

Sequence to Sequence МОДЕЛИ И МЕХАНИЗМ ВНИМАНИЯ

Олег Шлякко
18 апреля 2018

План лекции

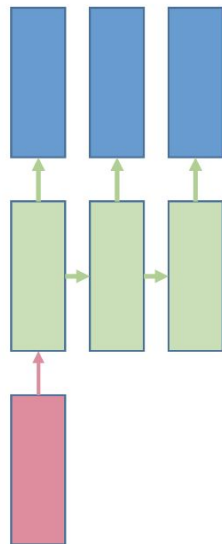
1. Задачи Sequence to Sequence
2. Архитектура энкодер-декодер
3. Механизм внимания
4. Tips & Tricks
5. **Разбор примера Machine Translation**

Рекуррентная нейросеть

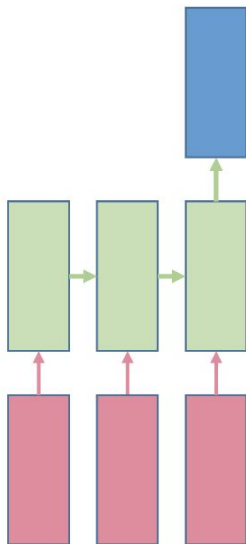
Один к одному



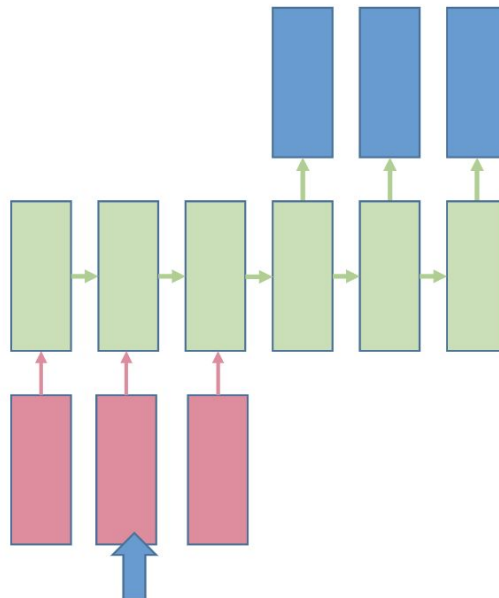
Один ко многим



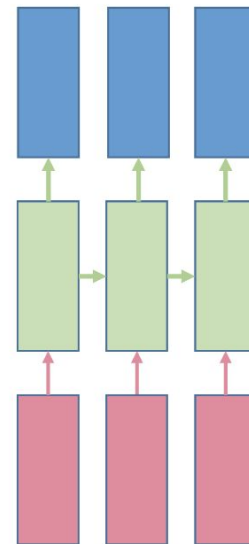
Многие к одному



Многие ко многим



Многие ко многим



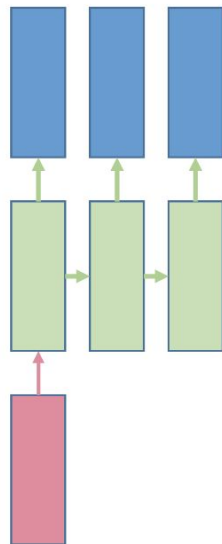
Переводчик, чат боты
Вход – текст. Выход - текст

Рекуррентная нейросеть

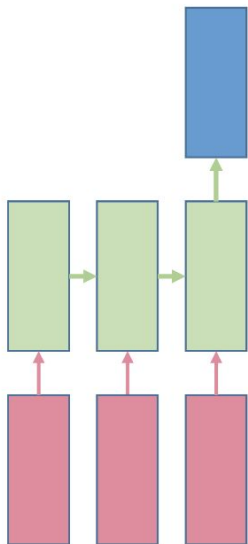
Один к одному



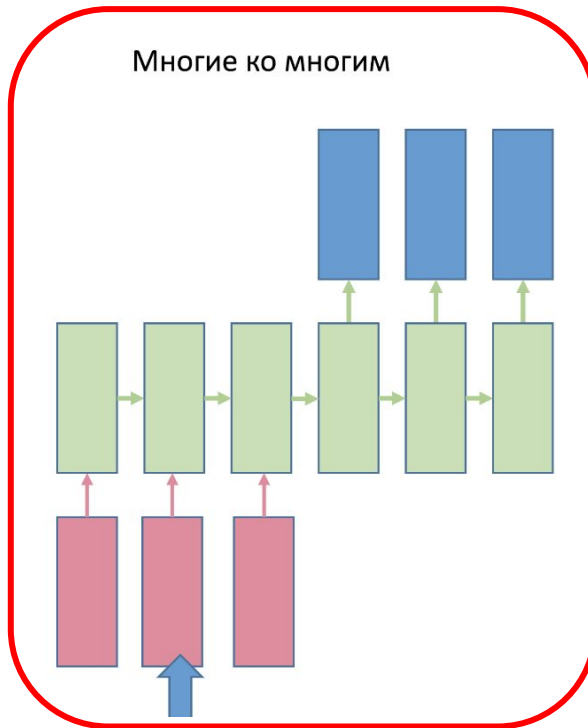
Один ко многим



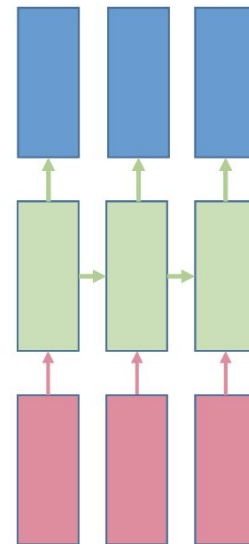
Многие к одному



Многие ко многим



Многие ко многим

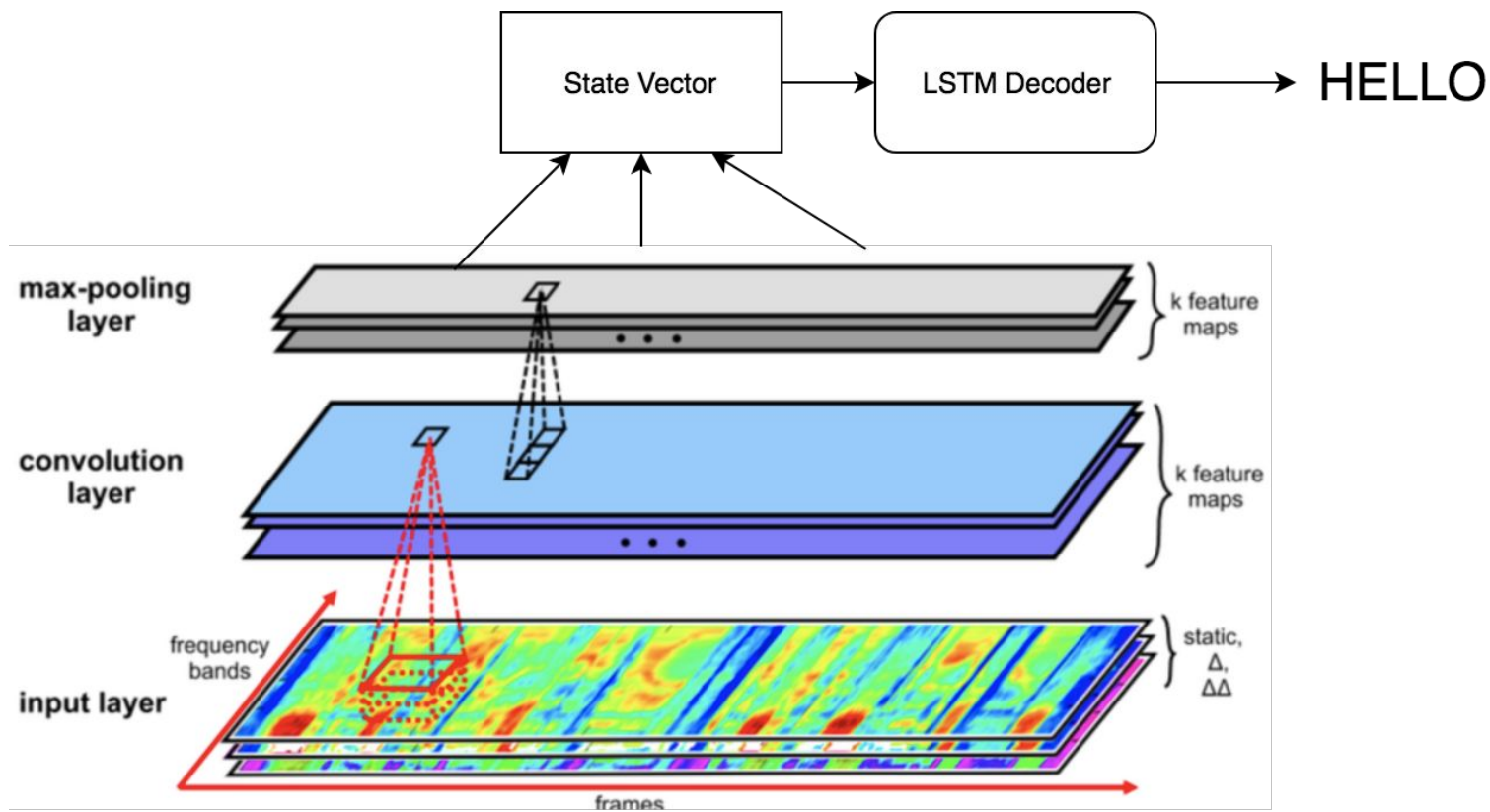


len(input) != len(output)

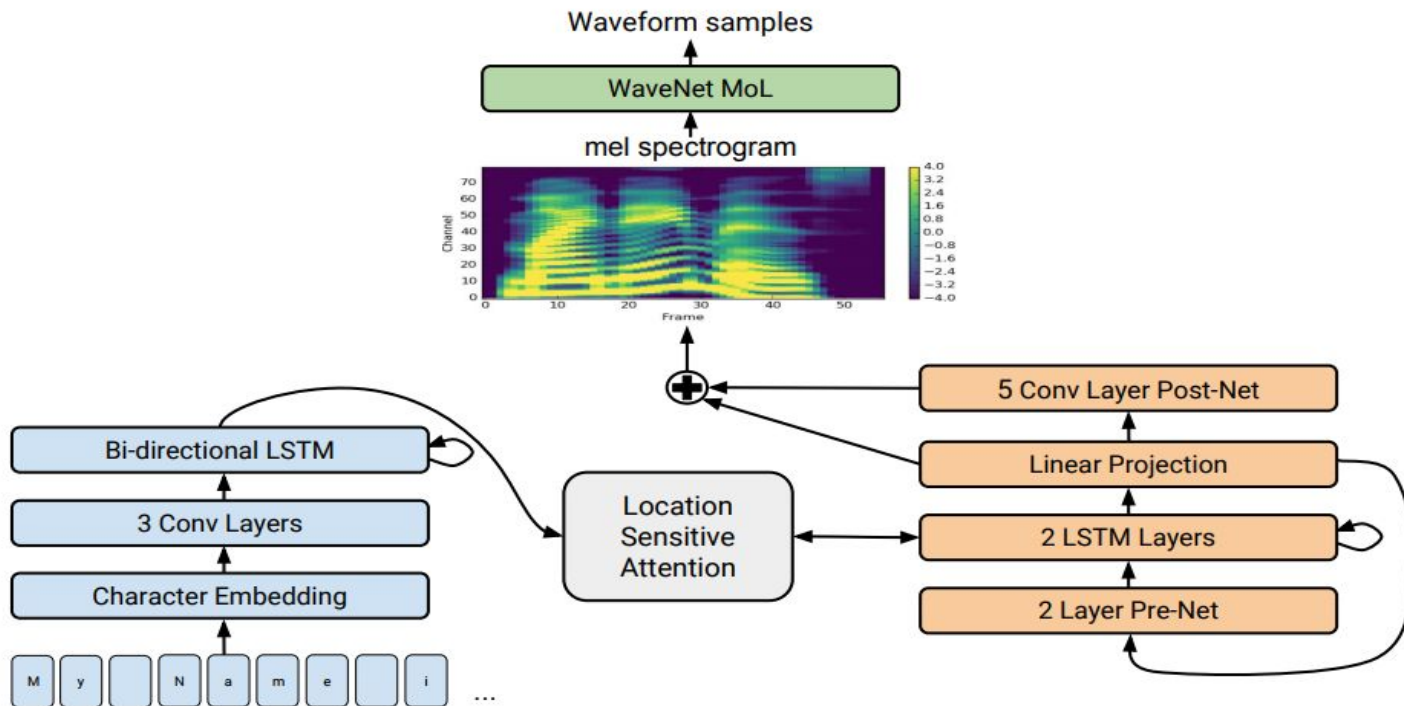
Задачи Sequence to Sequence

1. Распознавание речи (spectrum \rightarrow text)
2. Синтез речи (text \rightarrow waveform)
3. Рукописный ввод (image sequence \rightarrow text)
4. Машинный перевод (text \rightarrow text)
5. Чатботы (text \rightarrow text)
6. Суммаризация (text \rightarrow text)

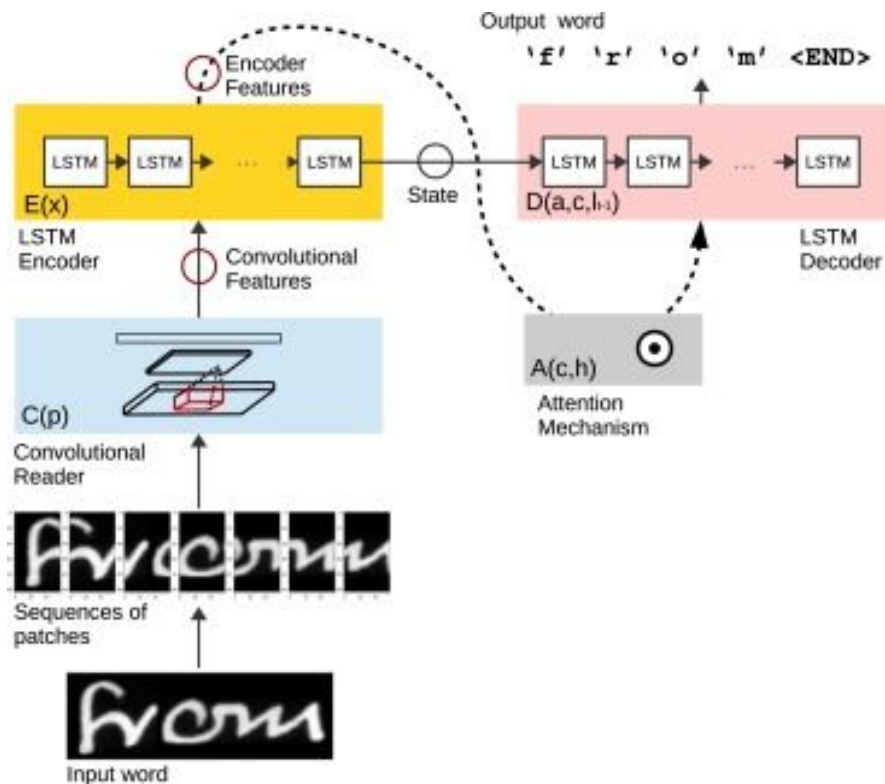
Speech recognition



Speech Synthesis



Рукописный ввод



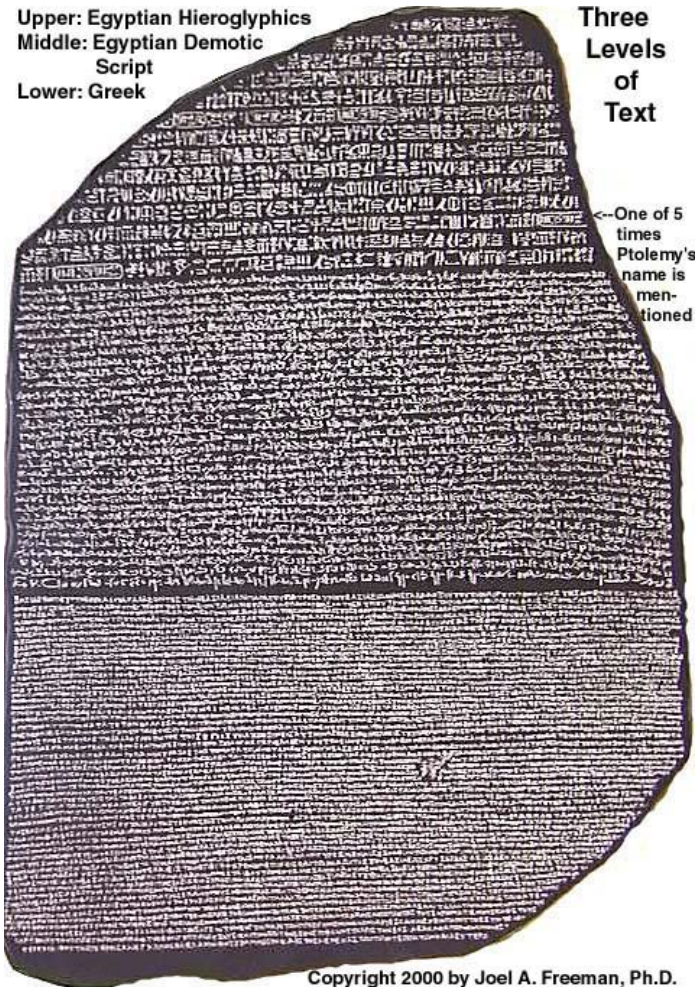
Задача перевода

Rosetta Stone ---->

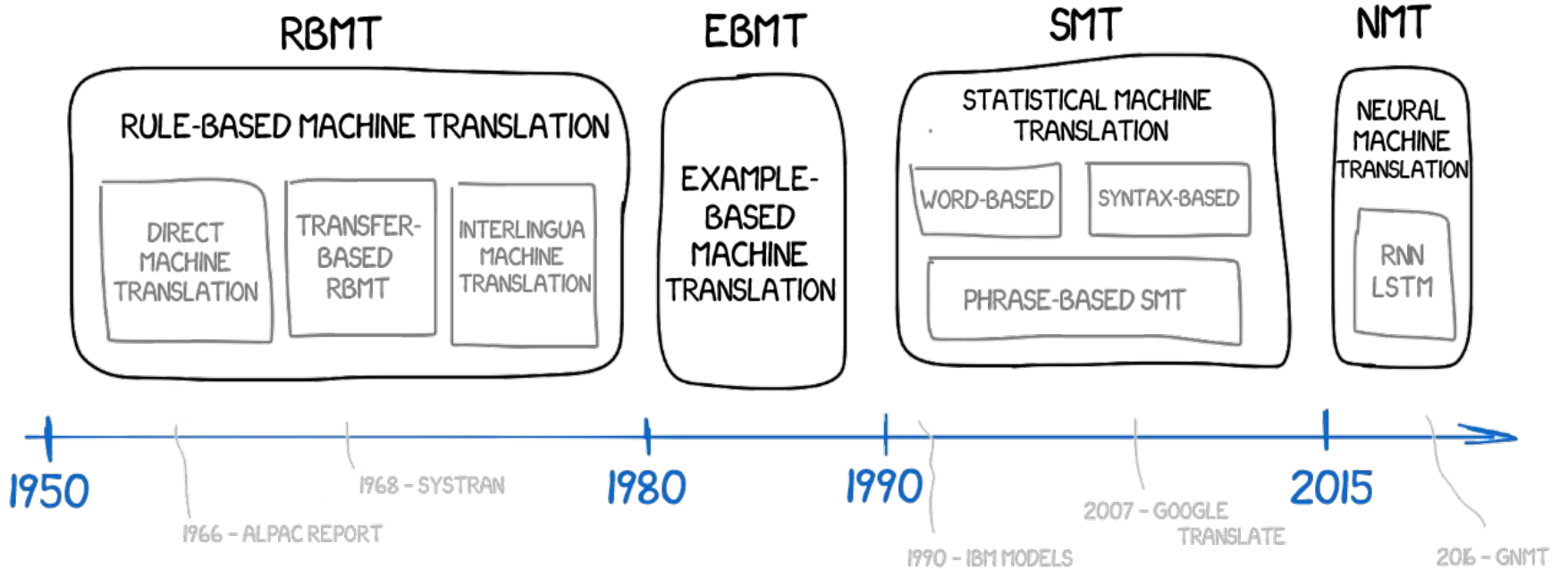
Параллельный корпус, найден в 1799 г.

Позволил расшифровать

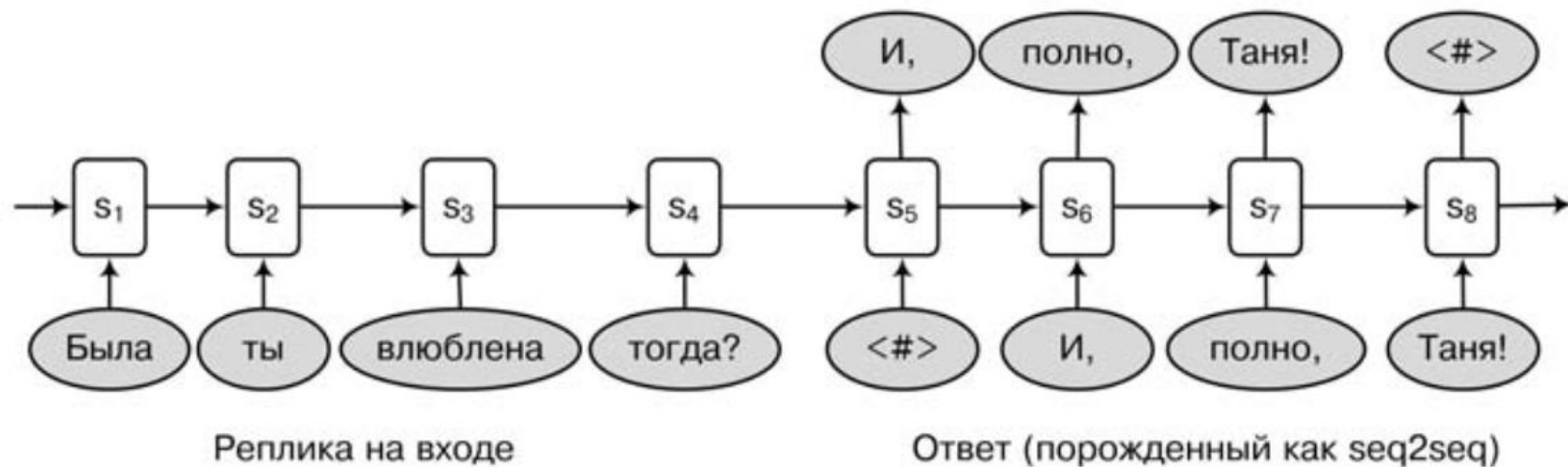
египетские иероглифы



A BRIEF HISTORY OF MACHINE TRANSLATION



Чатботы



RNN Sequence-to-sequence model

Google, Sutskever et al. 2014

Encoder

$$e_t = x_t * W_{emb}$$

$$h_t = rnn(h_{t-1}, e_t)$$

$$context = h_T$$

Decoder

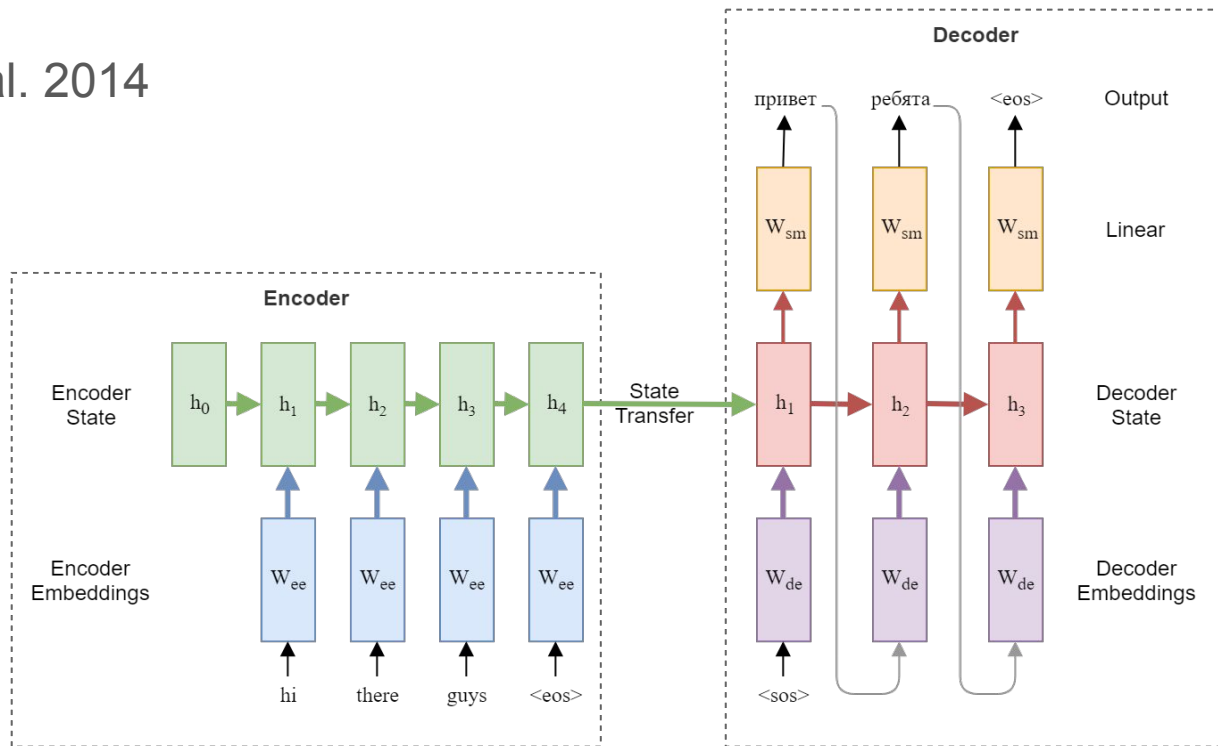
$$h_0 = context$$

$$y_0 = \langle sos \rangle$$

$$e_t = y_{t-1} * W_{emb}$$

$$o_t, h_t = rnn(h_{t-1}, e_t)$$

$$y_t = softmax(o_t * W_{sm})$$



RNN Sequence-to-sequence model

Cho et al. 2014

Encoder (same)

$$e_t = x_t * W_{emb}$$

$$h_t = rnn(h_{t-1}, e_t)$$

$$context = h_T$$

Decoder

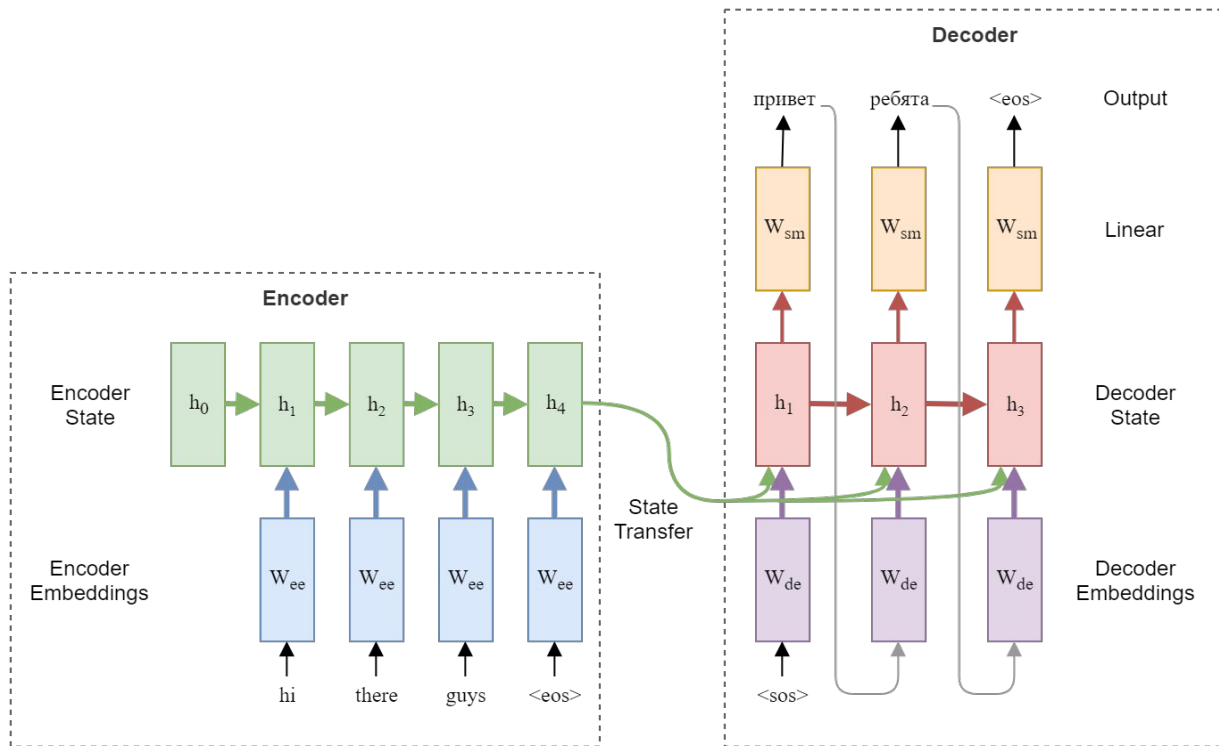
$$c_t = [e_t; context]$$

$$y_0 = \langle sos \rangle$$

$$e_t = y_{t-1} * W_{emb}$$

$$o_t, h_t = rnn(h_{t-1}, c_t)$$

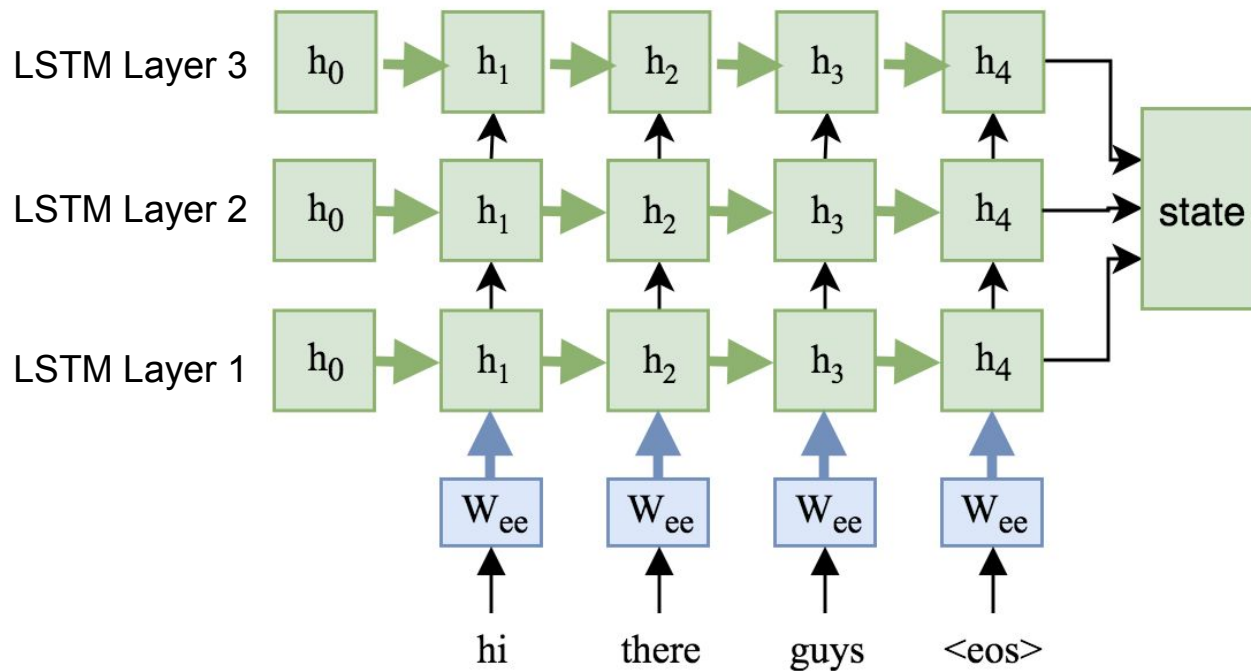
$$y_t = softmax(o_t * W_{sm})$$



RNN Sequence-to-sequence model

Улучшения:

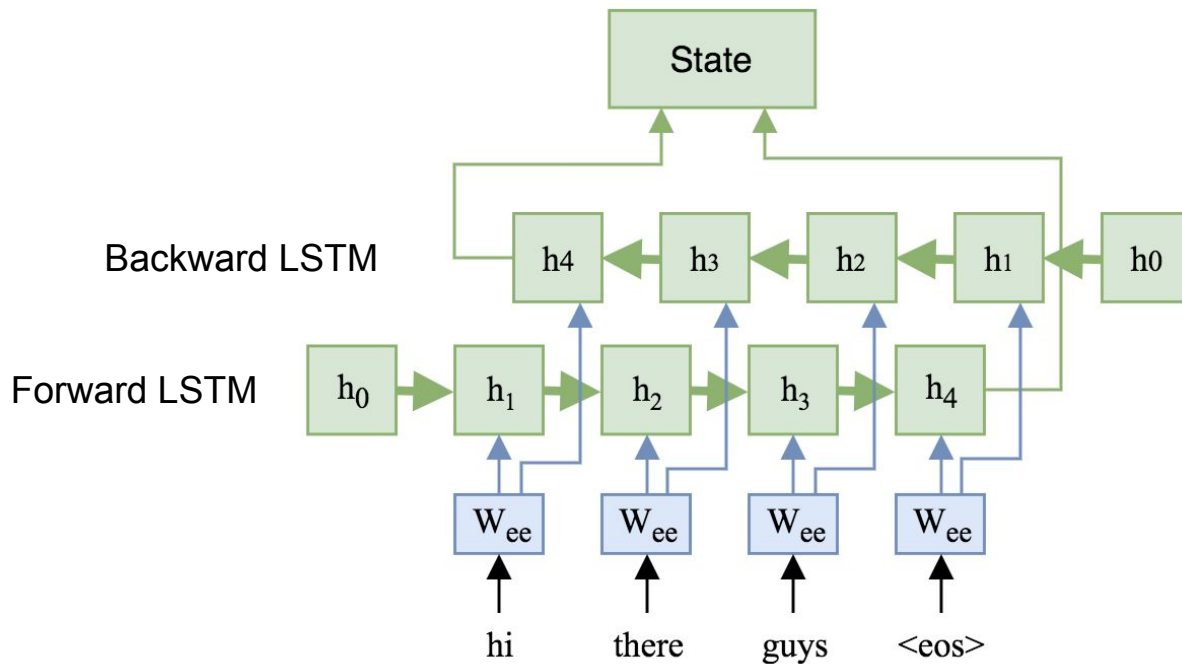
1. Deep Encoder
2. Deep Decoder



RNN Sequence-to-sequence model

Улучшения:

Bidirectional Encoder



RNN Sequence-to-sequence model

Проблемы:

1. Размер стейта фиксирован
2. Изменения из начала последовательности затираются
3. Не все входные токены одинаково значимы
4. Просто взять стейты со всех шагов декодера - слишком много данных

RNN Sequence-to-sequence model

Решение:

Внимание

Механизм внимания, мотивация

Xu et al. 2015

Show, Attend and Tell:

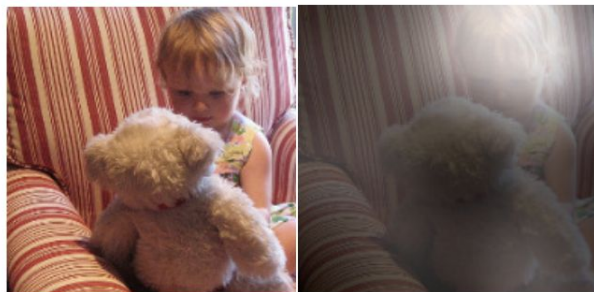
Neural Image Caption Generation
with Visual Attention.



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A little girl sitting on a bed with
a teddy bear.

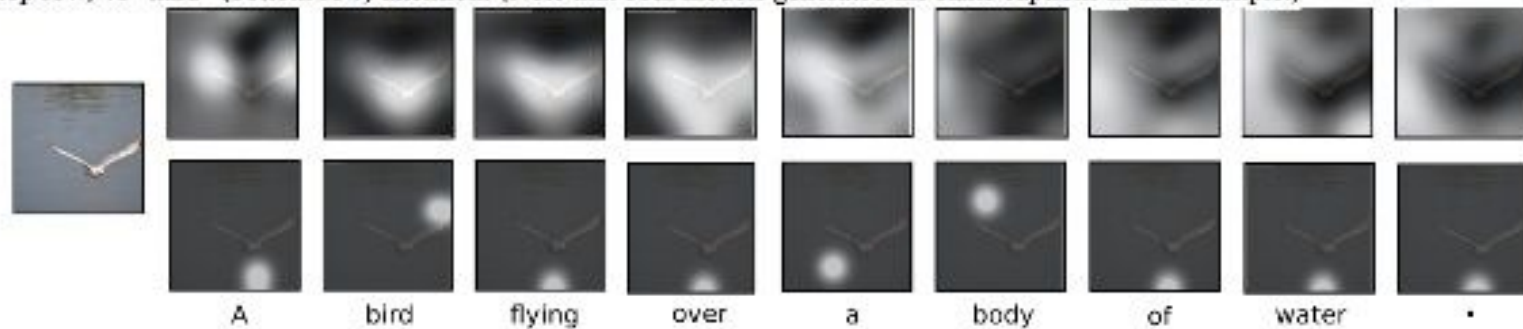


A group of people sitting on a boat
in the water.

<https://arxiv.org/abs/1502.03044>

“Soft” vs “Hard” Attention

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. “soft” (top row) vs “hard” (bottom row) attention. (Note that both models generated the same captions in this example.)



Soft vs Hard Attention

Hard

1. Выбор одной/н областей
2. Получаем сэмпингом из softmax
3. Не дифференцируем
4. Нужно учить с помощью RL
5. А значит тяжело учиться

Soft

1. Взвешенная сумма областей
2. Дифференцируемый
3. А значит обучаем через backprop

Механизм внимания, мотивация

В случае машинного перевода



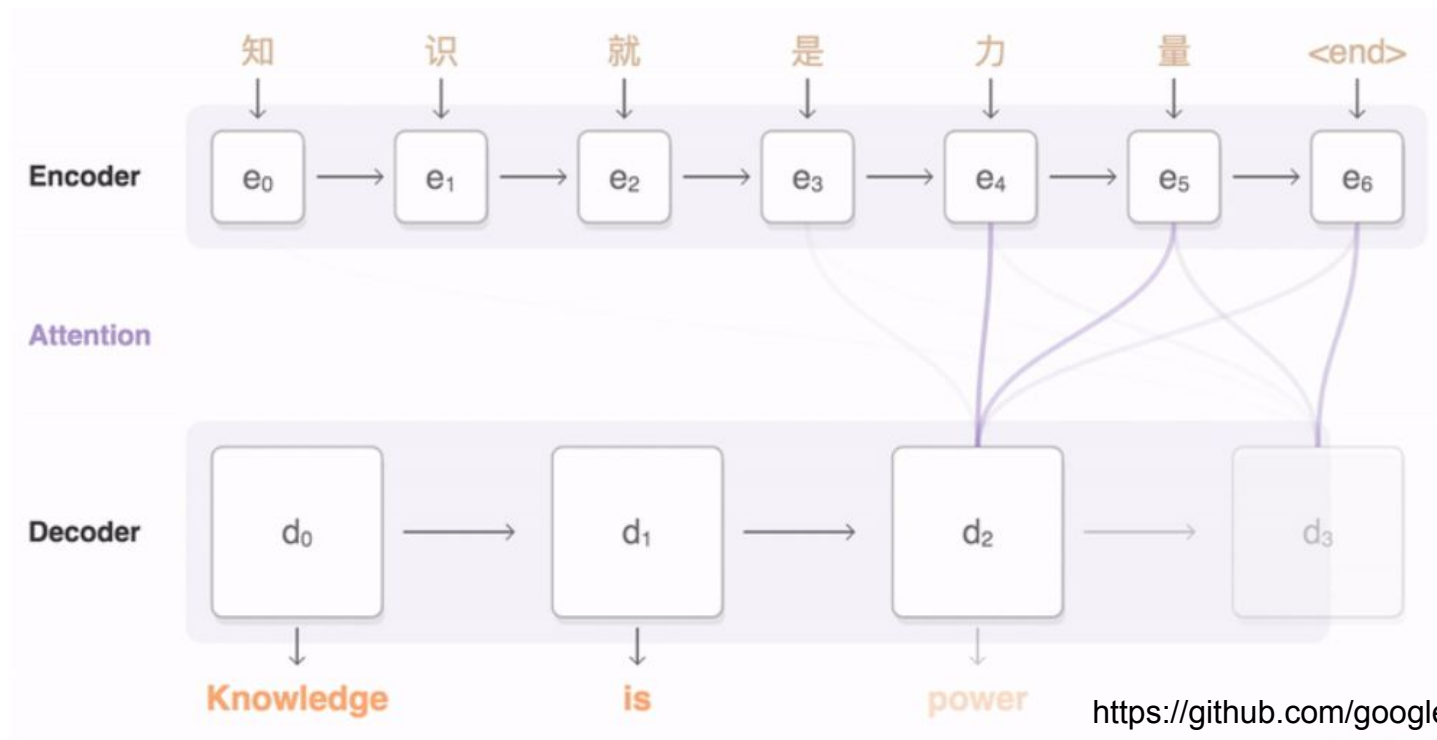
Economic growth has slowed down in recent years .



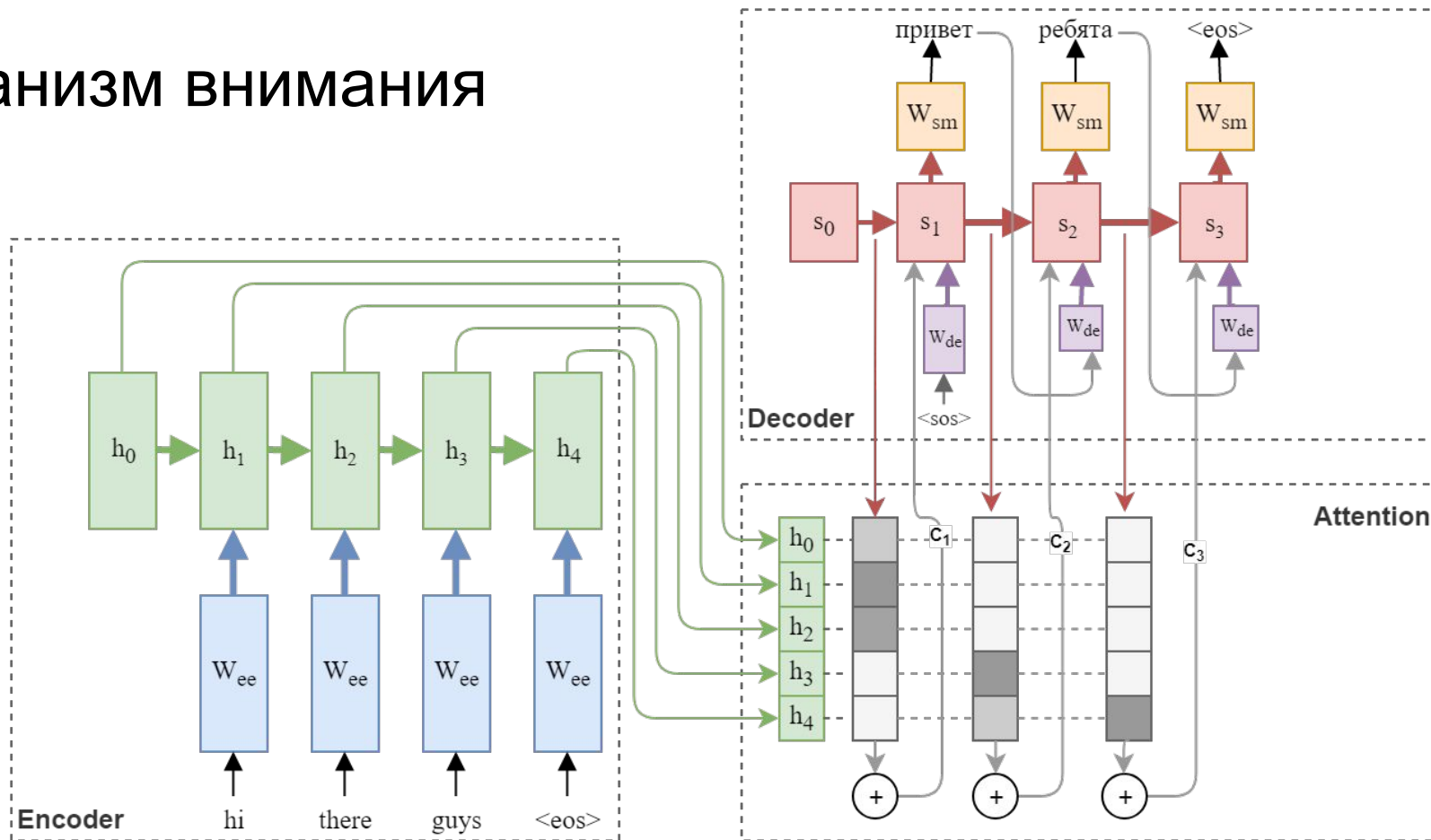
La croissance économique s' est ralentie ces dernières années .

(Edge thicknesses represent the attention weights found by the attention model)

Механизм внимания, мотивация



Механизм внимания



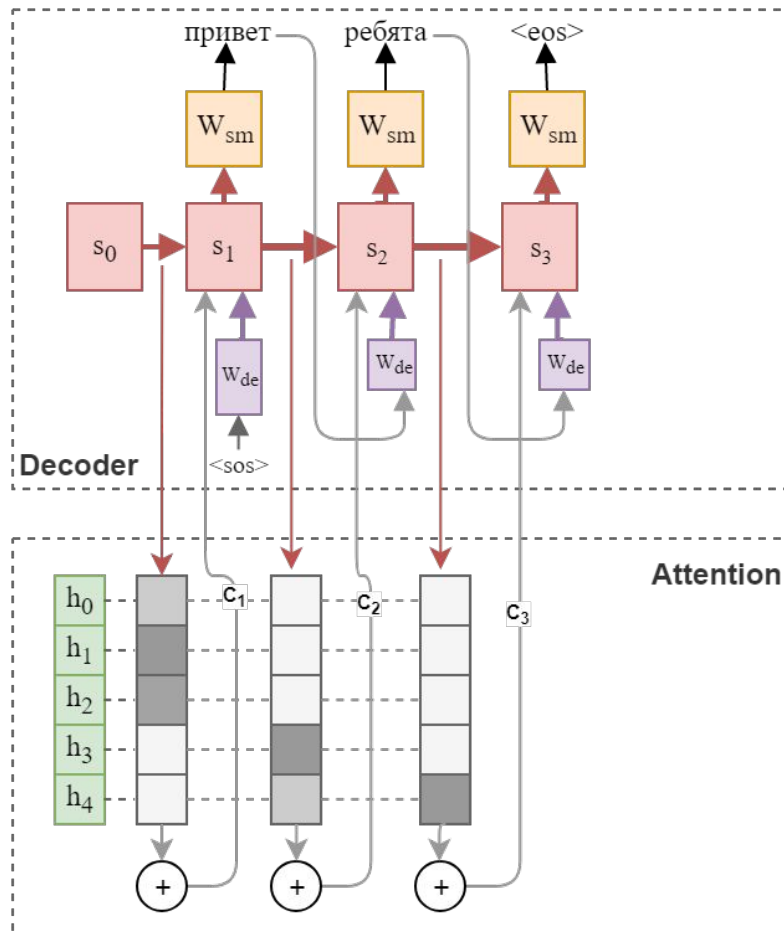
Механизм внимания

Bahdanau et al. 2014

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

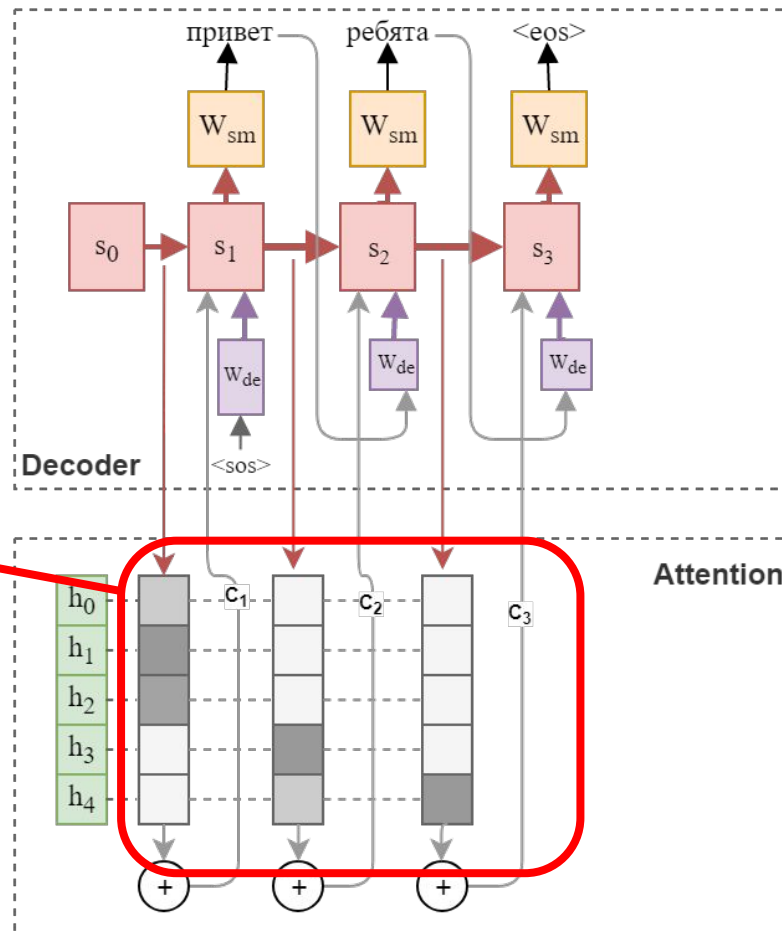
$$e_{ij} = \text{attention}(s_{i-1}, h_j)$$



Механизм внимания

Bahdanau et al. 2014

Карта внимания или alignment слов



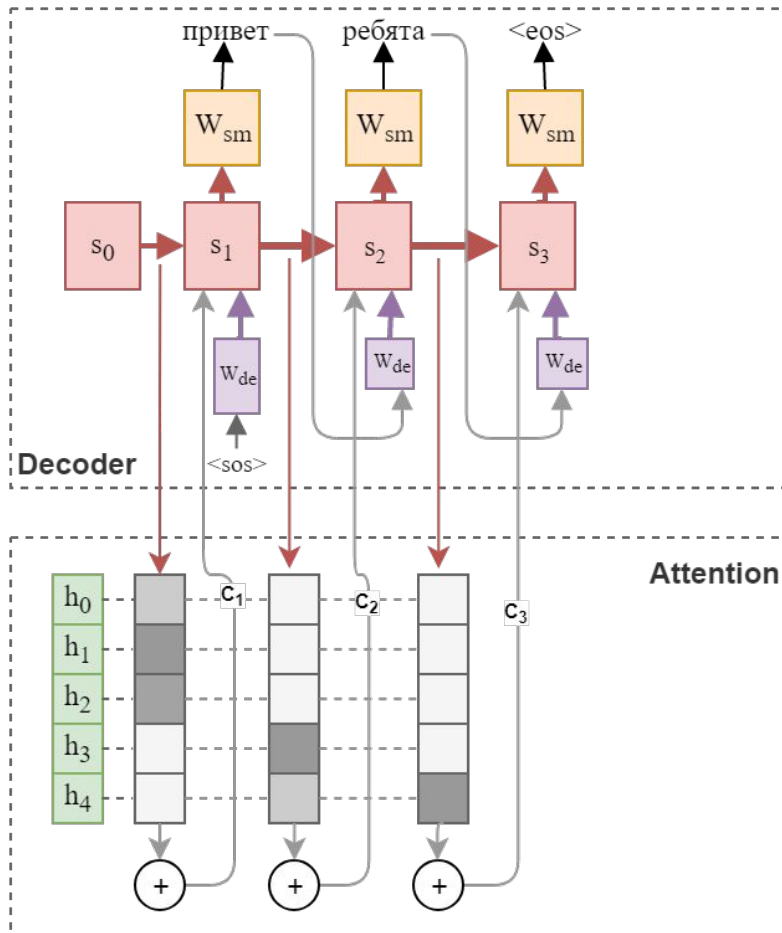
Механизм внимания

Bahdanau et al. 2014

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = \text{attention}(s_{i-1}, h_j)$$



Attention function

$$attention(s, h) = \begin{cases} h^\top * s & \text{Dot Product} \\ h^\top * W_a * s & \text{General} \\ v_a^\top * \tanh(W_a[h; s]) & \text{Additive} \end{cases}$$

Практические нюансы

1. Wordpiece models and character-based models
2. Pretrained embeddings
3. Multihead Attention
4. Teacher Forcing
5. Beam Search

Wordpiece models

Проблемы словаря

1. большой размер эмбеддингов и софтмакс слоя (сотни тысяч)
2. неизвестные слова при инференсе, приходится заменять на UNKNOWN токен

Решение

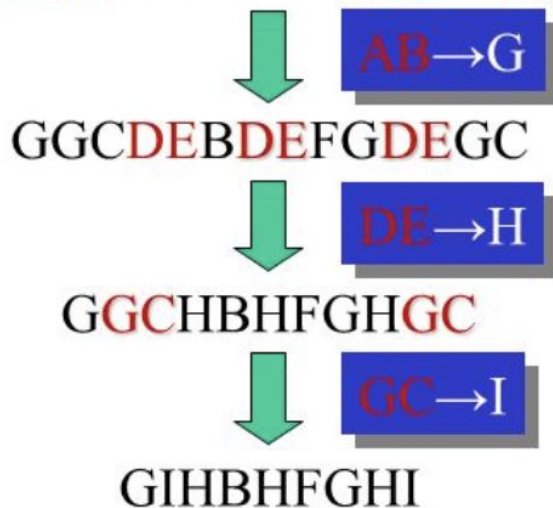
Давайте разбивать предложения на характерные части, которые меньше чем слово, но больше чем буква.

Идея пришла из сегментации корейских и японских предложений, где нет явной границы между словами.

Pretrained embeddings

Wordpiece models, BPE - byte-pair encoding

Text: $T = \text{ABABCDEBDEFABDEABC}$



$X_1 = \text{A};$

$X_2 = \text{B};$

$X_3 = \text{C};$

$X_4 = \text{D};$

$X_5 = \text{E};$

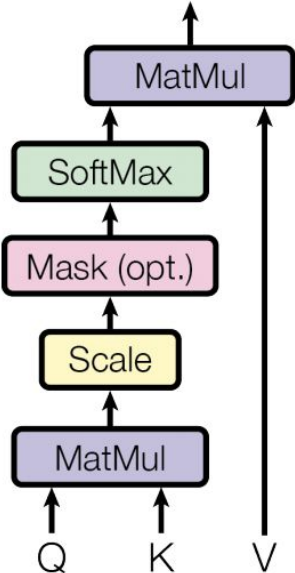
$X_6 = \text{F};$

$X_7 = X_1 \cdot X_2$

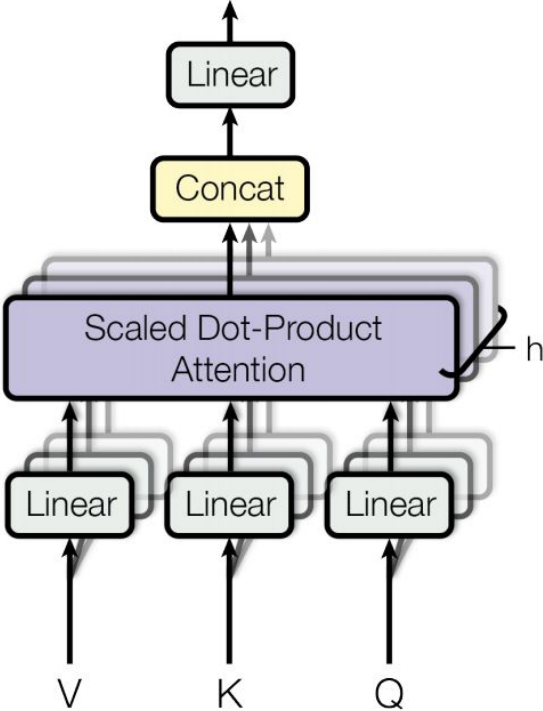
$X_8 = X_4 \cdot X_5$

Multihead Attention

Scaled Dot-Product Attention

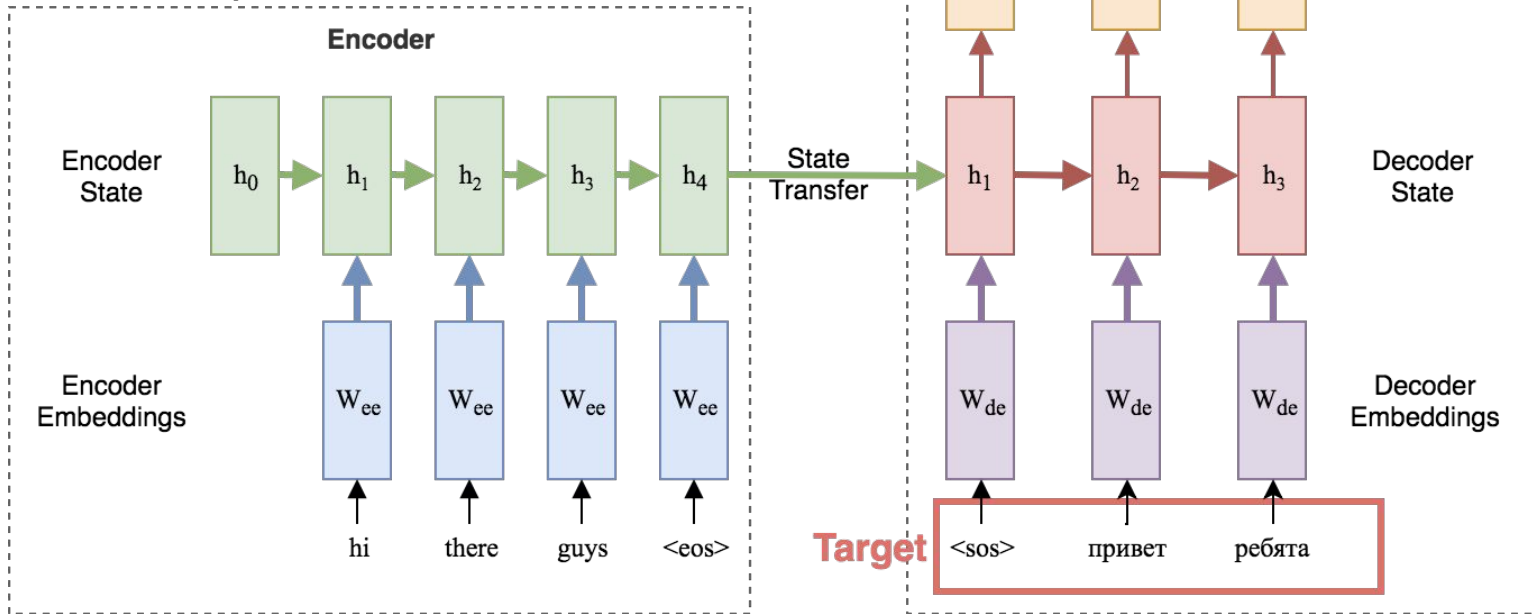


Multi-Head Attention

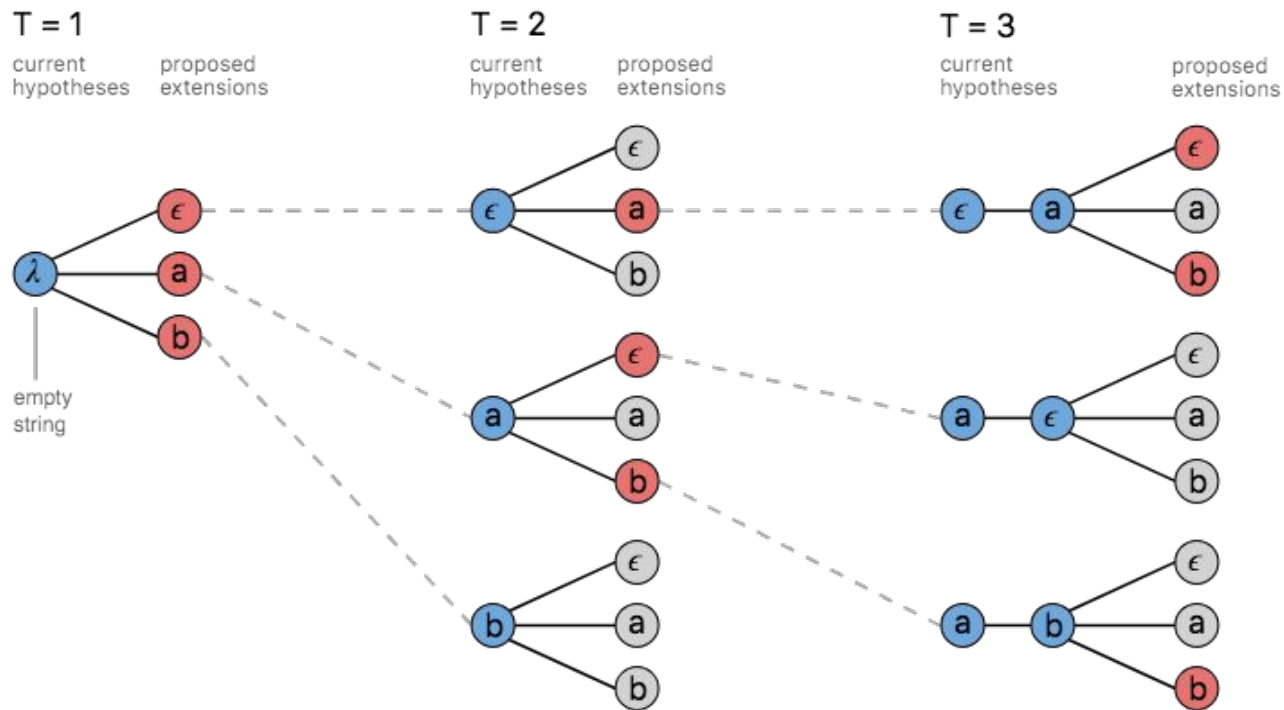


Teacher Forcing

Подаем на вход декодера не прошлый выход,
а верный символ из таргета



Beam Search



Beyond attention

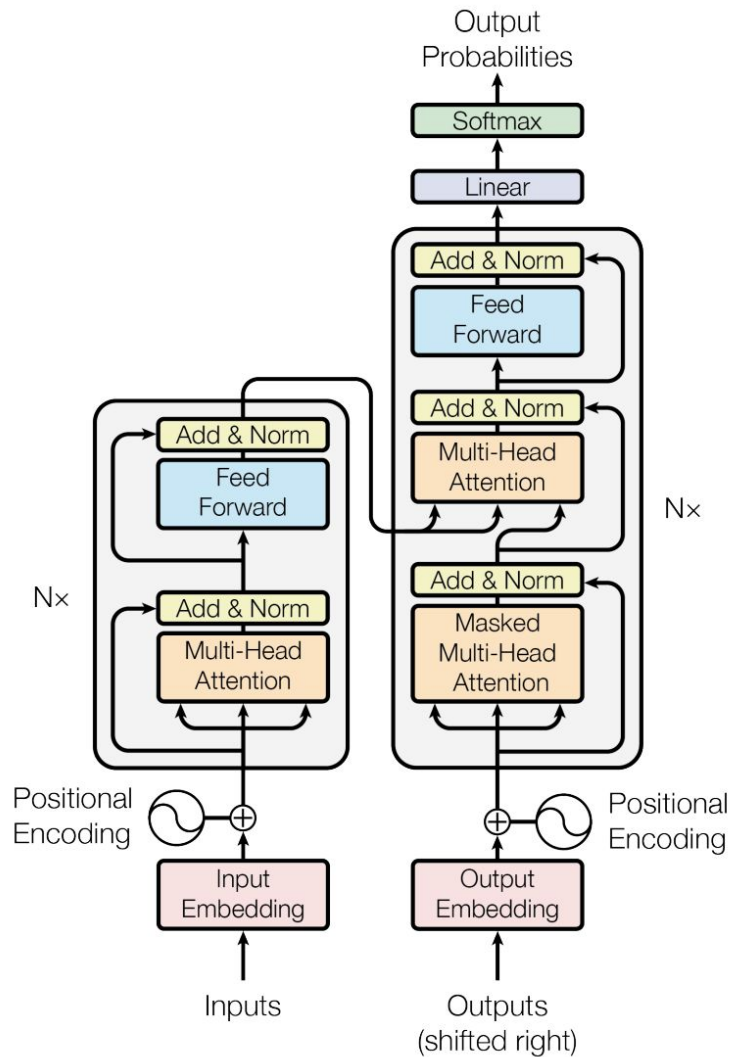
- Attention позволяет построить текущее состояние с учетом всего прошлого последовательности.
- Одинаково хорошо учитывает данные как из далекого прошлого, так и близкого.
- Как правило не содержит информации об относительном расположении определенных данных в последовательности, но это решаемо.
- Зачем тогда RNN, которая обновляет стейт последовательно и потому хуже учитывает далекое прошлое?

Transformer

Attention is all you need, Vaswani et al. 2017

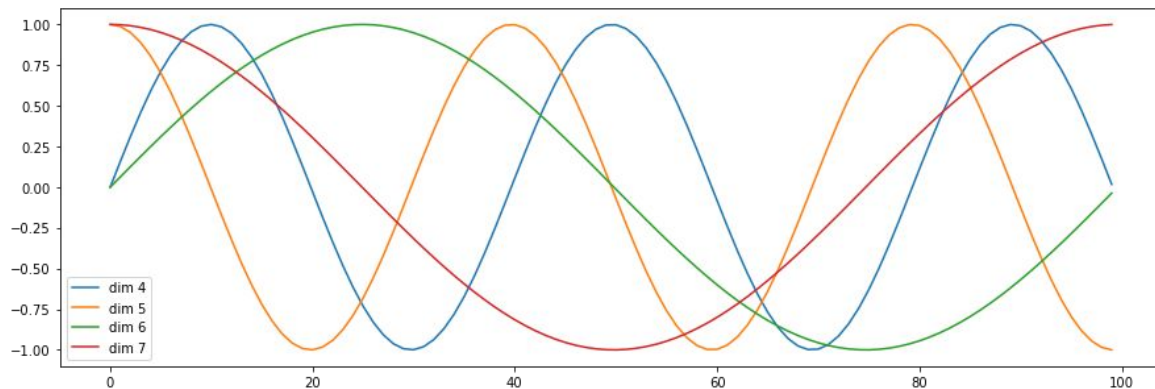
<https://arxiv.org/abs/1706.03762>

Self-attention instead of recurrence



Positional encoding

1. Sinusoidal encoding



2. Learned positional embeddings. Position index -> embedding layer -> vector

Что ещё посмотреть?

- <https://distill.pub/2016/augmented-rnns/>
- <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>
- https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html
- https://nlp.stanford.edu/pubs/emnlp15_attn.pdf
- <https://www.youtube.com/watch?v=IxQtK2SjWWM> (Stanford Deep NLP)