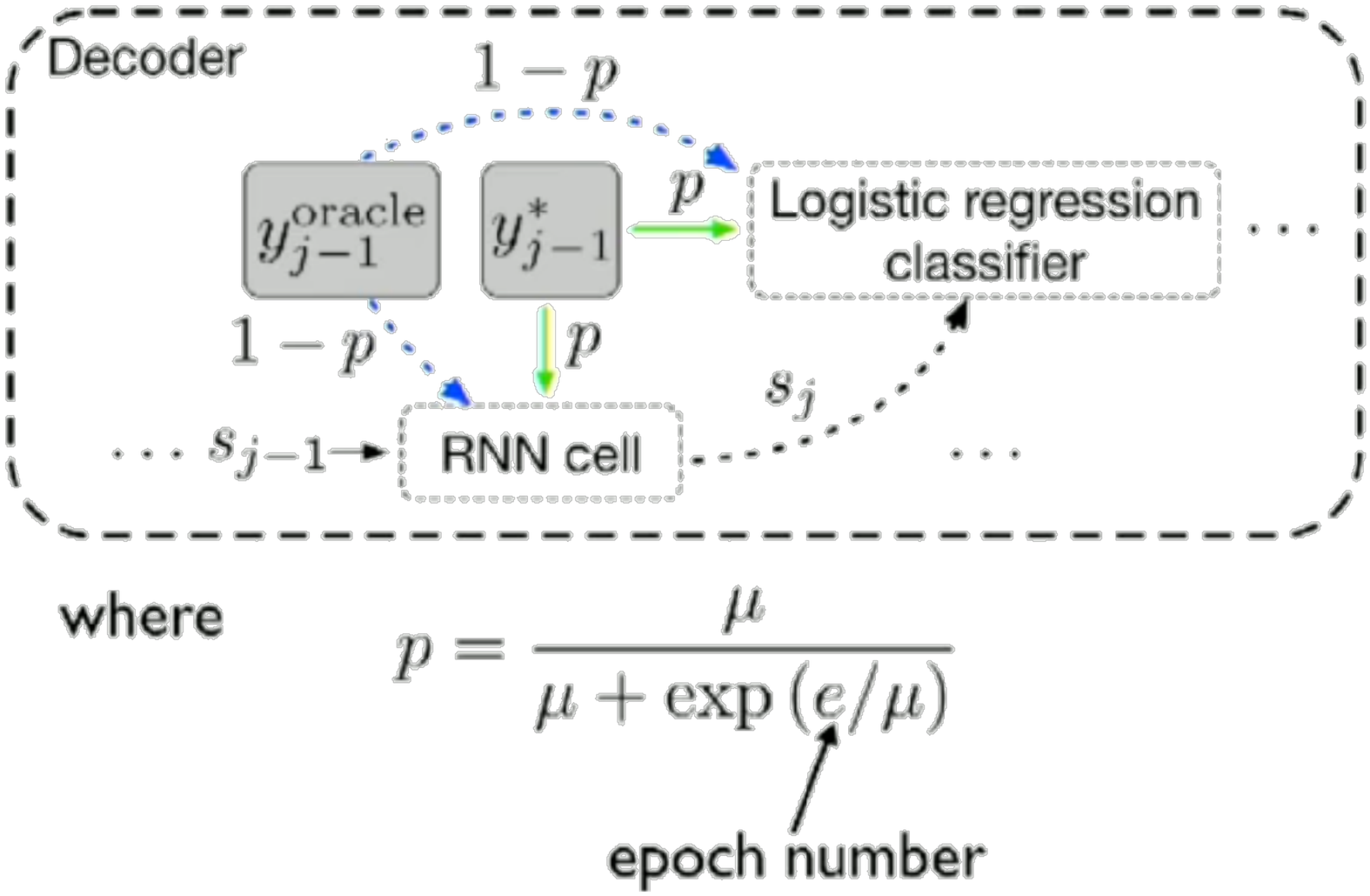
Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Sentence-level Oracle

- **Sentence-level Oracle (SO)**
  - Generate the top-k translation by beam search
  - Rerank the top-k translation with BLEU
  - Select the top 1

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Context Sampling with Decay



where

$$p = \frac{\mu}{\mu + \exp\left(e/\mu\right)}$$

epoch number

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Training Loss

$$\mathcal{L}(\theta) = -\sum_{n=1}^{N} \sum_{j=1}^{|\mathbf{y}^n|} \log P_j^n \left[y_j^{*n} \right]$$

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Experiment: Chinese-English Translation

Systems	Average
RNNsearch	37.73
+ word oracle	38.94
+ noise	39.50
+ sentence oracle	39.56
+ noise	**40.09**

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

More Experiments

Systems	Architecture	MT03	MT04	MT05	MT06	Average
Existing end-to-end NMT systems						
Tu et al. (2016)	Coverage	33.69	38.05	35.01	34.83	35.40
Shen et al. (2016)	MRT	37.41	39.87	37.45	36.80	37.88
Zhang et al. (2017)	Distortion	37.93	40.40	36.81	35.77	37.73
Our end-to-end NMT systems						
this work	RNNsearch	37.93	40.53	36.65	35.80	37.73
	+ SS-NMT	38.82	41.68	37.28	37.98	38.94
	+ MIXER	38.70	40.81	37.59	38.38	38.87
	+ OR-NMT	**40.40**‡†*	**42.63**‡†*	**38.87**‡†*	**38.44**‡	**40.09**
	Transformer	46.89	47.88	47.40	46.66	47.21
	+ word oracle	47.42	48.34	47.89	47.34	47.75
	+ sentence oracle	**48.31***	**49.40***	**48.72***	**48.45***	**48.72**

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Conclusions of This Work

- Sampling as context from the ground truth and the generated oracle can mitigate exposure bias.
- The sentence-level oracle is better than the word-level oracle.
- Gumbel noise can help improve translation quality.

Wen et al. Bridging the Gap between Training and Inference for Neural Machine Translation, ACL 2019 (Best paper award)

Todo

- **Suggested Readings:**
  - Sequence to Sequence Learning with Neural Networks (original seq2seq NMT paper)
  - Neural Machine Translation by Jointly Learning to Align and Translate (original seq2seq+attention paper)
  - Massive Exploration of Neural Machine Translation Architectures (practical advice for hyperparameter choices)
  - Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
  - Bridging the Gap between Training and Inference for Neural Machine Translation
  - A Theoretical Analysis of the Repetition Problem in Text Generation

Next lecture: Transformer and BERT

Todo: Mini-Lectures

- **Mini-Lectures** related to Pre-trained Language Models
- **Each team presents a paper** about this topic (about **18 minutes**)
- Accounts for **3 points** (equivalent to 3 reading assignments)
- Candidate paper list has been released. Bid your paper choice early!
- **Let us know each other better!**

	Category	A.K.A.	Paper
2	General		Paradigm Shift in Natural Language Processing
3			
4	Pretrained Models	RoBERTa	RoBERTa: A Robustly Optimized BERT Pretraining Approach
5		GPT2	Language Models are Unsupervised Multitask Learners
6		GPT3	Language Models are Few-Shot Learners
7		XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding
8		ELECTRA	ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators
9		BART	BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
10		T5	Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
11		ERNIE 3.0	ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation
12		UniLMv2	UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training
13			
14	Multilingual		Cross-Lingual Ability of Multilingual BERT: An Empirical Study
15		XLM-R	Unsupervised Cross-lingual Representation Learning at Scale
16			
17	Multimodality	CLIP	Learning Transferable Visual Models From Natural Language Supervision
18		data2vec	data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language
19			
20	Understanding BERT	LAMA	Language Models as Knowledge Bases?
21			What Does BERT Look At? An Analysis of BERT's Attention
22			What do you learn from context? Probing for sentence structure in contextualized word representations
23			What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models
24			
25	Fine-tuning		Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping
26		DAPT/TAPT	Don't Stop Pretraining: Adapt Language Models to Domains and Tasks
27			
28	Tuning with Prompt	LPAQA	How Can We Know What Language Models Know?
29		PET	It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners
30		Prefix-Tuning	Prefix-Tuning: Optimizing Continuous Prompts for Generation
31		Prompt-Tuning	The Power of Scale for Parameter-Efficient Prompt Tuning
32		LM-BFF	Making Pre-trained Language Models Better Few-shot Learners
33		AutoPrompt	AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts
34		FLAN	Finetuned Language Models are Zero-Shot Learners
35			
36	AI + X with NLP-based ideas	AlphaCode	Competition-Level Code Generation with AlphaCode
37		AlphaFold2	Highly accurate protein structure prediction with AlphaFold

https://docs.google.com/spreadsheets/d/1Zu1xzbQzVybStiKfOZiam9oIVEhlzsPf_-Z_097swQ0/edit#gid=0

References

1. https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html
2. **Stanford CS224N, Spring 2020:** http://web.stanford.edu/class/cs224n/, lecture 8
3. **University of Maryland, CMSC 470**: http://www.cs.umd.edu/class/fall2018/cmsc470//lectures/ lecture 18
4. https://vimeo.com/385255721
5. https://vas3k.com/blog/machine_translation/

Thanks! Q&A

Bang Liu
Email: bang.liu@umontreal.ca
Homepage: http://www-labs.iro.umontreal.ca/~liubang/