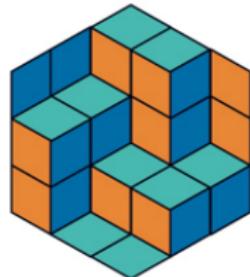


Лекция 5. Модели внимания и трансформеры

Александр Юрьевич Авдюшенко

МКН СПбГУ

17 марта 2022



Факультет
математики
и компьютерных
наук
СПбГУ

Пятиминутка

- ▶ Опишите (или нарисуйте) общую архитектуру автокодировщика
- ▶ Какое основное преимущество метода self-supervised learning?
- ▶ Выпишите функции потерь в GAN

Простейшая рекуррентная нейронная сеть (vanilla RNN)

напоминание

Оставляем один скрытый вектор h :

$$h_t = f_W(h_{t-1}, x_t)$$

В качестве функции f_W задаём линейное преобразование с нелинейной «сигмоидой» по компонентам:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Недостатки

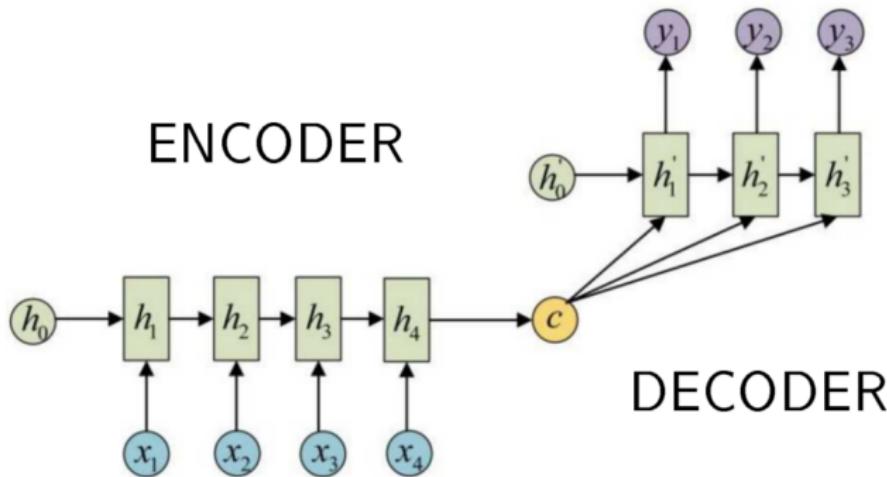
1. длины входного и выходного сигнала должны совпадать
2. «читаем» вход только слева-направо, не смотрим вперёд
3. как следствие, не подходит для задач machine translation, question answering и т.п.

RNN для синтеза последовательностей (seq2seq)

$X = (x_1, \dots, x_n)$ — входная последовательность

$Y = (y_1, \dots, y_m)$ — выходная последовательность

$c \equiv h_n$ кодирует всю информацию про X для синтеза Y



$$h_i = f_{in}(x_i, h_{i-1})$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$

$$y_t = f_y(h'_t, y_{t-1})$$

- ▶ h_n лучше помнит конец последовательности, чем начало
- ▶ чем больше n , тем труднее упаковать всю информацию в c
- ▶ придётся контролировать затухание и взрывы градиента
- ▶ RNN трудно распараллеливается

$$h_i = f_{in}(x_i, h_{i-1})$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$

$$y_t = f_y(h'_t, y_{t-1})$$

- ▶ h_n лучше помнит конец последовательности, чем начало
- ▶ чем больше n , тем труднее упаковать всю информацию в c
- ▶ придётся контролировать затухание и взрывы градиента
- ▶ RNN трудно распараллеливается

Вопрос

Как можно исправить часть перечисленных проблем?

$$h_i = f_{in}(x_i, h_{i-1})$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$

$$y_t = f_y(h'_t, y_{t-1})$$

- ▶ h_n лучше помнит конец последовательности, чем начало
- ▶ чем больше n , тем труднее упаковать всю информацию в c
- ▶ придётся контролировать затухание и взрывы градиента
- ▶ RNN трудно распараллеливается

Вопрос

Как можно исправить часть перечисленных проблем?

Подсказка

Как люди воспринимают информацию?

Посчитаем количество пасов

Баскетбол!

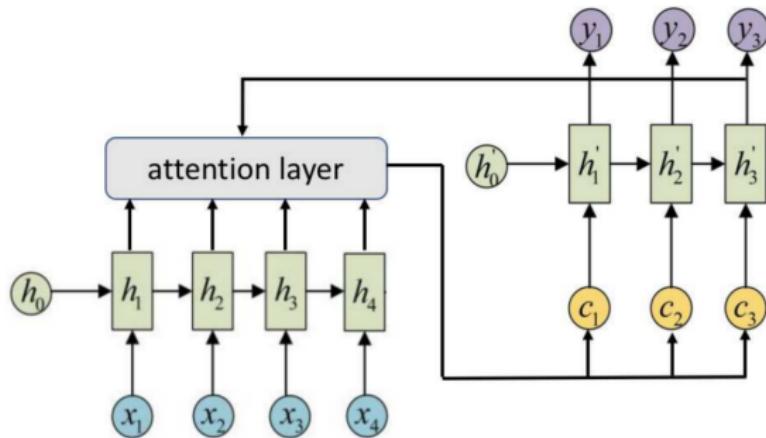
RNN с вниманием (attention mechanism)

$a(h, h')$ — функция сходства состояний входа h и выхода h'
(например, скалярное произведение или $\exp(h^T h')$ и другие)

α_{ti} — важность входа i для выхода t (attention score),

$$\sum_i \alpha_{ti} = 1$$

c_t — вектор входного контекста для выхода t (context vector)



$$h_i = f_{in}(x_i, h_{i-1})$$

$$\alpha_{ti} = \text{norm}_i a(h_i, h'_{t-1}), \text{ norm}_i(p) = \frac{p_i}{\sum_j p_j}$$

$$c_t = \sum_i \alpha_{ti} h_i$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c_t)$$

$$y_t = f_y(h'_t, y_{t-1}, c_t)$$

- ▶ можно вводить обучаемые параметры в a и c
- ▶ можно отказаться от рекуррентности как по h_i , так и по h'_t

Bahdanau et al. Neural machine translation by jointly learning to align and translate. 2015

Применение моделей внимания

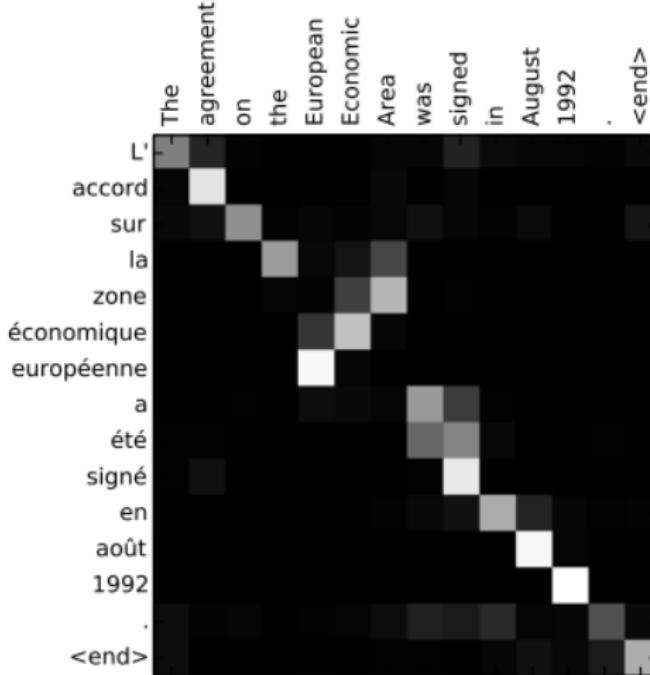
Преобразование одной последовательности в другую, seq2seq

- ▶ Машинный перевод
- ▶ Ответы на вопросы (question answering)
- ▶ Суммаризация текста (text summarization)
- ▶ Аннотация изображений, видео (multimedia description)
- ▶ Распознавание речи
- ▶ Синтез речи

Обработка последовательности

- ▶ Классификация текстовых документов
- ▶ Анализ тональности документа

Внимание в машинном переводе

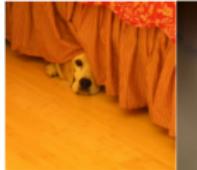


Интерпретируемость модели внимания: визуализация матрицы α_{ti}

Внимание для аннотирования изображений



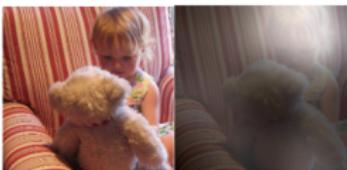
A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



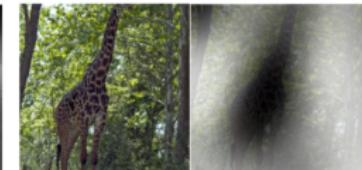
A stop sign is on a road with a mountain in the background.



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



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



A giraffe standing in a forest with trees in the background.

При генерации слова в описании изображения визуализация показывает, на какие области изображения модель обращает внимание.

Kelvin Xu et al. Show, attend and tell: neural image caption generation with visual attention. 2016

Функции сходства векторов

$a(h, h') = h^T h'$ — скалярное произведение

$a(h, h') = \exp(h^T h')$ — norm превращается в SoftMax

$a(h, h') = h^T W h'$ — с матрицей обучаемых параметров W

$a(h, h') = w^T \tanh(Uh + Vh')$ — аддитивное внимание с w, U, V

Линейные преобразования векторов query, key, value:

$$a(h_i, h'_{t-1}) = (W_k h_i)^T (W_q h'_{t-1}) / \sqrt{d}$$

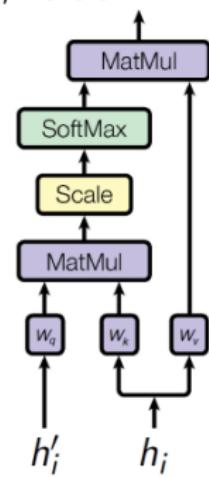
$$\alpha_{ti} = \text{SoftMax}_i a(h_i, h'_{t-1})$$

$$c_t = \sum_i \alpha_{ti} W_v h_i$$

W_q $d \times \text{dim}(h')$, W_k $d \times \text{dim}(h)$, W_v $d \times \text{dim}(h)$ —

матрицы весов линейных нейронов

Возможно упрощение $W_k \equiv W_v$



Dichao Hu. An introductory survey on attention mechanisms in NLP problems. 2018

Формула внимания

q — вектор-запрос, для которого хотим вычислить контекст
 $K = (k_1, \dots, k_n)$ — векторы-ключи, сравниваемые с запросом
 $V = (v_i, \dots, v_n)$ — векторы-значения, образующие контекст
 $a(k_i, q)$ — оценка релевантности (сходства) ключа k_i запросу q
 c — искомый вектор контекста, релевантный запросу

Модель внимания

Это 3х-слойная сеть, вычисляющая выпуклую комбинацию значений v_i , релевантных запросу q :

$$c = \text{Attn}(q, K, V) = \sum_i v_i \text{SoftMax}_i a(k_i, q)$$

$c_t = \text{Attn}(\mathcal{W}_q h'_{t-1}, \mathcal{W}_k H, \mathcal{W}_v H)$ — пример с предыдущего слайда, где $H = (h_1, \dots, h_n)$ — входные векторы, h'_{t-1} — выходной

Внутреннее внимание или «самовнимание» (self-attention):

$c_i = \text{Attn}(\mathcal{W}_q h_i, \mathcal{W}_k H, \mathcal{W}_v H)$ — частный случай, когда $h' \in H$

Многомерное внимание (multi-head attention)

Идея: J разных моделей внимания совместно обучаются выделять различные аспекты входной информации (например, части речи, синтаксис, фразеологизмы):

$$c^j = \text{Attn}(W_q^j q, W_k^j H, W_v^j H), j = 1, \dots, J$$

Варианты агрегирования выходного вектора:

$$c = \frac{1}{J} \sum_{j=1}^J c^j \text{ — усреднение}$$

$$c = [c^1 \dots c^J] \text{ — конкатенация}$$

$$c = [c^1 \dots c^J] W \text{ — чтобы вернуться к нужной размерности}$$

Регуляризация: чтобы аспекты внимания были максимально различны, строки $J \times n$ матриц A , $\alpha_{ji} = \text{SoftMax}_i a(W_k^j h_i, W_q^j q)$, декоррелируются ($\alpha_s^T \alpha_j \rightarrow 0$) и разреживаются ($\alpha_j^T \alpha_j \rightarrow 1$):

$$\|AA^T - I\|^2 \rightarrow \min_{\{W_k^j, W_q^j\}}$$

Не рассматриваем подробно в этой лекции

- ▶ иерархическое внимание (например, для классификации документов): слова \in предложения \in документы
- ▶ Graph Attention Network (GAT): многоклассовая multi-label классификация вершин графа

Z. Yang, A. Smola et al. Hierarchical attention networks for document classification. 2016

Petar Velickovic et al. Graph Attention Networks. ICLR-2018



Трансформер для машинного перевода

Трансформер (transformer) — это нейросетевая архитектура на основе моделей внимания и полносвязных слоёв, без RNN

Схема преобразований данных в машинном переводе:

- ▶ $S = (w_1, \dots, w_n)$ — слова предложения на входном языке
↓ обучаемая или пред-обученная векторизация слов ↓
- ▶ $X = (x_1, \dots, x_n)$ — эмбединги слов входного предложения
↓ трансформер-кодировщик ↓
- ▶ $Z = (z_1, \dots, z_n)$ — контекстные эмбединги
↓ трансформер-декодировщик ↓
- ▶ $Y = (y_1, \dots, y_m)$ — эмбединги слов выходного предложения
↓ генерация слов из построенной языковой модели ↓
- ▶ $\tilde{S} = (\tilde{w}_1, \dots, \tilde{w}_m)$ — слова предложения на выходном языке

Архитектура трансформера-кодировщика

1. Добавляются позиционные векторы p_i :

$$h_i = x_i + p_i, \quad H = (h_1, \dots, h_n) \quad \begin{matrix} d = \dim x_i, p_i, h_i = 512 \\ \dim H = 512 \times n \end{matrix}$$

2. Многомерное самовнимание:

$$h'_i = \text{Attn}(W_q^j h_i, W_k^j H, W_v^j H) \quad \begin{matrix} j = 1, \dots, J = 8 \\ \dim h'_i = 64 \\ \dim W_q^j, W_k^j, W_v^j = 64 \times 512 \end{matrix}$$

3. Конкатенация:

$$h'_i = \text{MH}_j(h'_i) \equiv [h_i^1 \cdots h_i^J] \quad \dim h'_i = 512$$

4. Сквозная связь + нормировка уровня:

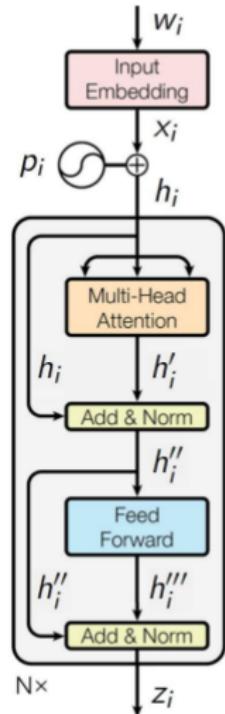
$$h''_i = \text{LN}(h'_i + h_i; \mu_1, \sigma_1) \quad \dim h''_i, \mu_1, \sigma_1 = 512$$

5. Полносвязная 2x-слойная сеть FFN:

$$h'''_i = W_2 \text{ReLU}(W_1 h''_i + b_1) + b_2 \quad \begin{matrix} \dim W_1 = 2048 \times 512 \\ \dim W_2 = 512 \times 2048 \end{matrix}$$

6. Сквозная связь + нормировка уровня:

$$z_i = \text{LN}(h'''_i + h''_i; \mu_2, \sigma_2) \quad \dim z_i, \mu_2, \sigma_2 = 512$$



Дополнения и замечания

- ▶ ещё таких блоков $N = 6$, $h_i \rightarrow \square \rightarrow z_i$ соединены последовательно
- ▶ вычисления легко распараллеливаются по x_i
- ▶ возможно использование пред-обученных эмбедингов x_i
- ▶ возможно обучение эмбедингов x_i слов $w_i \in V$
- ▶ слой нормировки уровня (Layer Normalization, LN),
 $x, \mu, \sigma \in \mathbb{R}^d$

$$LN_s(x; \mu, \sigma) = \sigma_s \frac{x_s - \bar{x}}{\sigma_x} + \mu_s$$

$$\bar{x} = \frac{1}{d} \sum_{s=1}^d x_s, \sigma_x^2 = \frac{1}{d} \sum_{s=1}^d (x_s - \bar{x})$$

Позиционное кодирование (positional encoding)

Позиции слов i кодируются векторами p_i так, что

- ▶ чем больше $|i - j|$, тем больше $\|p_i - p_j\|$
- ▶ число позиций не ограничено

Например,

$$c_j = \text{Attn}(q_j, K, V) = \sum_i (v_i + w_{i \boxminus j}^V) \text{SoftMax}_i a(k_i + w_{i \boxminus j}^k, q_j),$$

где $i \boxminus j = \max(\min(i - j, \delta), -\delta)$ — усеченная разность,
 $\delta = 5..16$

Shaw, Uszkoreit, Vaswani. Self-attention with relative position representations. 2018

Архитектура трансформера декодировщика

Авторегressiveный синтез последовательности:

$y_0 = \langle \text{BOS} \rangle$ — эмбединг символа начала;
для всех $t = 1, 2, \dots$:

1. Маскирование «данных из будущего»:

$$h_t = y_{t-1} + p_t; \quad H_t = (h_1, \dots, h_t)$$

2. Многомерное самовнимание:

$$h'_t = \text{LN} \circ \text{MH}_j \circ \text{Attn}(W_q^j h_t, W_k^j H_t, W_v^j H_t)$$

3. Многомерное внимание на кодировку Z :

$$h''_t = \text{LN} \circ \text{MH}_j \circ \text{Attn}(W_q^j h'_t, W_k^j Z, W_v^j Z)$$

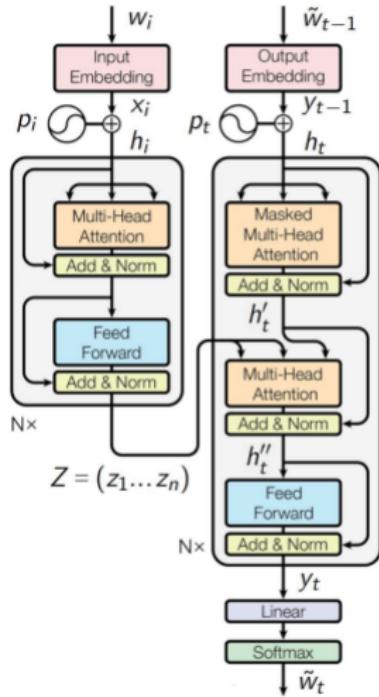
4. Двухслойная полносвязная сеть:

$$y_t = \text{LN} \circ \text{FFN}(h''_t)$$

5. Линейный предсказывающий слой:

$$p(\tilde{w}|t) = \text{SoftMax}_{\tilde{w}}(W_y y_t + b_y)$$

генерация $\tilde{w}_t = \arg \max_{\tilde{w}} p(\tilde{w}|t)$ пока $\tilde{w}_t \neq \langle \text{EOS} \rangle$



Vaswani et al. (Google) Attention is all you need. 2017.

Критерии обучения и валидации для машинного перевода

Критерий для обучения параметров нейронной сети W по обучающей выборке предложений S с переводом \tilde{S} :

$$\sum_{(S, \tilde{S})} \sum_{\tilde{w}_t \in \tilde{S}} \ln p(\tilde{w}_t | t, S, W) \rightarrow \max_W$$

Критерий оценивания моделей (недифференцируемый) по выборке пар предложений «перевод S , эталон S_0 »:
BiLingual Evaluation Understudy:

$$\text{BLEU} = \min \left(1, \frac{\sum \text{len}(S)}{\sum \text{len}(S_0)} \right) \times \text{mean}_{(S_0, S)} \left(\prod_{n=1}^4 \frac{\# n\text{-грамм из } S, \text{ входящих в } S_0}{\# n\text{-грамм в } S} \right)^{\frac{1}{4}}$$



BERT — Bidirectional Encoder Representations from Transformers

Трансформер BERT — это кодировщик без декодировщика, предобучаемый для решения широкого класса задач NLP
Схема преобразований данных в задачах NLP:

- ▶ $S = (w_1, \dots, w_n)$ — слова предложения на входном языке
↓ обучение эмбедингов вместе с трансформером ↓
- ▶ $X = (x_1, \dots, x_n)$ — эмбединги слов входного предложения
↓ трансформер-кодировщик ↓
- ▶ $Z = (z_1, \dots, z_n)$ — контекстные эмбединги
↓ дообучение на конкретную задачу ↓
- ▶ Y — выходной текст / разметка / классификация и т.п.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (Google AI Language) BERT: pre-training of deep bidirectional transformers for language understanding. 2019.

Критерий MLM (masked language modeling) для обучения BERT

Критерий маскированного языкового моделирования MLM строится автоматически по текстам (self-supervised learning):

$$\sum_S \sum_{i \in M(S)} \ln p(w_i | i, S, \mathbf{W}) \rightarrow \max_{\mathbf{W}}$$

где $M(S)$ — подмножество маскированных токенов из S ,

$$p(w | i, S, \mathbf{W}) = \text{SoftMax}_{w \in V}(\mathbf{W}_z z_i(S, \mathbf{W}_T) + \mathbf{b}_z)$$

— языковая модель, предсказывающая i -й токен предложения S , $z_i(S, \mathbf{W}_T)$ — контекстный эмбединг i -го токена предложения S на выходе трансформера с параметрами \mathbf{W}_T ,
 \mathbf{W} — все параметры трансформера и языковой модели

Критерий NSP (next sentence prediction) для обучения BERT

Критерий предсказания связи между предложениями NSP строится автоматически по текстам (self-supervised learning):

$$\sum_{(S, S')} \ln p(y_{SS'} | S, S', W) \rightarrow \max_W,$$

где $y_{SS'} = [\text{за } S \text{ следует } S']$ — бинарная классификация пары предложений,

$$p(y|S, S', W) = \operatorname{SoftMax}_{y \in \{0,1\}} (W_y \tanh(W_s z_0(S, S', W_T) + b_s) + b_y)$$

— вероятностная модель классификации пар (S, S') ,
 $z_0(S, S', W_T)$ — контекстный эмбединг токена $\langle \text{CLS} \rangle$ для пары предложений, записанной в виде $\langle \text{CLS} \rangle S \langle \text{SEP} \rangle S' \langle \text{SEP} \rangle$

Ещё несколько замечаний про трансформеры

- ▶ Fine-tuning: для дообучения на задаче задаётся модель $f(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборка $\{S\}$ и критерий $\mathcal{L}(S, f) \rightarrow \max$
- ▶ Multi-task learning: для дообучения на наборе задач $\{t\}$ задаются модели $f_t(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборки $\{S\}_t$ и сумма критериев $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \rightarrow \max$
- ▶ GLUE, SuperGLUE, Russian SuperGLUE — наборы тестовых задач на понимание естественного языка
- ▶ Трансформеры обычно строятся не на словах, а на токенах, получаемых BPE (Byte-Pair Encoding) или WordPiece

Ещё несколько замечаний про трансформеры

- ▶ Fine-tuning: для дообучения на задаче задаётся модель $f(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборка $\{S\}$ и критерий $\mathcal{L}(S, f) \rightarrow \max$
- ▶ Multi-task learning: для дообучения на наборе задач $\{t\}$ задаются модели $f_t(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборки $\{S\}_t$ и сумма критериев $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \rightarrow \max$
- ▶ GLUE, SuperGLUE, Russian SuperGLUE — наборы тестовых задач на понимание естественного языка
- ▶ Трансформеры обычно строятся не на словах, а на токенах, получаемых BPE (Byte-Pair Encoding) или WordPiece
- ▶ Первый трансформер: $N = 6, d = 512, J = 8$, весов 65M
- ▶ BERT_{BASE}, GPT-1: $N = 12, d = 768, J = 12$, весов 110M
- ▶ BERT_{LARGE}, GPT-2: $N = 24, d = 1024, J = 16$, весов 340M-1000M

Ещё несколько замечаний про трансформеры

- ▶ Fine-tuning: для дообучения на задаче задаётся модель $f(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборка $\{S\}$ и критерий $\mathcal{L}(S, f) \rightarrow \max$
- ▶ Multi-task learning: для дообучения на наборе задач $\{t\}$ задаются модели $f_t(Z(S, \mathbf{W}_T), \mathbf{W}_f)$, выборки $\{S\}_t$ и сумма критериев $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \rightarrow \max$
- ▶ GLUE, SuperGLUE, Russian SuperGLUE — наборы тестовых задач на понимание естественного языка
- ▶ Трансформеры обычно строятся не на словах, а на токенах, получаемых BPE (Byte-Pair Encoding) или WordPiece
- ▶ Первый трансформер: $N = 6, d = 512, J = 8$, весов 65M
- ▶ BERT_{BASE}, GPT-1: $N = 12, d = 768, J = 12$, весов 110M
- ▶ BERT_{LARGE}, GPT-2: $N = 24, d = 1024, J = 16$, весов 340M-1000M
- ▶ GPT-3: весов 175 млрд
- ▶ Gopher (DeepMind): 280 млрд
- ▶ Turing-Megatron (Microsoft + Nvidia): 530 млрд

Резюме

- ▶ Модели внимания сначала встраивались в RNN или CNN, но оказалось, что они самодостаточны
- ▶ На их основе разработана архитектура Трансформера, различные варианты которого (BERT, GPT-3, XLNet, ELECTRA и др.) на данный момент являются SotA в задачах обработки естественного языка, и не только его
- ▶ Доказано, что модель внимания multi-head self-attention (MHSA) эквивалентна свёрточной сети [Cordonnier, 2020]

Vaswani et al. Attention is all you need. 2017.

Dichao Hu. An Introductory Survey on Attention Mechanisms in NLP Problems. 2018.

Xipeng Qiu et al. Pre-trained models for natural language processing: A survey. 2020.

Cordonnier et al. On the relationship between self-attention and convolutional layers. 2020

Что ещё можно посмотреть?

- ▶ Лекция К.В. Воронцова «Модели внимания и трансформеры»
- ▶ GPT-3 на википедии: <https://en.wikipedia.org/wiki/GPT-3>
- ▶ Григорий Сапунов о трансформерах в 2021
- ▶ Николай Григорьев (SE @ Deepmind) об истории и современных языковых моделях, 14 февраля 2022
- ▶ Семинар ММЦ Эйлера и МКН о формализации математических доказательств в Lean и глубоком обучении