Основы глубинного обучения

Ульянкин Филипп

30 ноября 2023 г.

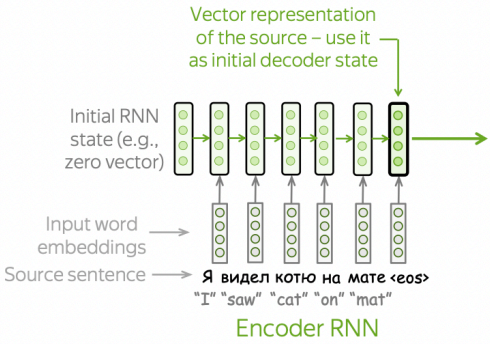
Лекция 10: seq2seq модели и механизм внимания

Agenda

- Seq2seq модели
- Механизмы внимания
- Attention is all you need
- Трансформеры
- BERT и GPT

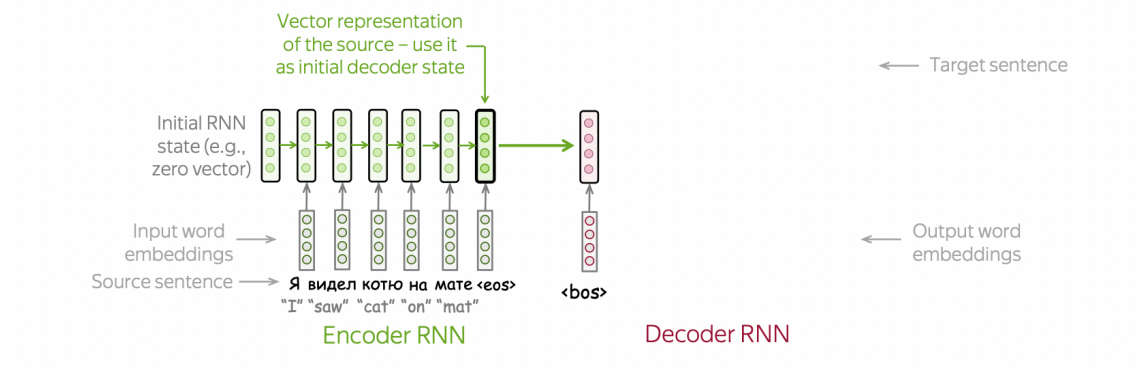
Seq2Seq перевод

Простейшая модель: RNN энкодер-декодер



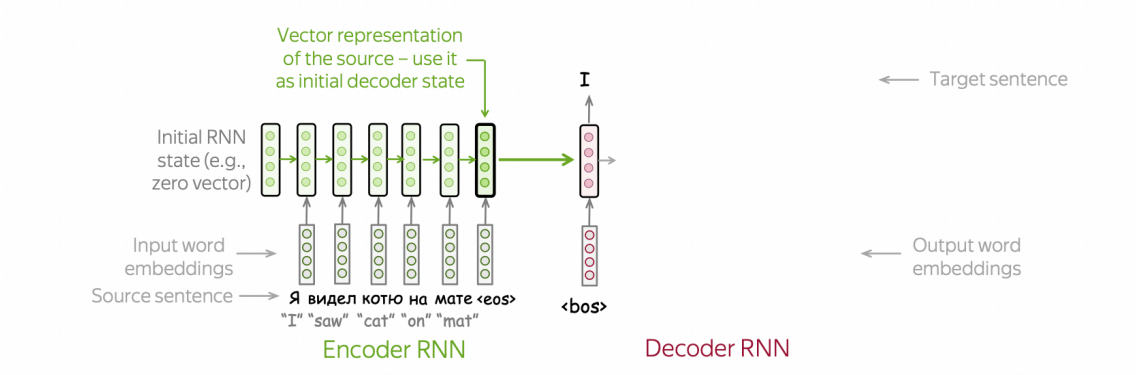
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



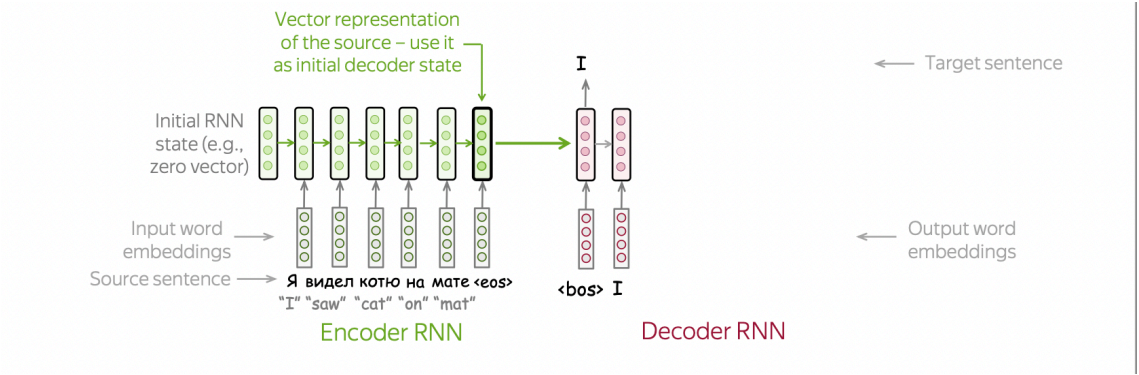
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



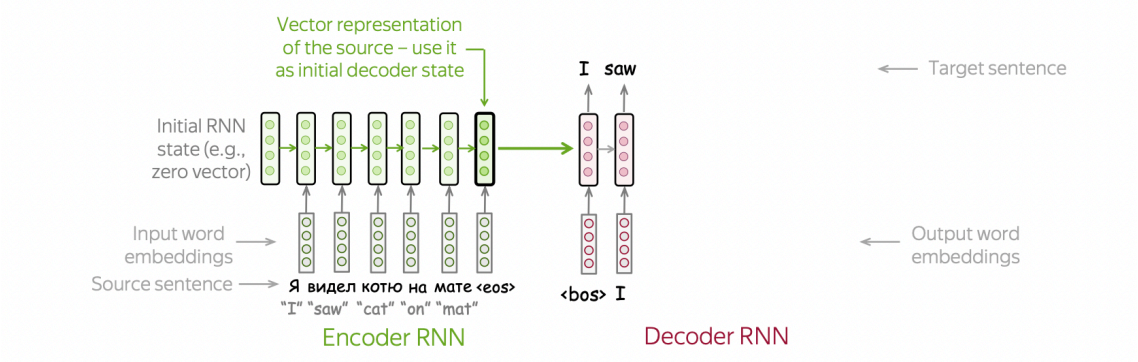
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



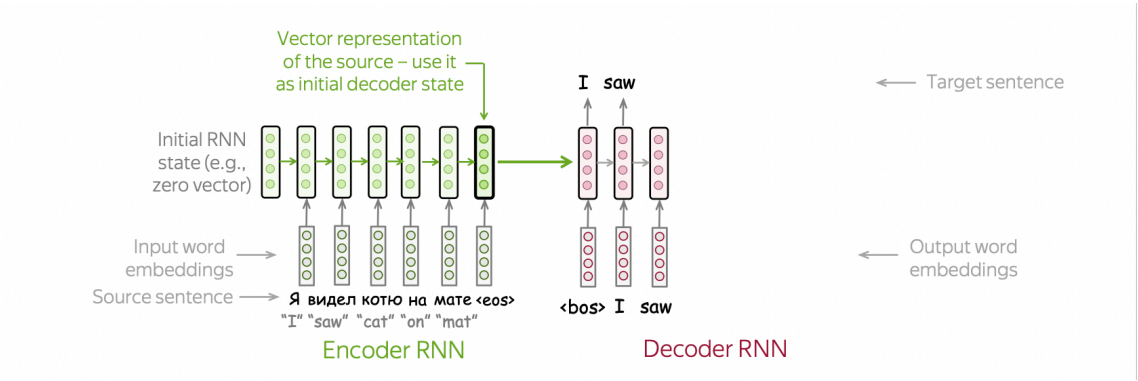
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



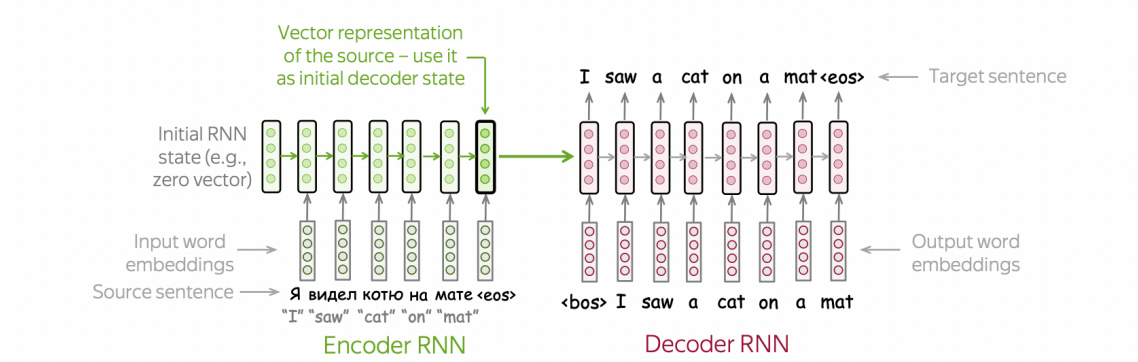
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



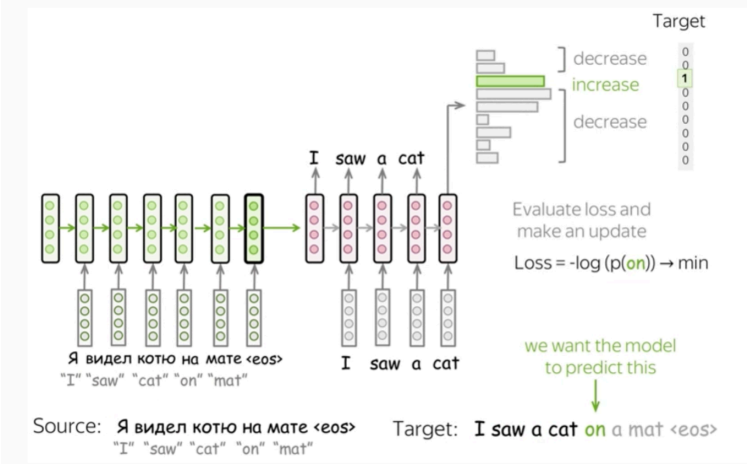
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Простейшая модель: RNN энкодер-декодер



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Как обучить модель?



https://lena-voita.github.io/resources/lectures/seq2seq/general/seq2seq_training_with_target.mp4

Как сделать прогноз?

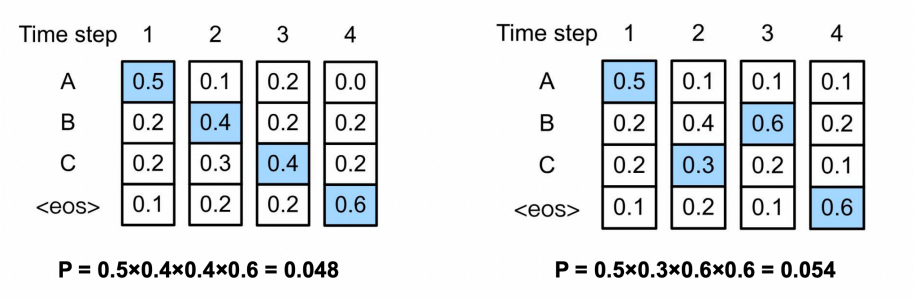
- Жадно (greedy decoding): на каждом шаге берём токен с самой большой вероятностью

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

- Наш прогноз это последовательность, жадный способ на каждом шаге выбирает локальный оптимум

- Перебирать все траектории, чтобы найти глобальный оптимум очень дорого

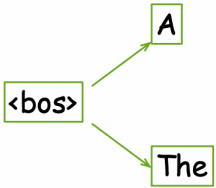
Неоптимальность Greedy Decoding



Beam Search

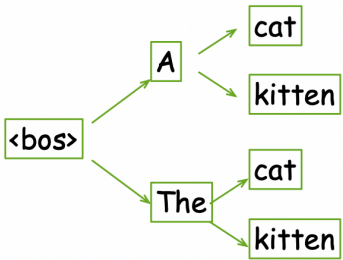
- Давайте поддерживать несколько самых вероятных траекторий
- Такая стратегия называется Beam Search
- Число траекторий, которое мы помним будет гиперпараметром (больше 10 брать не стоит)

Beam Search



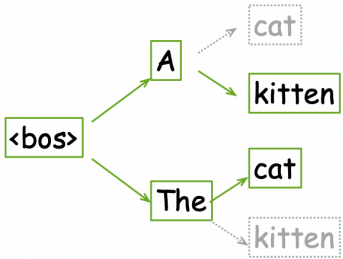
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



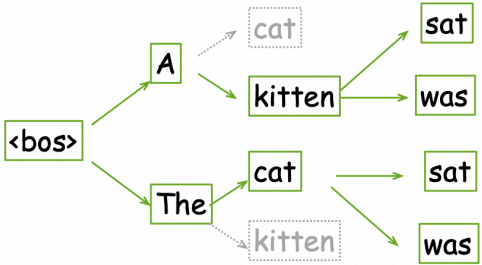
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



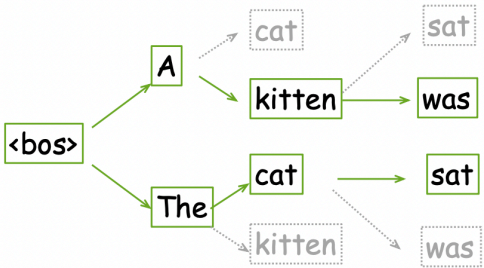
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



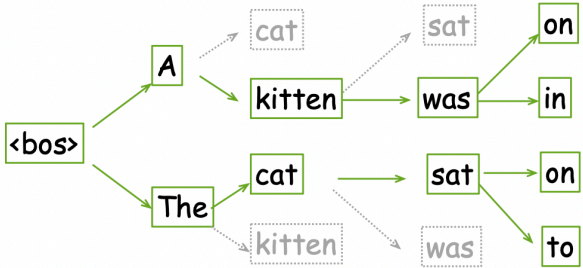
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



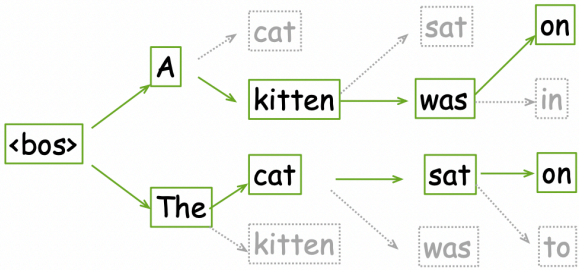
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



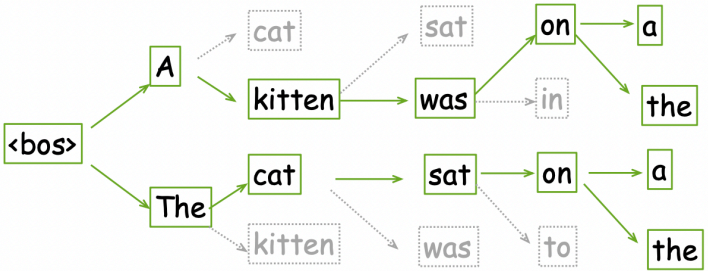
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



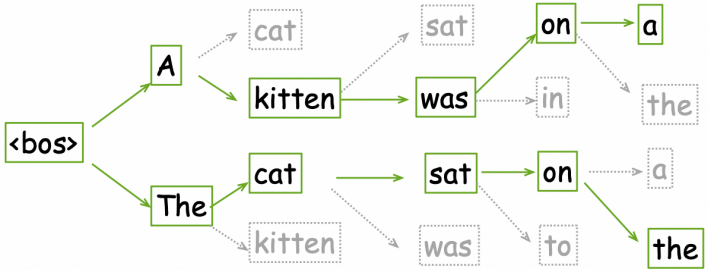
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



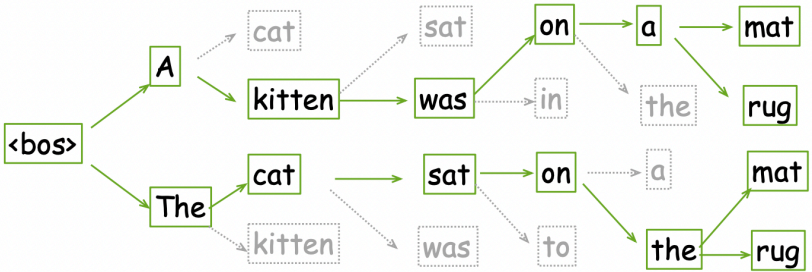
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



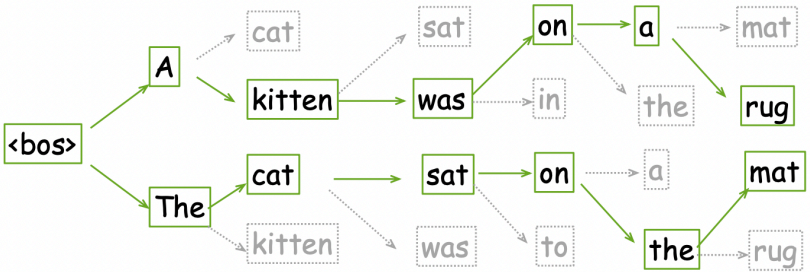
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



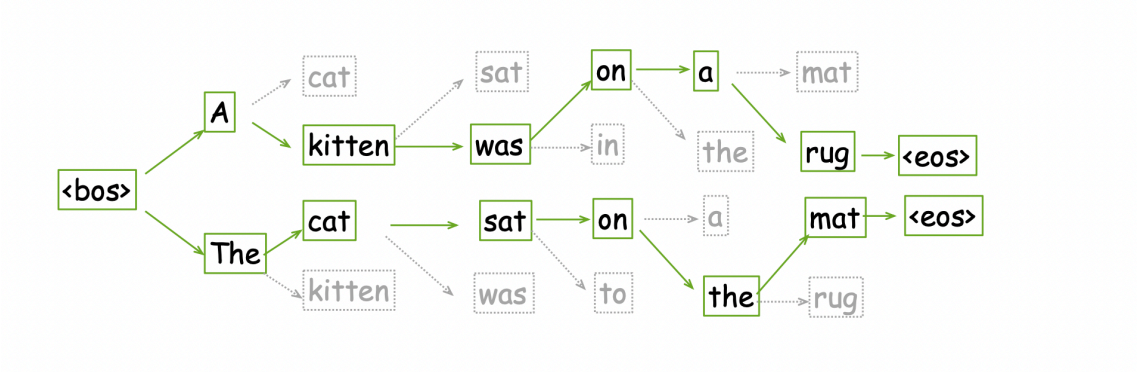
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



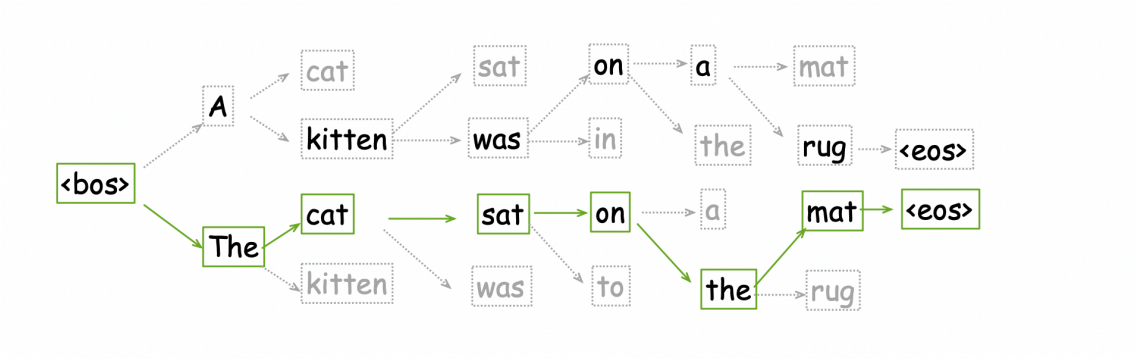
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Beam Search



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

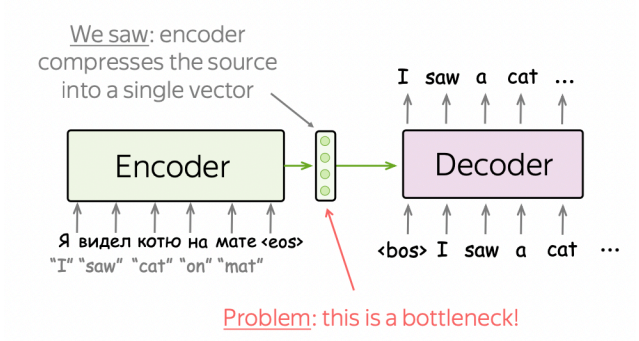
Beam Search



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Механизмы внимания

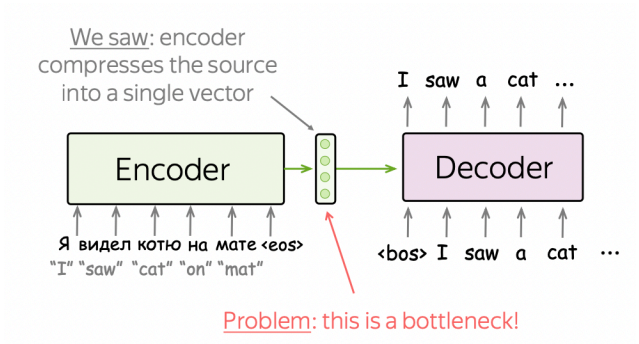
В чём проблема простой RNN модели?



- Сжать предложение в вектор сложно
- Декодеру может понадобиться информация о разных частях предложения для его расшифровки

https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

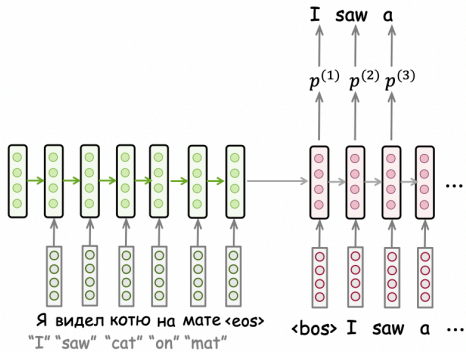
В чём проблема простой RNN модели?



- Attention: на разных шагах надо позволить модели фокусироваться на разных входных токенах

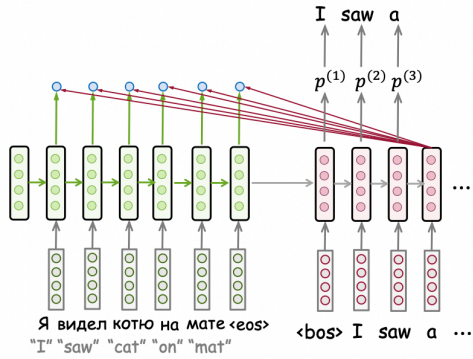
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



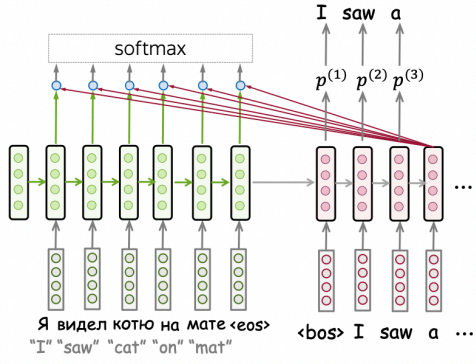
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



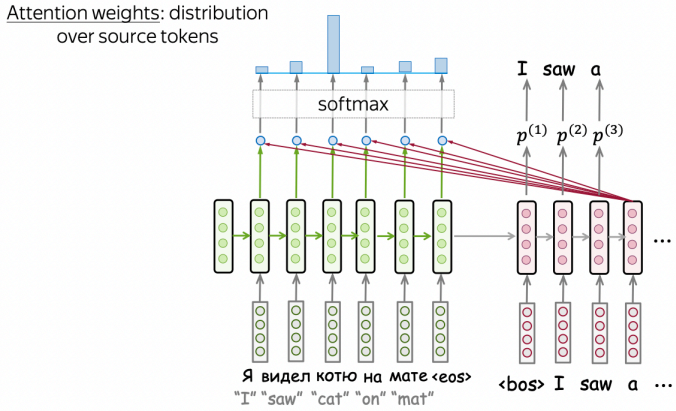
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



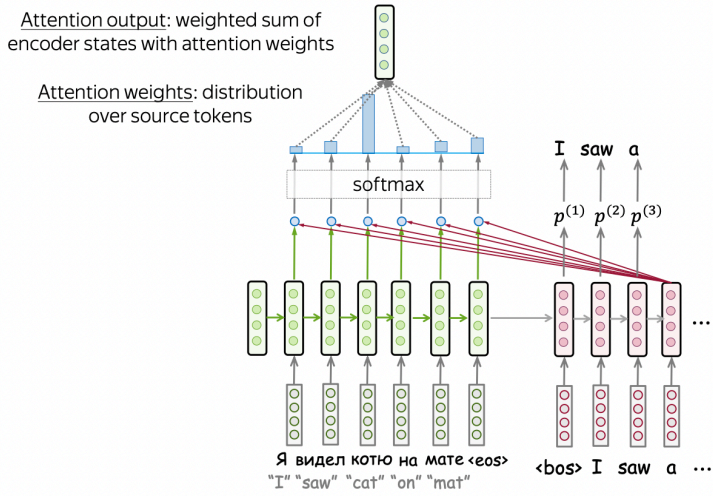
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



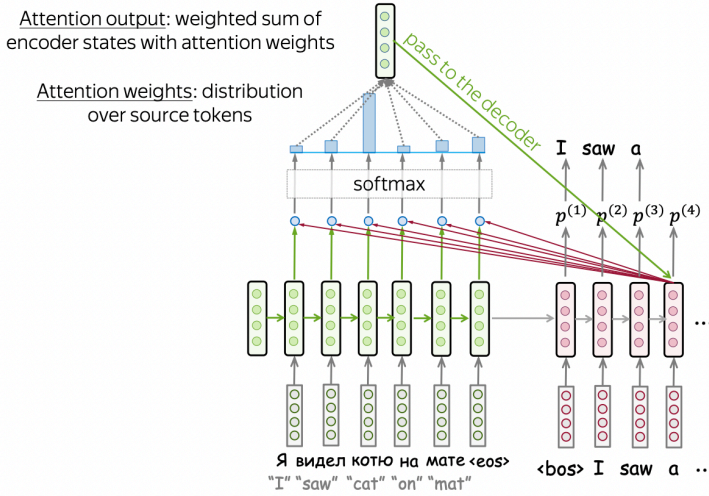
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



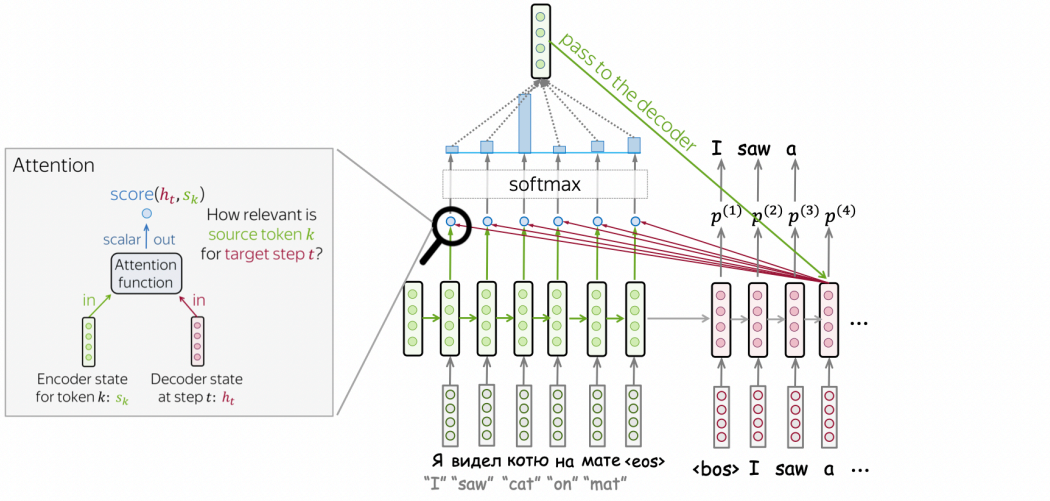
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention



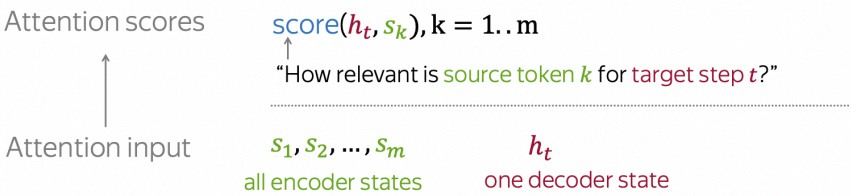
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Computation Pipeline

Attention input $s_1, s_2, \dots, s_m$ (all encoder states) h_t (one decoder state)

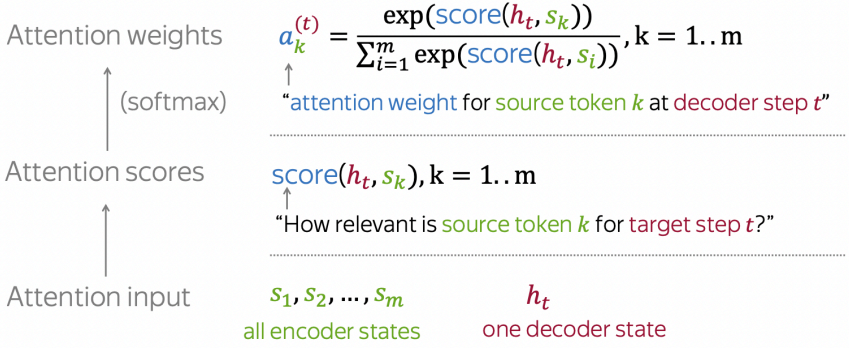
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Computation Pipeline



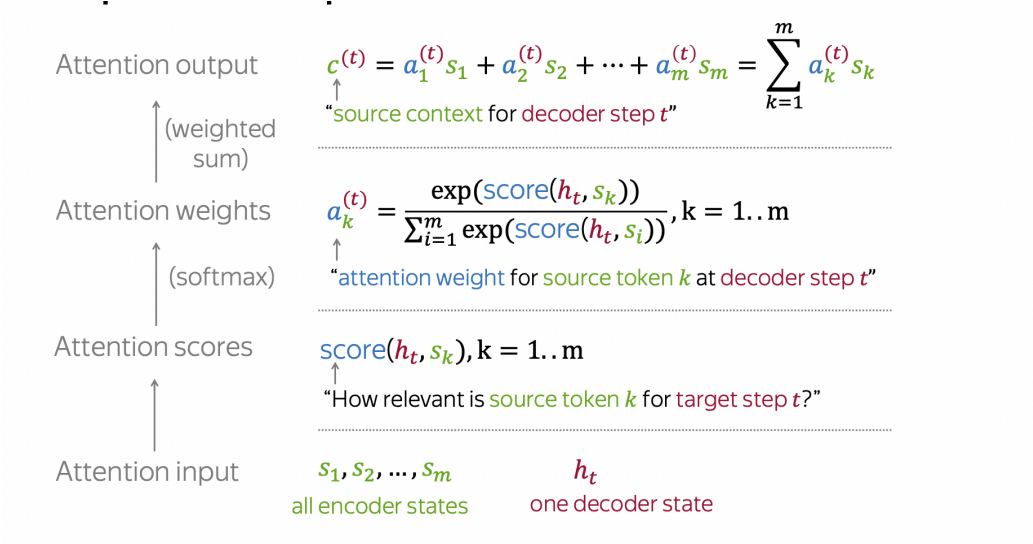
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Computation Pipeline



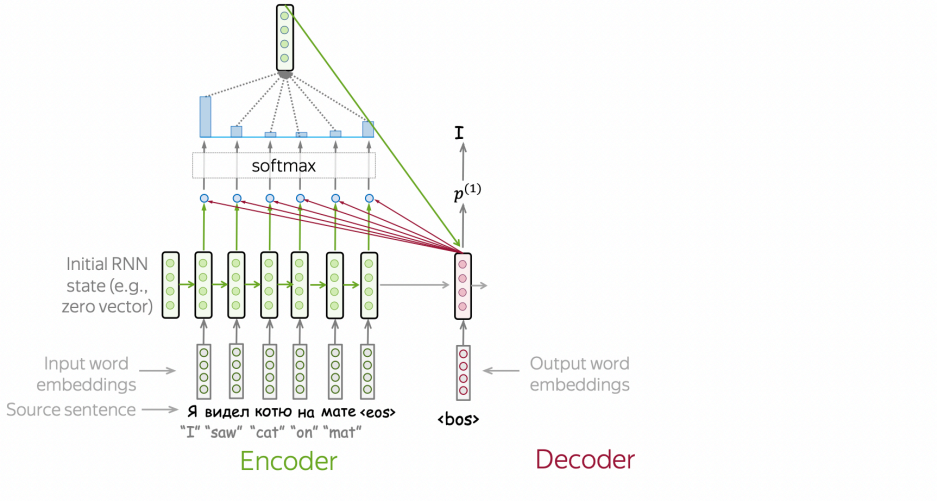
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Computation Pipeline



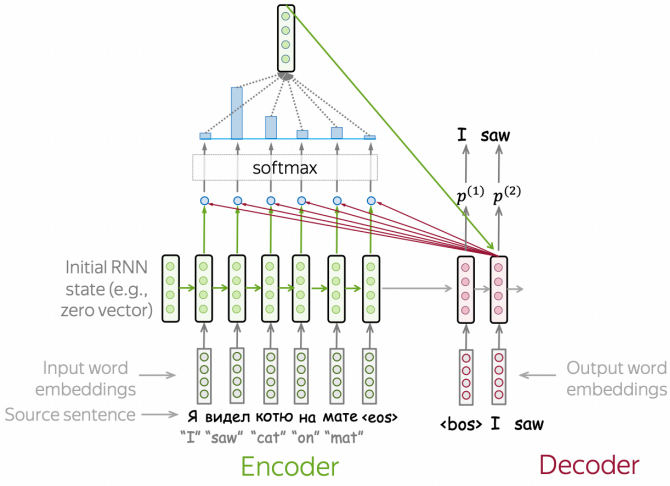
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



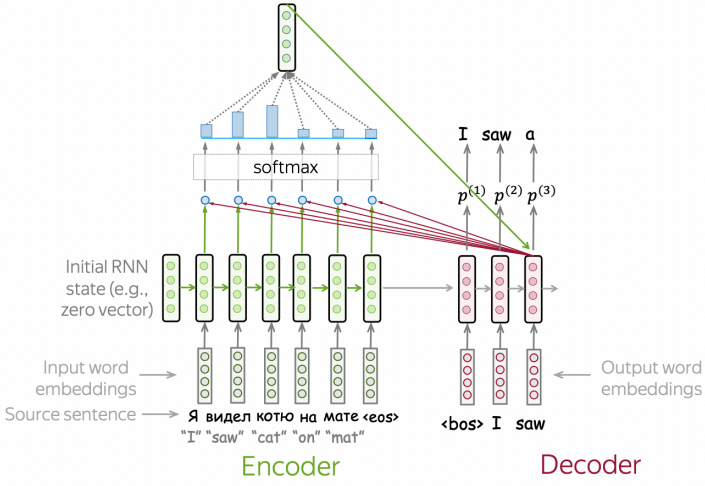
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



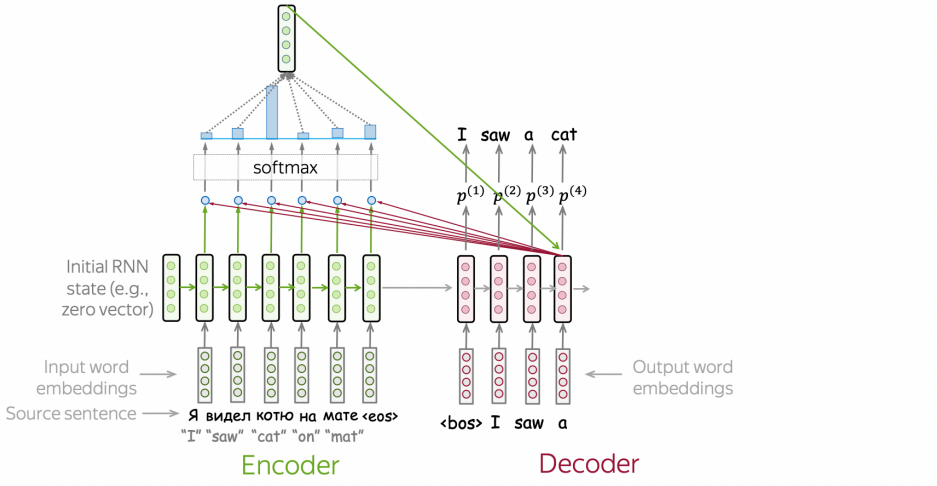
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



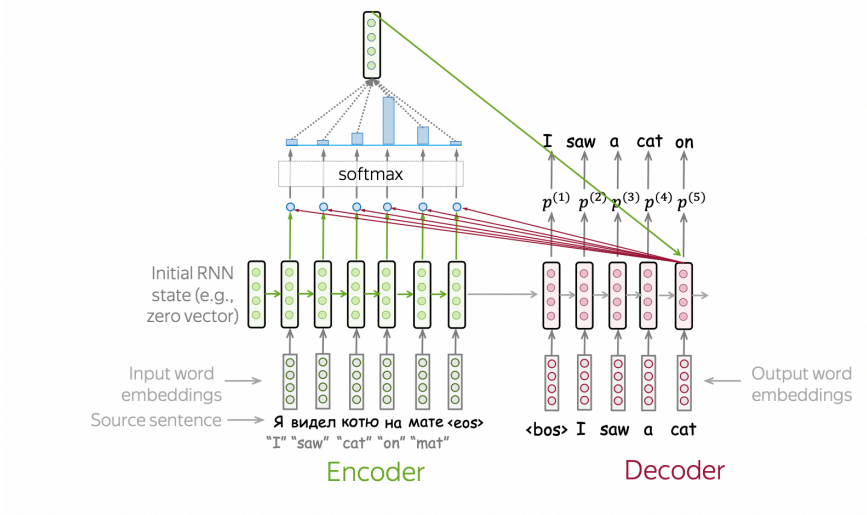
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



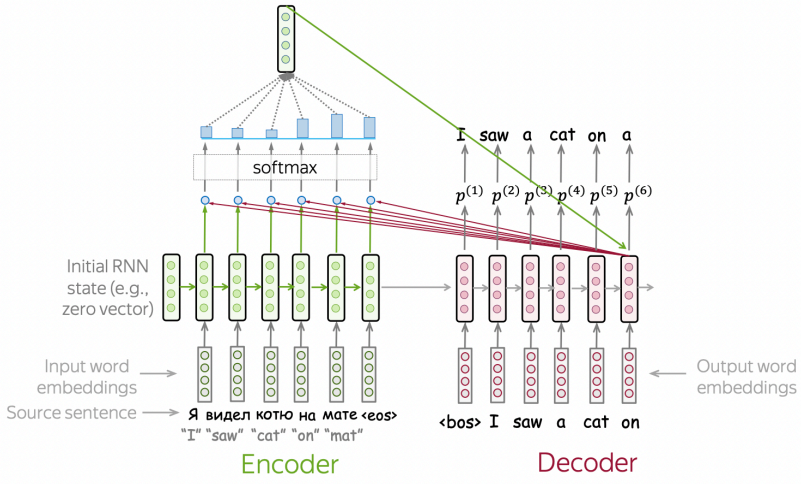
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



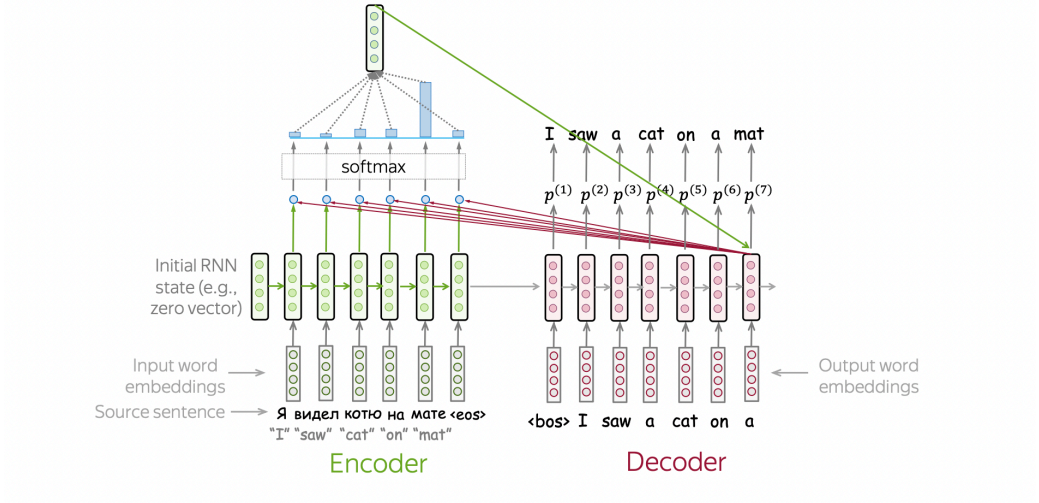
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



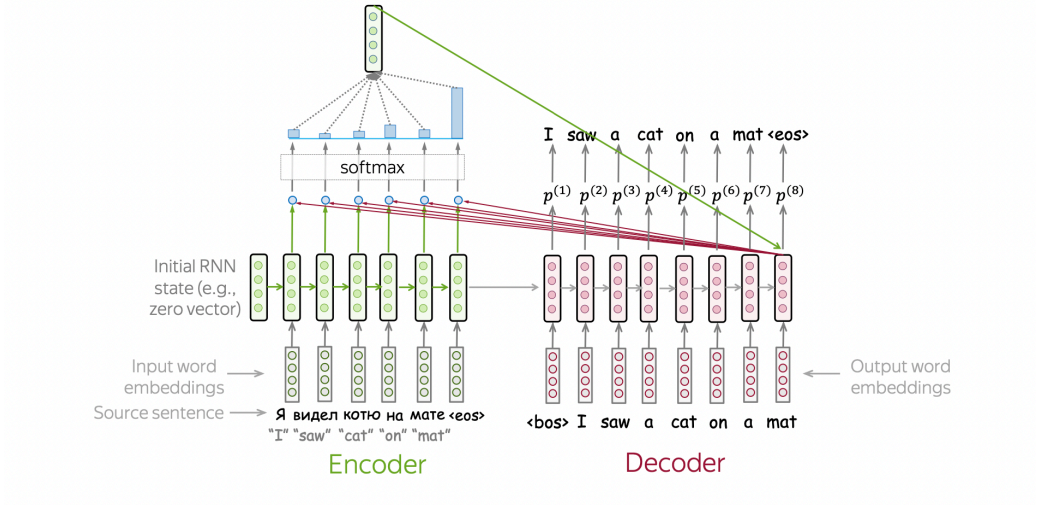
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



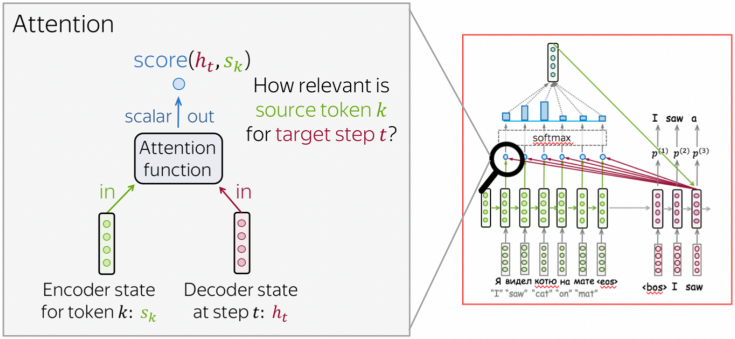
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Внимание в декодере



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

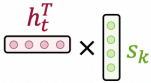
Attention Score Functions

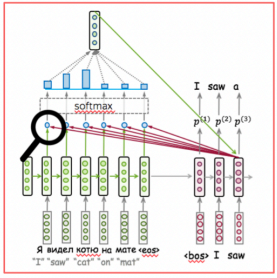


https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention Score Functions

- Dot-product: $\text{score}(h_t, s_k) = h_t^T s_k$

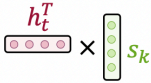




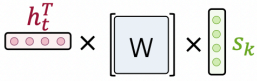
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

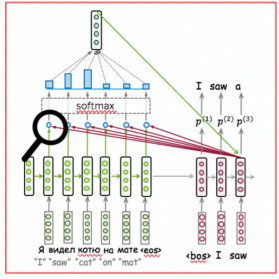
Attention Score Functions

- Dot-product: $\text{score}(h_t, s_k) = h_t^T s_k$



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

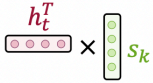




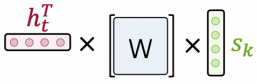
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention Score Functions

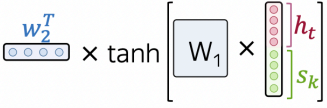
- Dot-product: $\text{score}(h_t, s_k) = h_t^T \, s_k$

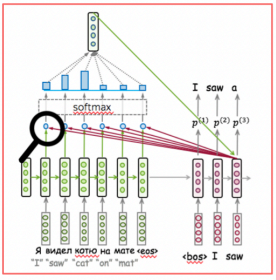


- Bilinear: $\text{score}(h_t, s_k) = h_t^T \, W s_k$



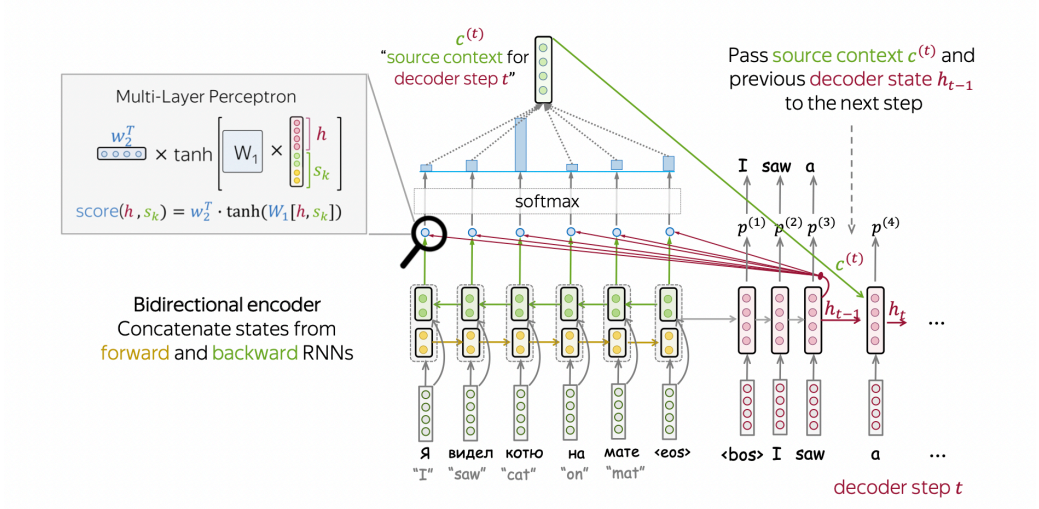
- Multi-Layer Perceptron: $\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$





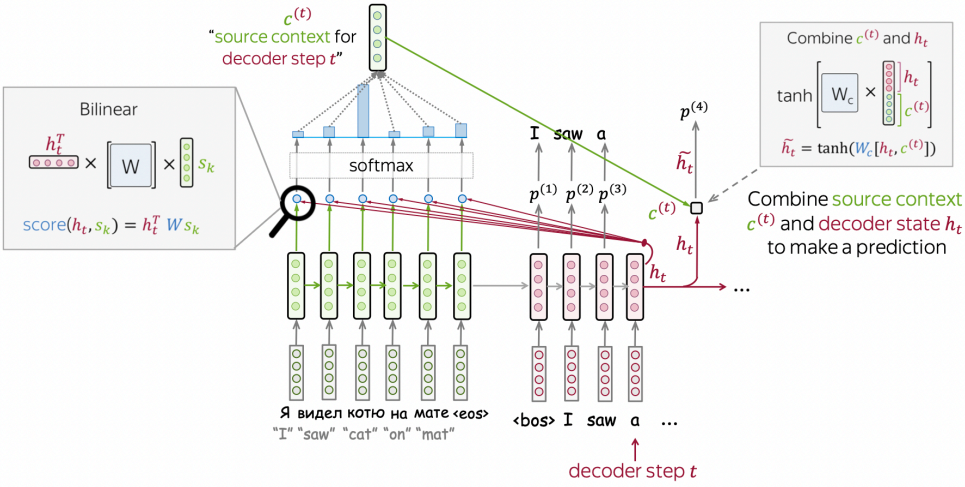
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Bahdanau Model (the original attention model)



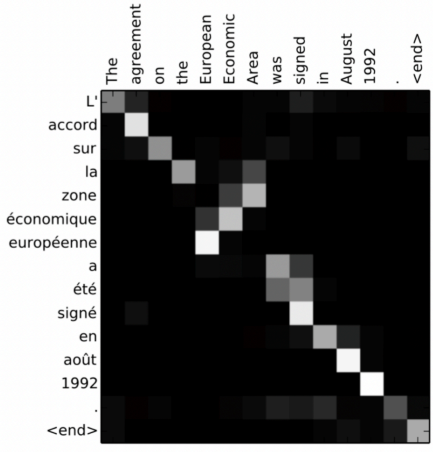
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

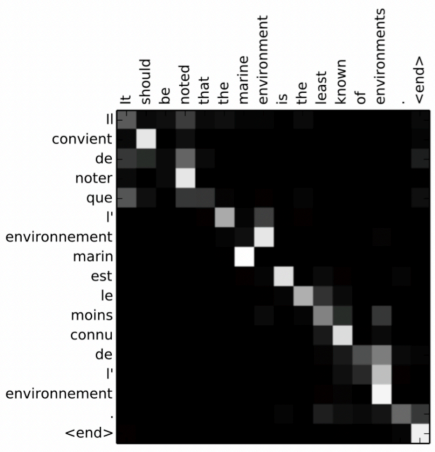
Luong Model



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Attention





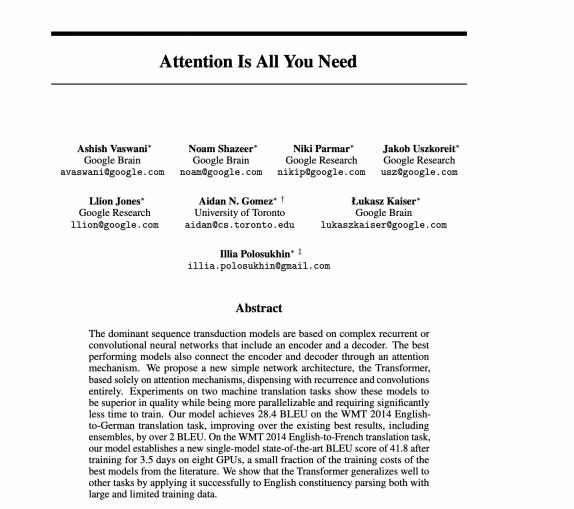
https://arxiv.org/pdf/1409.0473.pdf

Attention is All You Need!

Attention is All You Need! (2017)

- RNN это очень долго! Всегда, чтобы найти следующий токен, надо знать предыдущий
- Backward pass идёт ещё и через время :(
- Transformer — нейросетевая архитектура для задач seq2seq, основанная исключительно на полносвязных слоях
- Превзошла существовавшие seq2seq архитектуры как по качеству, так и по скорости работы
- Основной элемент — multi-head self-attention

Attention is All You Need! (2017)



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

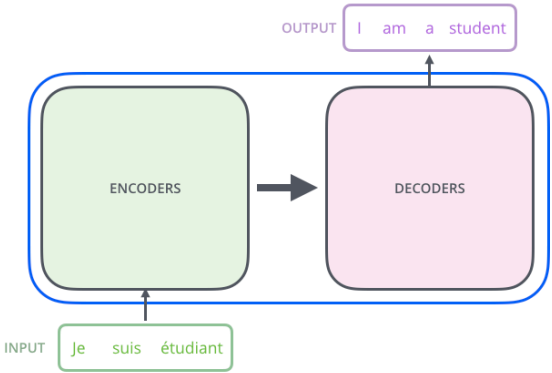
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

https://arxiv.org/abs/1706.03762

Transformer

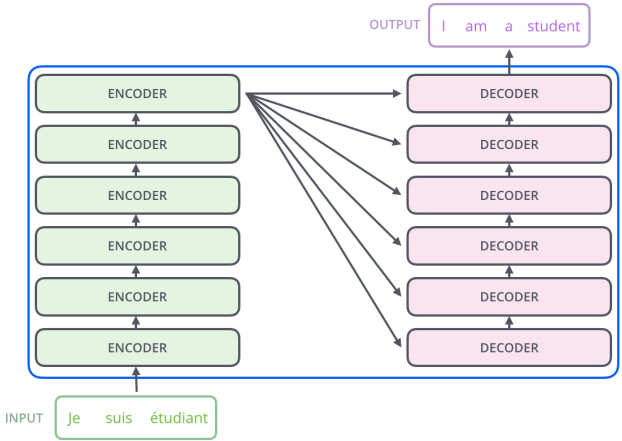
Верхнеуровнего - это просто энкодер и декодер



http://jalammar.github.io/illustrated-transformer/

Transformer

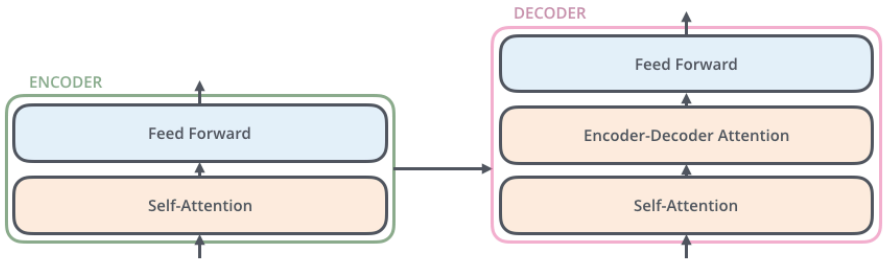
Энкодер и декодер состоят из одинаковых блоков; веса во всех блоках разные



http://jalammar.github.io/illustrated-transformer/

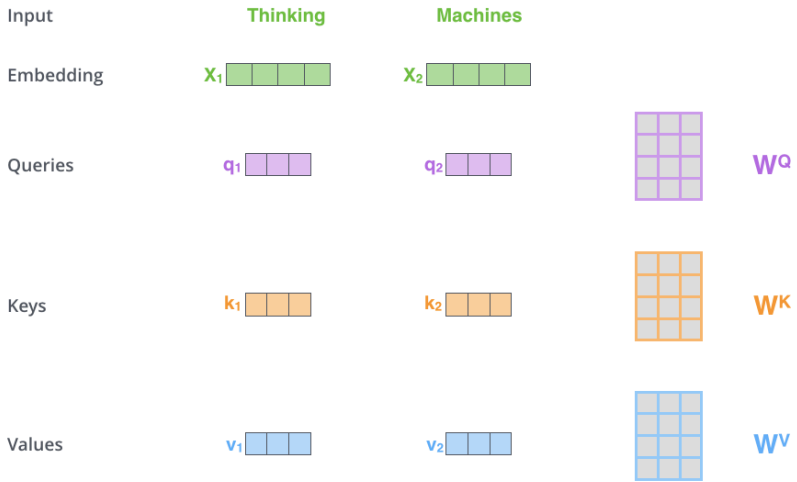
Transformer

В энкодере происходят две вещи: сначала вход прогоняется через self-attention, а затем — через полносвязный слой. В декодере помимо обычного self-attention есть ещё и attention из энкодера.



http://jalammar.github.io/illustrated-transformer/

Self-attention

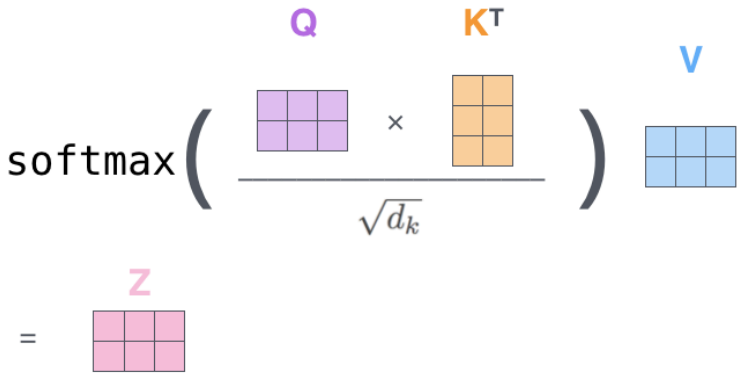


Абстракции

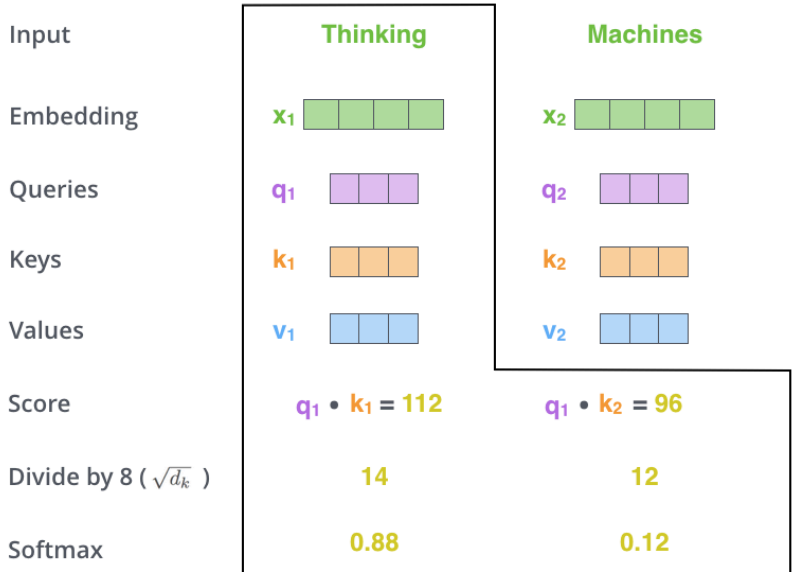
- Для каждого входного слова считаются три вектора: Query, Key и Value
- Матрицы W^Q, W^K, W^V обучаются вместе с моделью
- Value - то, что мы знаем об этом слове
- Query, Key помогают искать связи между словами, мы ходим по всем словам и пытаемся понять насколько они связаны между собой
- Query - мое текущее слово, Key - мое слово с которым я сравниваю себя

Self-attention

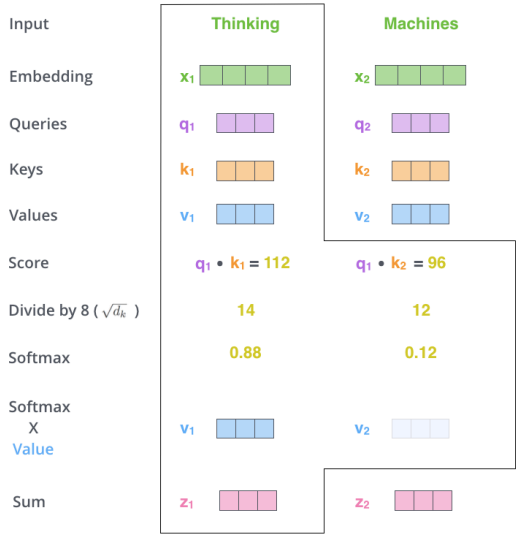
Цель этого слоя — сложить Value с некоторыми весами



Более детально



Более детально



Query, Key, Value

Each vector receives three representations (“roles”)

$W_Q \times = =$ Query: vector from which the attention is looking

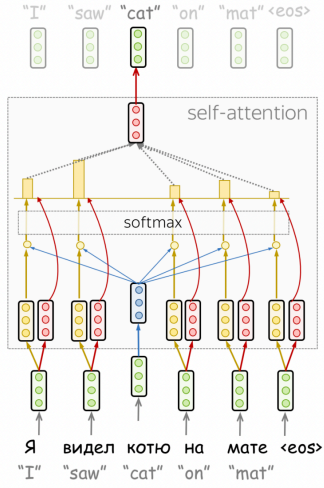
“Hey there, do you have this information?”

$W_K \times = =$ Key: vector at which the query looks to compute weights

“Hi, I have this information – give me a large weight!”

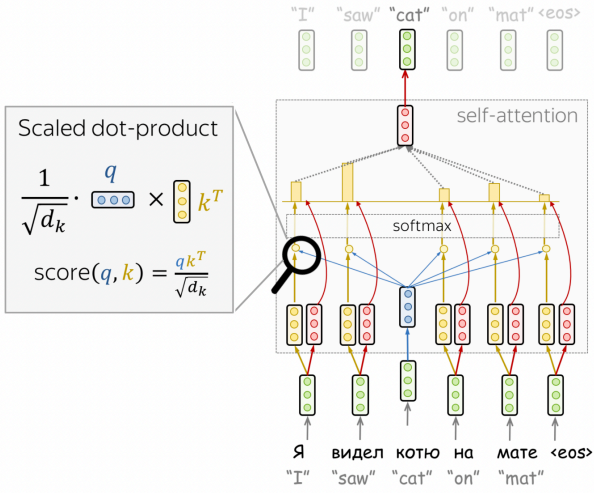
$W_V \times = =$ Value: their weighted sum is attention output

“Here’s the information I have!”



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Query, Key, Value

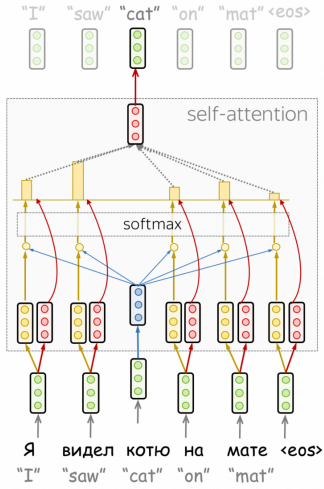


https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Query, Key, Value

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

from (q) · to (k, v) · vector dimensionality of K, V

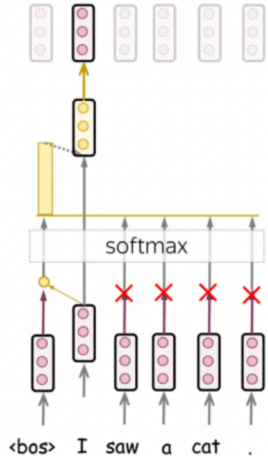


https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Masked Self-Attention

In the decoder, we forbid looking at future tokens – we don't know them

Note: in training, decoder processes all target tokens at once – without masks, it would see future



update token representation

gather context

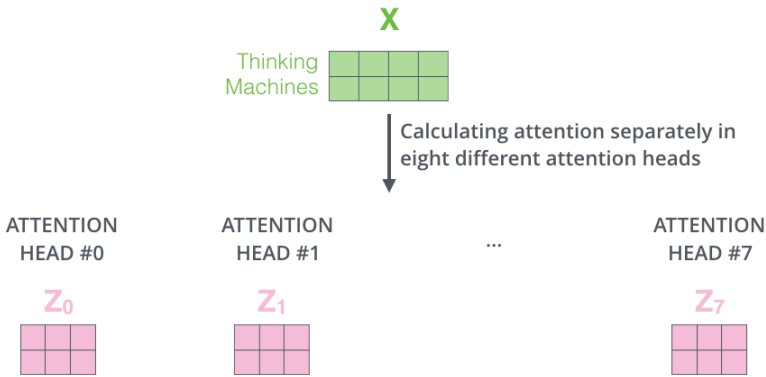
"look" at the previous tokens (future tokens are masked out)

https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

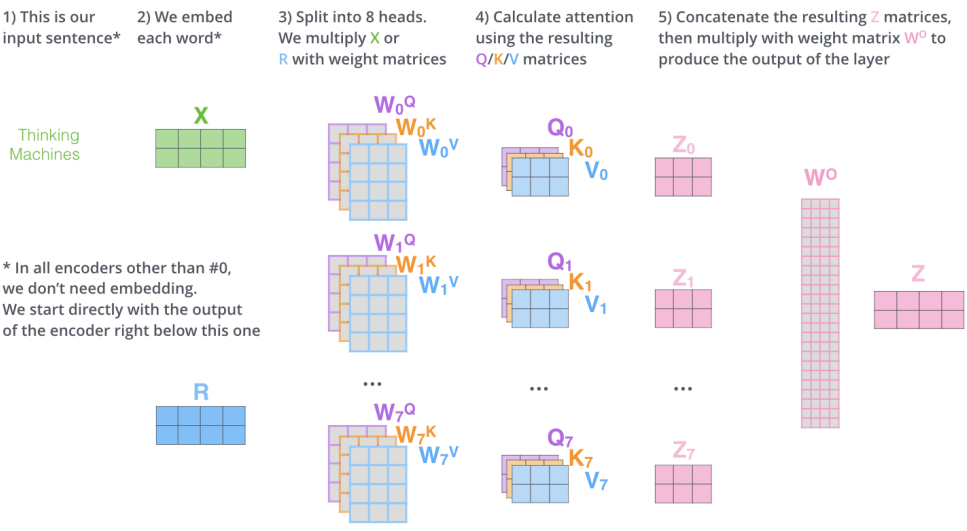
Зверь с кучей голов

Зверь с кучей голов

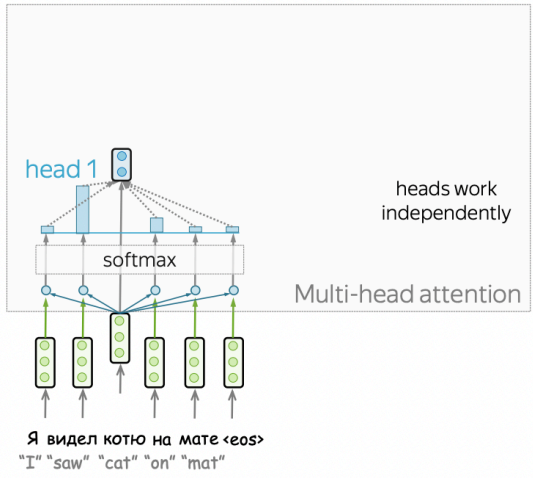
Несколько голов обеспечивают разное внимание



Слой целиком

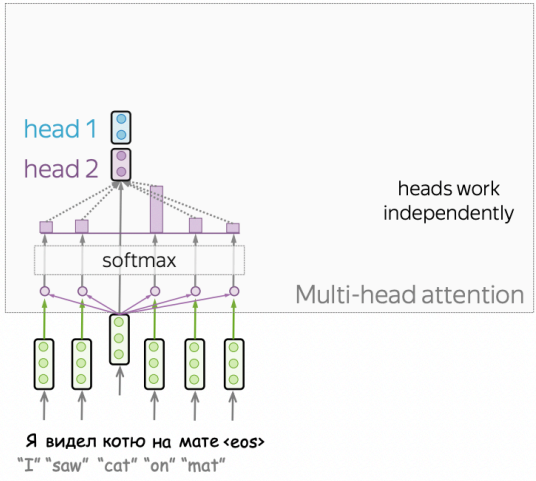


Multi-Head Attention



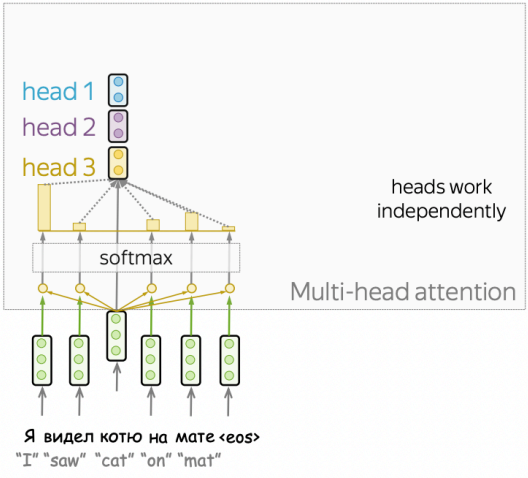
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention



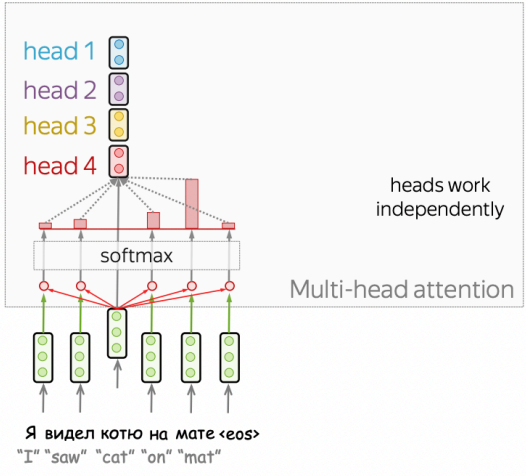
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention



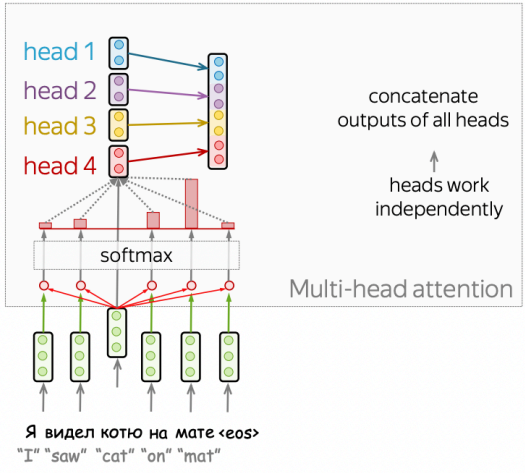
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention



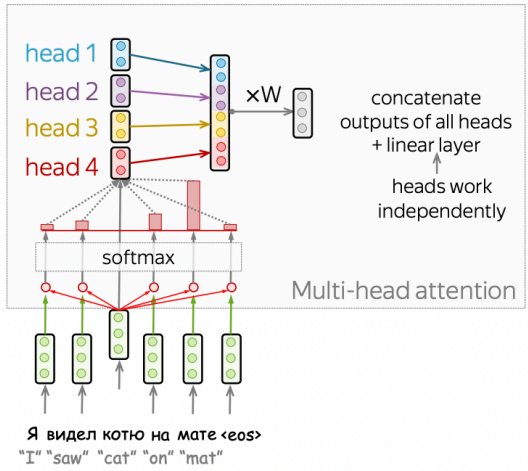
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention



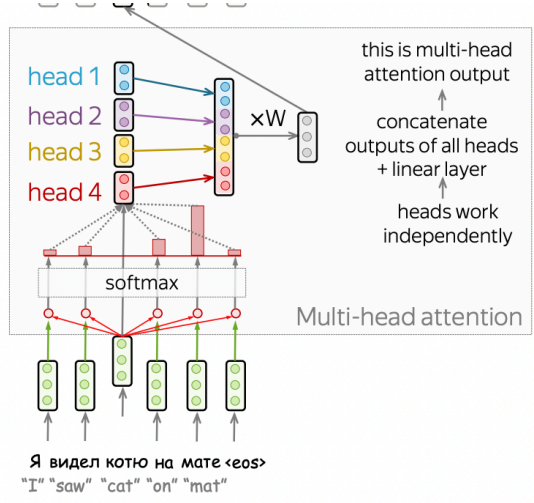
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention



https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Multi-Head Attention

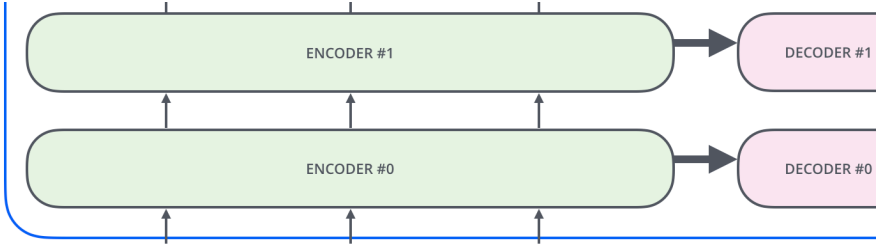


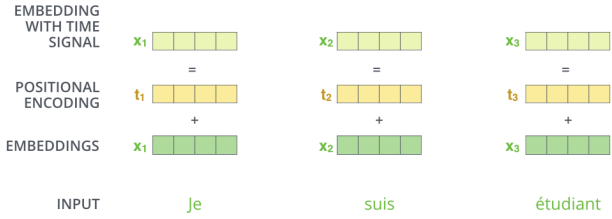
https://github.com/yandexdataschool/nlp_course/tree/2021/week04_seq2seq

Positional Encoding

Positional encoding

Для учёта позиции можно приплюсовать дополнительный вектор





Positional encoding

Можно закодировать позицию с помощью синусоиды или какого-нибудь OHE-вектора

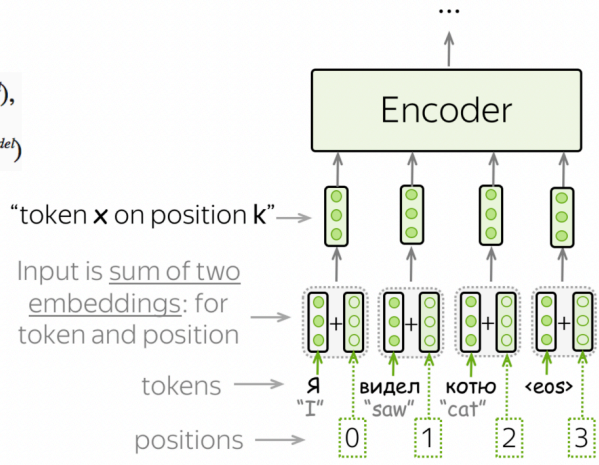
Positional encoding

Fixed encodings:

$$\mathrm{PE}_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$

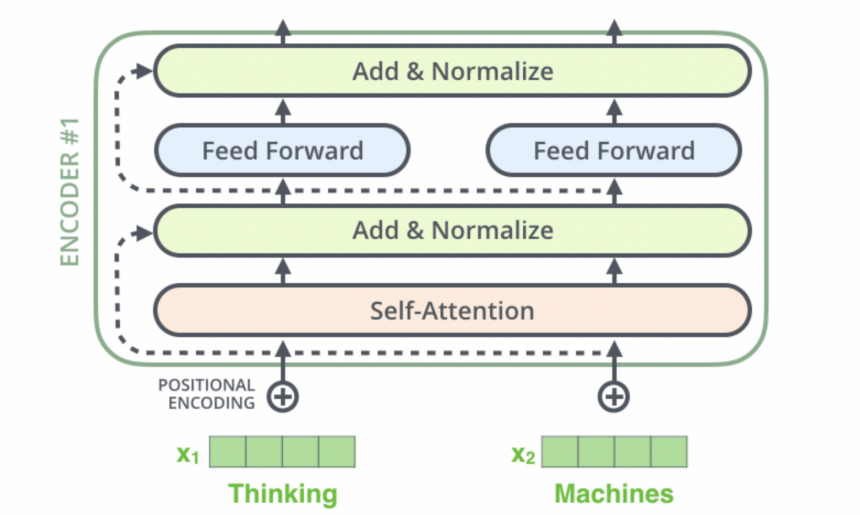
$$\mathrm{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

pos – position, i - dimension



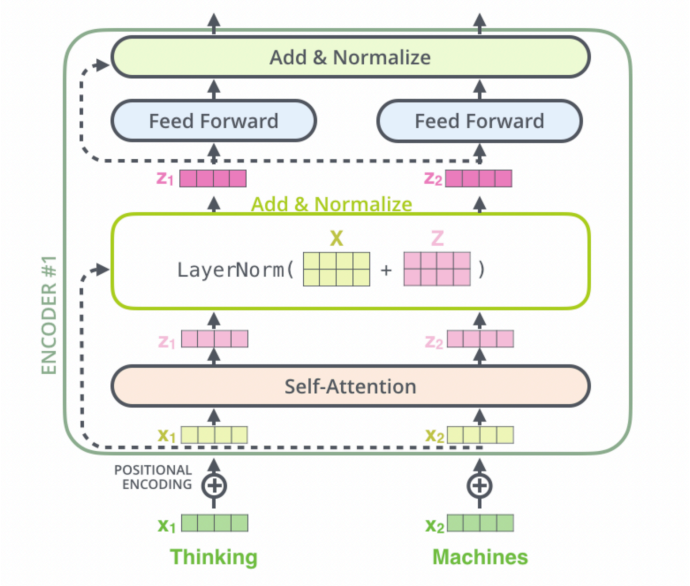
Энкодер и декодер

Что происходит в энкодере

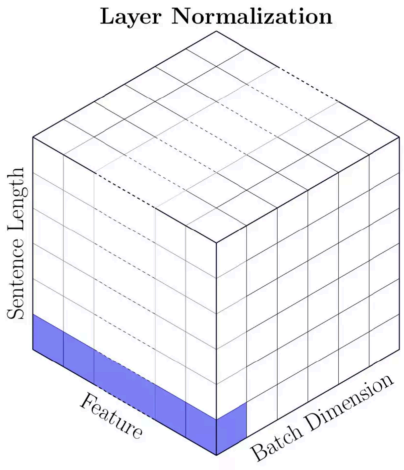


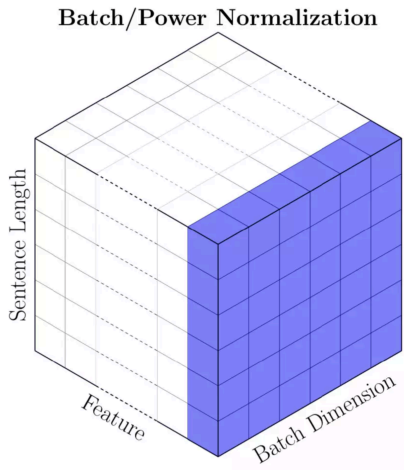
http://jalammar.github.io/illustrated-transformer/

Что происходит в энкодере

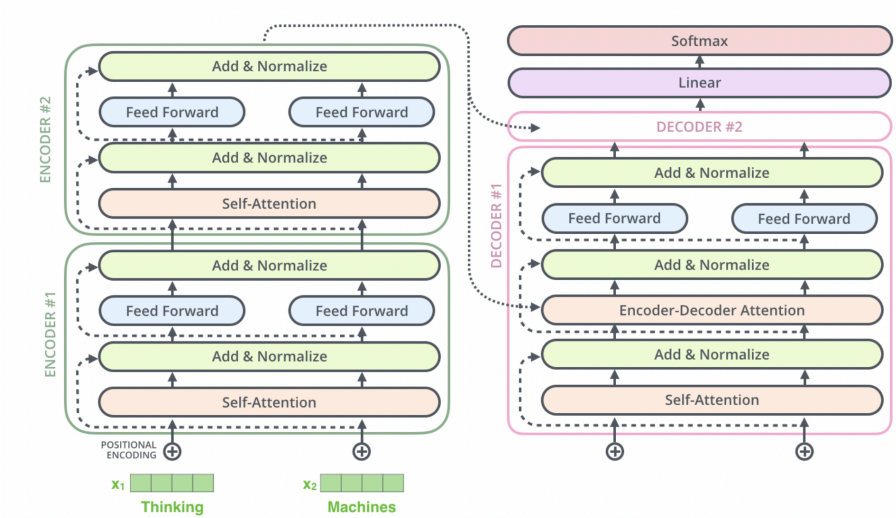


Layer Normalization



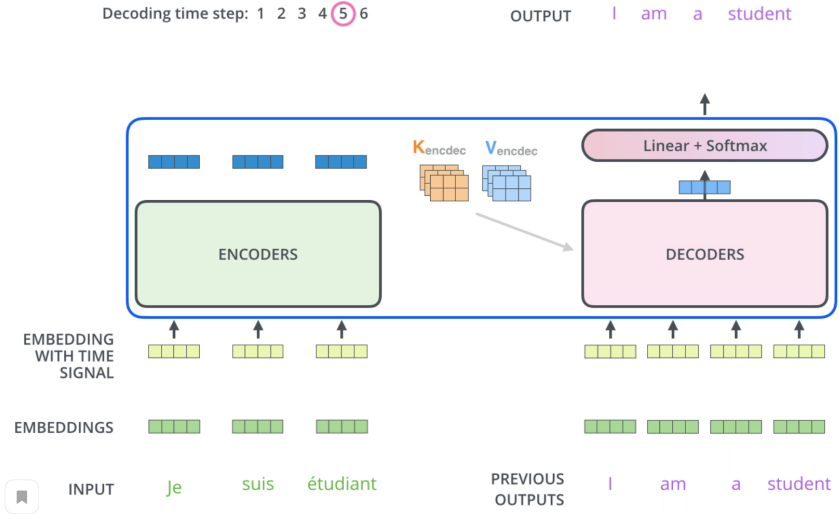


Две модели рядом



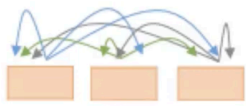
http://jalammar.github.io/illustrated-transformer/

Что происходит в декодере?



http://jalammar.github.io/illustrated-transformer/

Encoder-Encoder attention



Encoder Self-Attention

The one we have discussed earlier:

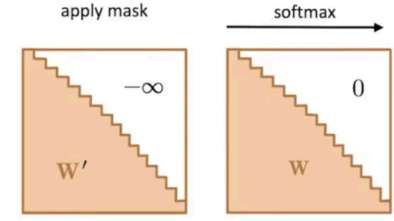
- Multi-head
- Scaled dot-product
- Self-attention: Q, K, V are computed from the same input matrix X

Update embeddings of tokens from the input sequence

Decoder-Decoder attention

Masked Decoder Self-Attention

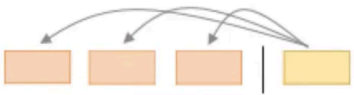
We need mask attention matrix to not look at future tokens during training:



$$W'_{ij} \leftarrow -\infty \text{ if } j > i \qquad W = \text{softmax}(W')$$

Attend to the previous words in the generated output sequence

Encoder-Decoder attention



This is not self-attention:

- Q from decoder
- K, V from encoder

Attend to tokens from the input sequence relevant for the generation of the next output token

Обучение

Trainig details

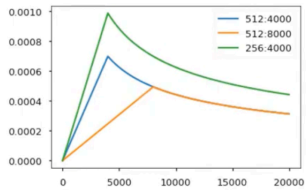
- Loss - standard NLL (cross-entropy) - lower is better:

$$NLL(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t|t_{t-1})$$

Instead perplexity is usually reported - higher is better:

$$Perplexity(y_{1:M}) = 2^{\frac{1}{M} NLL(y_{1:M})}$$

lr schedule



- Teacher forcing for decoder
- Adam optimizer
- Learning rate schedule with warm-up:

$$lr = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup^{-1.5})$$

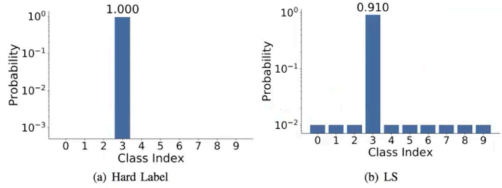
Trainig details

- Label smoothing

$$y_{ls} = (1 - \alpha) \cdot y_{hot} + \alpha / K$$

- Residual dropout - to the output of each sub-layer (before add+norm) + to the sums of the embeddings and the positional encodings
- BPE/Word-piece (shared embeddings for input/output)
- Model averaging (average last k checkpoints - SWA)

label smoothing

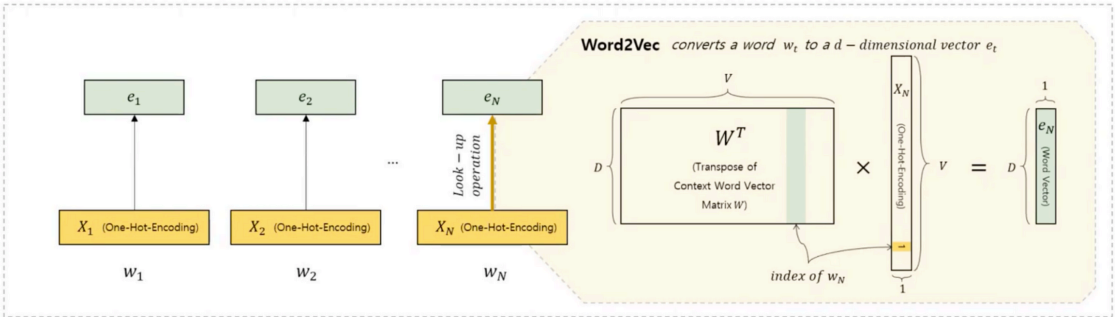


Contextualized Word Embeddings

Embeddings in NPL

- Большие объёмы неразмеченных данных в интернете в разных доменах (книги, новости, википедия, иные тексты из интернет-страниц)
- Размеченных данных мало. Качественная разметка дорогая и долгая
- Много вычислительных ресурсов, GPU, TPU, фреймворки распределённх вычислений
- Можем ли мы как-то заиспользовать имеющиеся ресурсы?

Individual embeddings



word2vec, GLOVE, fasttext …

Contextualised embeddings

- Обучаем большую модель трансформер на какой-нибудь unsupervised задаче на очень больших данных (очень долго, порядка нескольких недель);
- Дообучаем модель на конкретную задачку на малом корпусе размеченных данных (очень быстро, порядка 1 часа на одной ГПУ).

Contextualised embeddings

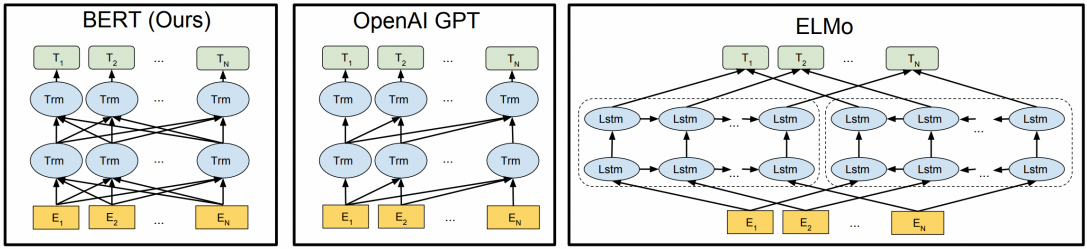


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

https://arxiv.org/pdf/1810.04805.pdf

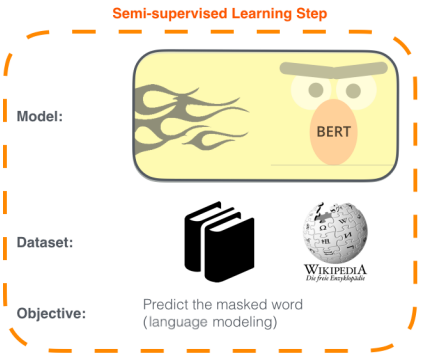
BERT (2018)

- Bidirectional Encoder Representations from Transformers
- Предобученный на искутсвенно придуманной задаче энкодер
- Его можно файн-тьюнить для решения других задач

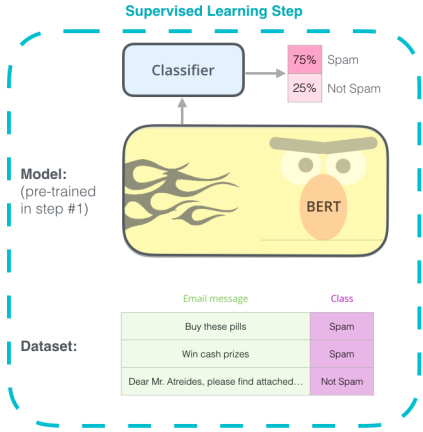
BERT (2018)

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



2 - Supervised training on a specific task with a labeled dataset.



http://jalammar.github.io/illustrated-bert/

BERT

- Masked Language Model: выкидываем часть слов (обычно 15%), модель должна их восстановить:
  - 80% меняем на токен <MASK>
  - 10% меняем на случайные
  - 10% не меняем

  Мы предсказываем пропуски, считаем Loss по всем 15%. Случайные и неизменные токены позволяют не переобучиться под токен <MASK>.

- Next Sentence Prediction: подаём на вход два предложения, модель должна понять в каком порядке они идут. Правда ли второе является продолжением первого. Эта задача помогает модели понять концепцию предложения.

CLS-токен

Пример токенезированного предложения:

[CLS] my dog is cute [SEP] he likes play ##ing [SEP]

- [CLS] — специальный токен, который мы добавляем к началу любого предложения, он несет информацию о всей входной последовательности (а вход состоит из двух предложений)
- [SEP] — специальный токен, который говорит, что слева от него первое предложение, а справа — второе
- ## — специальный токен, обозначающий, что это кусок слова

CLS-токен

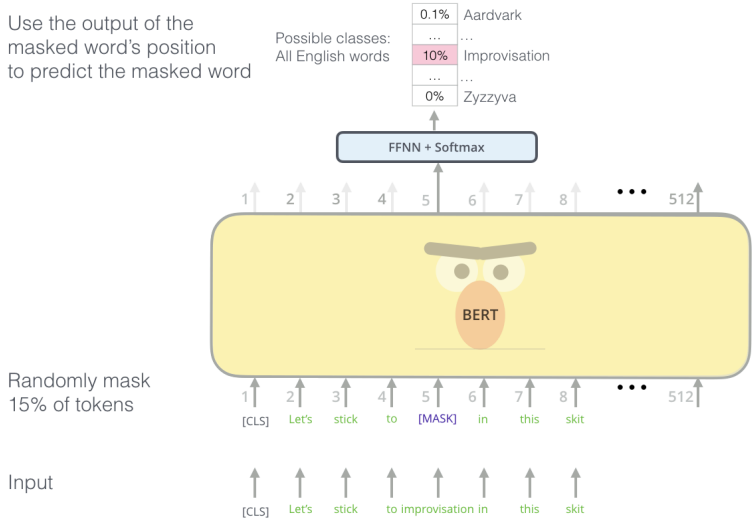
Пример токенезированного предложения:

[CLS] my dog is cute [SEP] he likes play ##ing [SEP]

Мы токенезируем предложение, для каждого токена учим эмбеддинг. Поскольку у нас трансформер, мы добавляем к эмбеддингам слов позиционный эмбеддинг и дополнительно добавляем эмбеддинг, отвечающий за то, в каком из двух предложений мы находимся. Все три эмбеддинга суммируются.

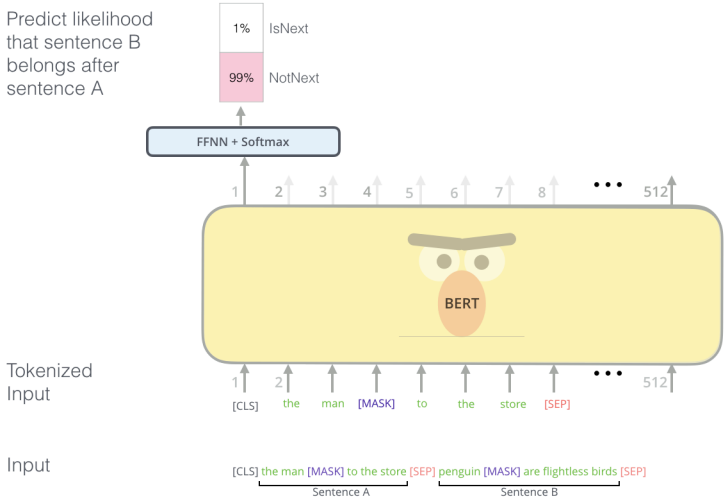
$$E = E_{word} + E_{pos} + E_{sent}$$

BERT



http://jalammar.github.io/illustrated-bert/

BERT



http://jalammar.github.io/illustrated-bert/

Как применять (finetuning)

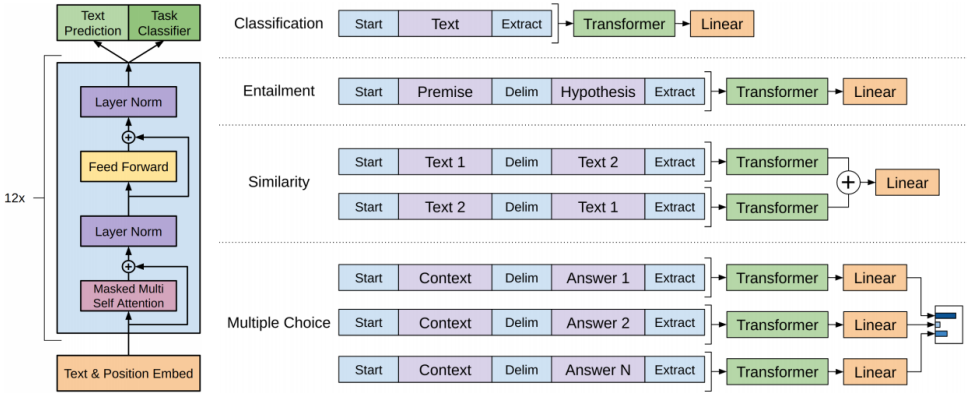


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

RoBERTa: A Robustly Optimized BERT (2019)

BERT был не дотюнен

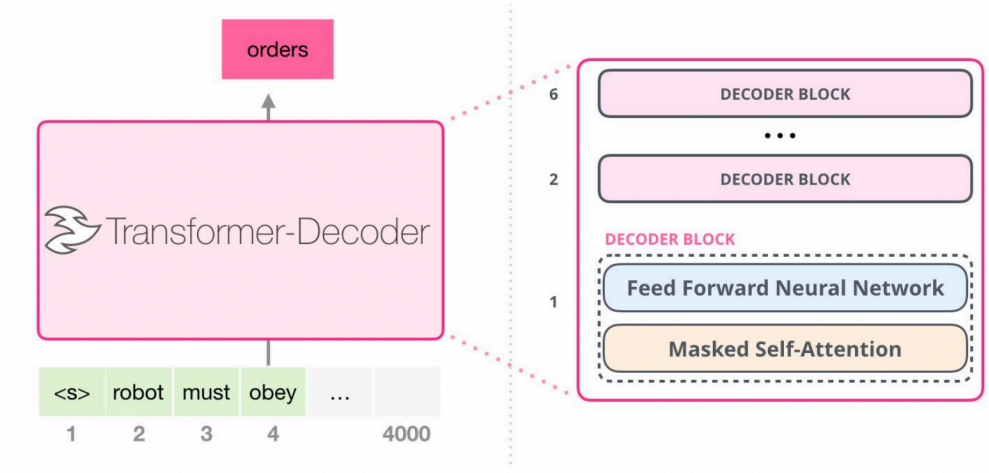
- Взять больше данных, тренировать дольше
- Next sentence prediction лишний
- Более длинные предложения
- Большие батчи
- Динамическое маскирование

https://arxiv.org/abs/1907.11692

GPT (Generative Pre-trained Transformer)

- Языковая модель на декодере трансформера
- Умеет генерить продолжение текста. Настолько хорошо, что OpenAI отказался её публиковать и устроил мощный PR
- Публикует понемногу, начиная с маленьких моделей
- Разные языковые модели на трансформерах можно попробовать здесь: https://transformer.huggingface.co/

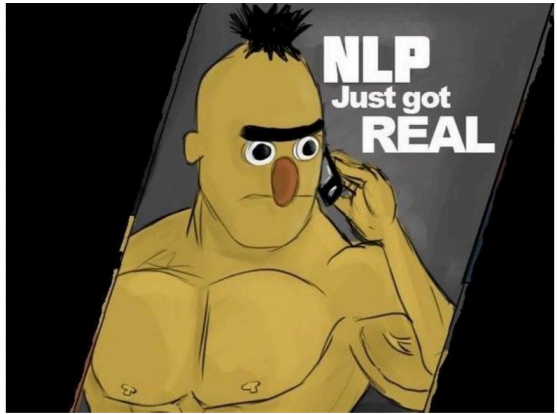
GPT



https://github.com/openai/gpt-2

Language Model Zoo

- ELMo
- ULMFiT
- GPT
- BERT (BioBERT, ClinicalBERT, …)
- ERNIE
- XLNet
- KERMIT
- ERNIE 2.0
- GPT-2
- ALBERT
- …



https://arxiv.org/pdf/1810.04805.pdf