Аттеншен, Трансформеры и улица Сезам
(CMMO, 2020)

Руслан Хайдуров

Майнор ИАД
НИУ ВШЭ

December 10, 2020

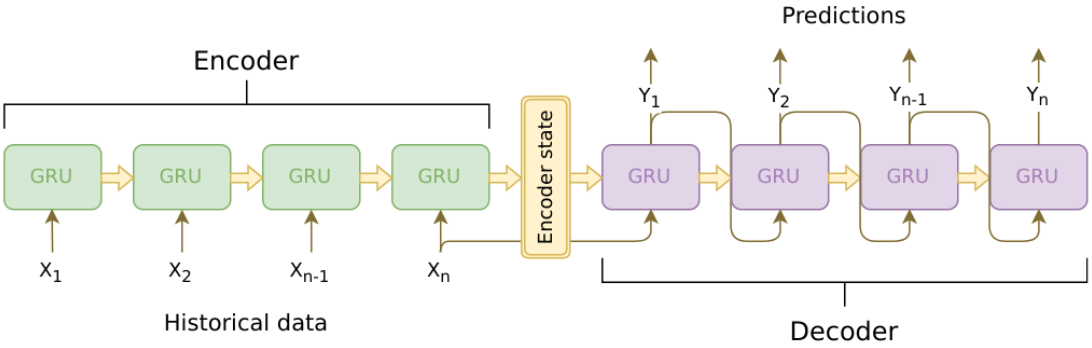
Ради чего мы собрались?

- Recap: RNN+Attention;
- Трансформеры: месть полносвязных;
- ELMO, BERT и прочие персонажи «Улицы Сезам»;
- Практические советы по использованию перечисленного.

Что делать с «забыванием» контекста?

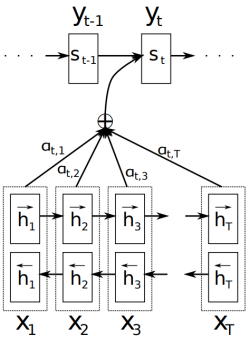
В чём проблема?

- В rnn-based способах решения seq2seq задач предложение произвольной длины кодируется в вектор фиксированного размера;
- **В длинных предложениях теряется контекст.**

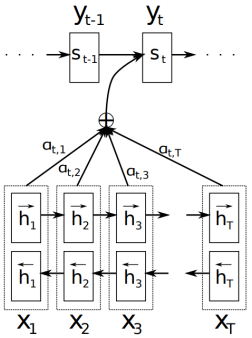


Механизм внимания (attention)

- При ручном переводе человек смотрит лишь на релевантный контекст каждого слова;
- **Идея:** *научить* нейросеть смотреть в нужные места исходной последовательности;
- Подавать на вход декодеру не итоговый hidden state, а все hidden state'ы.



Attention: explained



- Коэффициенты α_{ij} вычисляются по формуле

$$\alpha_{ij} = \frac{\exp(\mathrm{sim}(h_i, s_{j-1})}{\sum_{k=1}^{T} \exp(\mathrm{sim}(h_k, s_{j-1}))}$$

Attention: explained

- Коэффициенты α_{ij} вычисляются по формуле

$$\alpha_{ij} = \frac{\exp(\mathrm{sim}(h_i, s_{j-1})}{\sum_{k=1}^{T} \exp(\mathrm{sim}(h_k, s_{j-1}))}$$

- Дополнительный вход декодеру вычисляется по формуле:

$$c_j = \sum_{i=1}^{T} \alpha_{ij} h_i$$

- Скрытое состояние после декодинга и аутпут зависят от c_j, s_j и $\hat{y}_{j-1}$, то есть это некая функция $g(\hat{y}_{j-1}, s_j, c_j)$.

Как выбрать функцию sim

- Смысл функции sim — похожесть слов;
- Dot product:

$$\mathrm{sim}(h, s) = h^T s$$

- Additive attention:

$$\mathrm{sim}(h, s) = w^T \tanh(W_h h + W_s s)$$

- Multiplicative attention:

$$\mathrm{sim}(h, s) = h^T W s$$

W, W_s, W_h — обучаемые параметры.

Section 2

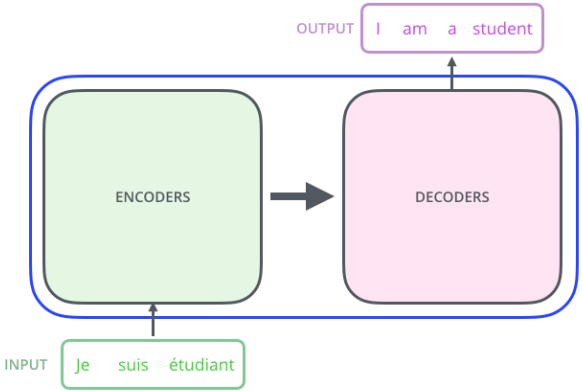
Как сделать seq2seq быстрее без потерь качества?

Attention is All You Need! (Waswani, et al., 2017)

- **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.

Transformer

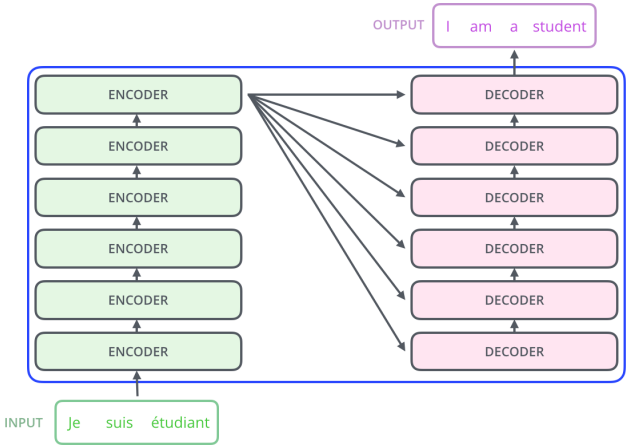
Верхнеуровнево это просто энкодер-декодер



[0]http://jalammar.github.io/illustrated-transformer/

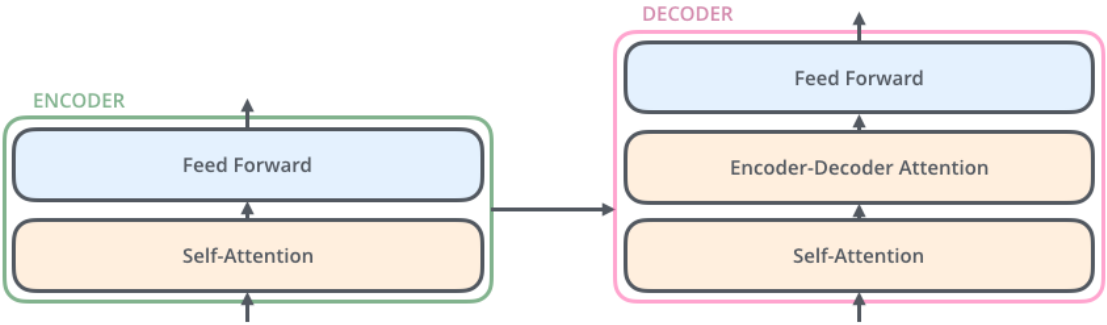
Transformer

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



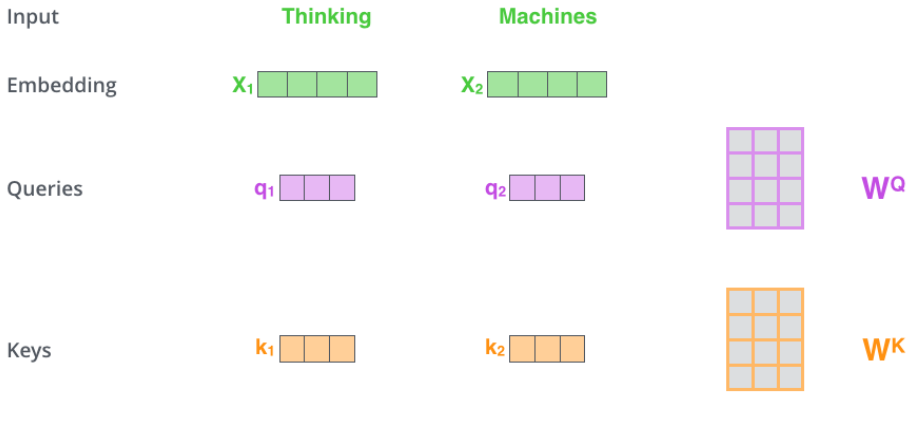
Transformer

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.



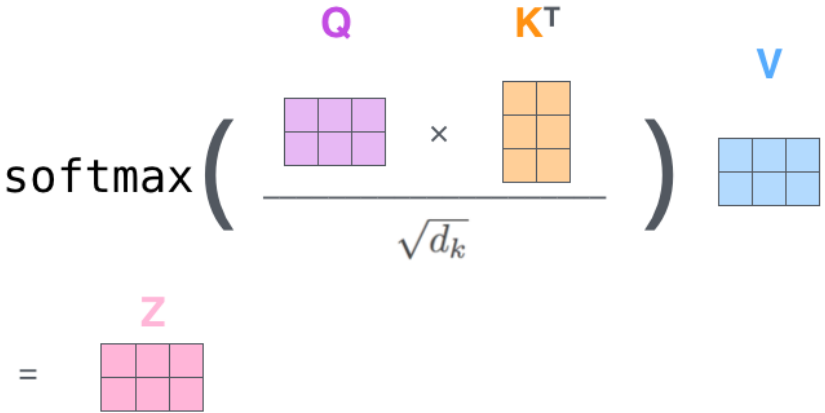
Self-attention

Для каждого входного слова считаются три вектора: Query, Key и Value (матрицы W^Q, W^K, W^V обучаются вместе с моделью).



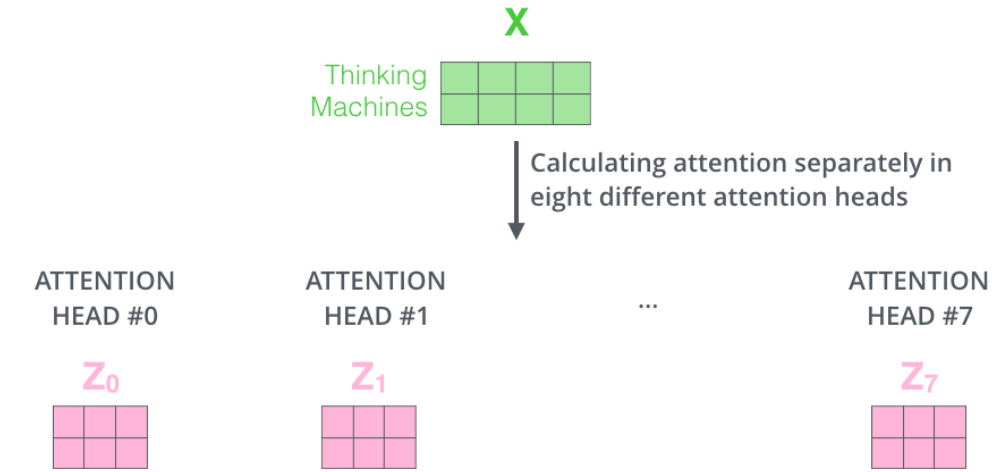
Self-attention

Цель этого слоя — сложить Value с некоторыми весами, образующими выпуклую комбинацию.



Multi-head Attention

Несколько голов обеспечивают разное внимание

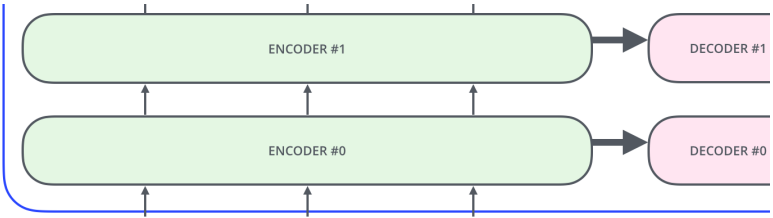


Positional Encoding

Для учёта позиции слова в предложении входные эмбеддинги преобразуются по следующему правилу

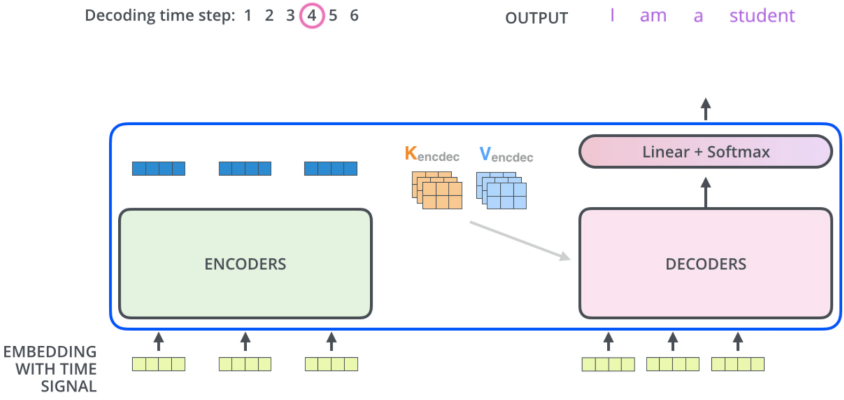
$$t_{pos,2i} = \sin(pos/(10000)^{2i/d_{mode}})$$

$$t_{pos,2i+1} = \cos(pos/(10000)^{(2i+1)/d_{mode}})$$



Ещё несколько хитростей

- Выходы последнего слоя энкодера подаются на вход всех промежуточных слоёв декодера (в encoder-decoder attention) в качестве Key и Value;
- В качестве Query — уже сгенерированные слова.



Transformer: модификации.

К настоящему времени человечество достигло следующего:

- 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?

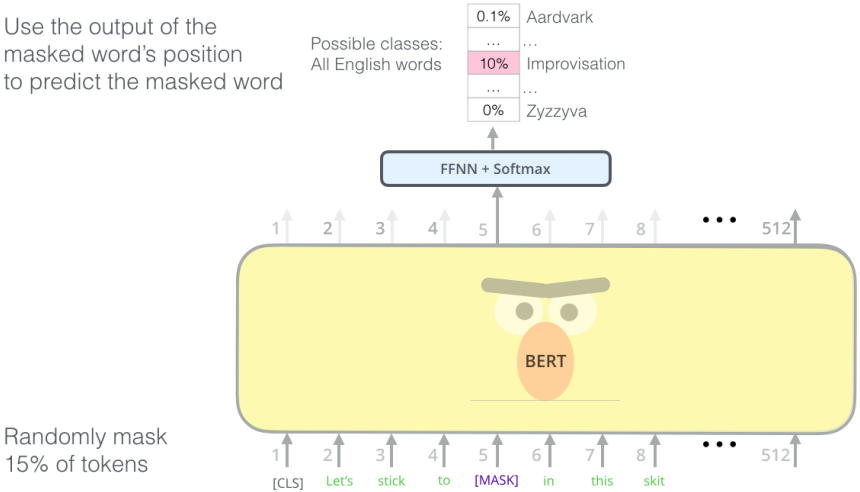
Transformer: модификации.

Да, можем! Использовать будем semi-supervised learning.

1. Обучаем большой трансформер на какой-нибудь unsupervised задаче на очень больших данных (очень долго, порядка нескольких недель на 64 гпу);
2. Дообучаем трансформер на малом корпусе размеченных данных (очень быстро, порядка 1 часа на одной ГПУ).

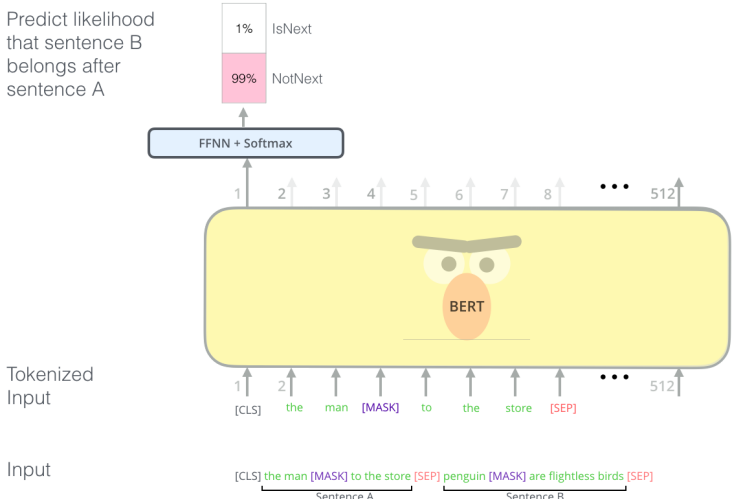
BERT (Devlin, et al., 2018)

Идея Берта: предобучать энкодер из трансформера на задаче Masked Language Modeling.



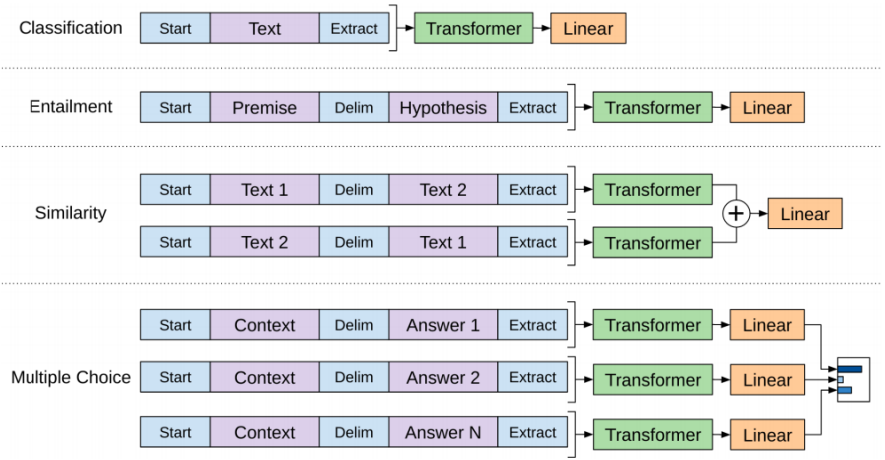
BERT (Devlin, et al., 2018)

а также на задаче Next Sentence Prediction:



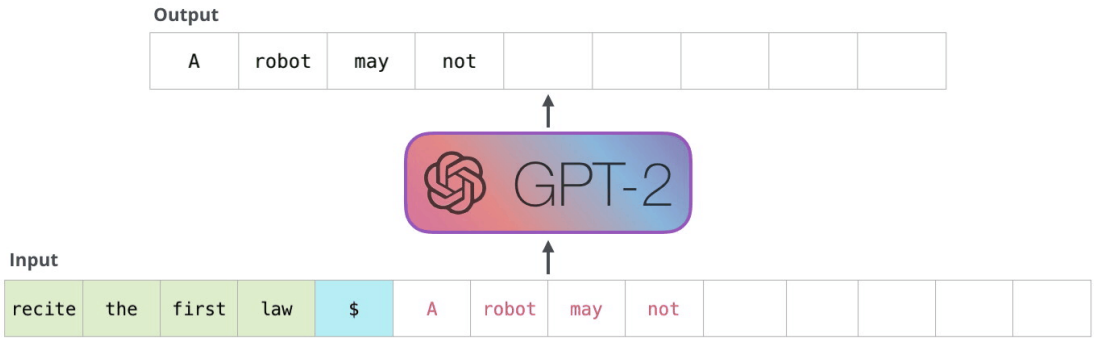
BERT (Devlin, et al., 2018)

После обучения unsupervised обучения на больших данных дообучить в supervised-режиме:



OpenAI GPT

Предобучение на большом корпусе текстов задаче Language Modeling



Полезности

- Трансформер и его слои встроены в pytorch;
- https://pytorch.org/tutorials/beginner/transformer_tutorial.html — официальный гайд по трансформеру от создателей pytorch;
- https://transformer.huggingface.co/ — поболтать с трансформером;
- Библиотеки: allennlp, fairseq, transformers, tensorflow-text — множествореализованных методов для трансформеров методов NLP.