

# Лекция 10

## Нейронные сети для обработки естественного языка

Храбров Кузьма

1 апреля 2024 г.



Не забывайте  
отмечаться  
и оставлять  
отзыв



# План лекции

1. Предисловие
2. Представления слов
3. Задачи NLP
4. Механизмы внимания
5. Модель Transformer
6. Модель GPT и ее развитие
7. Модели BERT и ее развитие
8. Transformer для картинок

# Предисловие

# NLP, что мы уже знаем

Многие задачи обработки текста являются классическими задачами машинного обучения: классификацией (например, определение языка, определение токсичности) и регрессии (оценка похожести текстов, тональность отзыва). Тут с некоторой оговоркой применимы подходы на основе полносвязных сетей, с которыми мы познакомились в лекциях 1 и 2.

Тем не менее тексты имеют понятную структуру последовательностей, а значит к ним применимы одномерные свертки (лекция 3) и рекуррентные модели (лекция 9).

## **GPT-4 Unveiled by OpenAI, Aces Professional Exams Including Bar Exam, AP exams, and other Pre-university Exams**

By [Leo Wong DQ](#) - Mar 14, 2023

# GPT-3.5/ChatGPT

K

Придумай смешную историю для лекции про архитектуру трансформеров.



Однажды в галактике, далекой-далекой, на планете, где обитали только роботы, решили они построить новую школу. Роботы были умными и продвинутыми, поэтому они решили использовать самую передовую технологию - трансформеры!

Они создали огромный трансформер, который должен был превращаться в школу. Но в процессе разработки, у них возникла проблема: они забыли спроектировать двери! Роботы были в ужасе - как же ученики будут заходить в школу, если нет дверей?

Однако, они нашли выход из ситуации - они добавили в трансформер специальный режим "телеportации". Теперь ученики могли просто "телеportироваться" внутрь школы, не беспокоясь о дверях.

Когда они открыли новую школу, все роботы были в восторге! Но когда первые ученики попытались телепортироваться внутрь, произошло нечто невероятное - вместо того, чтобы попасть внутрь школы, они оказались на другой планете!

# GPT-3.5/ChatGPT

K

нет, трансформер это архитектура нейронной сети, придумай историю про него.



Конечно, извините за недопонимание! Вот история про архитектуру трансформеров:

👍 🔍

Однажды в компании, занимающейся искусственным интеллектом, работали два разработчика - Джон и Мэри. Они работали вместе над созданием новой нейронной сети, которая должна была распознавать и классифицировать различные объекты.

Джон был заинтересован в использовании сверточных нейронных сетей, а Мэри предложила использовать рекуррентные нейронные сети. Оба они имели свои аргументы в пользу своих вариантов.

Но однажды, когда они обсуждали этот вопрос, случилось нечто странное - они наткнулись на старую игрушку-трансформера, которая была забыта в углу их офиса. И тогда они подумали - почему бы не использовать идею трансформации в нейронной сети?

Так и родилась идея архитектуры трансформеров. Они создали нейронную сеть, которая могла использовать механизмы трансформации входных данных, чтобы получать более точные результаты. Они использовали технику многоканальной свертки и многоголовое внимание, чтобы улучшить точность классификации объектов.

⟳ Regenerate response

# Представления слов

# Представление слов

## Задача

Сопоставить каждому слову  $w$  из словаря  $V$  вектор  $e(w)$ .

Подходы:

- ▶ One-hot encoding
- ▶ Counts
- ▶ CBOW
- ▶ Skip-grams

## One-hot encoding

Кодируем слово  $w_i$  вектором  $[0, 0, \dots, 0, \underbrace{1}_{i}, 0, \dots, 0]^T$  Плюсы:

- ▶ Просто реализовать
- ▶ Можно использовать разреженное представление

Минусы:

- ▶ Не учитывает близость слов
- ▶ Огромная размерность

# Counts

... and the cute **kitten** purred and then ...

... the cute **furry** cat purred and miaowed ...

... that small **kitten** miaowed and she ...

... the loud **furry** dog ran and bit ...

Словарь: bit, cute, furry, loud, miaowed, purred, ran, small

**kitten**: cute, purred, small, miaowed  $\Rightarrow [0, 1, 0, 0, 1, 1, 0, 1]^T$

**cat**: cute, furry, miaowed  $\Rightarrow [0, 1, 1, 0, 1, 0, 0, 0]^T$

**dog**: loud, furry, ran, bit  $\Rightarrow [1, 0, 1, 1, 0, 0, 1, 0]^T$

$$sim(w_1, w_2) = \frac{\langle w_1, w_2 \rangle}{\|w_1\| \cdot \|w_2\|}$$

## Embedding matrix

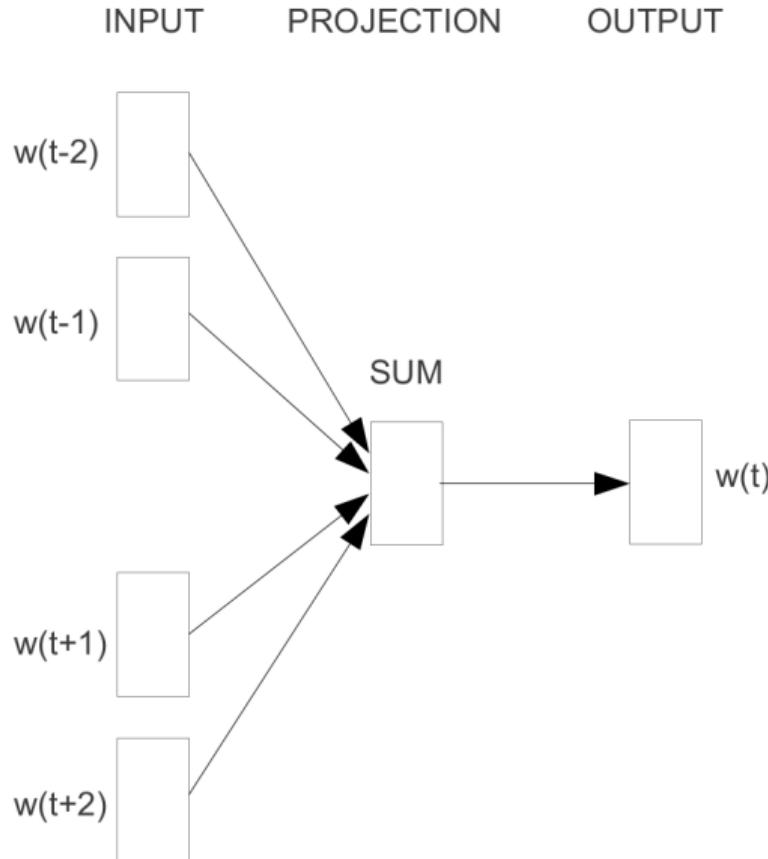
Матрица представлений:

$$E = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_{|V|} \end{bmatrix}$$

Каждая строка — представление одного слова.

Идея: обучим матрицу  $E$  при помощи нейронной сети.

# Continuous bag of words



Предсказываем пропущенное слово по контексту.

Представление слова:

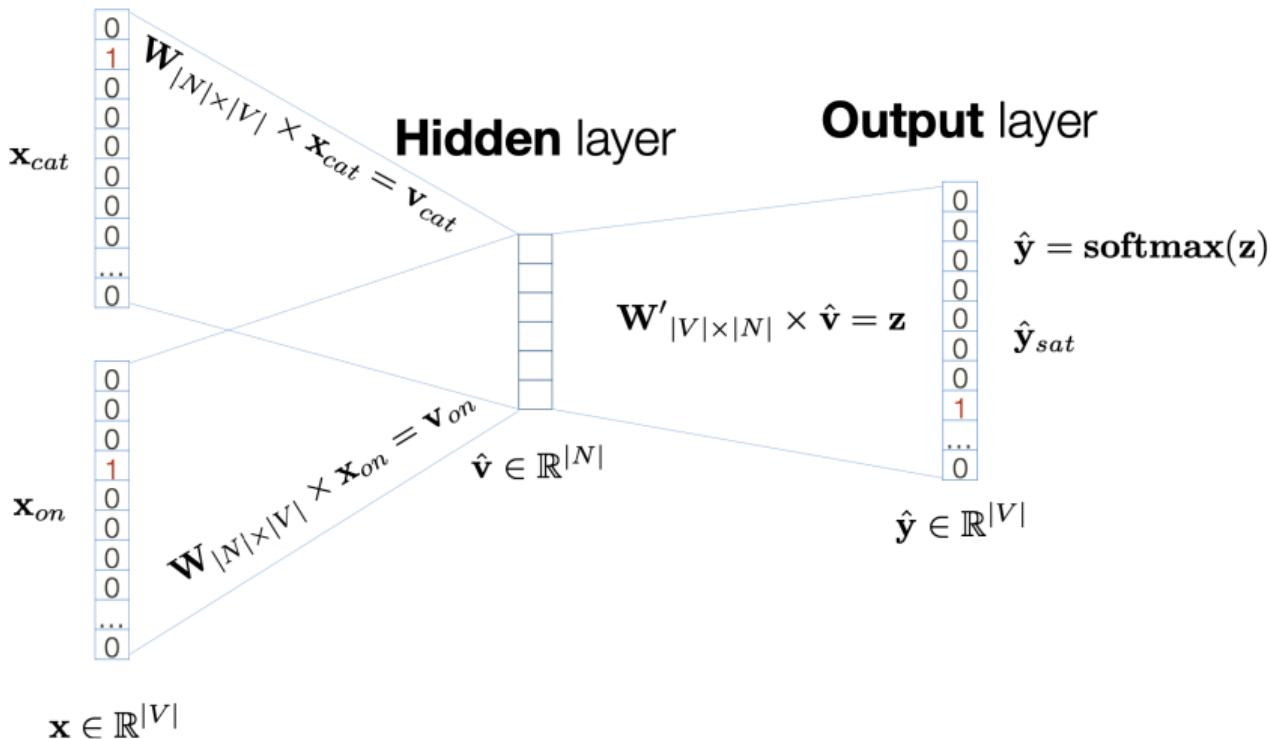
$$h_t = W \sum_{w \in \text{context}(w_t)} \text{one\_hot}(w)$$

$$P(w_i | \text{context}(w_i)) = \text{softmax}(W' h) [i]$$

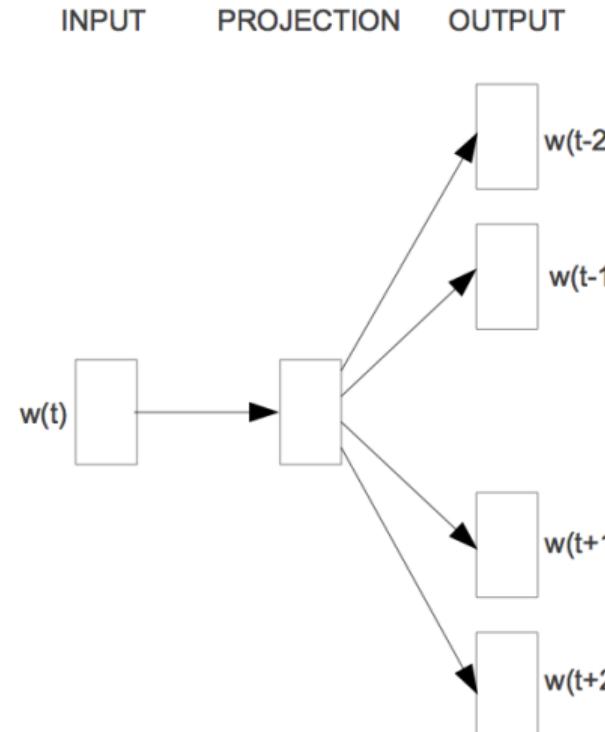
Функция потерь:  $L = -\log P(w_i | \text{context}(w_i))$

# Continuous bag of words. Details

## Input layer

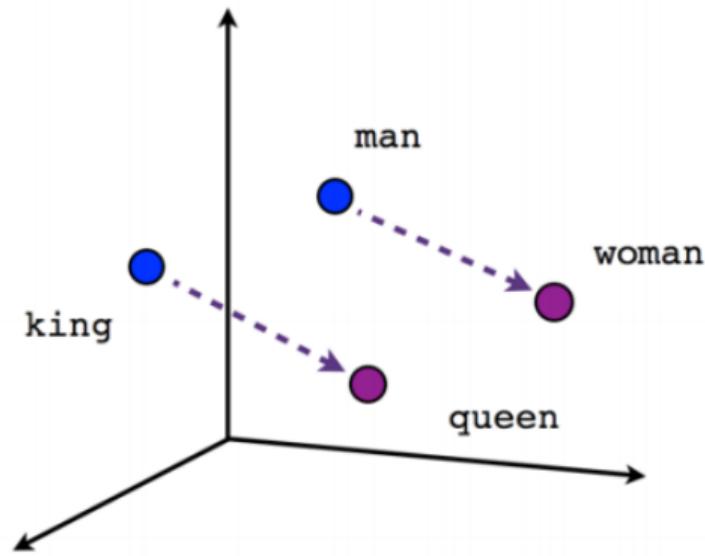


# Skip-gram

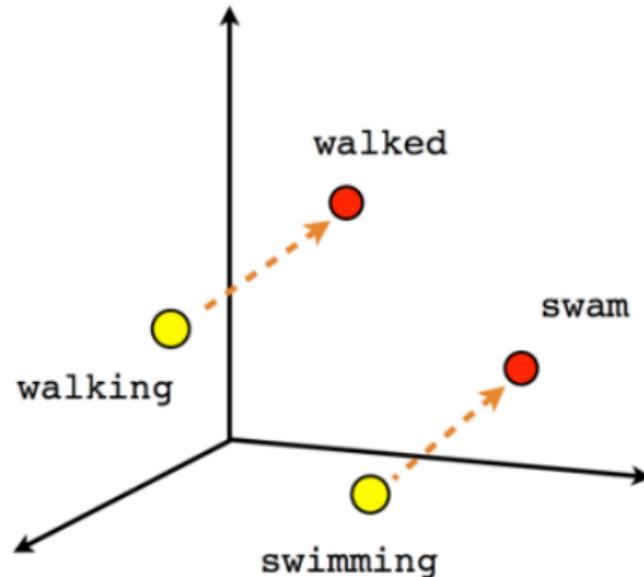


Предсказываем контекст по пропущенному слову

## word2vec: арифметика



Male-Female



Verb tense

# Токенизация подслов

Три общих алгоритма:

- ▶ Кодирование пар байтов (BPE) (Sennrich et al., 2016)
- ▶ Unigram Language Model (ULM) (Taku Kudo, 2018  
<https://arxiv.org/pdf/1804.10959.pdf> )
- ▶ WordPiece ( Schuster, Nakajima <https://static.googleusercontent.com/media/research.google.com/ja/pubs/archive/37842.pdf>)

Все имеют 2 части:

- ▶ Выделение токенов: необработанный корпус -> словарь (набор токенов).
- ▶ Сегментатор маркеров, который берет необработанное тестовое предложение и разбивает его в соответствии с этим словарем.

# Byte Pair Encoding. Идеи<sup>1</sup>

Исходный словарь - набор отдельных символов А, Б, В, Г,...,а, б, в, г.....

Алгоритм получения словаря:

Повторять пока не было сделано  $k$  слияний:

- ▶ выбрать два символа, которые чаще всего соседствуют в обучающем корпусе (скажем, «А», «Б»),
- ▶ добавляем в словарь новый объединенный символ «АБ» и заменяем все соседние «А» и «Б» в корпусе на «АБ».

На тестовых данных превращаем буквы в токены из словаря:

- ▶ Жадно
- ▶ В том порядке, в котором токены были получены во время построения словаря

---

<sup>1</sup><https://arxiv.org/abs/1508.07909>

# Byte Pair Encoding.

## Byte Pair Encoding Data Compression Example

aaabdaaabac

aaabdaaabac Replace Z = aa

ZabdZabac Replace Y = ab

ZYdZYac Replace X = ZY

XdXac Final compressed string

Replacement Table

Byte pair	Replacement
X	ZY
ab	Y
aa	Z

# Задачи NLP

# Задачи NLP

- ▶ Информационный поиск (Information retrieval)
- ▶ Классификация текстов
- ▶ Генерация текста
- ▶ Диалоговые системы

# Задачи NLP

- ▶ Детекция спама / Классификация тематики / Анализ тональности
- ▶ Распознавание именованных сущностей (Named entity recognition, NER)
- ▶ Relation extraction
- ▶ Parts of speech tagging
- ▶ Ранжирование
- ▶ Суммаризация
- ▶ Ответы на вопросы
- ▶ "Понимание прочитанного"(Reading comprehension)
- ▶ Speech-to-text / Text-to-speech
- ▶ Image-to-text / Text-to-image

## Question answering (Ответы на вопросы)

Мэри взяла там молоко  
Джон переехал в спальню  
Сандра вернулась на кухню  
Мэри пошла в коридор  
Джон взял там футбольный мяч  
Джон пошел в коридор  
Джон отложил футбольный мяч  
Мэри пошла в сад  
Джон пошел на кухню  
Сандра вышла в коридор  
Дэниел вышел в коридор  
Мэри отказалась от молока  
**Где молоко?**

Ответ: **в саду**

# Visual QA<sup>2</sup>

Who is wearing glasses?

man



woman

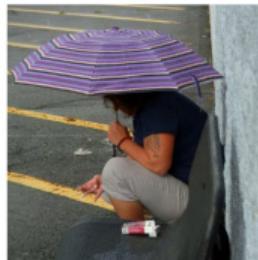


Is the umbrella upside down?

yes



no



Where is the child sitting?

fridge



arms



How many children are in the bed?

2



1



<sup>2</sup>Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering (CVPR 2017)

# Visual QA<sup>3</sup>

Answer: No



Answer: Yes



complementary scenes



Tuple: <girl, walking, bike>

## Анализ тональности

### Отзыв положительный или отрицательный?

y=1 Мне очень понравился этот фильм. Никогда раньше ничего подобного не видел!!!))

y=0 Ужасно! Ушла с середины фильма, тк больше невозможно было смотреть.

y=0 Ну да, конечно. Просто отличный фильм.

# Анализ тональности

## Отзыв положительный или отрицательный?

$y=1$  Мне очень понравился этот фильм. Никогда раньше ничего подобного не видел!!!))

$y=0$  Ужасно! Ушла с середины фильма, тк больше невозможно было смотреть.

$y=0$  Ну да, конечно. Просто отличный фильм.

Простой подход: Bag-of-words + Logistic regression

Какие есть проблемы у такого подхода?

# Анализ тональности

## Отзыв положительный или отрицательный?

$y=1$  Мне очень понравился этот фильм. Никогда раньше ничего подобного не видел!!!))

$y=0$  Ужасно! Ушла с середины фильма, тк больше невозможно было смотреть.

$y=0$  Ну да, конечно. Просто отличный фильм.

Простой подход: Bag-of-words + Logistic regression

Какие есть проблемы у такого подхода?

- ▶ Не учитывает сарказм
- ▶ Не учитывает схожесть слов (например, кот ↔ котенок)
- ▶ Не учитывает порядок слов

## Пример: SQuAD

Rank	Model	EM	F1
	Human Performance <i>Stanford University</i> (Rajpurkar & Jia et al. '18)	86.831	89.452
1	IE-Net (ensemble) <i>RICOH_SRCB_DML</i>	<b>90.939</b>	<b>93.214</b>
	Jun 04, 2021		
2	FPNet (ensemble) <i>Ant Service Intelligence Team</i>	90.871	93.183
	Feb 21, 2021		
3	IE-NetV2 (ensemble) <i>RICOH_SRCB_DML</i>	90.860	93.100
	May 16, 2021		
4	SA-Net on Albert (ensemble) <i>QIANXIN</i>	90.724	93.011
	Apr 06, 2020		

## Пример: SQuAD

*P6418* Термин Computer science (Компьютерная наука) появился в 1959 году в научном журнале Communications of the ACM, в котором Луи Фейн (Louis Fein) ратовал за создание Graduate School in Computer Sciences (Высшей школы в области информатики) . . . Усилия Луи Фейна, численного аналитика Джорджа Форсайта и других увенчались успехом: университеты пошли на создание программ, связанных с информатикой, начиная с Университета Пердью в 1962.

*P6418* The term "computer science" appears in a 1959 article in Communications of the ACM, in which Louis Fein argues for the creation of a Graduate School in Computer Science . . . Louis Fein's efforts, and those of others such as numerical analyst George Forsythe, were rewarded: universities went on to create such departments, starting with Purdue in 1962.

*Q11870* Когда впервые был применен термин Computer science ( Компьютерная наука )?

*Q11870* When did the term "computer science" appear?

*Q28900* Кто впервые использовал этот термин?

*Q28900* Who was the first to use this term?

*Q30330* Начиная с каого учебного заведения стали применяться учебные программы, связанные с информатикой?

*Q30330* Starting with wich university were computer science programs created?

# Современные задачи NLP. SuperGlue<sup>4</sup> <sup>5</sup>

Rank	Name	Model	URL	Score
1	JDExplore d-team	Vega v2		91.3
+	2 Liam Fedus	ST-MoE-32B		91.2
	3 Microsoft Alexander v-team	Turing NLR v5		90.9
	4 ERNIE Team - Baidu	ERNIE 3.0		90.6
	5 Yi Tay	PaLM 540B		90.4
+	6 Zirui Wang	T5 + UDG, Single Model (Google Brain)		90.4
+	7 DeBERTa Team - Microsoft	DeBERTa / TuringNLVRv4		90.3
	8 SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8
+	9 T5 Team - Google	T5		89.3

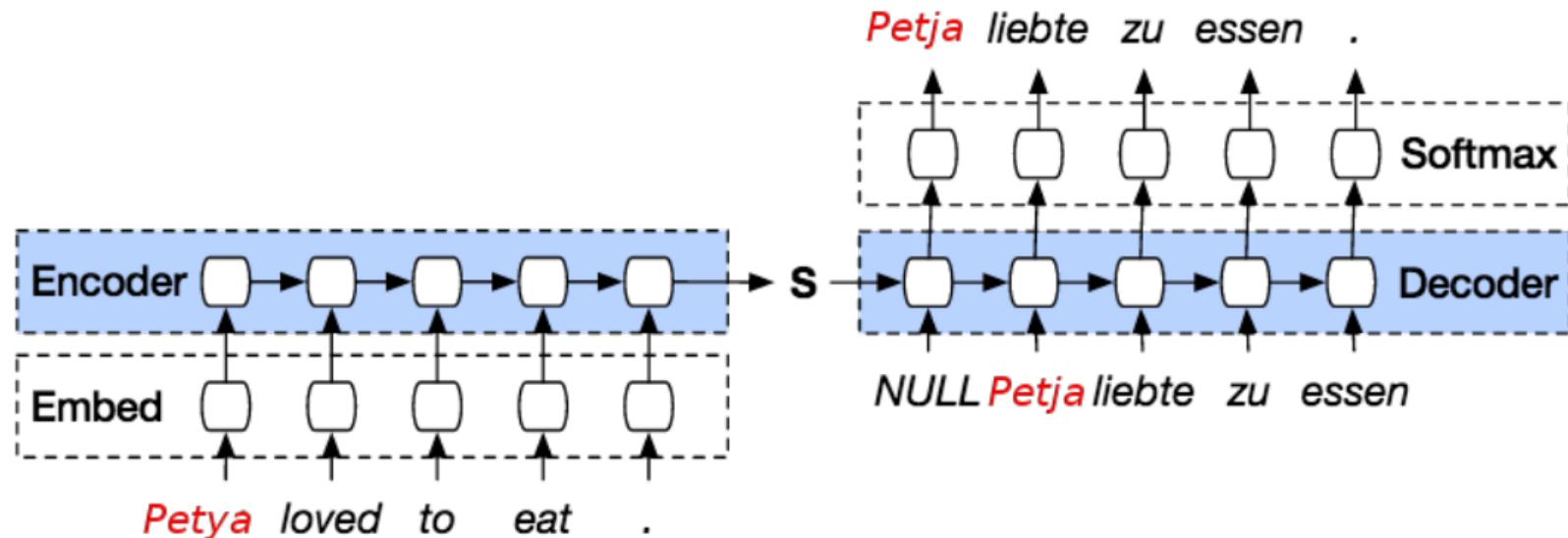
<sup>4</sup><https://super.gluebenchmark.com/leaderboard>

<sup>5</sup><https://russiansuperglue.com>

# Механизмы внимания

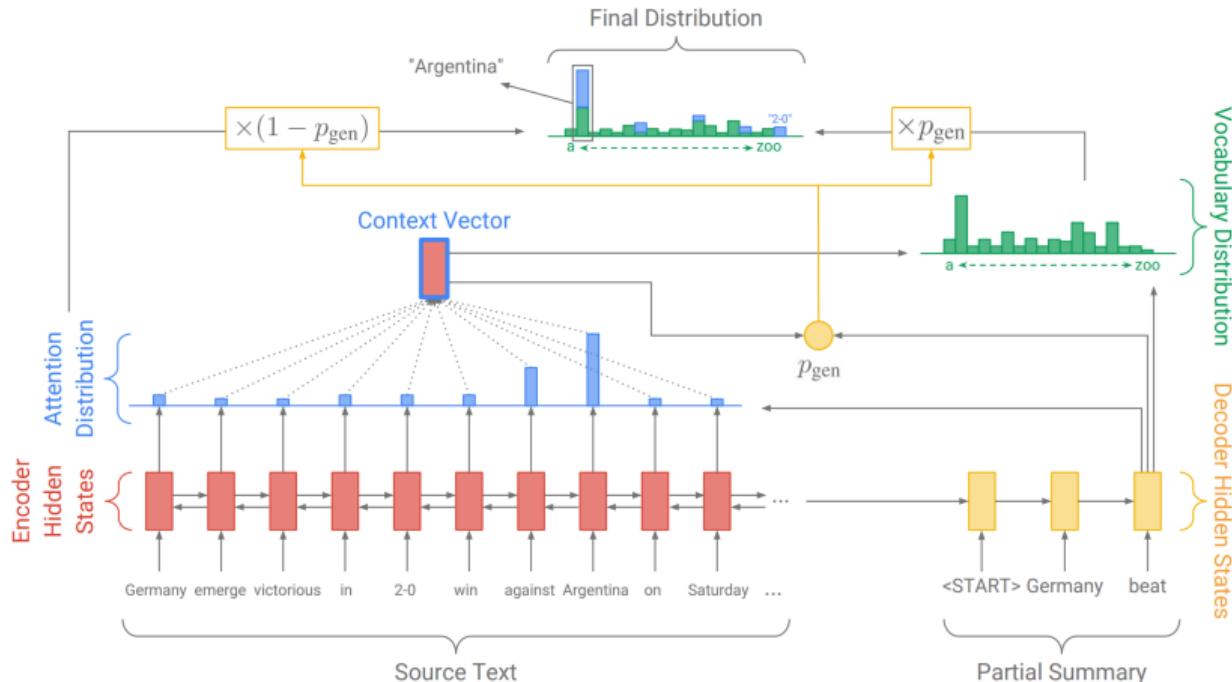
## RNN. Seq2Seq

Проблема: надо запоминать точные сущности из текста, например, имена, названия, ..., также для цитирования надо запомнить точный текст, при ограниченном размере эмбеддинга это почти невозможно.



# Attention<sup>6</sup>

Добавим Attention (Механизм внимания).



<sup>6</sup><https://arxiv.org/abs/1409.0473>

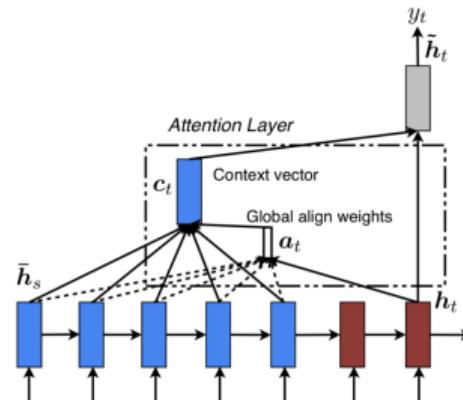
# Attention

$$\text{decoder}_i = \text{RNN}(\dots)$$

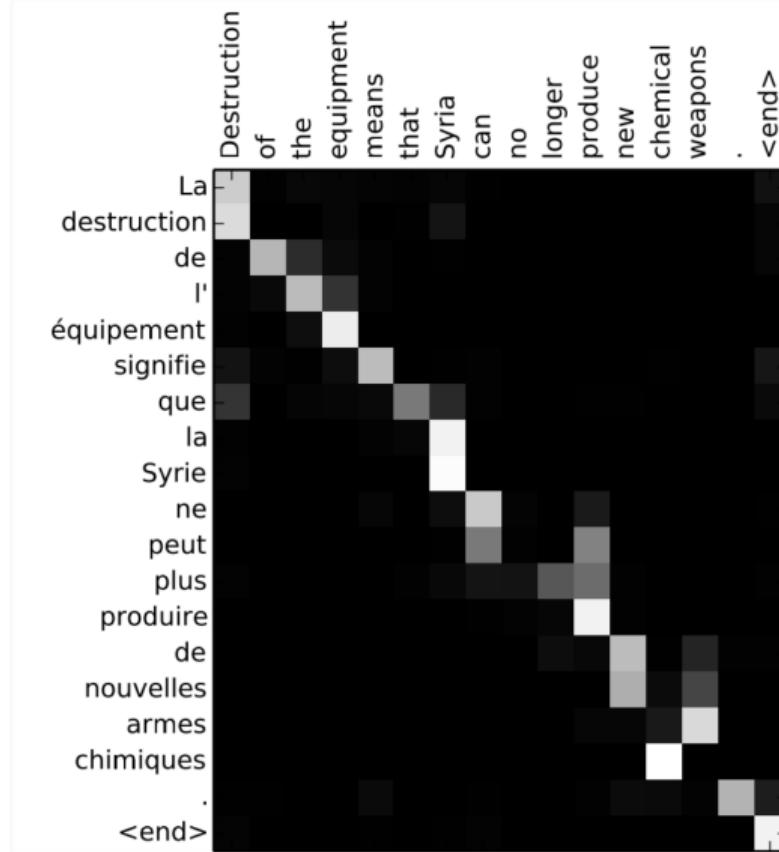
$$\text{attention\_score}_{ij} = \text{softmax}_j(\text{attention}(\text{decoder}_i, \text{encoder\_output}_j))$$

$$\text{context}_i = \sum_j \text{attention\_score}_j \cdot \text{encoder\_output}_j$$

$$\text{decoder\_output}_i = \text{softmax}(f(\text{decoder}_i, \text{context}_i))$$



# Attention



# Attention to images

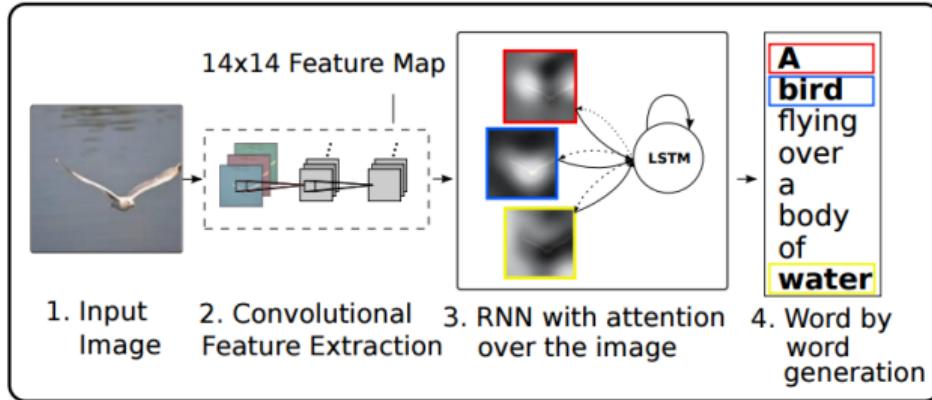
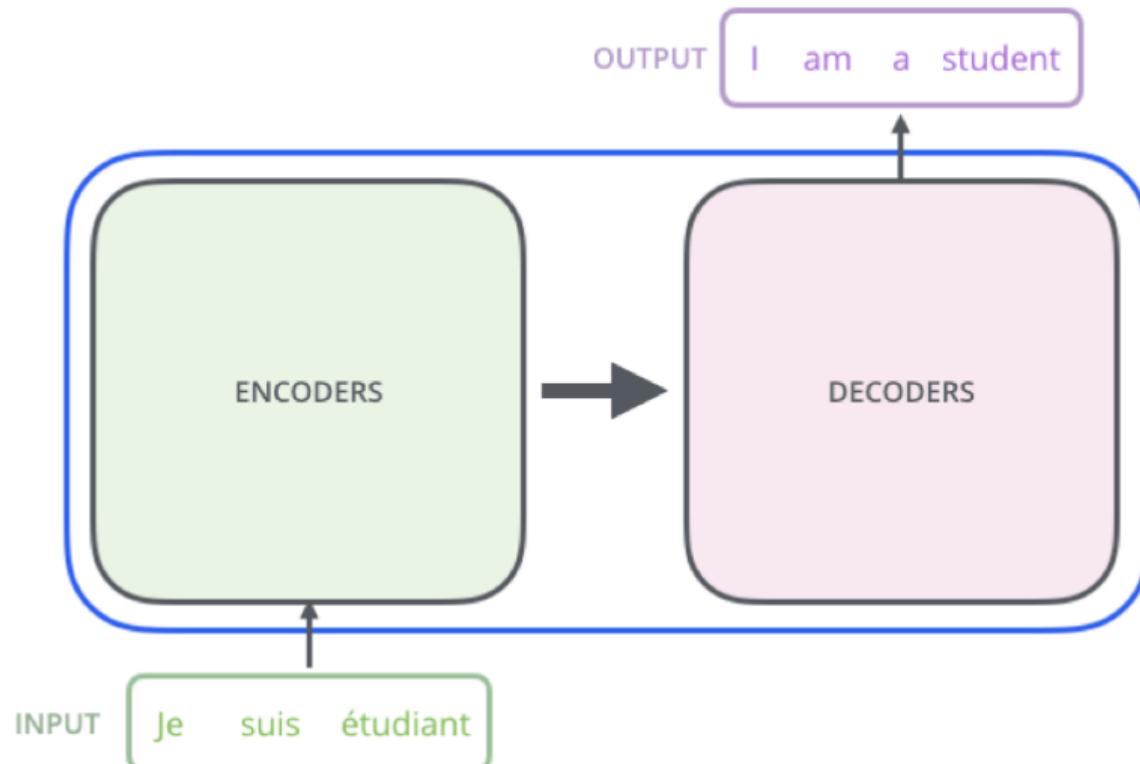


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicate the corresponding word)

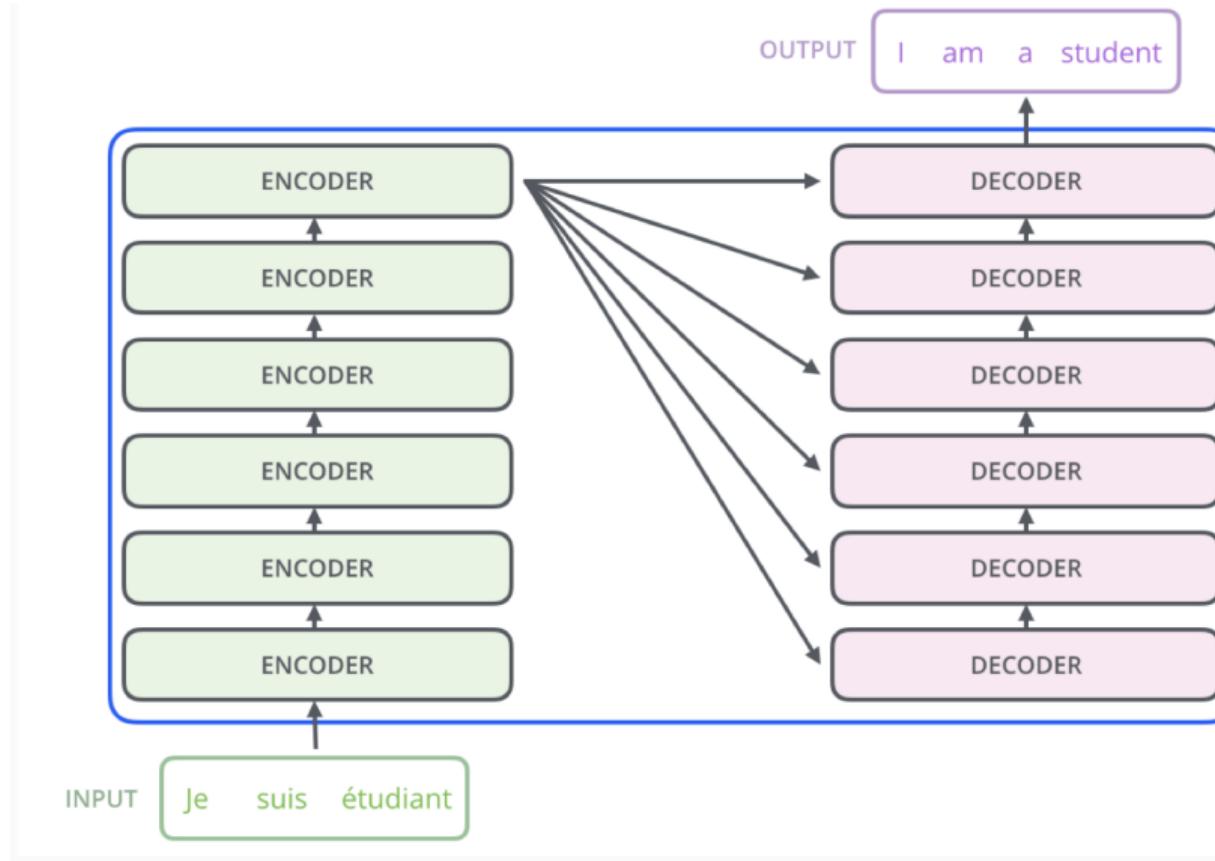


# Архитектура Transformer

# Transformer (Attention is all you need)



# Transformer (Attention is all you need)



## Transformer (Attention is all you need). Embeddings

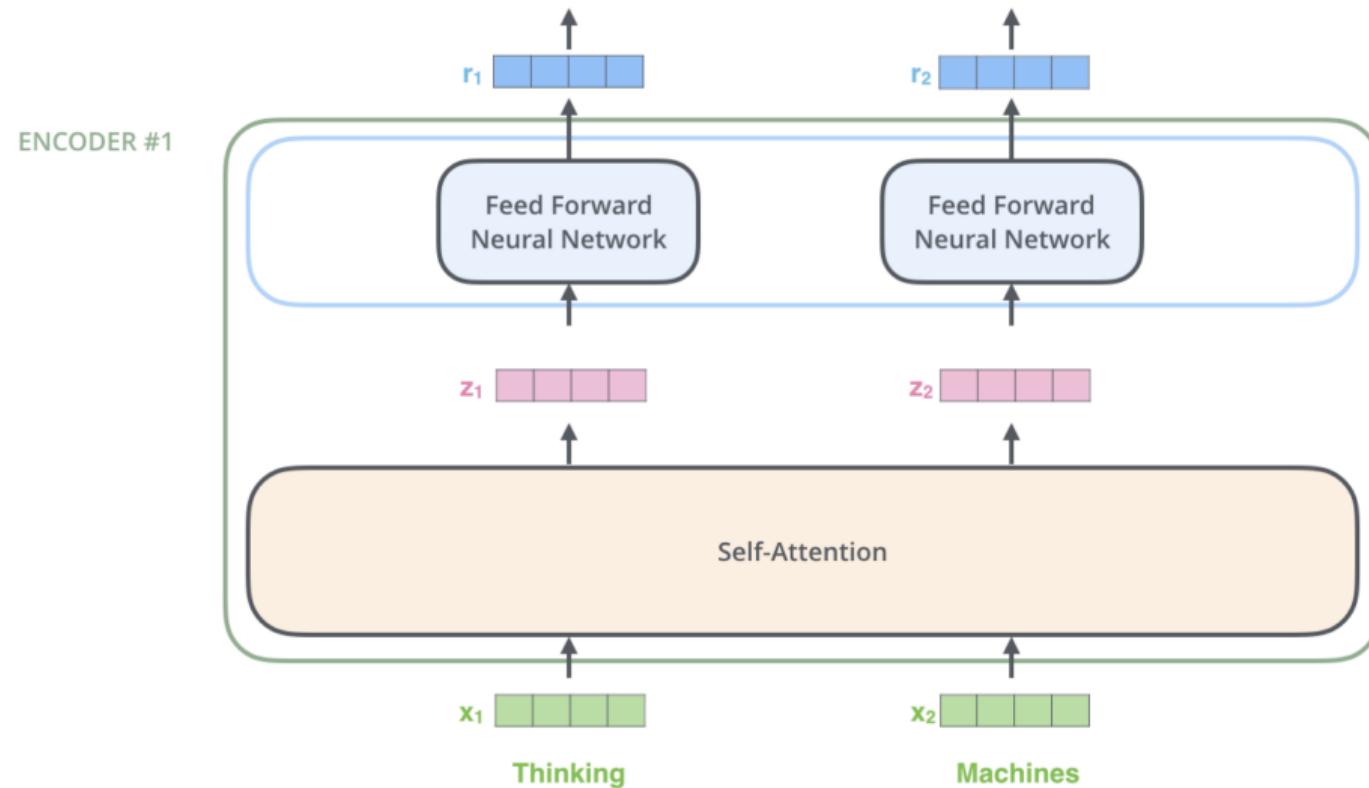
Последовательность слов, подаваемая в трансформер сначала разбивается на токены (слова, наборы букв или из подслов(BPE, SentencePiece, WordPiece, etc.)), которые подаются в Embedding слой.

Далее для учета порядка к полученным векторам добавляется некоторое смещение, зависящее от номера токена(Positional Encoding).

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

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

# Transformer (Attention is all you need)



# Transformer (Attention is all you need). Self-attention



$$\bar{q}_i = W^Q x_i$$

$$\bar{k}_i = W^K x_i$$

$$\bar{w}_i = W^V x_i$$

$$Z = \text{softmax} \left( \frac{1}{\sqrt{d}} Q K^T \right) V$$

$$z_i = \sum_j a_{ij} v_j, \quad a_{ij} = \text{softmax}_j \left( \frac{1}{\sqrt{d}} \langle q_i, k_j \rangle \right)$$

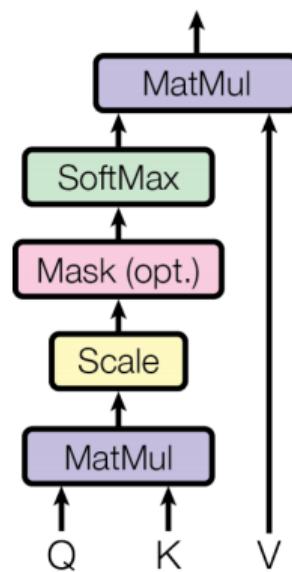
# Attention (Attention is all you need). MultiHead

Одна голова:  $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ .

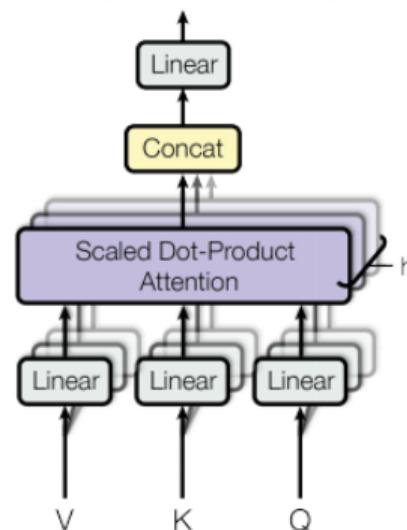
Параллелим:  $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$

Где  $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

One Head Attention

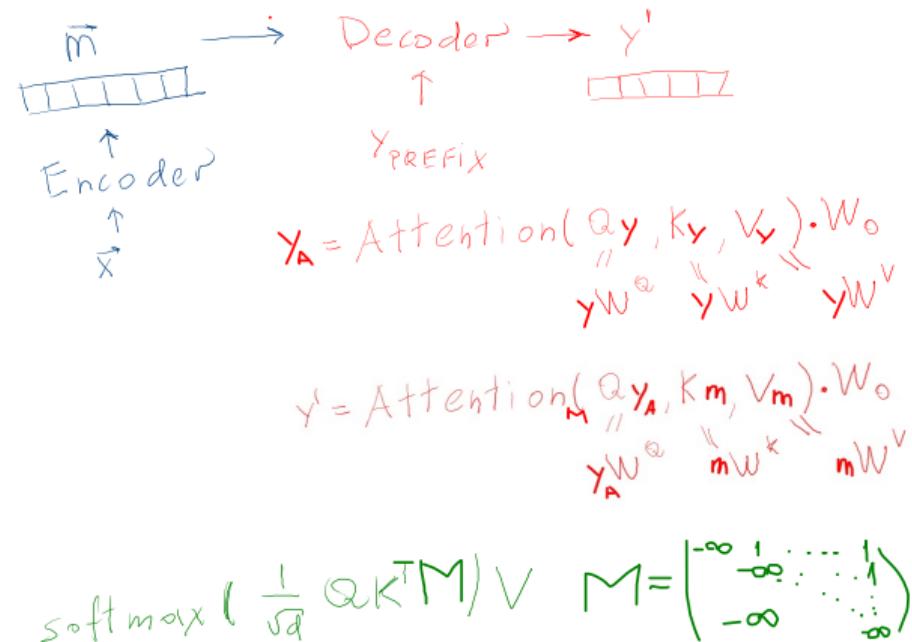


Multi-Head Attention

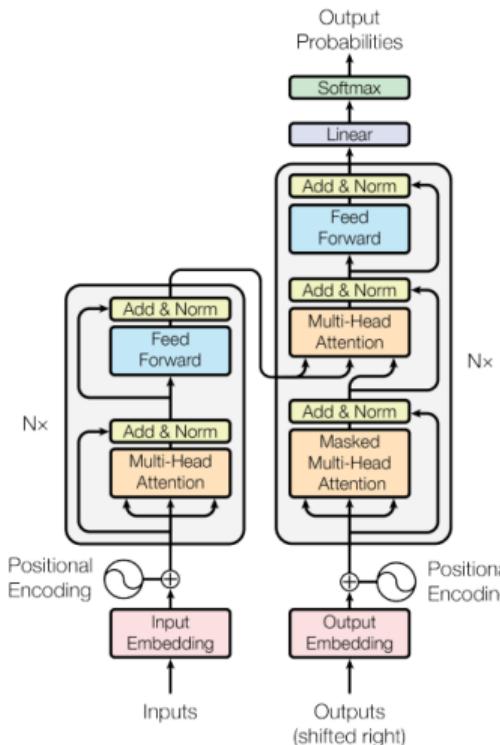


# Transformer (Attention is all you need). Decoder

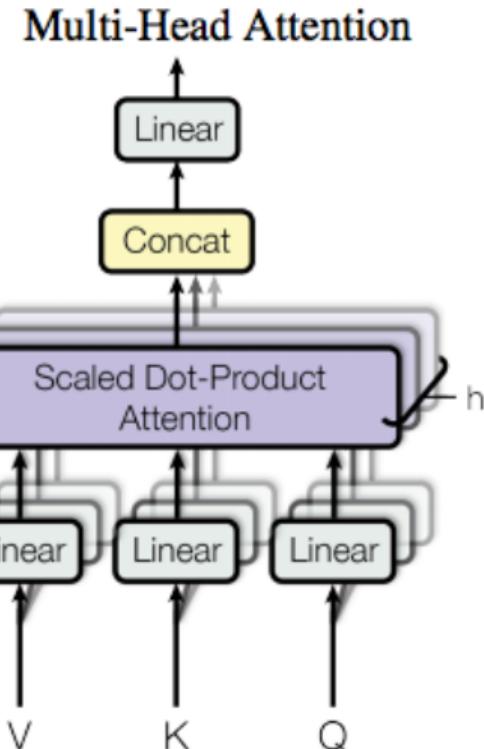
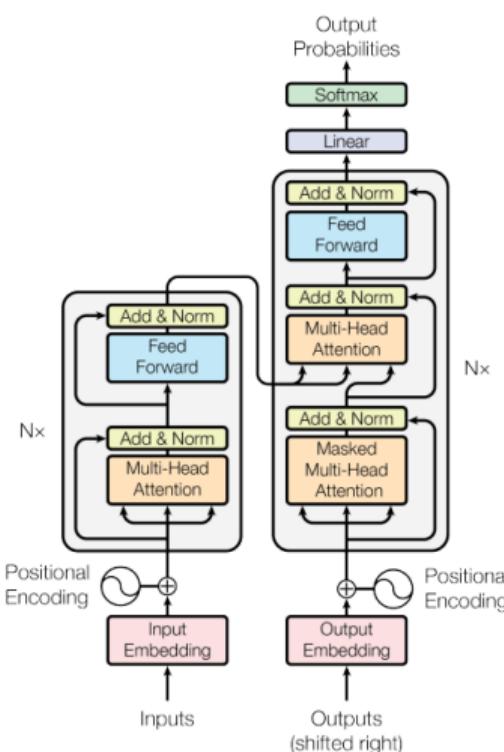
Decoder по сути устроен так же как энкодер, однако на вход в Decoder подается последний выход в Encoder ( $m$ ) + начало сгенерированной последовательности  $y_{prefix}$ . Также в Decoder-е используется Masked Self-Attention: мы запрещаем предыдущим токенам смотреть на следующие.



# Transformer (Attention is all you need)

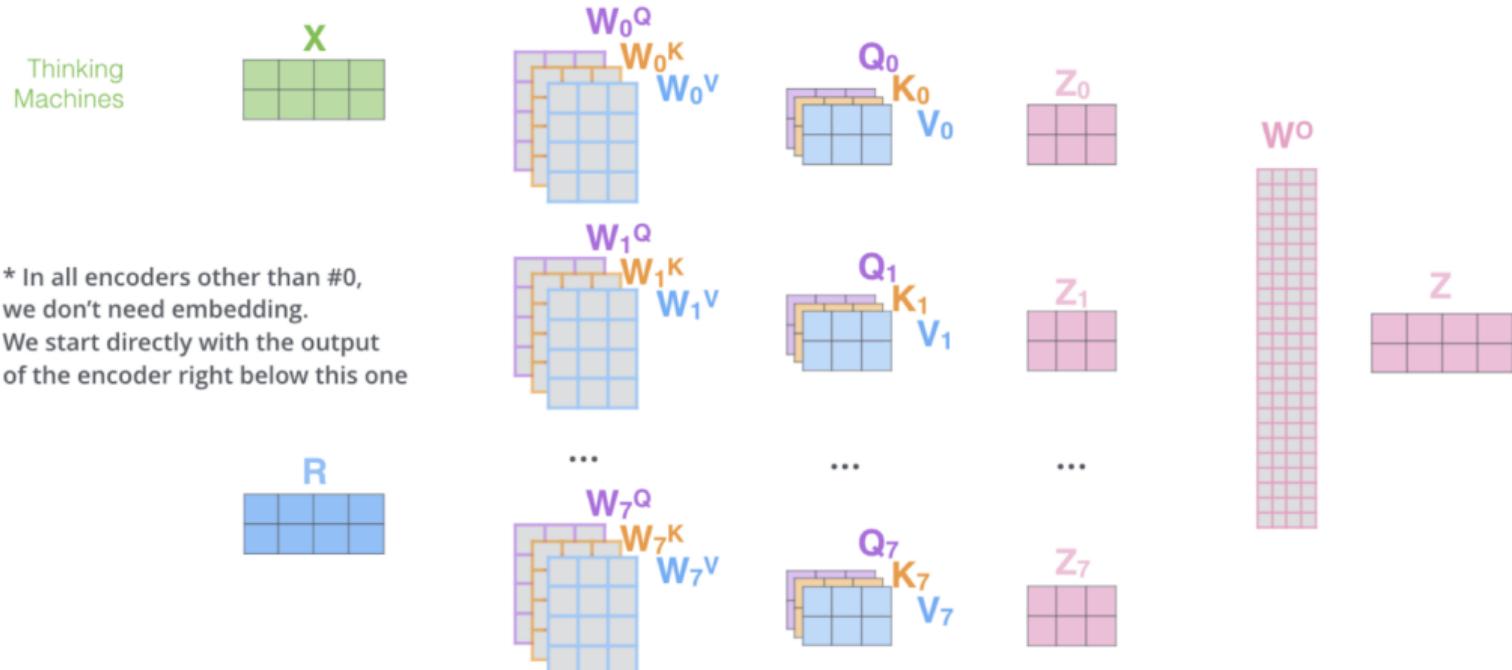


# Transformer (Attention is all you need)



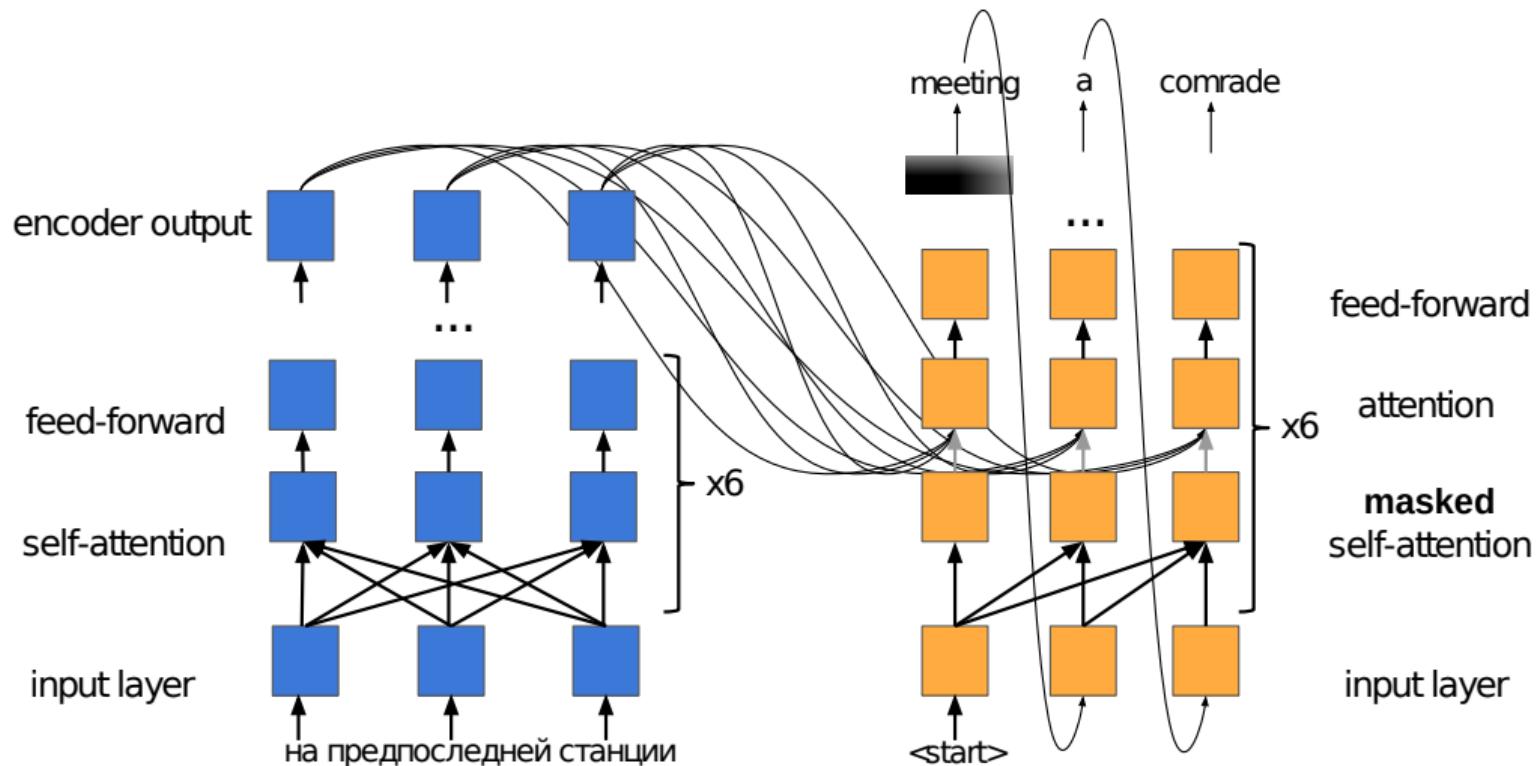
# Transformer (Attention is all you need). Full architecture

- 1) This is our input sentence\*
- 2) We embed each word\*
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices
- 4) Calculate attention using the resulting  $Q/K/V$  matrices
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^o$  to produce the output of the layer



# Transformer (Attention is all you need)<sup>7</sup>

Визуализация (gif)



# Transformer (Attention is all you need)

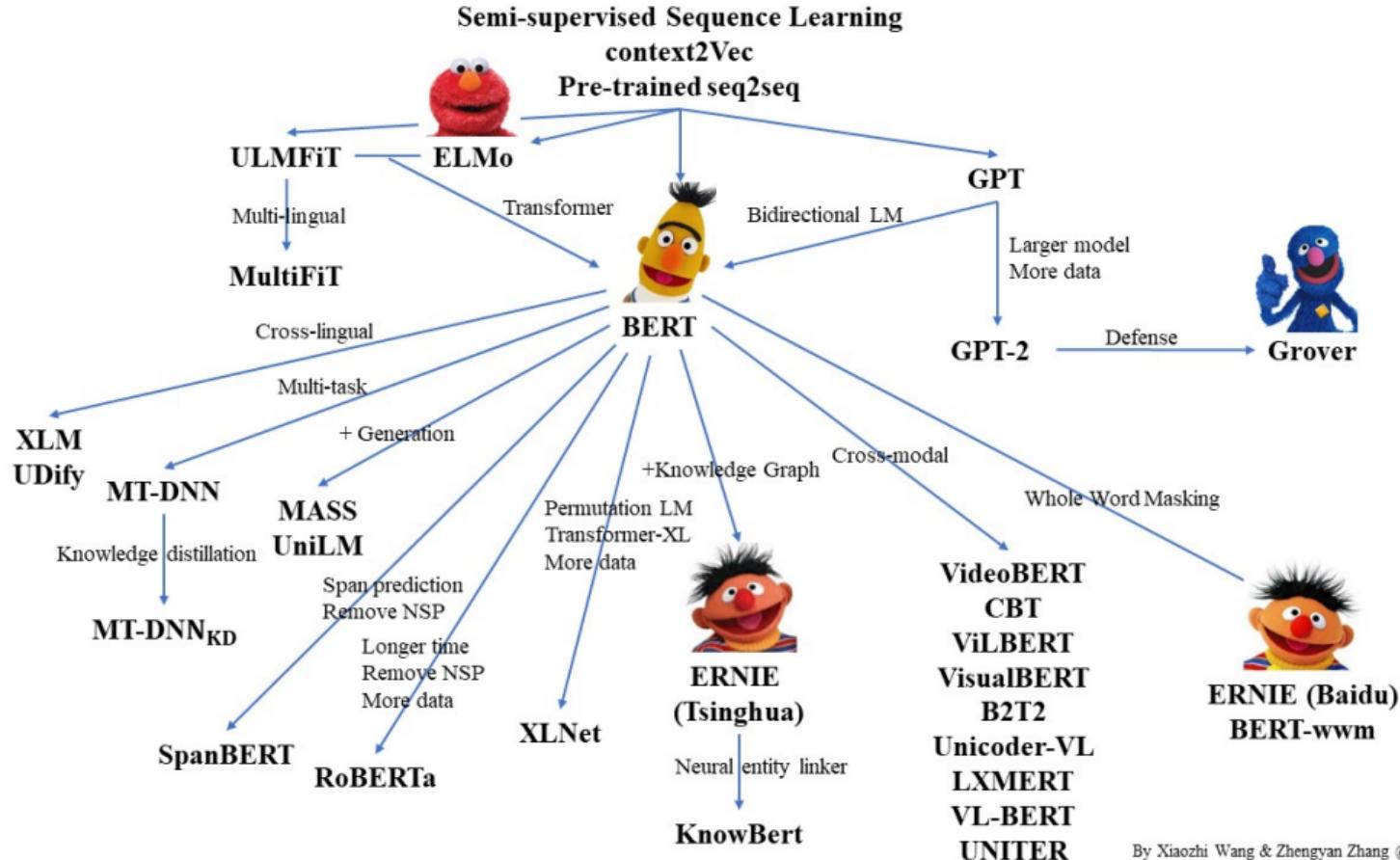
Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types.  $n$  is the sequence length,  $d$  is the representation dimension,  $k$  is the kernel size of convolutions and  $r$  the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

# Transformer (Attention is all you need)

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		<b><math>3.3 \cdot 10^{18}</math></b>
Transformer (big)	<b>28.4</b>	<b>41.8</b>		$2.3 \cdot 10^{19}$

# Transformer Family



# Архитектура GPT

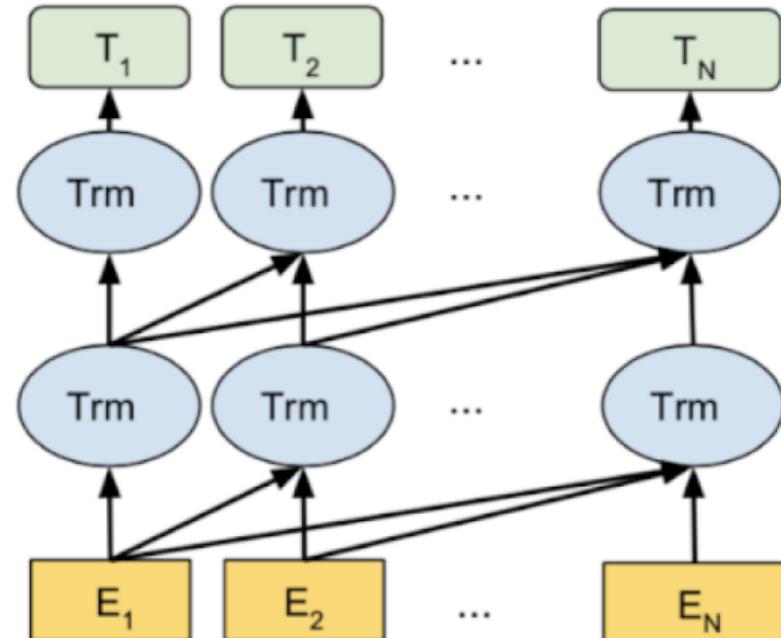
# GPT<sup>8</sup>

Language model:

$$P(w_{n+1} | w_1, \dots, w_n) = f(w_1, \dots, w_n)$$

В случае GPT используются слои трансформера.

Также для ускорения обучения мы можем за один проход учить сразу несколько предсказаний за счет масок. (каждая позиция может "смотреть" только назад)

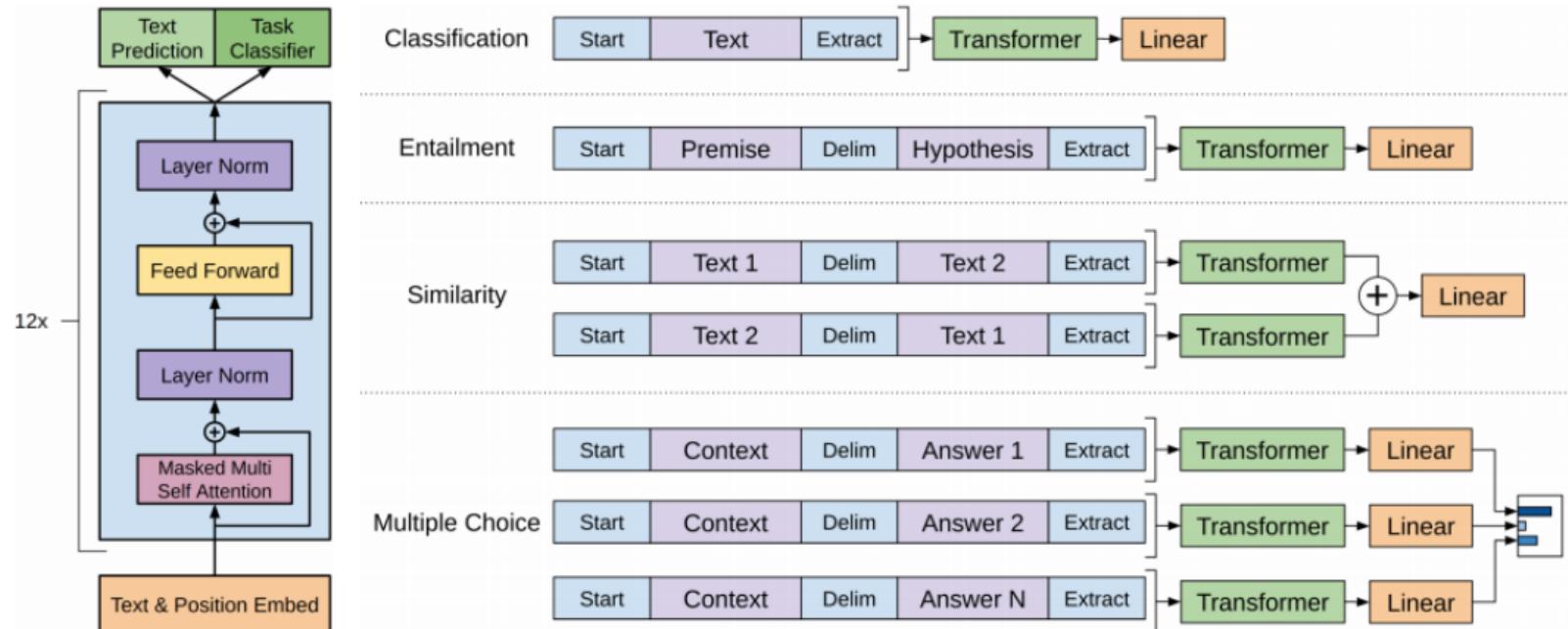


---

<sup>8</sup>[https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf)

# Multitask pretraining

Предобучаем модель на огромном корпусе →  
fine-tune на конкретную задачу.



# GPT-2<sup>9</sup> (TL;DR)

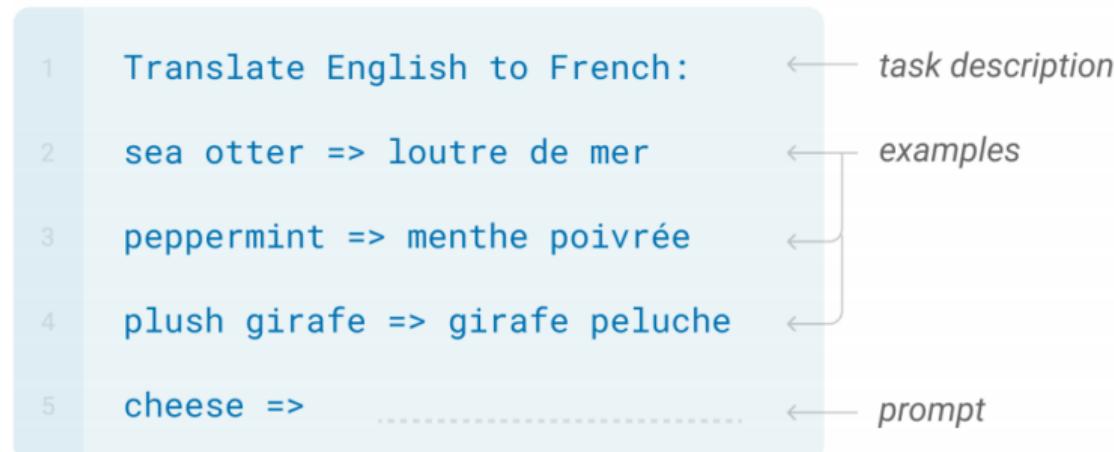
SYSTEM PROMPT (HUMAN-WRITTEN)	<p><i>A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.</i></p>
MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)	<p>The incident occurred on the downtown train line, which runs from Covington and Ashland stations.</p> <p>In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.</p>
	<p>"The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation," said Tom Hicks, the U.S. Energy Secretary, in a statement. "Our top priority is to secure the theft and ensure it doesn't happen again."</p>
	<p>The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials.</p>
	<p>The Nuclear Regulatory Commission did not immediately release any information.</p>
	<p>According to the release, the U.S. Department of Energy's Office of Nuclear Material Safety and Security is leading that team's investigation.</p>
	<p>"The safety of people, the environment and the nation's nuclear stockpile is our highest priority," Hicks said. "We will get to the bottom of this and make no excuses."</p>

<sup>9</sup>[https://d4mucfpksywv.cloudfront.net/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf)

# GPT-3<sup>10</sup> (zero-shot learning!)

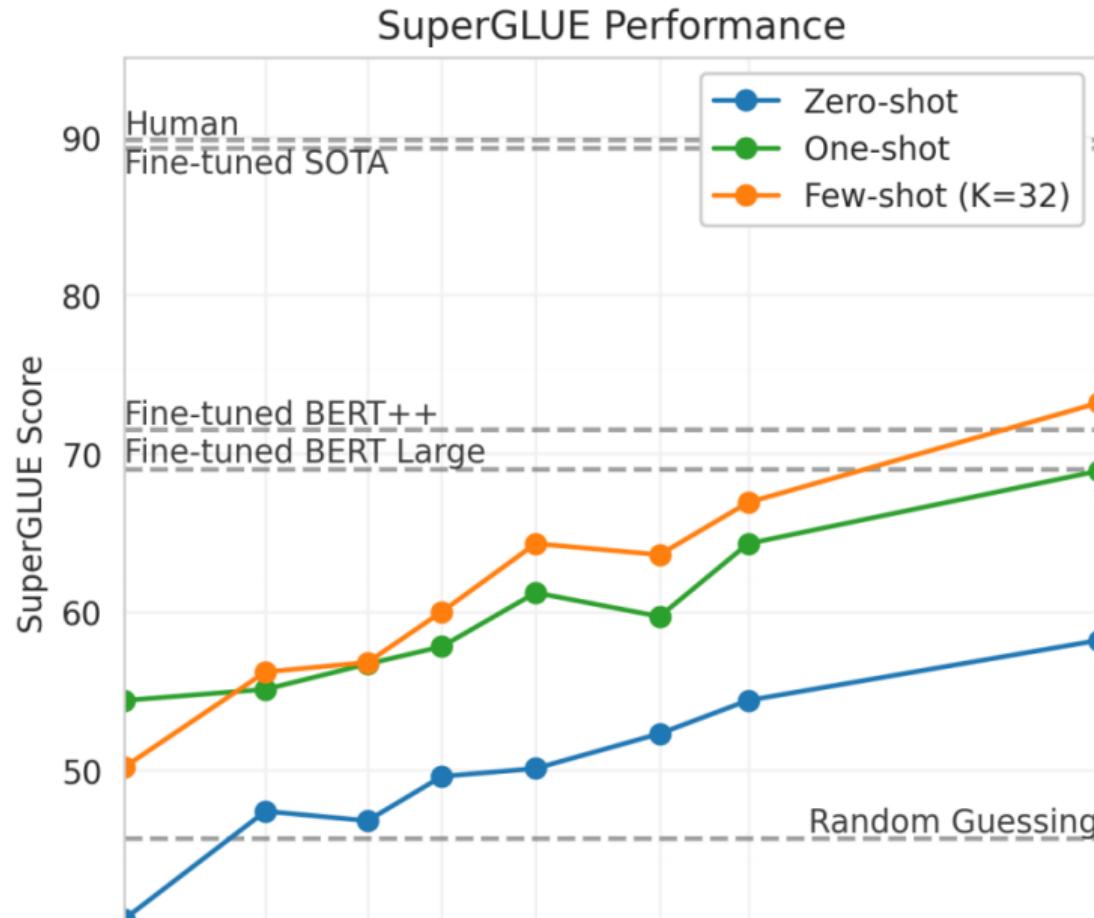
## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



<sup>10</sup><https://arxiv.org/abs/2005.14165>

# GPT-3 (zero-shot learning!)

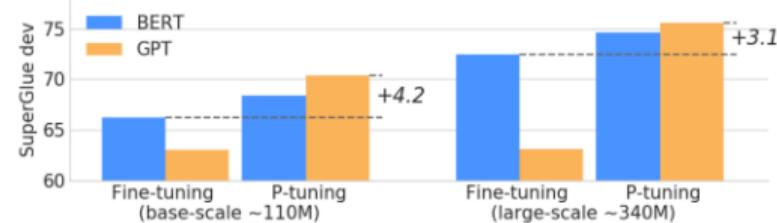


## P-tuning<sup>11</sup>

Prompt	P@1
[X] is located in [Y]. <i>(original)</i>	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08

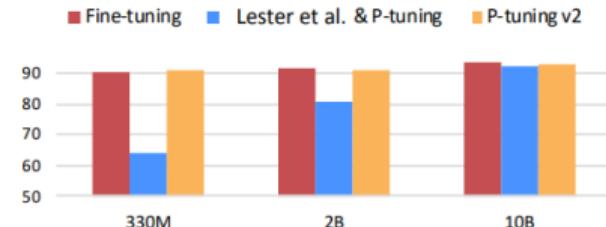
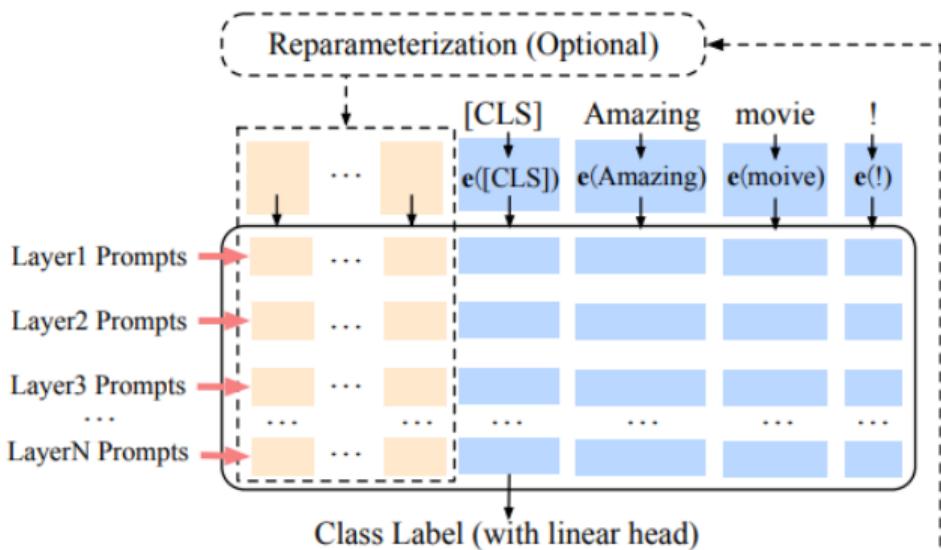
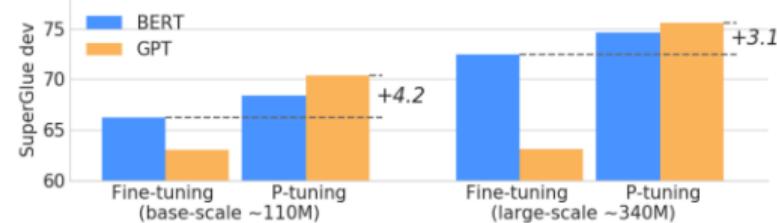
# P-tuning<sup>11</sup>

Prompt	P@1
[X] is located in [Y]. <i>(original)</i>	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08



# P-tuning<sup>11</sup>

Prompt	P@1
[X] is located in [Y]. (original)	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08



# GPT-3.5/ChatGPT

Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sampled from our prompt dataset.

Explain the moon landing to a 6 year old

A labeler demonstrates the desired output behavior.



Some people went to the moon...

This data is used to fine-tune GPT-3 with supervised learning.



Step 2

**Collect comparison data, and train a reward model.**

A prompt and several model outputs are sampled.

Explain the moon landing to a 6 year old

A Explain gravity...  
B Explain war...  
C Moon is natural satellite of...  
D People went to the moon...

A labeler ranks the outputs from best to worst.

D > C > A = B

This data is used to train our reward model.

RM  
D > C > A = B

Step 3

**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

Write a story about frogs

The policy generates an output.



Once upon a time...



The reward model calculates a reward for the output.

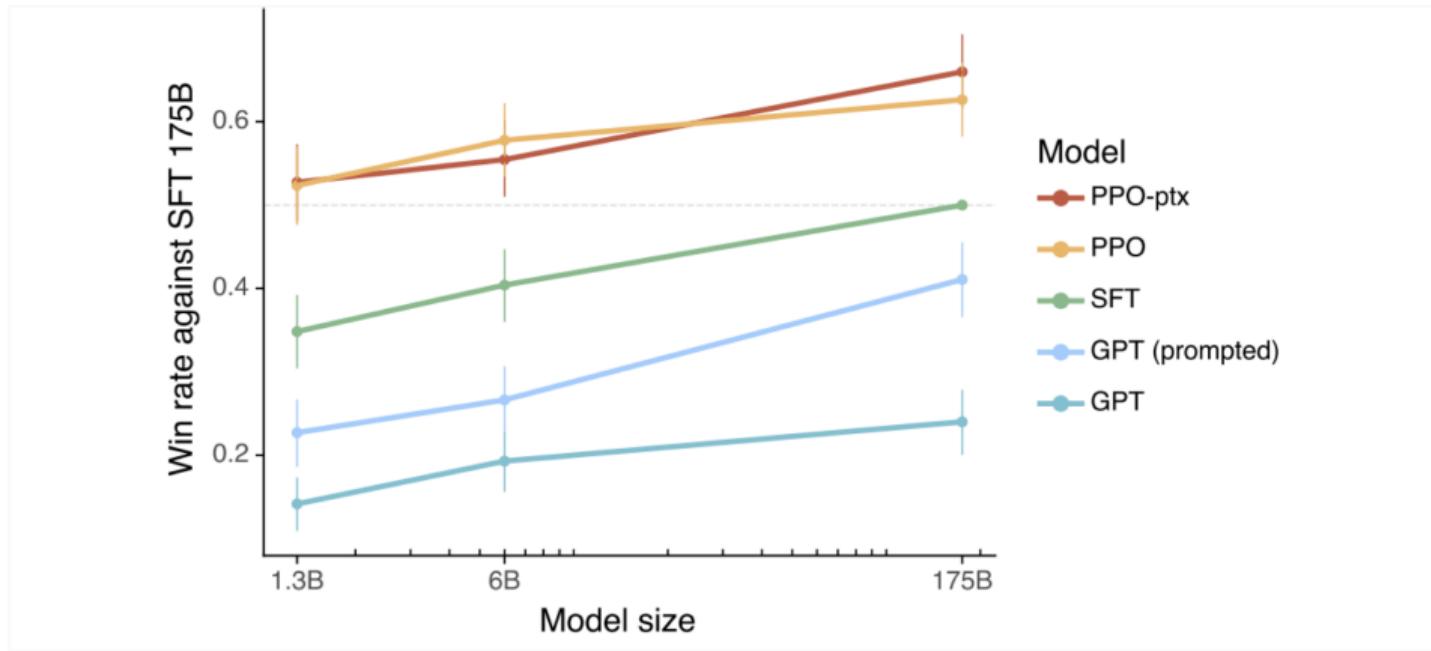


The reward is used to update the policy using PPO.

$r_k$

10

# GPT-3.5/ChatGPT



# GPT-3.5/ChatGPT

USER help me write a short note to introduce myself to my neighbor

CHATGPT Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,  
[Your Name]

USER can you make it more formal?

CHATGPT Dear Neighbor,

I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

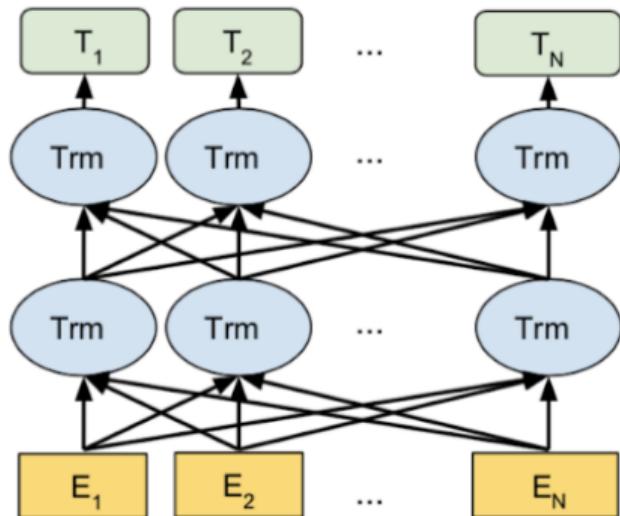
If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

Sincerely,  
[Your Name]

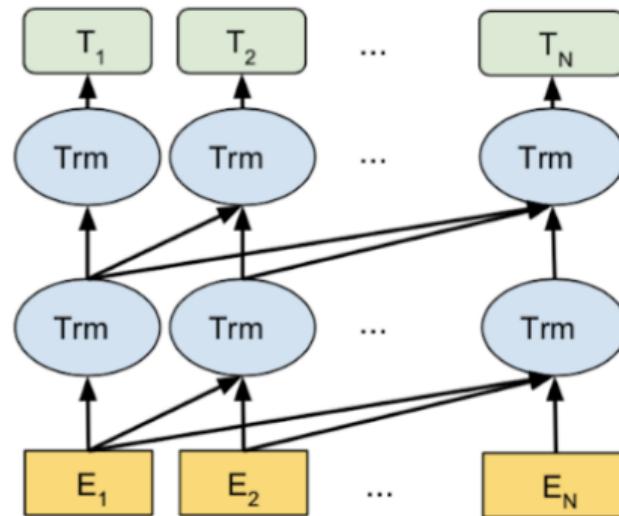
# Архитектура BERT

# BERT vs GPT

BERT (Ours)

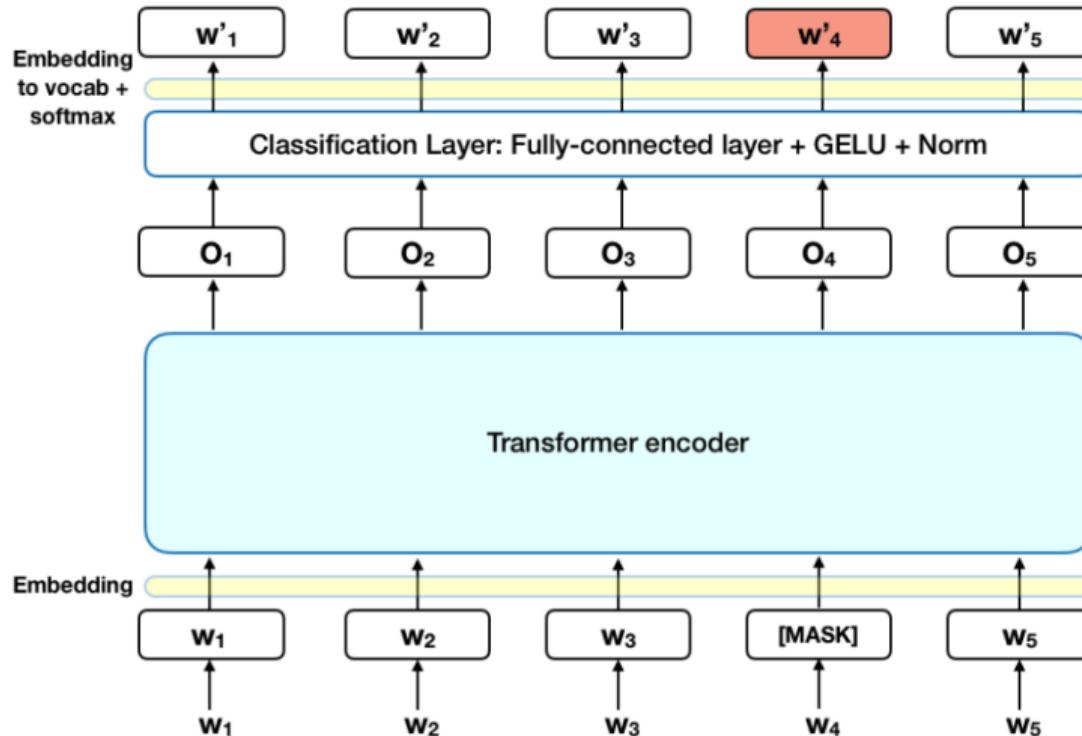


OpenAI GPT



# BERT<sup>12</sup>

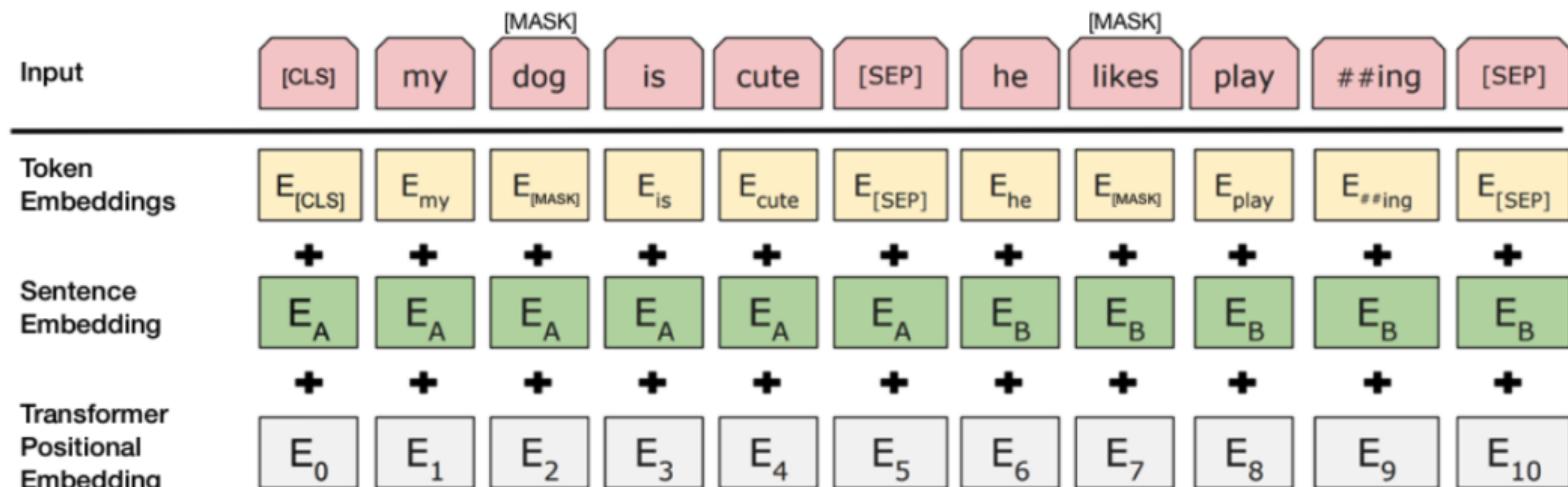
## Bidirectional Encoder Representations from Transformers



<sup>12</sup><https://arxiv.org/abs/1810.04805>

# BERT - детали

- ▶ MASK - некоторые слова заменяем на токен неизвестного слова и пытаемся их восстановить.
- ▶ NSP - Для пары предложений пытаемся предсказать, правда ли, что В следует за А. (берем В случайно в 50% случаев)  
Нужно для улучшения модели языка и вопросно-ответных задач.
- ▶ Обучающее множество включает всю английскую википедию и книги не защищенные авторским правом. Для большой модели надо 4 дня на 16-и cloud TPU.



# BERT - SOTA

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT <sub>BASE</sub> (Single)	80.8	88.5	-	-
BERT <sub>LARGE</sub> (Single)	84.1	90.9	-	-
BERT <sub>LARGE</sub> (Ensemble)	85.8	91.8	-	-
BERT <sub>LARGE</sub> (Sgl.+TriviaQA)	<b>84.2</b>	<b>91.1</b>	<b>85.1</b>	<b>91.8</b>
BERT <sub>LARGE</sub> (Ens.+TriviaQA)	<b>86.2</b>	<b>92.2</b>	<b>87.4</b>	<b>93.2</b>

(a) BERT на SQuAD v1.0 (найти сегмент с ответом)

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT <sub>BASE</sub>	81.6	-
BERT <sub>LARGE</sub>	<b>86.6</b>	<b>86.3</b>
Human (expert) <sup>†</sup>	-	85.0
Human (5 annotations) <sup>†</sup>	-	88.0

(b) BERT на SWAG (выбор из нескольких вариантов ответа)

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

## RoBERTa<sup>13</sup>

"We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it."

---

<sup>13</sup><https://arxiv.org/abs/1907.11692>

# RoBERTa<sup>13</sup>

"We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it."

- ▶ Объём - BERT обучался на 16GB текстов, мы будем учить на 160GB (включая датасет "хороших сайтов" GPT-2).
- ▶ NSP - учиться лучше на больших отрезках текста (параграфах, а не парах предложений), NSP не нужен! (без него на итоговых задачах не хуже, а иногда и лучше)
- ▶ Размер батча - оригинальный BERT учился на 256 примерах за раз, в работе показано, что лучше будет брать намного больший батч, например, 8К. (тут это был предел технических возможностей, есть работы, в которых увеличивали вплоть до 32K)
- ▶ RoBERTa - Robustly optimized BERT approach.

---

<sup>13</sup><https://arxiv.org/abs/1907.11692>

## ALBERT<sup>14</sup>

У BERT слишком много параметров ( $BERT_{large} \approx 334M$ ), на самом деле, столько не надо.

---

<sup>14</sup><https://arxiv.org/abs/1909.11942>

У BERT слишком много параметров ( $BERT_{large} \approx 334M$ ), на самом деле, столько не надо.

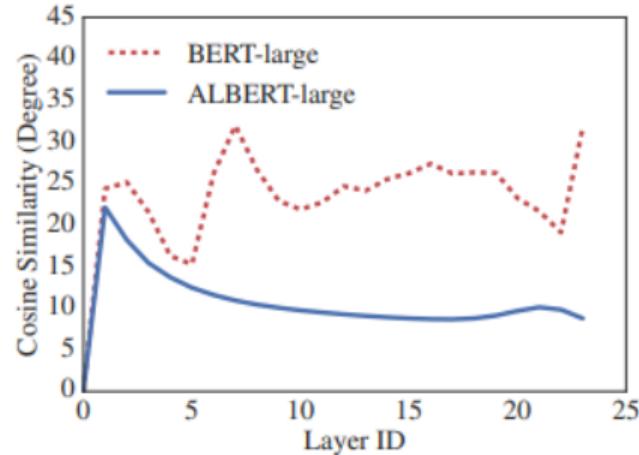
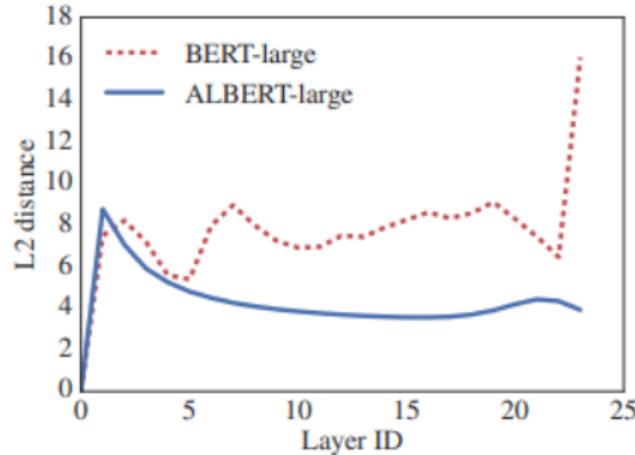
Как сокращать параметры:

- ▶ На эмбеддинги слов тратится слишком много параметров. Для лучшего качества мы хотим скрытые представления порядка  $H = 2048$ . При количестве слов около  $V = 30000$  (на самом деле, это под-слова, но об этом мы немного поговорим в конце) матрица  $V \times H$  получается слишком большой. Введем промежуточное представление размера  $E$  и факторизуем матрицу:  $V \times H = V \times E \times H$ . Получается меньше параметров и намного быстрее считать. (с помощью грид-серча  $E = 128$ )
- ▶ Давайте шарить веса на разных слоях трансформера. Вообще-то, давайте просто сделаем все слои одинаковыми (вспомните машину Больцмана). Оказывается, если применять этот слой много раз, эмбеддинги "стабилизируются".

---

<sup>14</sup><https://arxiv.org/abs/1909.11942>

# ALBERT



- ▶ NSP - он все-таки нужен, но другой. Используем SOP - предсказываем, правда ли, что в тексте A идет перед B, или нет.
- ▶ В итоге  $ALBERT_{xxlarge} \approx 235M$  меньше  $BERT_{large} \approx 334M$ , при этом считается всего в 3 раза дольше. (учится быстрее)

Советую прочитать статью, там невероятно многочисленных экспериментов, доказывающих все позиции, также некоторое осталось за рамками (например, про то, что Dropout делает хуже).

Есть проблемы с подходом BERT:

1. Токен маски (MASK) есть только во время предобучения, что создает смещение датасета во время файн-тюнинга
2. Когда мы пытаемся восстановить несколько слов, мы считаем, что эти слова независимы друг от друга, что не всегда правда

---

<sup>15</sup><https://arxiv.org/abs/1906.08237>

Есть проблемы с подходом BERT:

1. Токен маски (MASK) есть только во время предобучения, что создает смещение датасета во время файн-тюнинга
2. Когда мы пытаемся восстановить несколько слов, мы считаем, что эти слова независимы друг от друга, что не всегда правда

Давайте предсказывать слова по случайному контексту слева и справа от слова. Чтобы учиться предсказывать несколько слов, сгенерируем случайную перестановку и разрешим каждому слову смотреть только "назад".

Допустим, у нас есть предложение: [Разводные<sub>1</sub>, мосты<sub>2</sub>, в<sub>3</sub>, Санкт-<sub>4</sub>, Петербурге<sub>5</sub>].

Перемешаем слова: [в<sub>3</sub>, Разводные<sub>1</sub>, Петербурге<sub>5</sub>, Санкт-<sub>4</sub>, мосты<sub>5</sub>].

Хотим предсказать:

$$P(\text{Разводные}_1 | \text{в}_3), P(\text{Санкт-}_4 | \text{Разводные}_1, \text{в}_3, \text{Петербурге}_5)$$

---

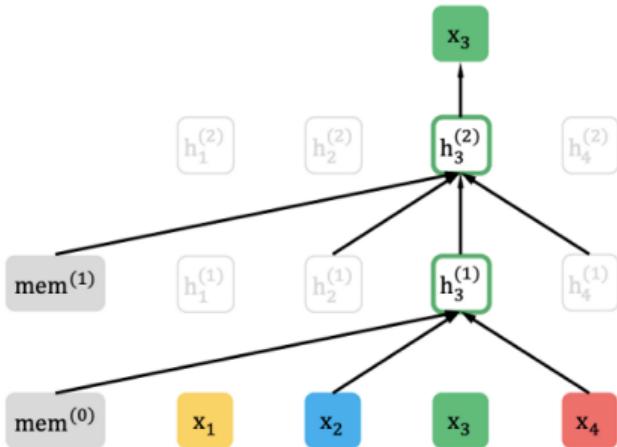
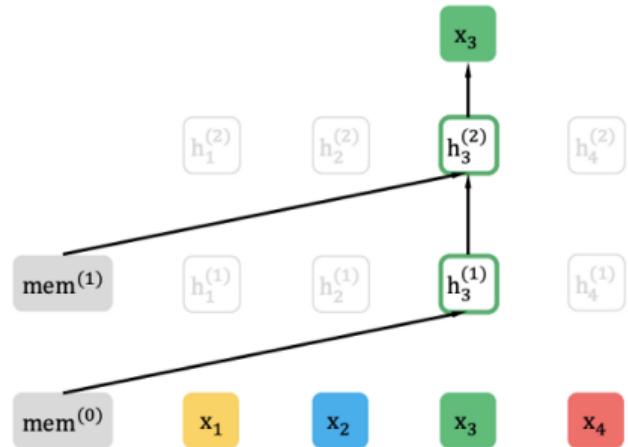
<sup>15</sup><https://arxiv.org/abs/1906.08237>

# XLNet

Моделируем:

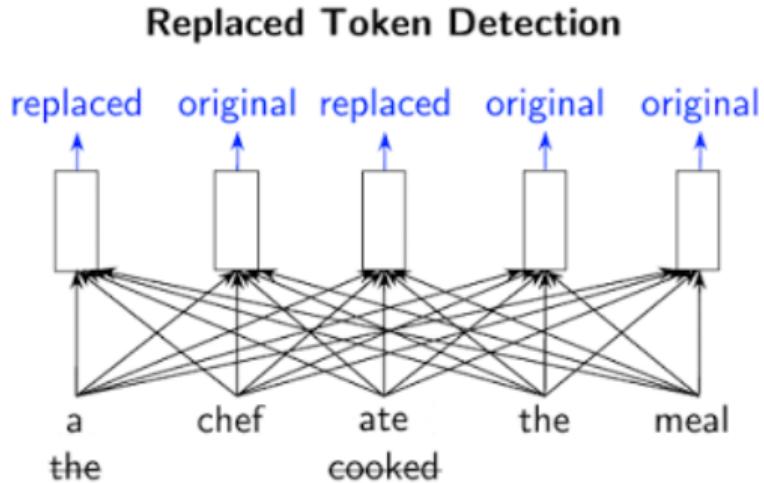
1. GPT -  $P(x_t | x_{i < t})$
2. BERT -  $P(x_t | x_{i \notin \{t, m_1, \dots\}})$
3. XLNet -  $P(x_t | x_{\sigma(i), i < \sigma^{-1}(t)})$

На самом деле, тут есть еще одна хитрость, чтобы понимать, какое слово предсказывать, но мы ее опустим.



# ELECTRA<sup>16</sup>

Восстанавливать замаскированные слова слишком просто, давайте искать замененные слова.

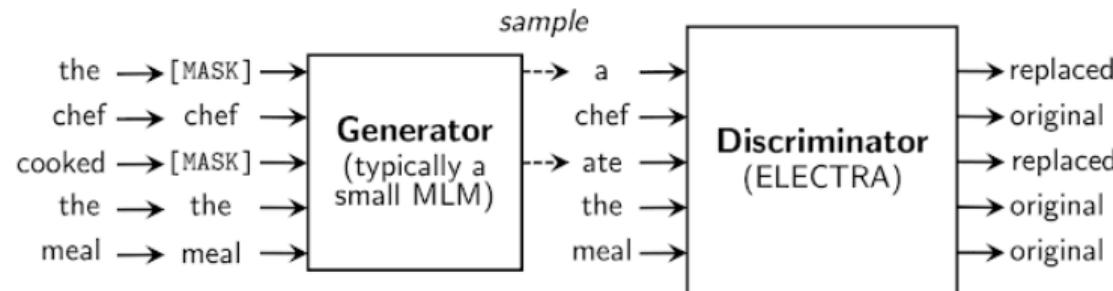
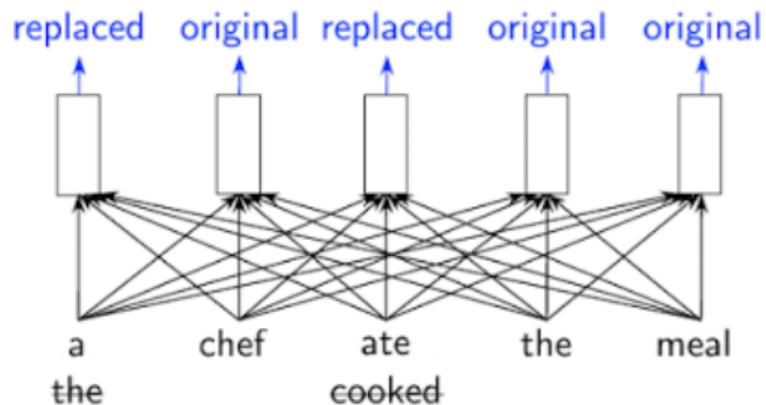


<sup>16</sup><https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html>

# ELECTRA<sup>16</sup>

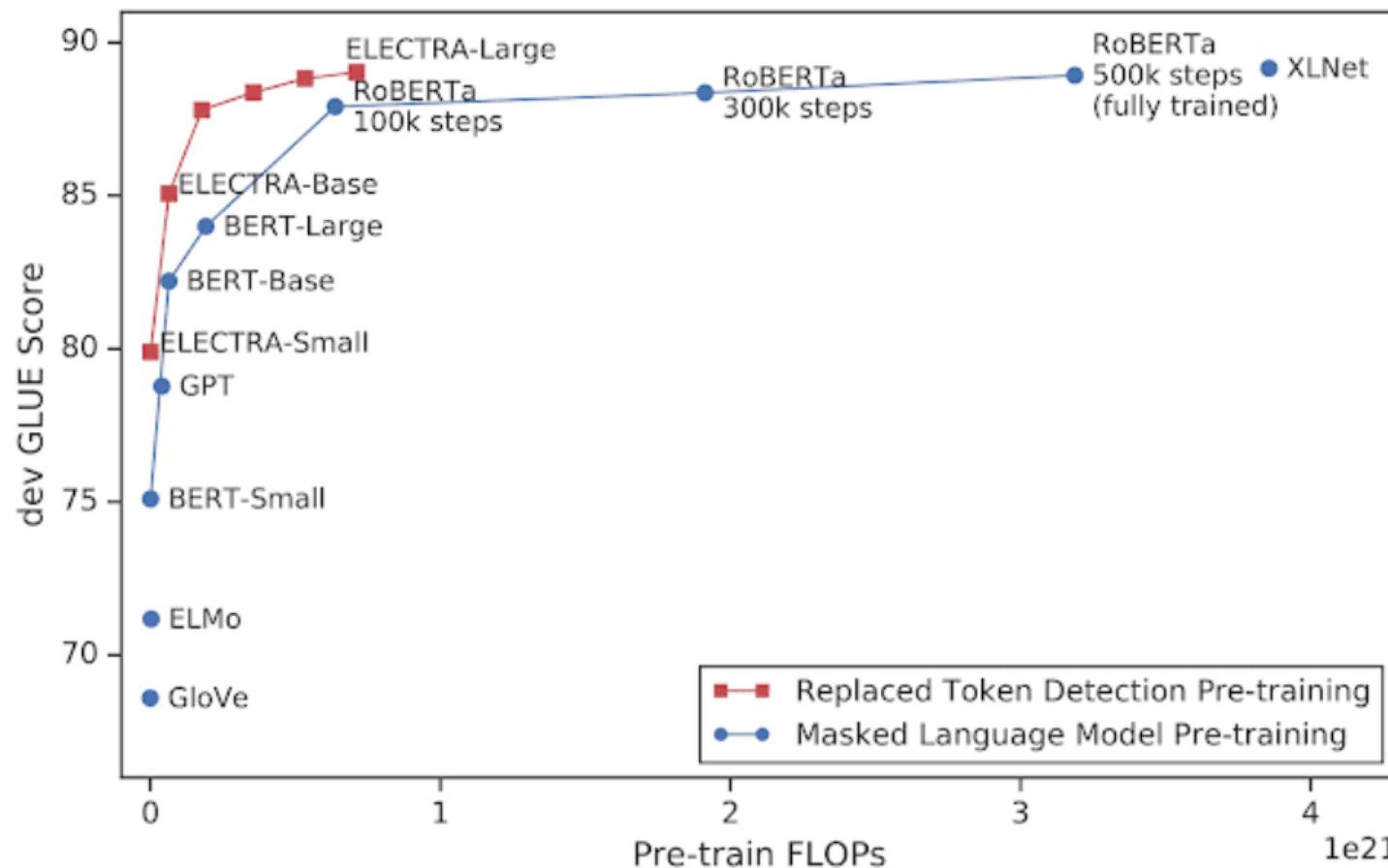
Восстанавливать замаскированные слова слишком просто, давайте искать замененные слова.

## Replaced Token Detection



<sup>16</sup><https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html>

# ELECTRA



## DeBERTa<sup>17</sup>

Авторы статьи решают проблему разделения (распутывания) смысловой информации токена и его расположения, вводя вектора ( $H_i$  и  $P_{i|j}$ ), контекстный вектор и вектор относительной позиции.

Для таких векторов записывается расширенная операция внимания:

$$\begin{aligned} A_{i,j} &= \{H_i, P_{i|j}\} \times \{H_j, P_{j|i}\}^\top \\ &= H_i H_j^\top + H_i P_{j|i}^\top + P_{i|j} H_j^\top + P_{i|j} P_{j|i}^\top \end{aligned}$$

$$Q_c = HW_{q,c}, K_c = HW_{k,c}, V_c = HW_{v,c}, Q_r = PW_{q,r}, K_r = PW_{k,r}$$

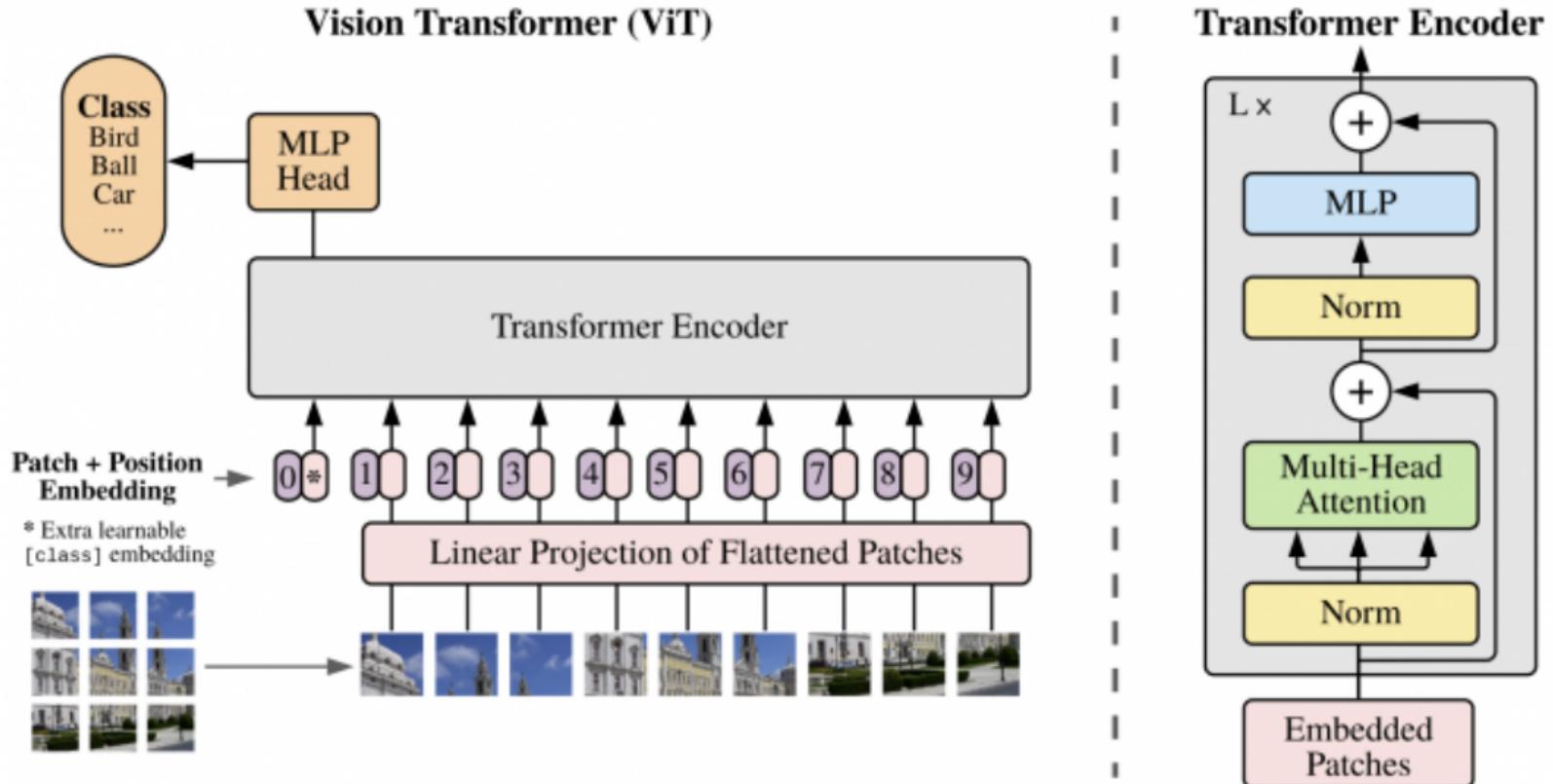
$$\tilde{A}_{i,j} = \underbrace{Q_i^c K_j^{c\top}}_{\text{(a) content-to-content}} + \underbrace{Q_i^c K_{\delta(i,j)}^r}^{\top}_{\text{(b) content-to-position}} + \underbrace{K_j^c Q_{\delta(j,i)}^r}^{\top}_{\text{(c) position-to-content}}$$

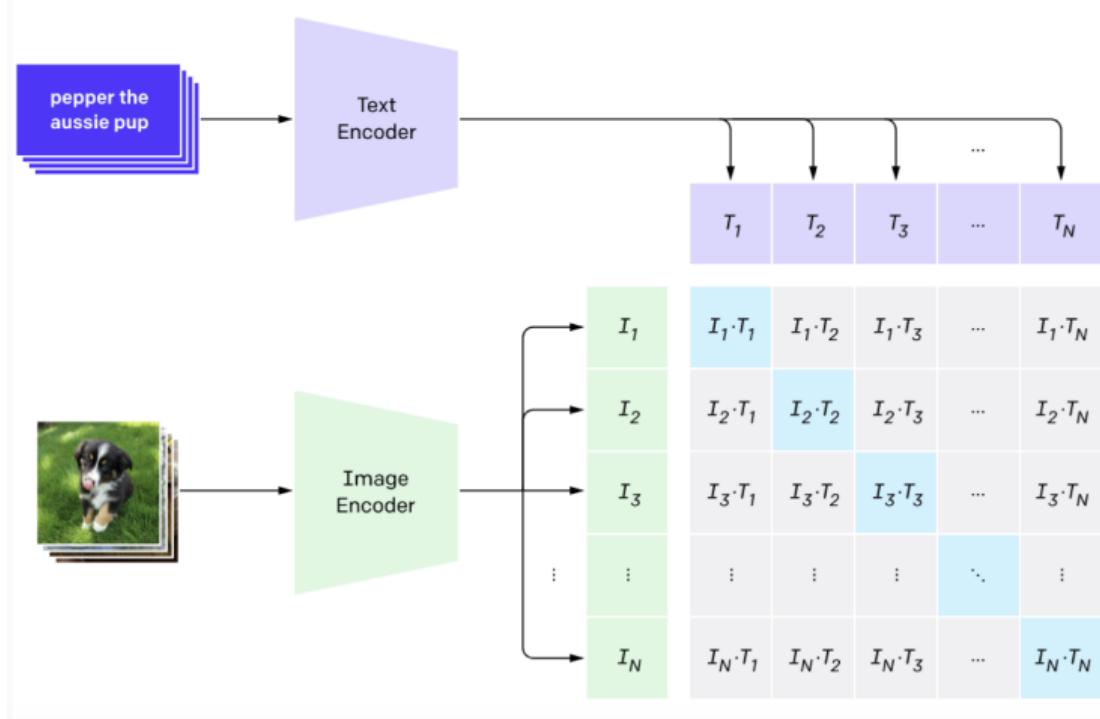
$$H_o = \text{softmax}\left(\frac{\tilde{A}}{\sqrt{3d}}\right)V_c$$

---

<sup>17</sup><https://arxiv.org/abs/2006.03654>

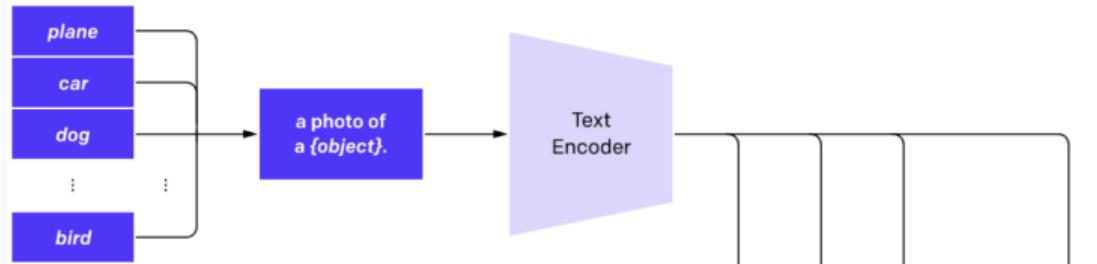
Transformer для картинок



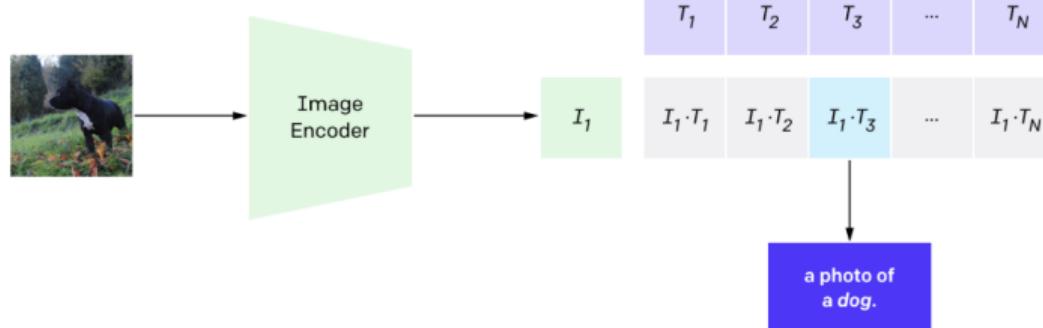
**1. Contrastive pre-training**

# CLIP

## 2. Create dataset classifier from label text



## 3. Use for zero-shot prediction



# Заключение

## С чем мы познакомились?

- ▶ Представления слов для алгоритмов машинного обучения (OneHot, Word2Vec, основанные на токенизации)
- ▶ Механизмы внимания
- ▶ Модель трансформера и ее улучшения

## За рамками лекции

- ▶ Дистилляция моделей
- ▶ Методы декодирования (лучевой метод (beamsearch), top-k)
- ▶ Технические подробности обучения огромных моделей
- ▶ Модель T5
- ▶ Модели для длинных последовательностей. Transformer XL, Reformer, Sparse Transformer, BigBird.

## Ссылки

- ▶ Attention Is All You Need <https://arxiv.org/abs/1706.03762>
- ▶ The Illustrated Transformer <https://jalammar.github.io/illustrated-transformer/>
- ▶ Курс Stanford NLP <https://web.stanford.edu/class/cs224n/>
- ▶ Курс Александра Дьяконова <https://github.com/Dyakonov/DL>
- ▶ Huggingface transformers <https://huggingface.co/docs/transformers>

# Вопросы



Будем  
ВКонтакте!

