

Глубокое обучение

Бекезин Никита

28 мая 2022

Занятие 8: Быстрое введение в Seq2Seq + Attention

Agenda

- Machine Translation до 2014 (SMT)
- Seq2seq
- Attention
- ELMO

Statistical Machine Translation

1990s-2010s: Statistical Machine Translation

- Core idea: Learn a **probabilistic model** from data
- Suppose we're translating French → English.
- We want to find **best English sentence y , given French sentence x**

$$\operatorname{argmax}_y P(y|x)$$

- Use Bayes Rule to break this down into **two components to be learnt separately**:

$$= \operatorname{argmax}_y P(x|y)P(y)$$



Learning alignment for SMT

- Question: How to learn translation model $P(x|y)$ from the parallel corpus?
- Break it down further: we actually want to consider

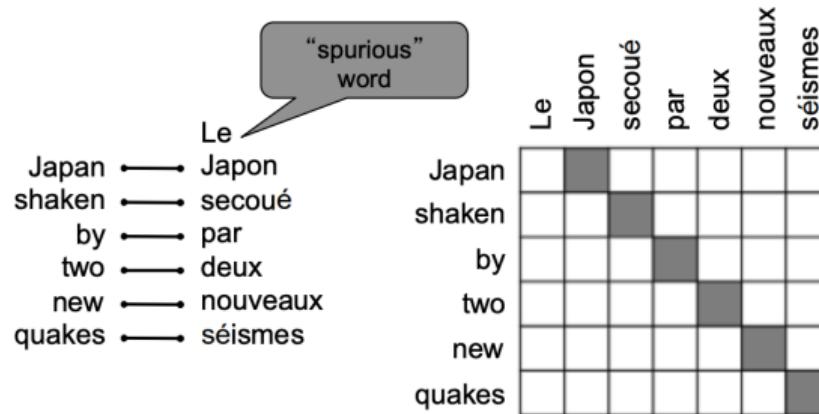
$$P(x, a|y)$$

where a is the **alignment**, i.e. word-level correspondence between French sentence x and English sentence y

What is alignment?

Alignment is the correspondence between particular words in the translated sentence pair.

- Note: Some words have no counterpart



1990s-2010s: Statistical Machine Translation

- SMT was a huge research field
- The best systems were extremely complex
 - Hundreds of important details we haven't mentioned here
 - Systems had many separately-designed subcomponents
 - Lots of feature engineering
 - Need to design features to capture particular language phenomena
 - Require compiling and maintaining extra resources
 - Like tables of equivalent phrases
 - Lots of human effort to maintain
 - Repeated effort for each language pair!

Neural Machine Translation: Seq2seq

Seq2seq

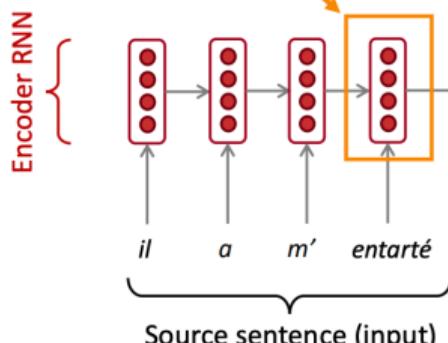
Seq2seq - это задача, в которой мы хотим предсказать по одной последовательности другую

Самая стандартная подобная задача - машинный перевод. Нейронные сети ворвались в эту сферу человеческого прогресса в 2014 году

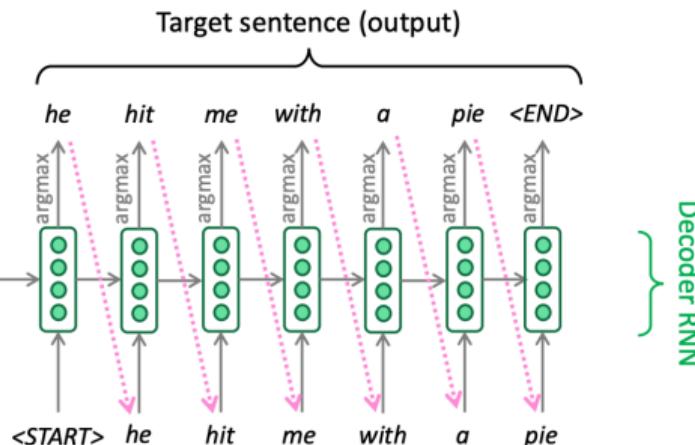
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.



Encoder RNN produces
an encoding of the
source sentence.



Decoder RNN is a Language Model that generates
target sentence, *conditioned on encoding*.

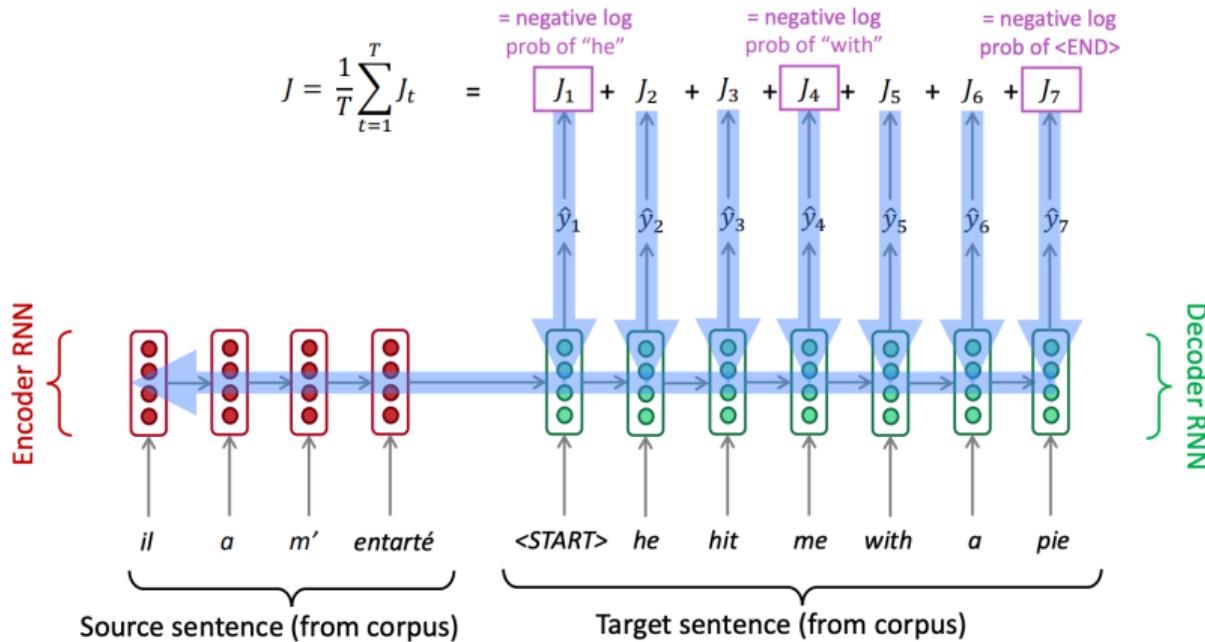
Note: This diagram shows test time behavior:
decoder output is fed in as next step's input

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Sequence-to-sequence is versatile!

- Sequence-to-sequence is useful for *more than just MT*
- Many NLP tasks can be phrased as sequence-to-sequence:
 - **Summarization** (long text → short text)
 - **Dialogue** (previous utterances → next utterance)
 - **Parsing** (input text → output parse as sequence)
 - **Code generation** (natural language → Python code)

Training a Neural Machine Translation system

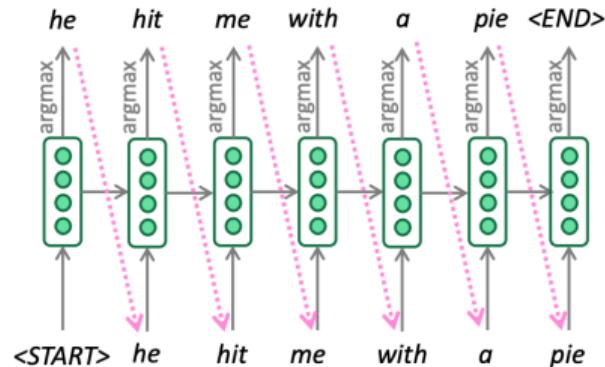


Seq2seq is optimized as a single system.
Backpropagation operates "end-to-end".

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is **greedy decoding** (take most probable word on each step)
- Problems with this method?**

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
 - Input: *il a m'entarté* (*he hit me with a pie*)
 - → *he* ____
 - → *he hit* ____
 - → *he hit a* ____ (whoops! no going back now...)
- How to fix this?

Beam search decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call *hypotheses*)
 - k is the beam size (in practice around 5 to 10)
- A hypothesis y_1, \dots, y_t has a score which is its log probability:

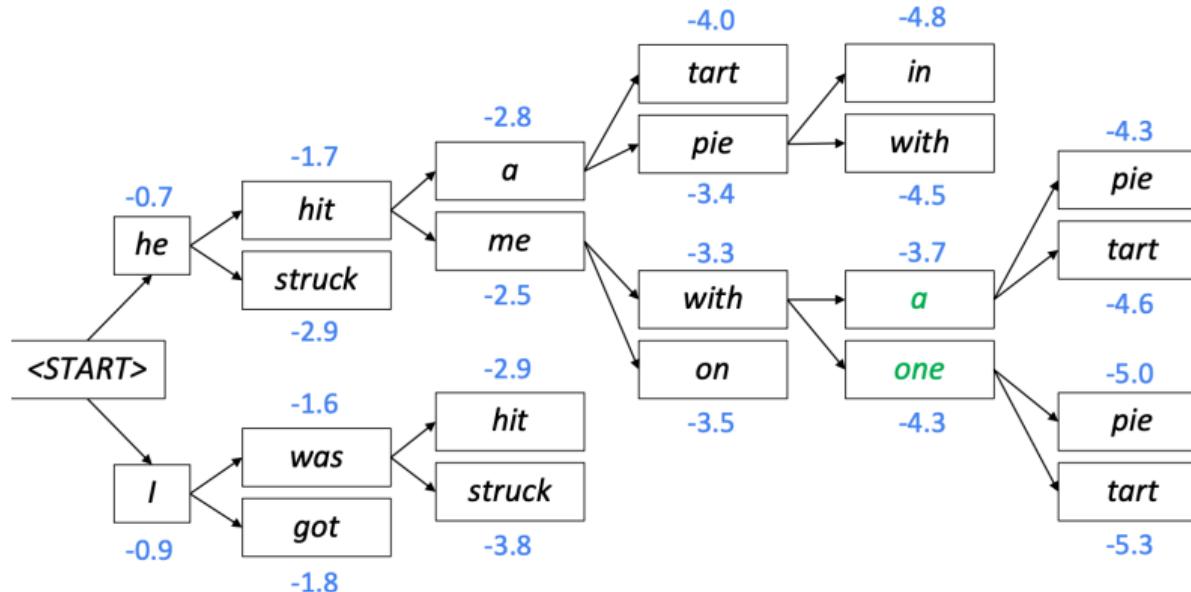
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- 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 much more efficient than exhaustive search!

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Beam search decoding: example

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



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

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces a **<END> token**
 - For example: *<START> he hit me with a pie <END>*
- In **beam search decoding**, different hypotheses may produce **<END> tokens on different timesteps**
 - When a hypothesis produces **<END>**, 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)

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?
- Each hypothesis y_1, \dots, y_t on our list has a score

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

- Problem with this: longer hypotheses have lower scores
- Fix: Normalize by length. Use this to select top one instead:

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

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

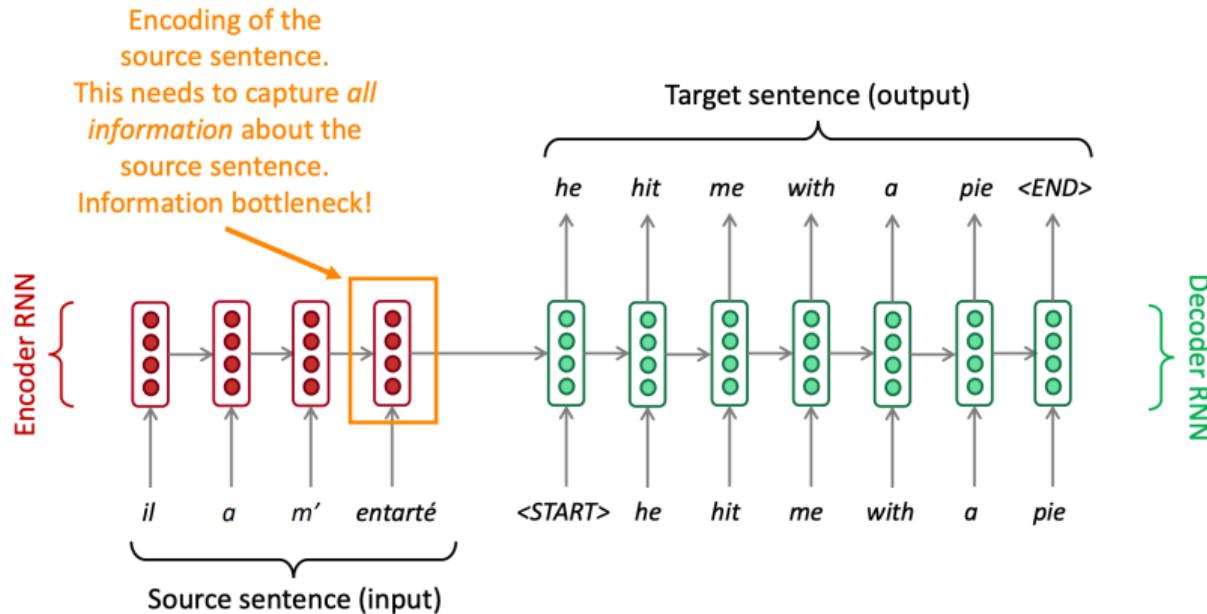
Attention!

Что можно улучшить в архитектуре Seq2seq?

Подсказка: Как связан decoder с encoder?

Attention

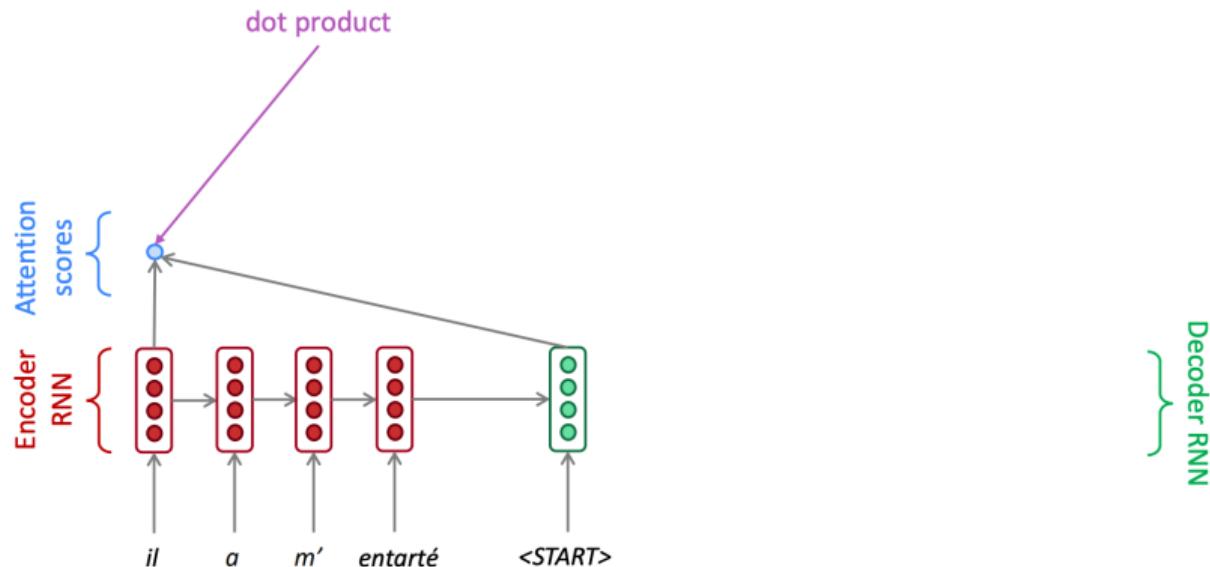
Sequence-to-sequence: the bottleneck problem



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

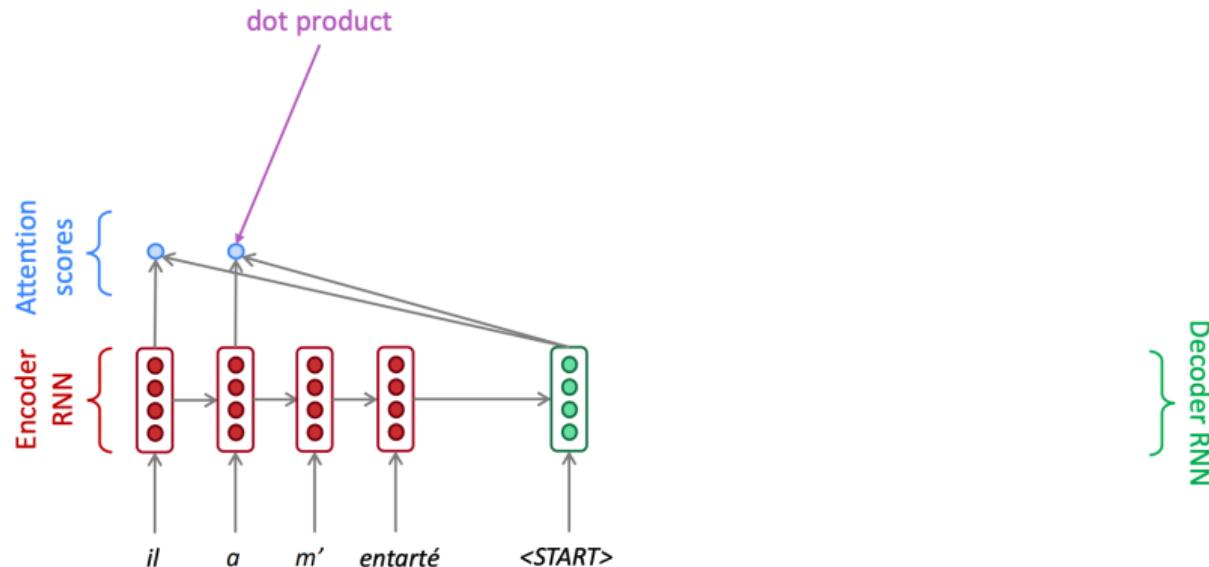
Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

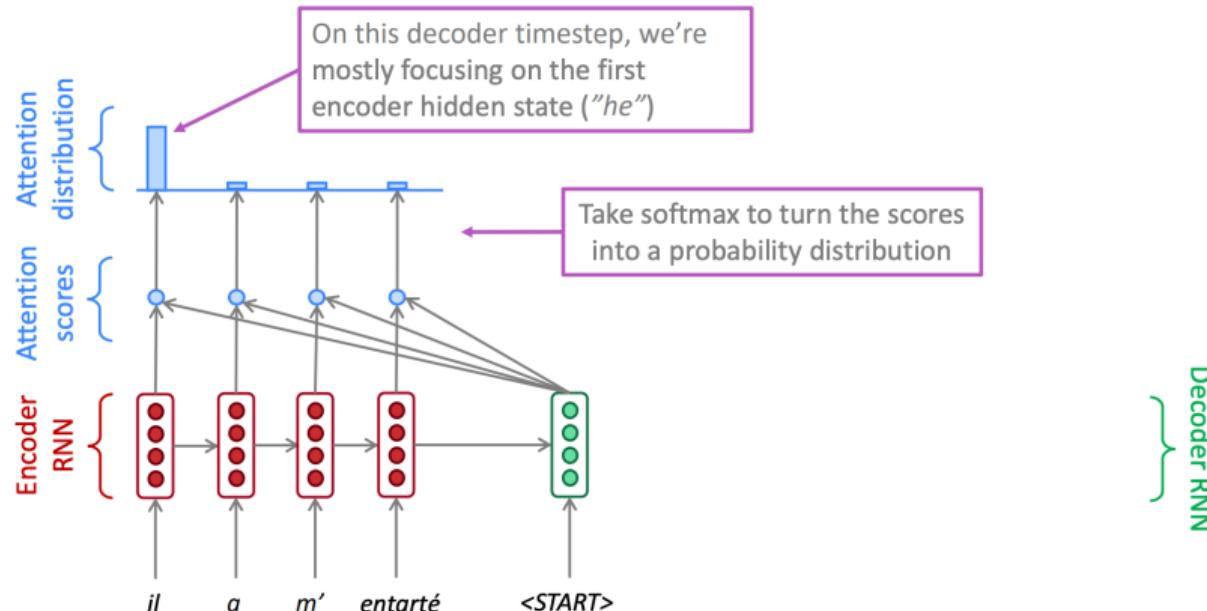
Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

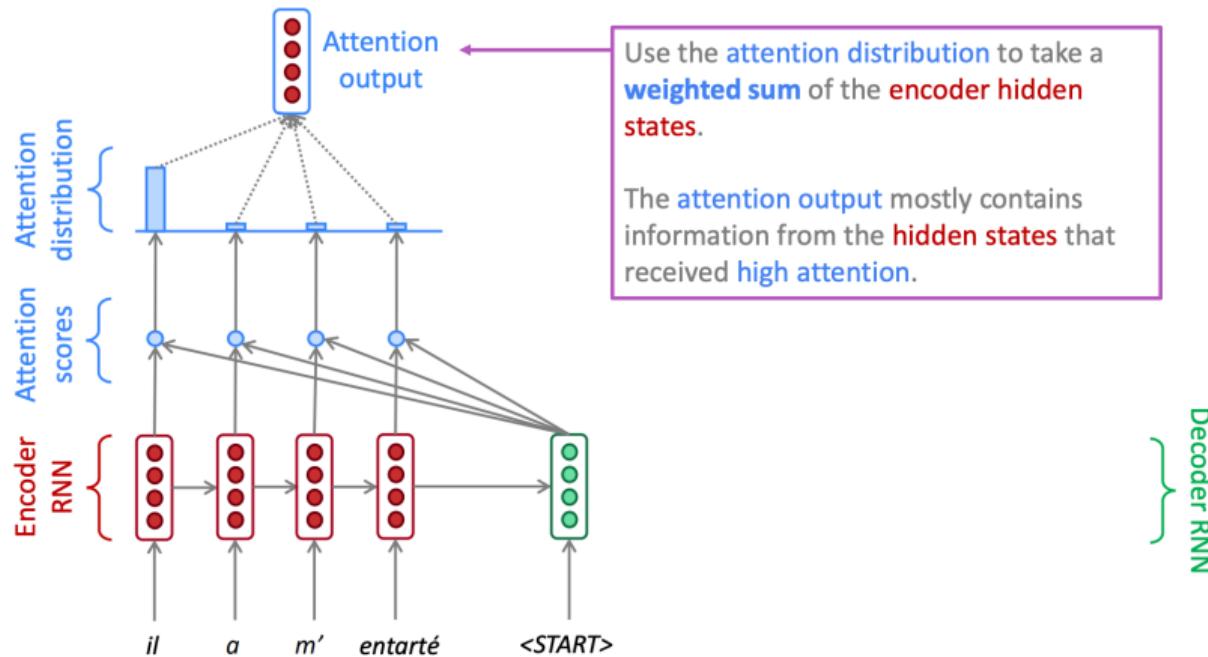
Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

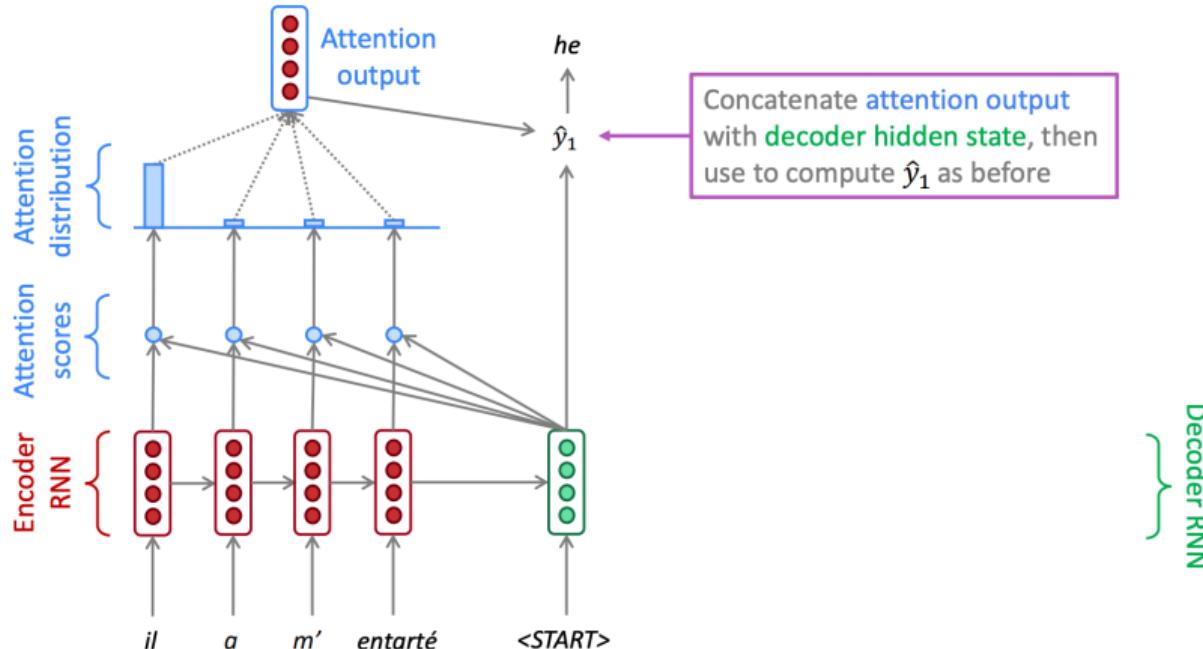
Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

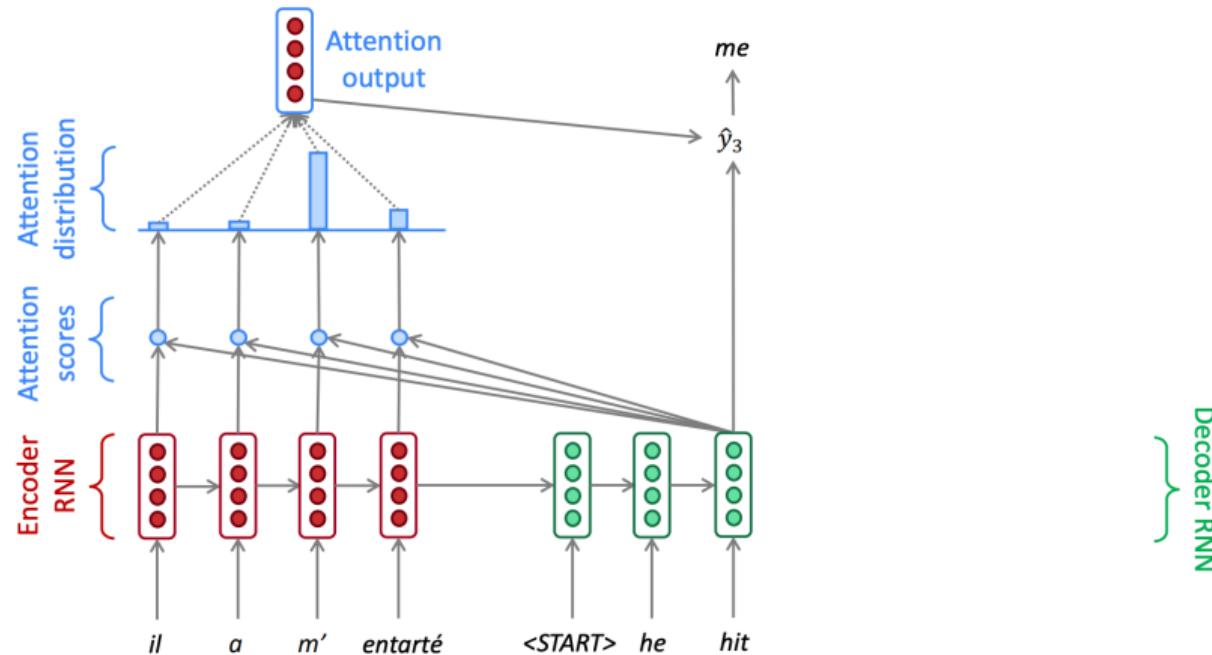
Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention

Sequence-to-sequence with attention



Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Интуиция за использованием Attention

- Взвешенная сумма - это выборочное выделение основной информации, хранящейся в словах исходного предложения. Выход с одной из ячеек декодера определяет на каких словах из исходного предложения нужно делать акцент.
- Attention - это способ получить представление о входных данных фиксированного размера из произвольного числа представлений в зависимости от рассматриваемого представления.

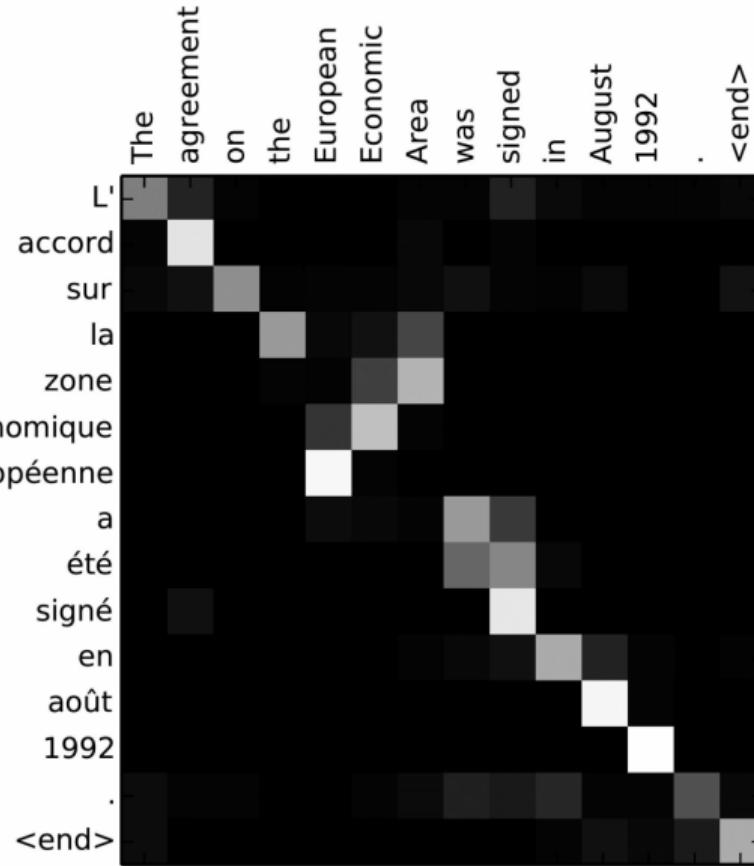
Вариации Attention

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

- Basic dot-product attention: $e_i = s^T \mathbf{h}_i \in \mathbb{R}$
 - Note: this assumes $d_1 = d_2$
 - This is the version we saw earlier
- Multiplicative attention: $e_i = s^T \mathbf{W} \mathbf{h}_i \in \mathbb{R}$
 - Where $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix
- Additive attention: $e_i = \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 s) \in \mathbb{R}$
 - Where $\mathbf{W}_1 \in \mathbb{R}^{d_3 \times d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\mathbf{v} \in \mathbb{R}^{d_3}$ is a weight vector.
 - d_3 (the attention dimensionality) is a hyperparameter

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

Attention



Как оценивать качество Машинного перевода?

- Модели в машинном переводе сравнивают по BLEU score
- BLEU - это метрика сравнения полученного машиной перевода и человеческого. Насколько хорошо перевод совпадает с истинным предложением.
- Из проблем данной метрики: если машина перевела корректно по смыслу, но не так, как заложено в таргете, то BLEU будет низкий

BLEU

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^n \text{precision}_i \right)^{1/n} \cdot 100\%$$

$$\text{brevity penalty} = \min \left(1, \frac{\text{output length}}{\text{reference length}} \right)$$

Источник: https://github.com/girafe-ai/ml-mipt/blob/master/week1_o3_Machine_Translation_and_Attention/ml-mipt_f20_lect103_Machine_Tranlation.pdf

BLEU

BLEU (Bilingual Evaluation Understudy) compares the machine-written translation to human-written translation, and computes a similarity score based on:

- n-gram precision
- brevity penalty

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

REFERENCE: Israeli officials are responsible for airport security

SYSTEM B: airport security Israeli officials are responsible
2-GRAM MATCH 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%

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^n \text{precision}_i \right)^{1/n} \cdot 100\%$$

Источник: https://github.com/girafe-ai/ml-mipt/blob/master/week1_o3_Machine_Translation_and_Attention/ml-mipt_f20_lect1o3_Machine_Tranlation.pdf

Краткий итог основной части лекции

1. Рассмотрели, что такое Machine translation
2. Познакомились с Seq2seq архитектурой
3. Поянли , что идея с вниманием является ключевой. С помощью нее можно тянуть информацию через всю последовательность токенов

Маленькая мысль в конце: мы разобрали сложные архитектуры на составляющие запчасти и разобрались с тем, как их соединить воедино. Современный моделист в нейронных сетях - это скорее инженер, нежели аналитик

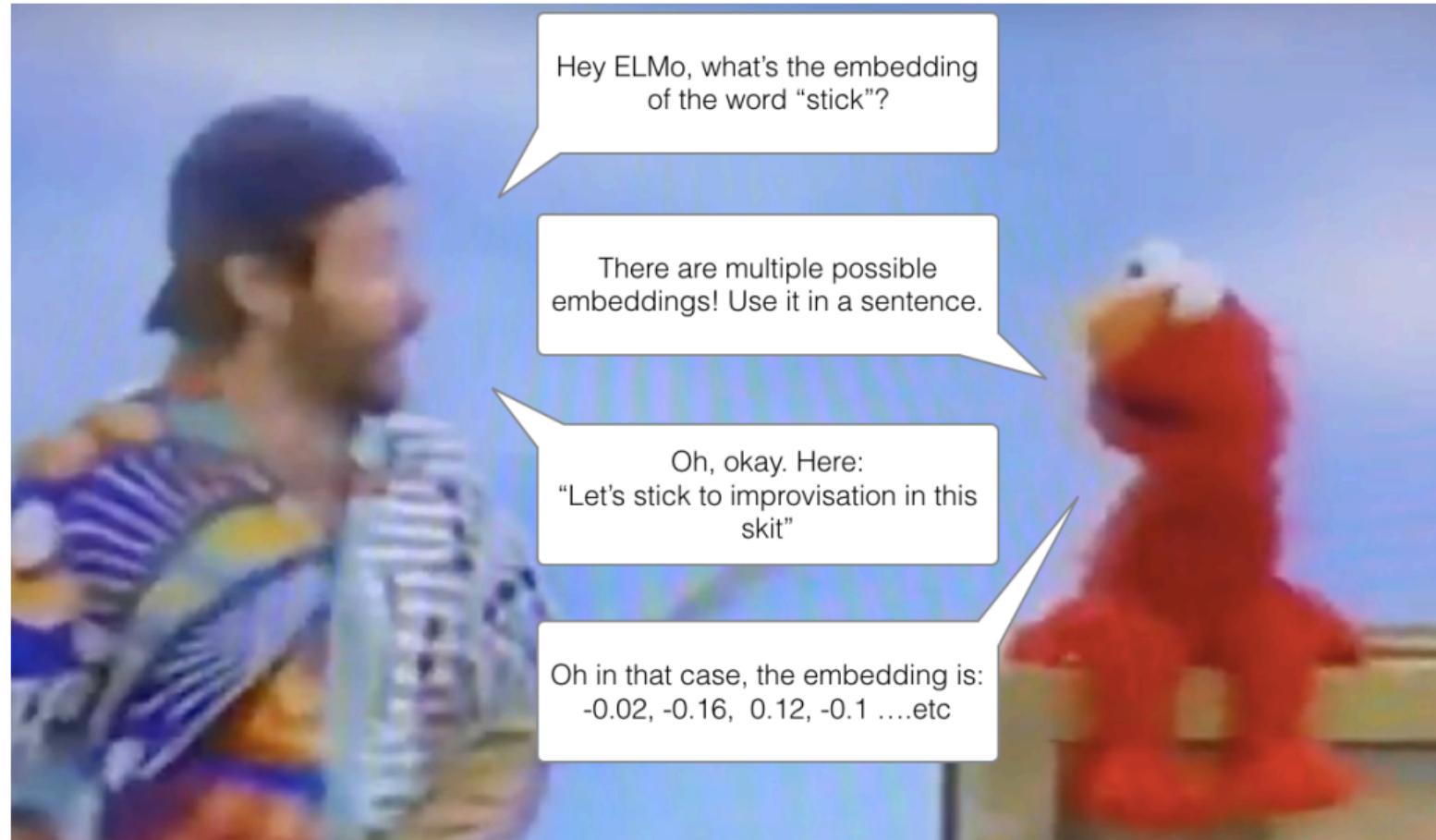
ELMO

Серия вопросов в зал

Как работают разные эмбединги?

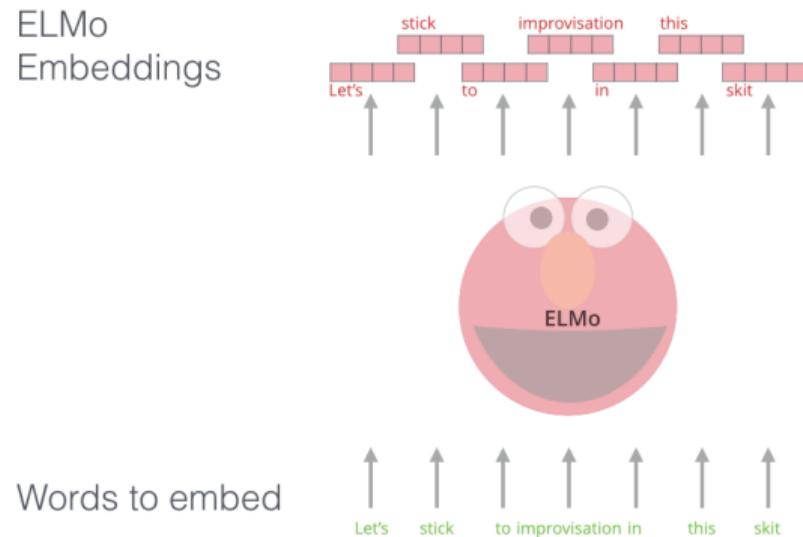
В чем, по вашему мнению, их главная проблема?

Картиночка про решение проблемы



ELMO

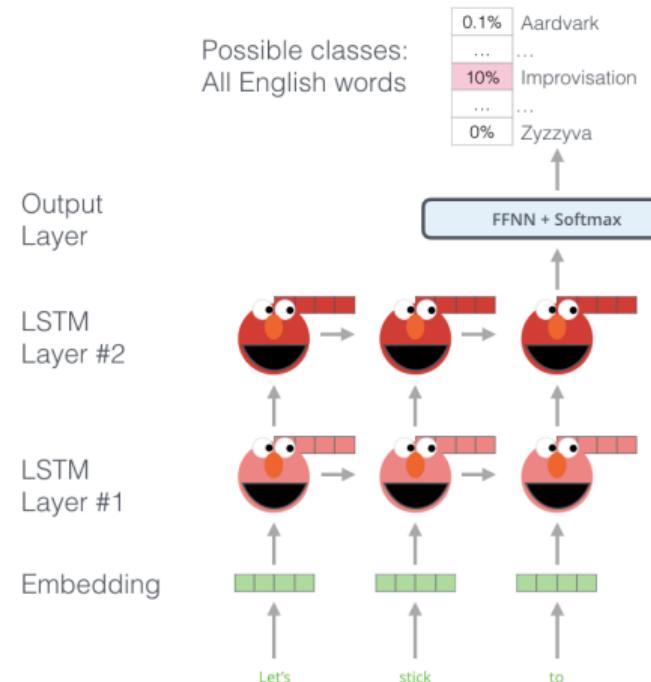
Захватываем контекст предложения через biderictional LSTM. Таким образом мы захватываем и контекст предложения (да, надо очень очень много данных, не обучайте это дома)



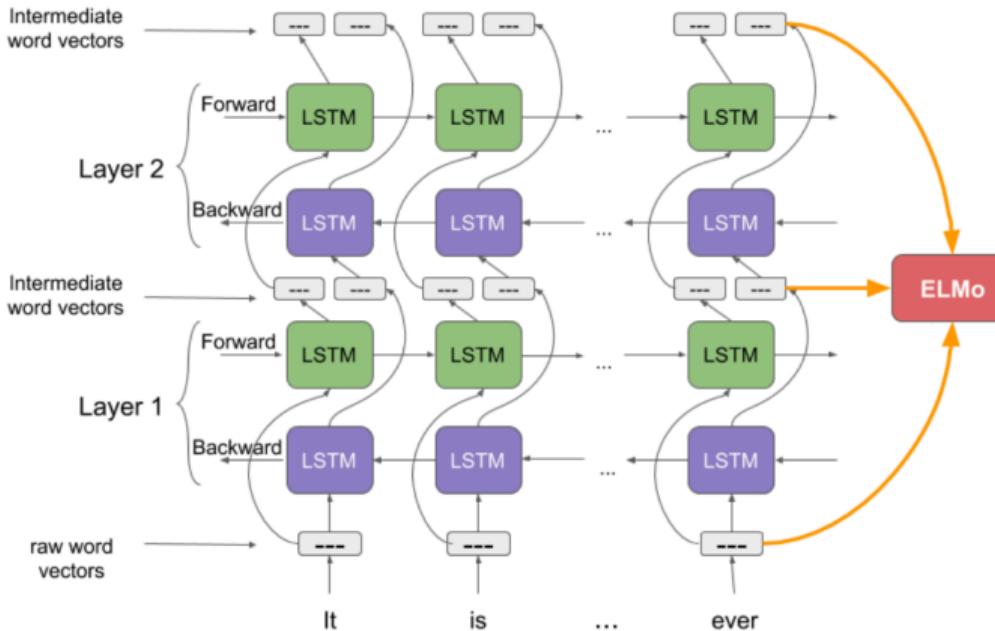
Рисунки на слайдах заимствованы [отсюда](#)

ELMO

Учится понимать язык ELMO следующим образом - оно берет большой дата сет и пытается предсказать следующее слово в предложении (Language Modeling task).



Архитектура ELMO



Source

ELMO

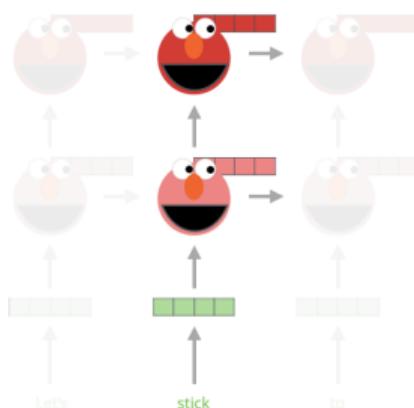
Применяем

Embedding of “stick” in “Let’s stick to” - Step #2

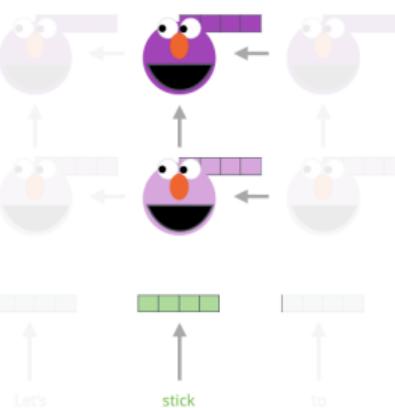
1- Concatenate hidden layers



Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of "stick" for this task in this context

Семинар