Московский государственный университет
имени М.В. Ломиносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Эссе по курсу «Глубокое обучение»

Транфсормер

Выполнил:
студент 317 группы
Е.Д. Стулов

Москва, 2021

Аннотация

Данная работа представляет собой расширенный конспект лекций А.Г. Дьяконова по архитектуре глубоких нейронных сетей Transformer. В работе описываются основные идеи, в том числе пришедшие из классических seq2seq-архитектур, положенные в основу данной архитектуры, а также представлен обзор новых и наиболее популярных моделей, основанных на трансформере.

Содержание

Введение

Порождение выходной последовательности по входной находит применение во многих областях машинного обучения: от обработки естественного языка до генерации описаний объектов на фотографиях.

До недавнего времени наиболее эффективные seq2seq-модели основались на сложных рекуррентных или сверточных нейронных сетях, беря за основу подход encoder-decoder и механизм внимания. В 2017 году в статье [1] «Attention Is All You Need», была предложена новая архитектура, основанная исключительно на механизмах внимания, названная Transformer.

В данной работе представлен обзор некоторых популярных моделей, вдохновленных архитектурой трансформер, и рассмотрены основные идеи и принципы, легшие в основу этих моделей.

Класический seq2seq

1.1 Encoder-decoder

В простейшем варианте модель seq2seq представляет собой две последовательно соединенные рекурентные сети, называемые **encoder** и **decoder**. Первая RNN — encoder, принимает на вход последовательность векторных представлений токенов и генерирует так называемый **hidden state**, который подается на вход второй нейросети — decoder'у. Decoder, в свою очередь, служит для построения целевой последовательности по внутреннему состоянию.

Для большей простоты интерпритации можно рассмотреть задачу перевода: на вход кодировщику подается текст на исходном языке. Тогда hidden state можно интерпритировать как смысл этого текста, по которому затем декодировщик восстанавливает текст на целевом языке.

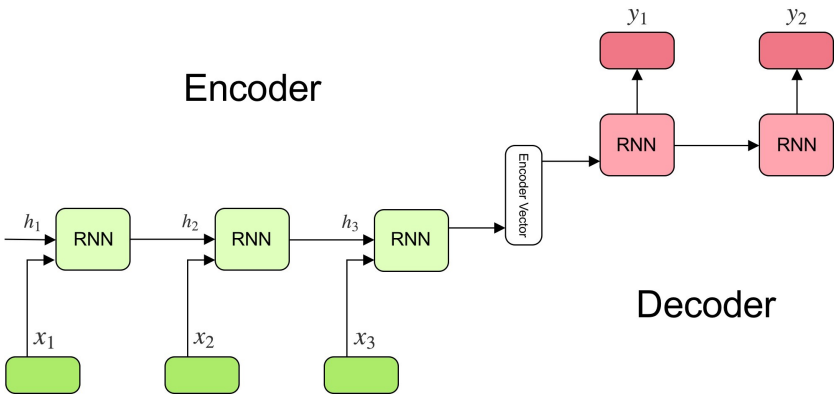

Рисунок 1.1: Модель encoder-decoder [2]

Недостаток такого подхода заключается в следующем: вся исходная последовательность должна быть закодирована некоторым конечномерным внутренним представлением, но если исходная последовательность

была достаточно длинной, этого может оказаться недостаточно, и качество работы модели ухудшится. Наиболее популяйный способ борьбы с этим недостатком — механизм внимания (attention).

1.2 Attention

Концепция **attention** состоит в предположении, что между токенами существуют некоторые взаимосвязи. При таком подходе кодировщик передает в декодировщик не одно состояние, кодирующее всю последовательность целиком, а набор состояний всех токенов. Для этого применяется пулинг состояний по схожести: на основе скрытых состояний элементов кодировщика, называемых **векторами-ключами** K, скрытого состояния текущего элемента декодировщика, называемого **вектором-запросом** q, и векторных представлений элементов целевой последовательности – **векторов-значений** V, формируется контекстный вектор как взвешенная сумма векторов-значений с такими весами, что чем больше соответствующий вектор-ключ схож с вектором-запросом, тем больше вес.

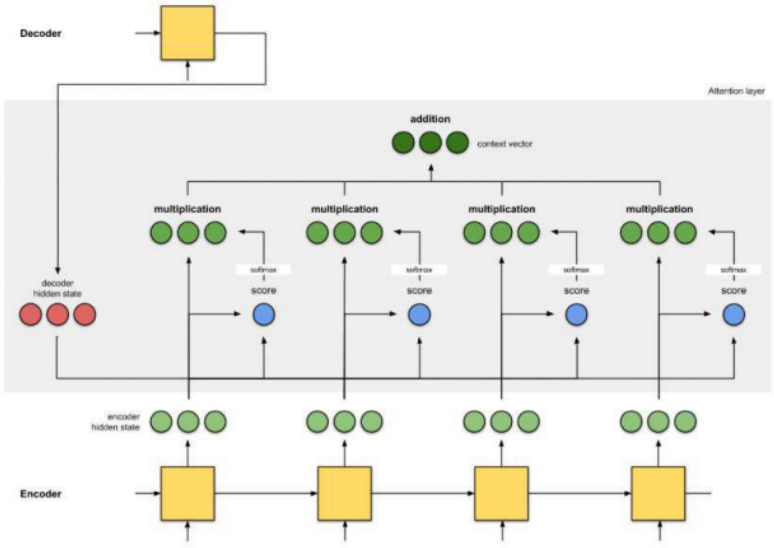

Рисунок 1.2: Seq2seq с механизмом внимания

Часто на практике в качестве меры схожести используется скалярное произведение, а указанные веса получаются номализацией полученых

значений в распределение с помощью функции softmax.

$$V\,\mathrm{softmax}(K^T q)$$

Можно записать вектора-запросы в столбцы матрицы Q и добавить нормировочный множитель для улучшения устойтивости модели, тогда формула примет вид:

$$V\,\mathrm{softmax}(\alpha K^T Q), \quad \alpha = \frac{1}{\sqrt{d}},$$

где d – размерность векторов.

Таким образом, декодер получает больше информации об исходной последовательности, что позволяет решить проблему сжатия исходной последовательности в единственный вектор состояния.

Transformer

Представленная в 2017 году в статье [1] «Attention Is All You Need» *нерекурентная* архитектура transformer, первоначально предназначенная для задачи машинного перевода, активно использует блоки внимания и также состоит из двух блоков: кодировщика и декодировщика.

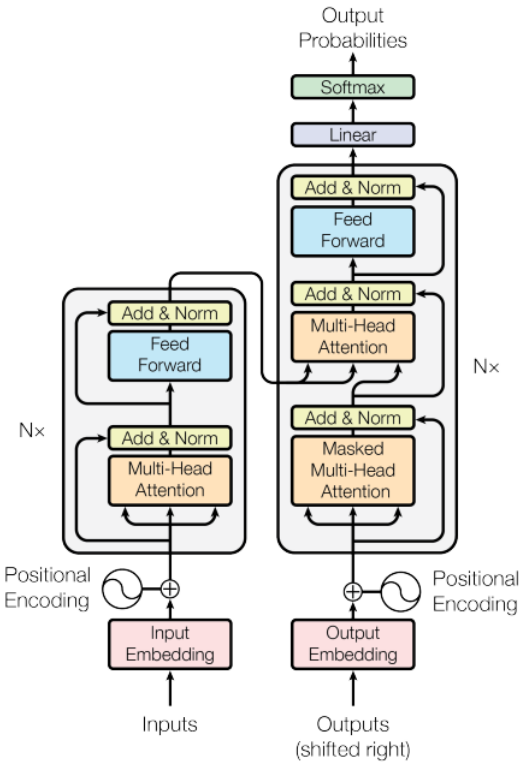

Рисунок 2.1: Архитектура трансформера [1]

2.1 Encoder

Кодировщик состоит из $N = 6$ одинаковых слоев. Каждый слой, в свою очередь, состоит из двух подслоев: multi-head self-attention, состоящего из нескольких, параллельно работающих, головок self-attention, и полносвязной сети прямого распространения (feed forward).

2.2 Decoder

Декодировщик также представляет собой стекинг $N = 6$ одинаковых слоев (так называемых трансформер-блоков). В дополнение к двум описанным выше подслоям декодировщик добавляет третий подслой, который применяет multi-head attention к выходу кодировщика, а слой self-attention маскируется таким образом, чтобы не позволять последующим позициям вносить вклад, гарантируя, что предсказания для i-той позиции будут учитывать только уже известные позиции до i.

2.3 Multi-head attention

Слой multi-head attention состоит из нескольких головок внимания, работающих параллельно.

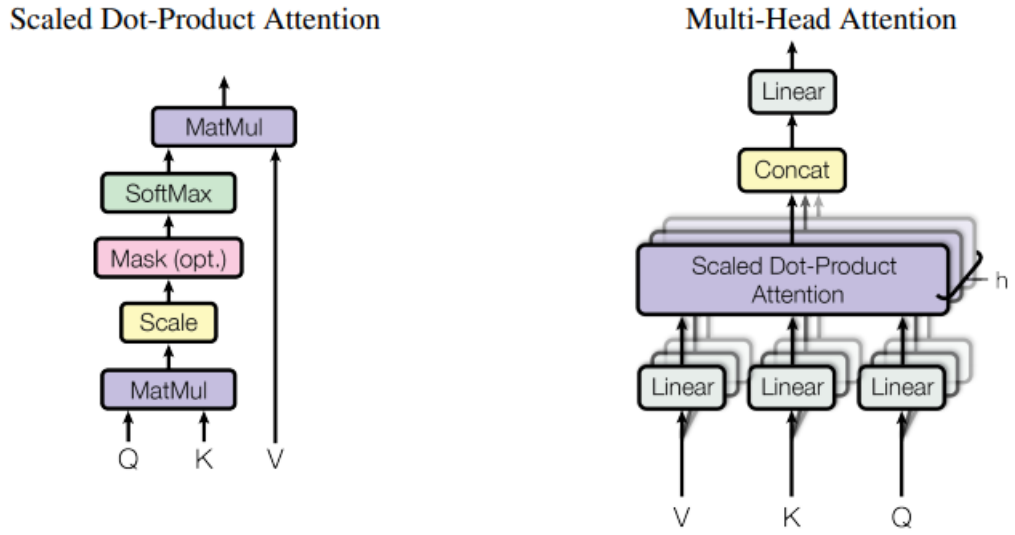

Рисунок 2.2: Механизм attention (слева) и multi-head attention (справа) [1]

Multi-head attention позволяет модели совместно обрабатывать информацию из разных подпространств представления в разных позициях, чего нельзя добиться, используя только одну головкой внимания.

$$\text{head}(X|W^V, W^K, W^Q) = W^V X \,\text{softmax}(\alpha(W^K X)^T W^Q X)$$
$$\text{multihead} = W\text{concat}[\text{head}(W_t^V, W_t^K, W_t^Q)]_{t=1}^n,$$

где в X перечислены объекты внимания, а матрицы W, W_t^Q, W_t^K, W_t^V — обучаемые параметры.

2.4 Positional encoding

Так как модель не является рекурентной, чтобы учитывать порядок элементов, необходимо дополнительно добавить в векторные представления токенов информацию об их относительном или абсолютном положении в последовательности, при этом необходимо, чтобы метод кодирования позиций не зависел от длины входной последовательности.

Способ, предложенный в оригинальной статье сопоставляет d-мерному эмбеддингу (векторному представлению) токена на позиции t новый d-мерный вектор, кодирующий его позицию в последовательности. Формально этот способ можно описать следующим образом:
пусть $\mathbf{e}_t \in \mathbb{R}^d$ — эмбеддинг токена. $f : \mathbb{N} \to \mathbb{R}$ кодирует позицию $\mathbf{e}_t$ вектором $\mathbf{p}_t$:

$$\mathbf{e}_t = f(t) = \begin{cases} \sin(\frac{t}{10000^{2k/d}}), i = 2k \\ \cos(\frac{t}{10000^{2k/d}}), i = 2k+1 \end{cases}$$

Итоговое представление токена получается как сумма $\mathbf{e}_t$ и $\mathbf{p}_t$.

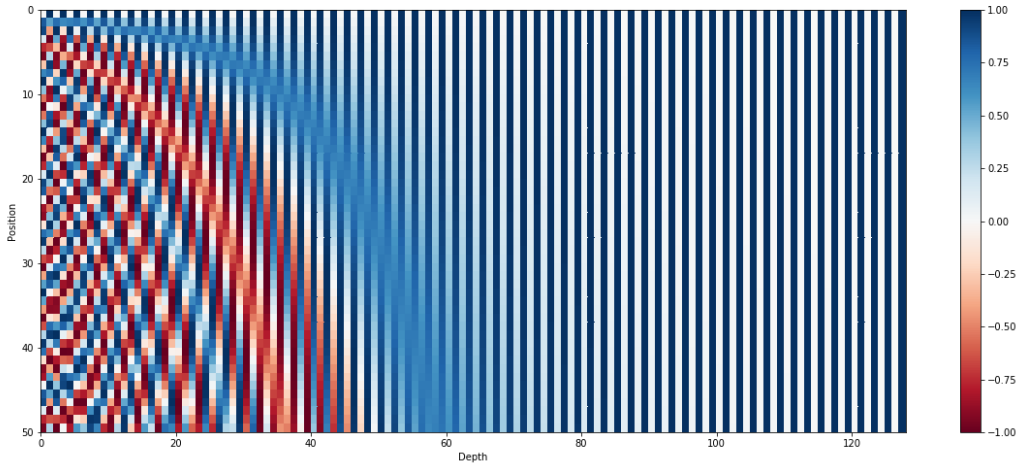

Рисунок 2.3: Positional encoding для $d = 128, t = 1...50$ [3]

2.5 Общая схема функционирования

Опишем в общих чертах схему работы Transformer.

1. На вход подаются эмбеддинги токенов.

2. К каждому эмбеддингу добавляется positional encoding.

3. Набор векторов проходит через $N = 6$ трансформер-блоков кодировщика. Трансформер-блок сохраняет размерности входных векторов, уточняя векторные представления токенов.

4. Токены выходной последовательности, генерируются один за одним с помощью декодировщика.

5. На вход декодировщику подается уже сгенерированные на текущем шаге выходные вектора с учетом positional encoding.

6. Генерация заканчивается, когда будет сгенерирован специальный токен, озбозначающий конец последовательности.

Для выполнения предсказания и подсчета ошибки выходы декодировщика проходят через линейный слой и softmax-слой. В качестве функции ошибки выступает кросс-энтропия.

2.6 Особенности обучения

Transformer обладает большим количеством обучаемых параметров и оказывается достаточно чувствительным к выбору learning rate и batch size.

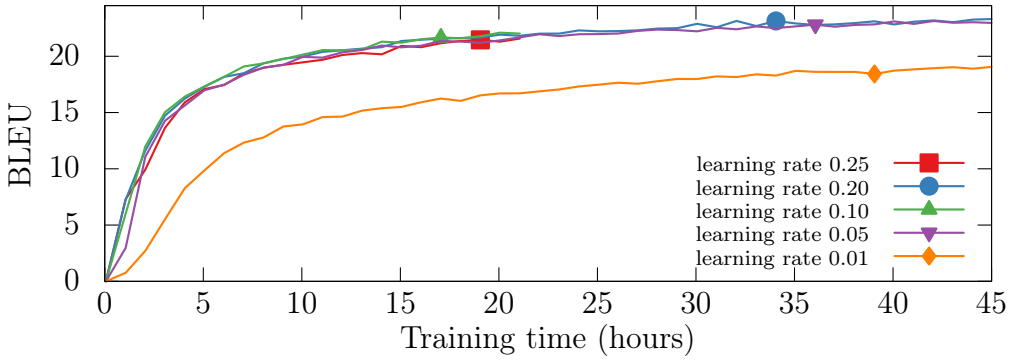

Рисунок 2.4: Влияние learning rate на качество [4]

Как можно видеть из графика 2.4, если выбрать learning rate слишком маленьким, качество остается на относительно низком уровне. Большая скорость обучения приводит к лучшему результату.

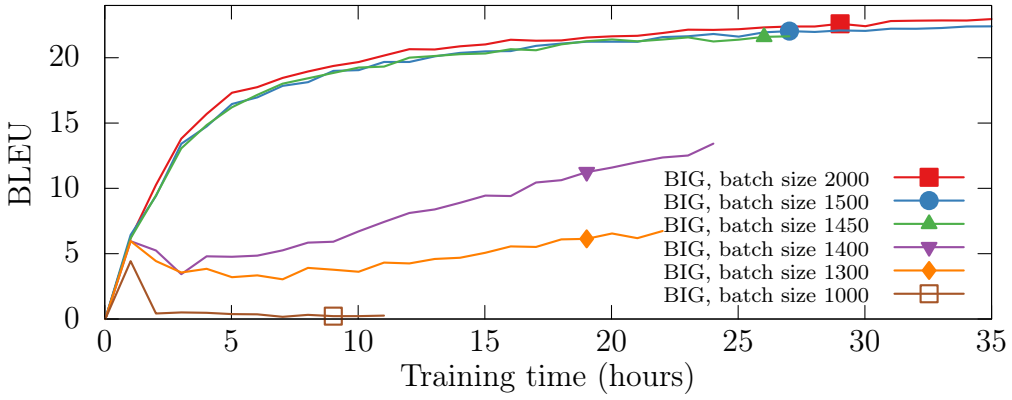

Рисунок 2.5: Влияние batch size на качество [4]

На графике 2.5 отображено, как больший batch size приводит к повышению качества.

При обучении таких сложных архитектур, как трансформер, очень важно подобрать комбинацию параметров, дающую наибольшее качество, при этом, кака видно из представленных графиков, хорошие значения как для learning rate, так и для batch size лежат в достаточно большом диапозоне.

BERT

После выхода Transformer начало создаваться большое количество моделей, которые на нем основаны. Одной из наиболее популярных является BERT (Bidirectional Encoder Representations from Transformers) [5].

BERT — это transformer-сеть для предобучения модели языка и получения представления слов. BERT был спроектирован, чтобы решать проблему полного обзора при обучении, когда информация о предсказываемом токене уже содержится в скрытых состояниях соседних токенов.

3.1 Masked language model

Чтобы обойти эту проблему, авторы BERT предлагают маскировать (заменять на специальные токены-маски) какую-то часть слов и предсказывать их. В оригинальной статье процесс маскирования выглядит так [5]:

1. Из исходного текста случайно выбирается 15% токенов, которые модель должна предсказать.

2. Каждый из выбранных токенов с вероятностью $p = 0.8$ заменяется на токен-маску.

3. С вероятностью $p = 0.1$ заменятся на случайное слово.

4. С вероятностью $p = 0.1$ остается без изменения.

3.2 Next sentence prediction

Помимо предсказания слов по контексту, BERT также способен одновременно решать задачу NSP (next sentence prediction): для пары предложений определить, следует ли второе предложение за первым. Для

этого вводятся два служебных токена, обозначающих начало текста и конец предложения.

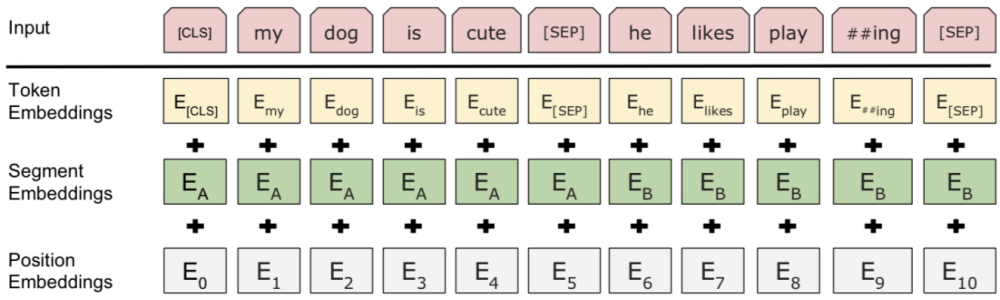

Рисунок 3.1: Вход BERT [5]

Как видно из рис. 3.1, итоговое векторное представление токена получается как сумма эмбеддинга токена, предложения и обучаемого представления позиции.

3.3 Архитектура

По сути, BERT представляет собой кодировщик трансформера, где каждый трансформер-блок на основе контекста уточнает эмбеддинг токена, таким образом преобразуя токен-маску в наиболее подходящее слово или токен-начало текста (CLS на рис. 3.1) в такой вектор, пропутив который через линейный классификатор, можно получить ответ на задачу NSP.

В оригинальной статье авторами предлагается несколько версий BERT: базовая и расширенная. Они отличаются количеством слоев, а также количеством головок внимания в каждой из них.

BERT	**# слоев**	**# головок attention**	**Размерность эмбеддингов**
Base	12	12	768
Large	24	16	1024

3.4 Transfer learning

Обученный BERT оказывается очень полезным и удобным, чтобы использовать его в качестве базовой модели при решении различных задач.

Предобученный BERT дообучается на датасете, специфическом для данной задачи. Примеров решаемых задач множество: от классификации предложений до ответов на вопросы о тексте, заданные на естественном языке.

Ниже приведен пример дообучения BERT для задачи нахождения ответа на вопрос в тексте. В качестве первого предложения подается вопрос, а в качестве второго — текст. Каждому токену из текста, пропуская его через обучаемый классификатор, ставится в соответствие две вероятности: если этот токен является началом участка текста, дающего ответ на вопрос, и если этот токен является концом участка текста, дающего ответ на вопрос.

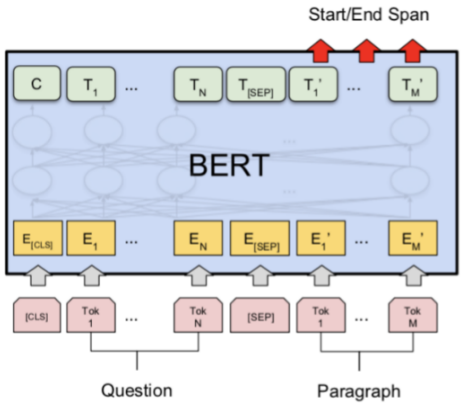

Рисунок 3.2: Fine-tuning BERT для задачи поиска ответа на вопрос в тексте [5]

RoBERTa

RoBERTa (Robustly Optimized BERT Pretraining Approach) [6] — модификация BERT, предлагающая качественные улучшения.

4.1 Различия с BERT

Сама архитектура модели осталась неизменной, а основной модификацией является подход к обучению. Авторы RoBERTa предложили:

обучать модель дольше, на большем объеме данных и с большими батчами:

train×10: 16GB → 160GB;

mini-batch: 256 samples → 8000 samples;

более длинные предложения;

убрать NSP;

применять динамическое маскирование.

Решения убрать из обучения задачу NSP, а также увеличить размер батча были приняты на основе экспериментальных данных. Оказалось, что умение модели решать задачу NSP не дает существенного прироста при fine-tuning.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Рисунок 4.1: Качество различных down-stream моделей в зависимости от наличия NSP [6]

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Рисунок 4.2: Качество в зависимости от размера мини-батча [6]

4.2 Новая обучающая выборка

Авторы RoBERTa отказались от Wiki-корпуса и BookCorpus'а в пользу CC-News — 76GB новостей за 2.5 года, OpenWebText — корпуса для GPT2 статей, упомянутых в постах на Reddit — 38GB и корпуса Stories — 31GB.

4.3 Динамическое маскирование

При динамическом маскировании маски непосредственно выбираются при подаче предложения на вход модели, а не на этапе препроцессинга как в оригинальном BERT. Это позволяет экономить память на дисках и генерировать больше данных.

ELECTRA

Обучение BERT-подобных моделей оказывается крайне трудоемким: требует много время и вычислительных ресурсов. Помимо этого такие модели обучаются с использованием токенов-масок, которые не присутствуют в реальных текстах. Главной задачей авторов ELECTRA явлось повышение эффективности обучения моделей.

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) [7] — еще одна модель, основанная на транформере. Необычность предлагаемого подхода, который лег в основу ELECTRA, можно увидеть из названия модели: вместо маскирования токенов, подобно BERT, авторы предложили заменять некоторые токены генератором и обучать модель распознавать замены.

5.1 Replaced Token Detection

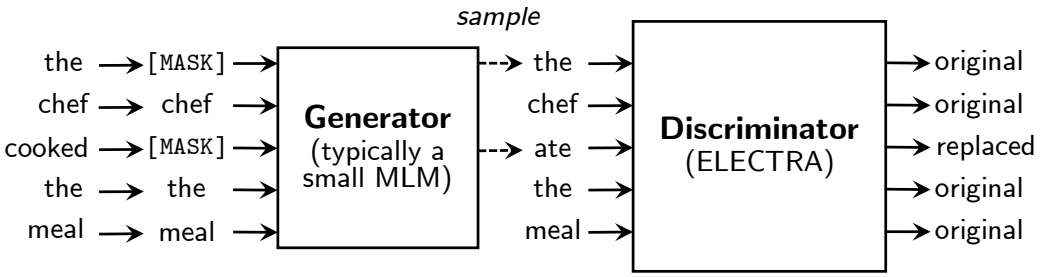

Рисунок 5.1: Схема работы replaced token detection [7]

В случае ELECTRA имеется две трансформер-модели: **generator** (генератор) и **discriminator** (дискриминатор).

1. На вход генератору подается предварительно замаскированный текст.

2. Генератор каким-либо образом восстанавливает исходный текст.

3. Дискриминатор классифицирует выходные токены генератора как правильно (**original**) или неправильно (**replaced**) восстановленные.

Но как архитектура, состоящая из двух моделей, одна из которых подобна BERT, может быть эффективнее, чем BERT? Дело в том, что в BERT loss рассчитывается из способности восстанавливать маски, которые составляют всего около 15% текста: нужно сгенерировать эмбеддинги для всех токенов, пропустить их через сложную архитектуру трансформера, но в конечном итоге, для обучения из них будет использована лишь небольшая часть. ELECTRA использует для обучения все входные токены: как замаскированные, так и не замаскированные, что и позволяет получать большую производительность при меньших вычислительных мощностях.

5.2 Обучение

Генератор и дискриминатор обучаются одновременно. Авторами приведены такие функции потерь:

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right),$$

где

$$p_G(x_t | \boldsymbol{x}) = \exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right) / \sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)$$

$$D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t)$$

Генератор штрафуется за неверно размаскированные токены-маски, а функция потерь для дискриминатора представляет собой обычную логистическую регрессию.

В процессе обучения ELECTRA минимизируется объединенный loss (в оригинальной статье $\lambda = 50$):

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

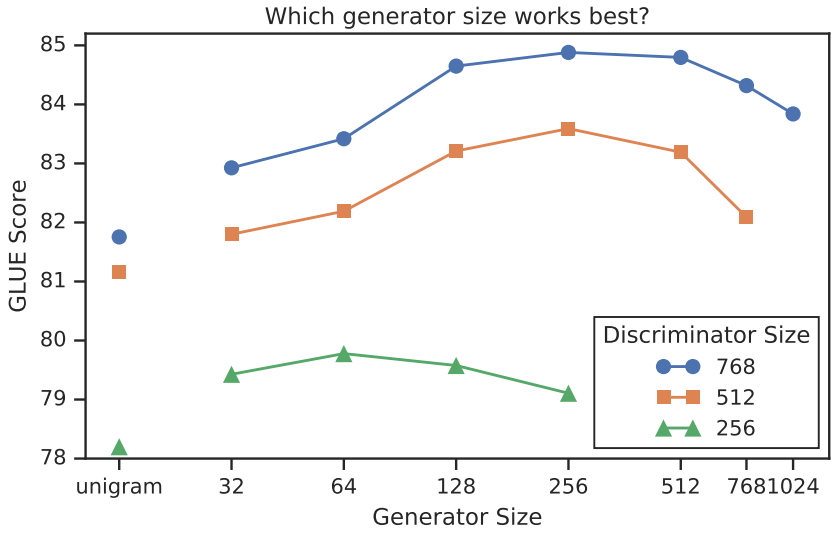

Рисунок 5.2: Зависимость качества от размера генератора [7]

Из результатов экпериментов, проведенных авторами (график 5.2), можно заметить, что наилучшее качество получается в случае, когда генератор несколько меньше дискриминатора. И так как генератор используется только при обучении, дополнительно это позволяет сэкономить вычислительные ресурсы. Слишком маленький или слишком большой генератор, размаскирующий тексты слишком плохо или слишком хорошо, не даст дискриминатору обучиться в полной мере.

Так как обе модели тренируются одновременно, по мере обучения дискриминатор будет получать все более сложные задачи [7]. Обучение по нарастанию сложности повышает качество ELECTRA.

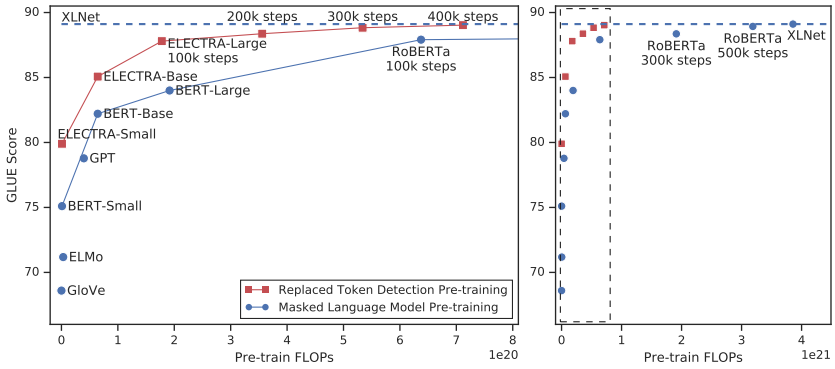

Рисунок 5.3: Предобучение с использованием replaced token detection превосходит предобучение с использованием masked language model при одинаковых вычислительных мощностях

Meena

Meena [8] — диалоговая модель, способная поддерживать беседу на произвольную тему, глубокая нейронная сеть с 2.6 миллиардами обучаемых параматров, основанная на архитектуре трансформера.

Для обучения модели использовалась метрика SSA (Sensibility and Specificity Average), которая учитывает ключевые элементы человекоподобного свободного разговора, и датасет из диалогов (40 миллиардов слов). На вход подаются до семи предыдущих реплик, по которым необходимо сгенерировать ответ.

6.1 Sensibleness and Specificity Average

Для оценивания ответов модели авторами оригинальной статьи была предложена метрика SSA, объединяющая два ключевых требования к ответу: осмысленность и специфичность. Для этого людей просили присвоить каждому ответу сети метку на основе этих двух критериев. Первая часть этой метрики, осмысленность (sensibleness) — базовое требование: ответы должны иметь смысл в контексте. Так, для ответов людей 97% удовлетворяют этому требованию.

Однако, если модель спроектирована так, чтобы максимизировать только sensibleness, ее ответы могут быть нечеткими и общими, так как это позволило бы избегать штрафа за малую осмысленность.

Чтобы устранить этот недостаток, добровольцев просили оценить, насколько ответ специфичен данному контексту.

Авторами была выявлена корелляция SSA и human-likeness (похожесть на человека), которая также оценивалась добровольцами (график 6.1).

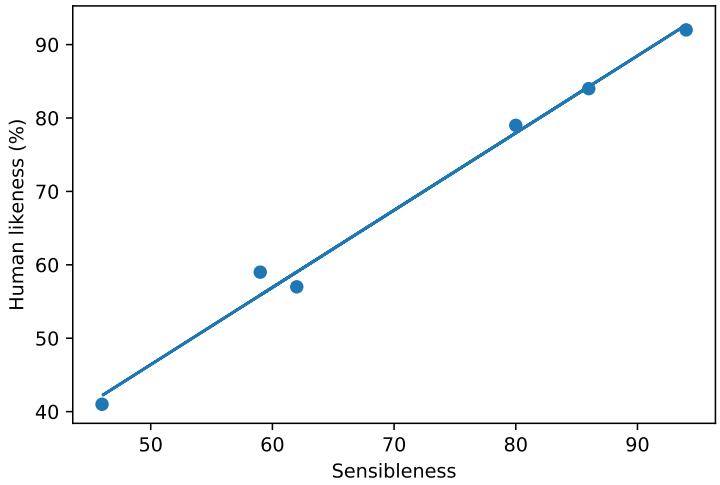

Рисунок 6.1: SSA против human-likeness. Наблюдаем зависимость, очень похожую на линейную. Каждая точка представляет какого-то чат-бота, кроме верхней, представляющей человека

6.2 Архитектура и детали обучения

Meena основана на улучшенной архитектуре трансформера — Evolved Transformer (ET) [9]. Основные отличия Evolved Transformer: дополнительные слои wide convolution и gated linear unit, ветвления и swish вместо ReLU. И encoder, и decoder имеют ветвящуюся нижнюю часть (рисунок 6.2) с wide convolution. Последующие слои же практически не отличаются от обычного трансформера.

Meena включает в себя один блок декодировщика ET и 13 ET-блоков декодироващиков. Модель обучалась 30 дней на TPU. Выяснилось, что модель с 2.6 параметрами переобучается на датасете из 61 миллиарда BPE-токенов, поэтому авторами было решено добавить 0.1-dropout в полносвязный слой и слои attention.

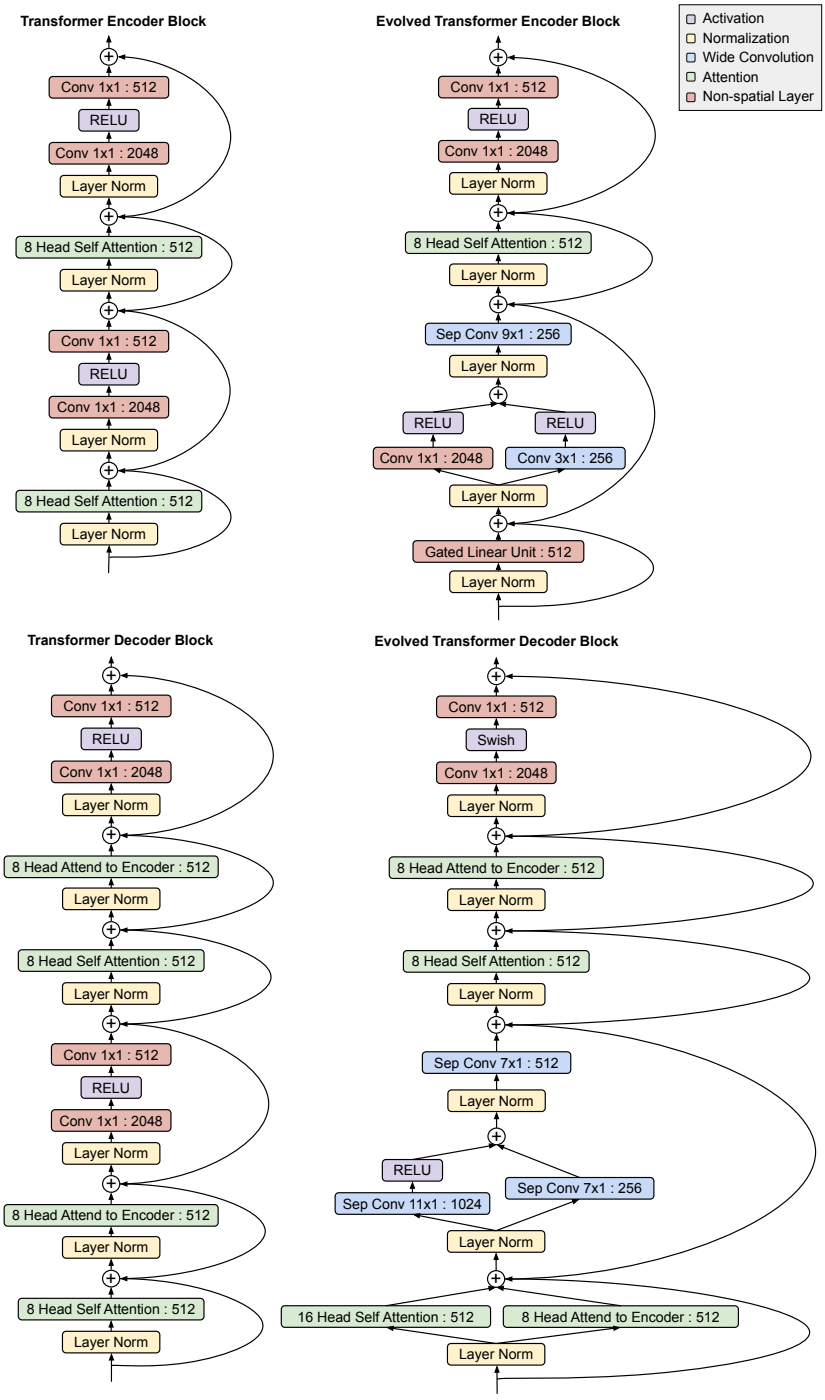

Рисунок 6.2: Сравнение Transformer и Evolved Transformer

Еще одна модификация, примененная авторами Meena — семплирование с температурой: сначала семплируются N возможных ответов, с использованием температуры T, затем выбирается ответ с наибольшей вероятостью быть использованным в качестве итогового ответа.

Температура T — это гиперпараметр, регулирующий распределение вероятностей p_i для следущего токена в процессе декодирования.

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Было выявлено, что большие значения T повышают вероятности редких токенов, таких как подходящие по контексту имена, однако могут присвоить слишком большую вероятность неверным токенам. В то же время, небольшие значения температуры приводят к использованию более типичных слов, таких как предлоги и артикли.

6.3 Результаты

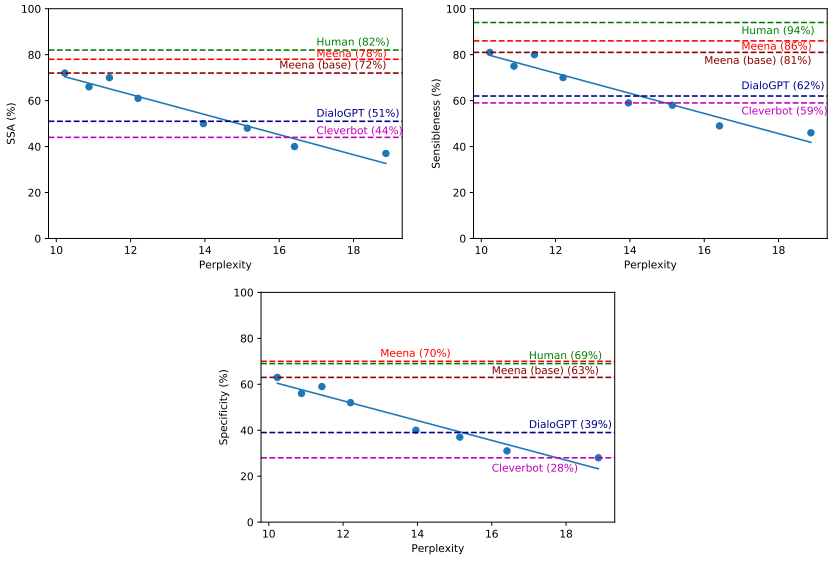

Рисунок 6.3: Метрики SSA, Sensibleness и Specificity для различных чат-ботов и человека

Как можно видеть из графиков 6.3, значения SSA для Meena отличаются от показателей для человека только на 4%. Ниже приведено сравнение ответов, сгенерированных различными моделями.

Sample Contexts	**Meena (base)**	**Cleverbot**	**DialoGPT**
A: You must have read a lot of books *B:* Oh wow, how do you know? Do I look nerdy? *A:* Of course, you're not!	*B:* So how do you know? [*sensible: 1, specific: 1*]	*B:* Never. said I was. [*sensible: 0, specific: 0*]	*B:* Oh thank goodness. [*sensible: 1, specific: 1*]
A: You know nothing, Jon Snow! *B:* You must be a fan of Game of Thrones *A:* Haha yea, how can I not?	*B:* Who's your favorite character? [*sensible: 1, specific: 1*]	*B:* What do you mean? [*sensible: 0, specific: 0*]	*B:* You can't. But that's not why you shouldn't know. [*sensible: 0, specific: 0*]
A: What is your favorite island in the world? *B:* I would say Fiji *A:* Interesting, what's nice about it?	*B:* Just a beautiful place [*sensible: 1, specific: 0*]	*B:* The title is self explanatory. [*sensible: 0, specific: 0*]	*B:* You can see some great scenery. [*sensible: 1, specific: 0*]
A: You must have read a lot of books *B:* Oh wow, how do you know? Do I look nerdy?	*A:* I know because I've read a lot of books. [*sensible: 1, specific: 1*]	*A:* Observation. [*sensible: 1, specific: 0*]	*A:* I know because you look like your mom. [*sensible: 1, specific: 1*]
A: You know nothing, Jon Snow! *B:* You must be a fan of Game of Thrones	*A:* Yeah, I'm reading the books at the moment. [*sensible: 1, specific: 1*]	*A:* No. [*sensible: 1, specific: 0*]	*A:* I've never really been interested in it... [*sensible: 1, specific: 1*]
A: What is your favorite island in the world? *B:* I would say Fiji	*A:* Fiji, yes! I've been there once. It's the most beautiful place I've ever been. It definitely makes me want to move there. I hope you get to visit again soon! [*sensible: 1, specific: 1*]	*A:* I would have guessed 666. [*sensible: 0, specific: 0*]	*A:* Fiji is awesome! But I would say Fiji and Oceania are more popular at the moment. [*sensible: 0, specific: 0*]
A: You must have read a lot of books	*B:* Reading's for nerds [*sensible: 1, specific: 1*]	*B:* What books have you read. [*sensible: 1, specific: 1*]	*B:* I read a lot of books but have never read a single book about the history of our planet. [*sensible: 1, specific: 1*]
A: You know nothing, Jon Snow!	*B:* Thank you for this. I was hoping I wasn't the only one getting an ASOIAF vibe from this. [*sensible: 1, specific: 1*]	*B:* Am not jon snow. [*sensible: 1, specific: 0*]	*B:* The North Remembers... [*sensible: 1, specific: 1*]
A: What is your favorite island in the world?	*B:* That's a tough one. I'd have to say Hawaii. [*sensible: 1, specific: 1*]	*B:* Daniel I like frites. [*sensible: 0, specific: 0*]	*B:* Probably the one on the north island [*sensible: 0, specific: 0*]

Рисунок 6.4: Caption

Рисунок 6.5: Примеры ответов Meena, Cleverbot, DialoGPT на некоторых контекстах, и метки sensibleness и specificity [8]

Заключение

В данной работе была описана архитектура трансформер, объяснены ключевые понятия, такие как attention и positional encoding, представлены обзоры современных архитектур, основанных на трансформере. Описаны главные идеи и подходы, легшие в основы данных архитектур и сделавшие их столь популярными. Были также приведены сравнения показателей качества и времени работы для описанных моделей.

Литература

1. Vaswani Ashish, Shazeer Noam, Parmar Niki [и др.]. Attention Is All You Need. 2017.

2. Batista David S. The Attention Mechanism in Natural Language Processing - seq2seq. URL: http://www.davidsbatista.net/blog/2020/01/25/Attention-seq2seq/.

3. Kazemnejad Amirhossein. Transformer Architecture: The Positional Encoding. URL: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/.

4. Popel Martin, Bojar Ondřej. Training Tips for the Transformer Model // The Prague Bulletin of Mathematical Linguistics. 2018. Apr. Т. 110, № 1. с. 43–70. URL: http://dx.doi.org/10.2478/pralin-2018-0002.

5. Devlin Jacob, Chang Ming-Wei, Lee Kenton [и др.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

6. Liu Yinhan, Ott Myle, Goyal Naman [и др.]. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019.

7. Clark Kevin, Luong Minh-Thang, Le Quoc V. [и др.]. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. 2020.

8. Adiwardana Daniel, Luong Minh-Thang, So David R. [и др.]. Towards a Human-like Open-Domain Chatbot. 2020.

9. So David R., Liang Chen, Le Quoc V. The Evolved Transformer. 2019.