

Дерунец Роман

Трансформеры Лекция 10

Внимание и ничего кроме внимания?

“Attention is all you need” (Vaswani et al., 2017)

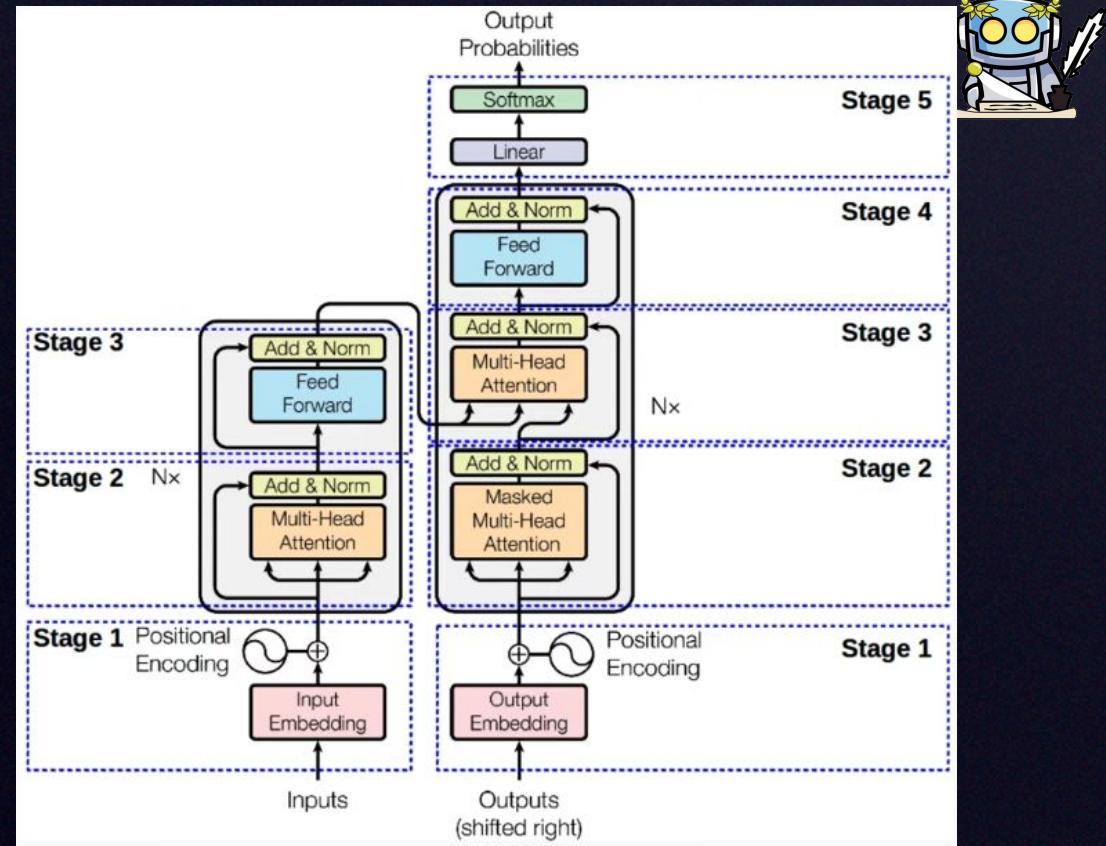


- Все может быть проще
- Основная идея — self-attention
- Оказалось, что подходит для всевозможных seq2seq задач
- Главная мотивация – уйти от кодирования вектором постоянной длины

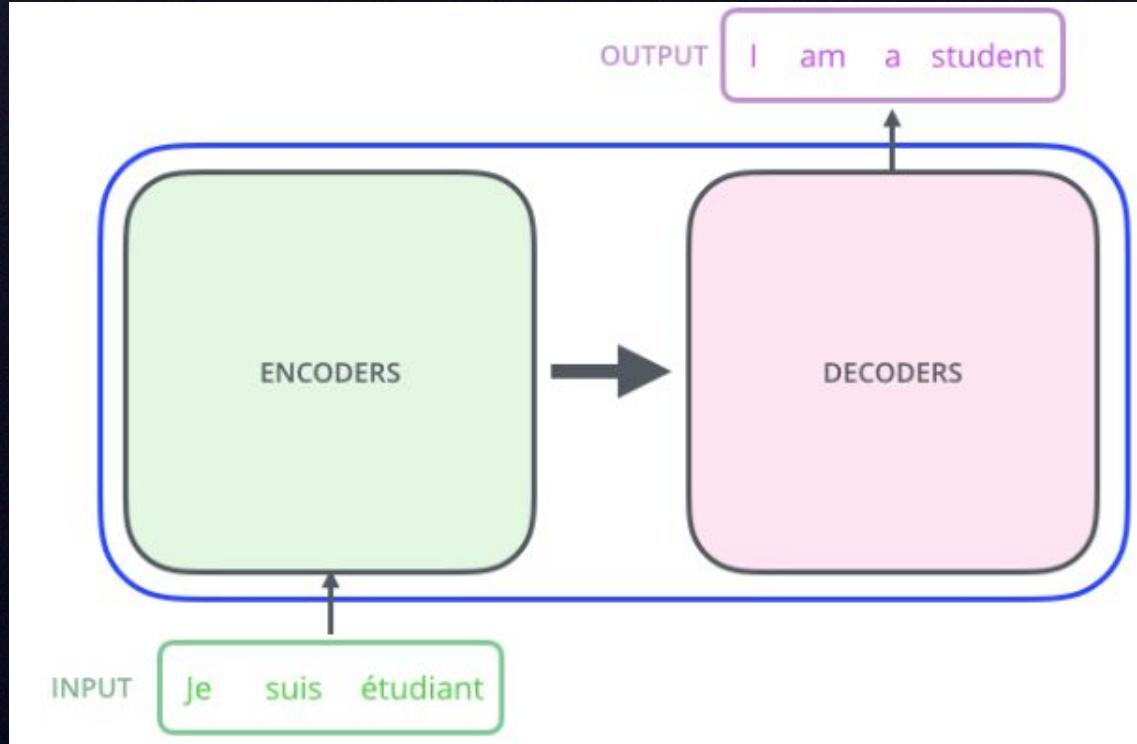
Transformer



Attention is all you need

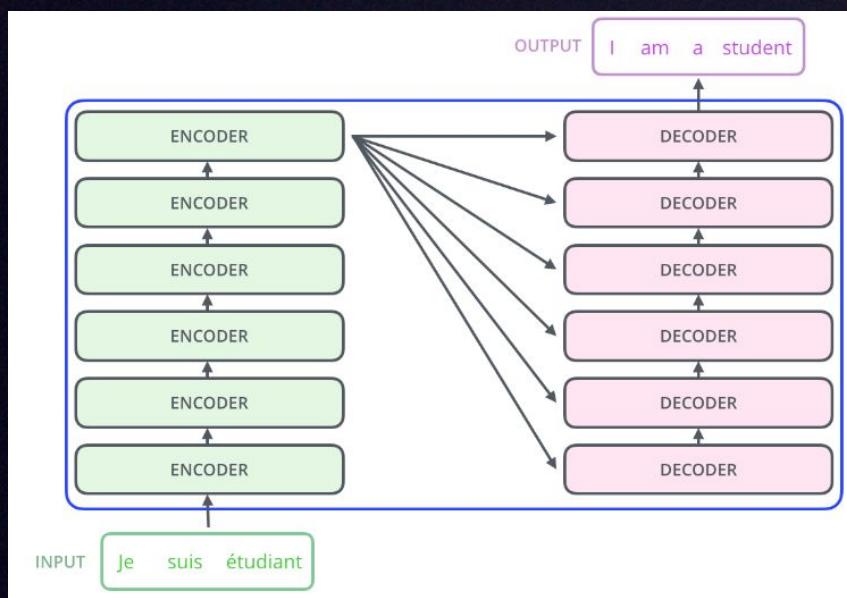


Transformer



Transformer

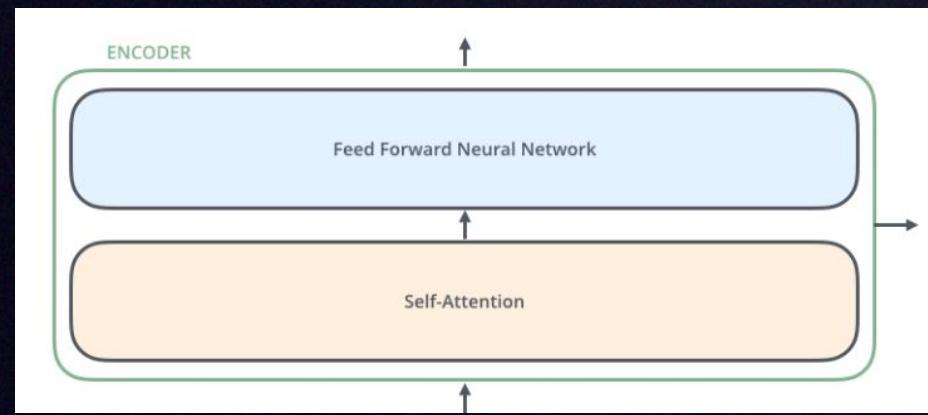
- 6 слоев энкодера
- 6 слоев декодера



Transformer



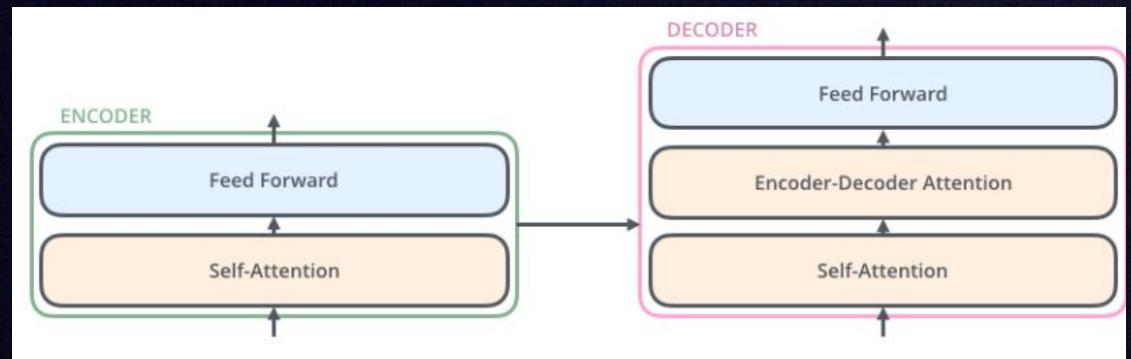
- У энкодера две части:
FFNN (применяемый к каждой позиции входа)
и Self-Attention



Transformer



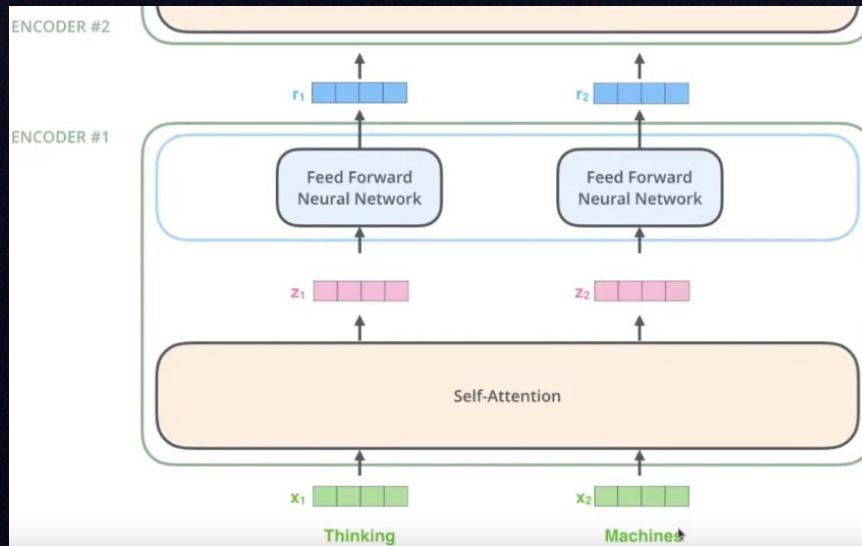
- У декодера ещё одна дополнительная часть – Enc-Dec Attention



Transformer



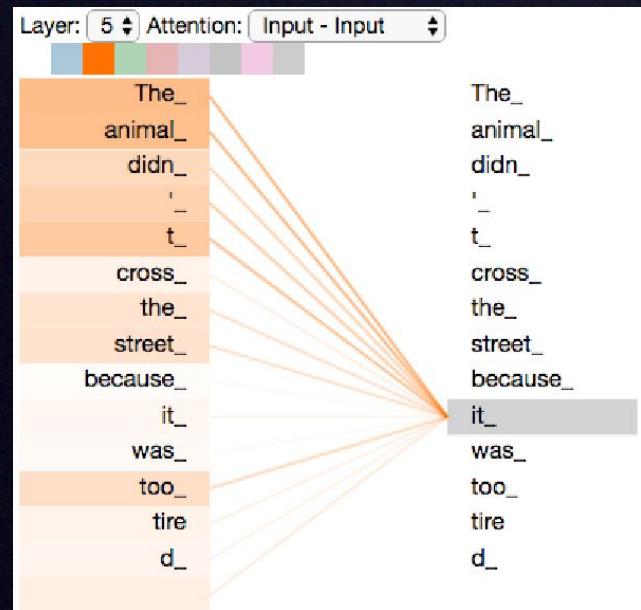
- Слова представляются векторами, в FFNN всё параллельно
- На вход энкодера идет уже векторное представление
- Encoder дальше улучшает векторы и строит более хорошее представление



Что такое self-attention



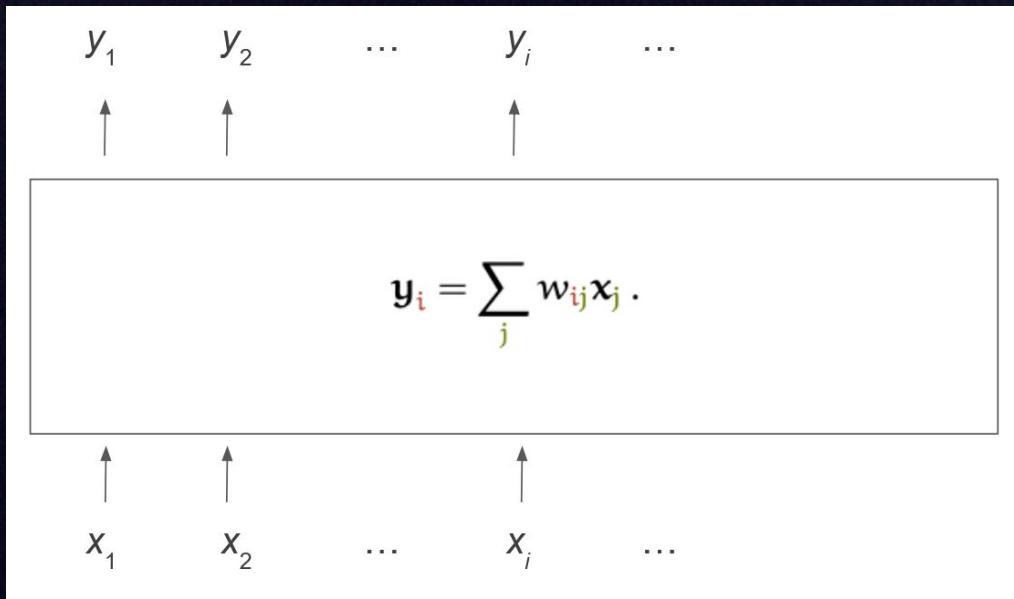
- Идея: обучить веса, с которыми обработка текущего слова будет учитывать другие слова



Что такое self-attention



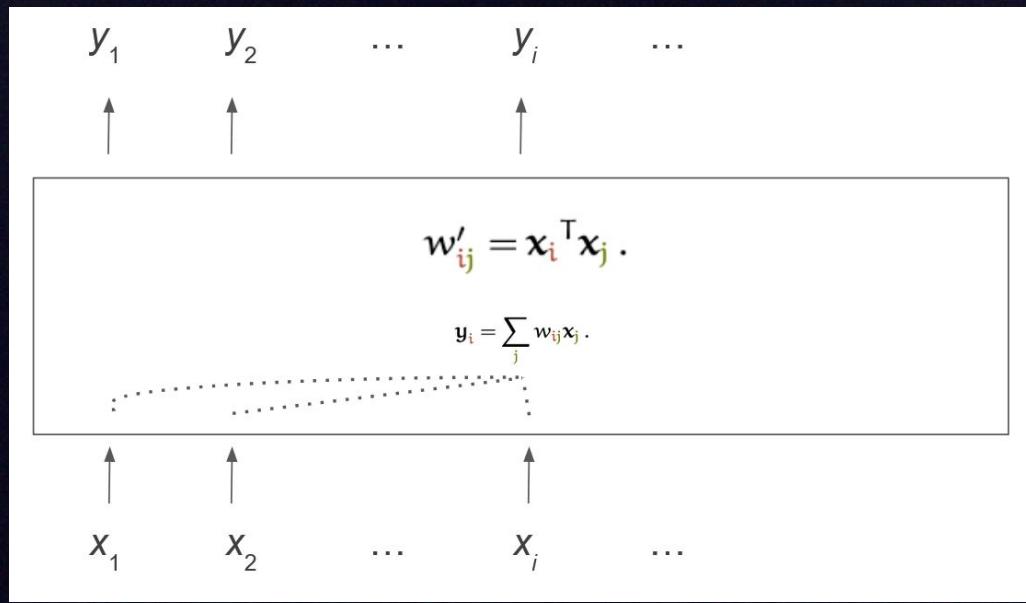
- Простое самовнимание



Что такое self-attention



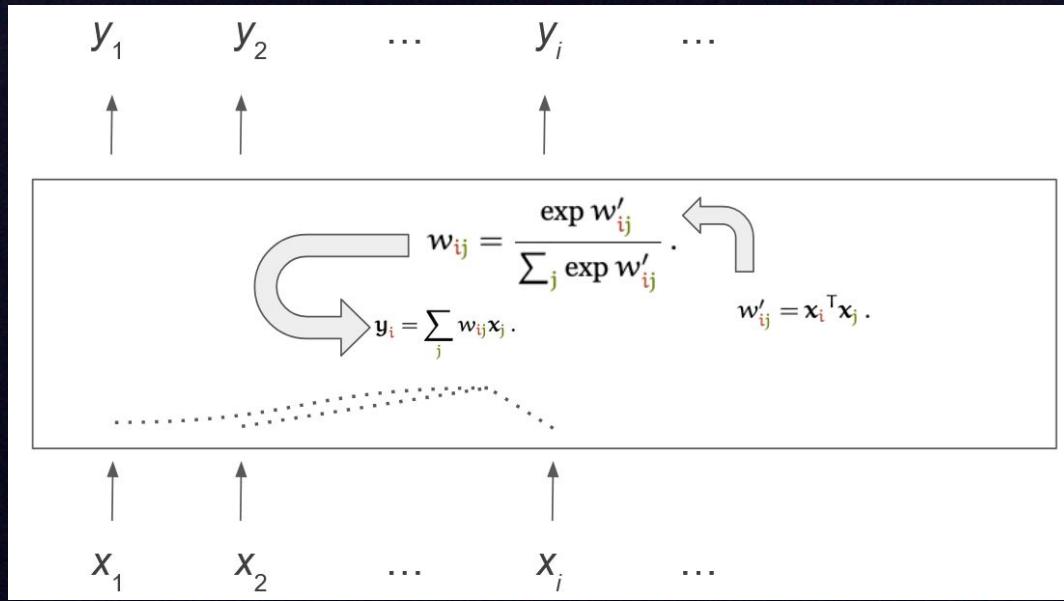
- Внимание! w_{ij} - это не вес!



Что такое self-attention



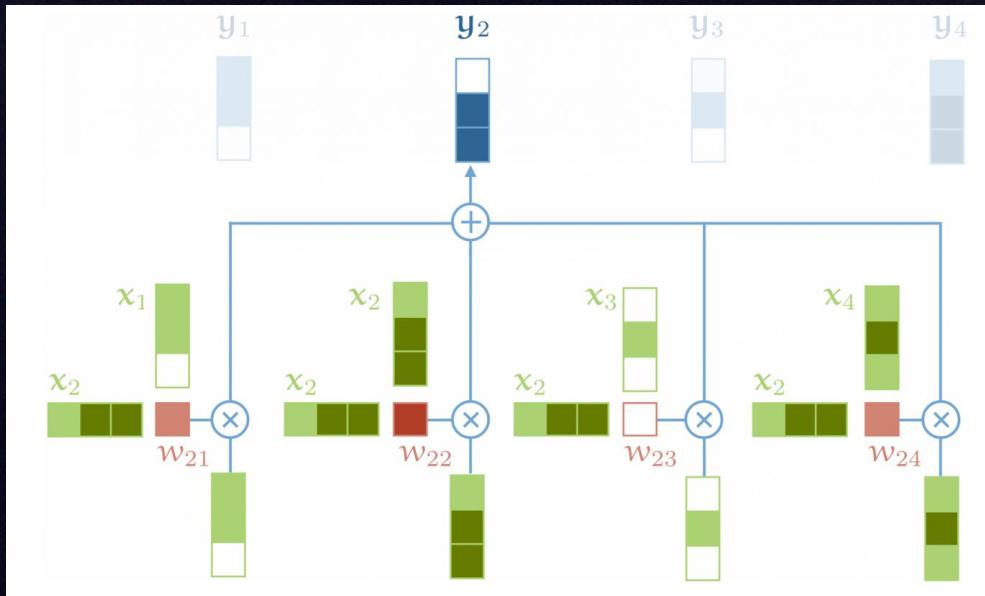
- Внимание! w_{ij} - это распределение вероятностей контекста!



Что такое self-attention



- Внимание! w_{ij} - это распределение вероятностей контекста!



Что такое self-attention



- x_i - **это запрос**: его сравнивают с каждым из других векторов для установки веса соответствующего ему y_j ;
- x_i - **это ключ**: его сравнивают с каждым из других векторов для установки весов других y_j ;
- x_i - **это значение**: он после установки весов участвует в вычислении взвешенной суммы

Что такое self-attention



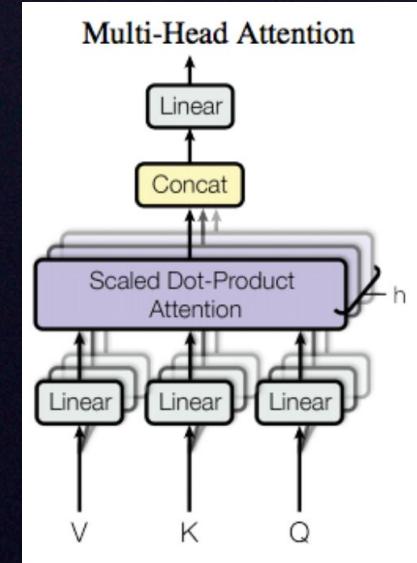
- Разделим роли запроса, ключа и значения

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i$$

$$w'_{ij} = q_i^T k_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

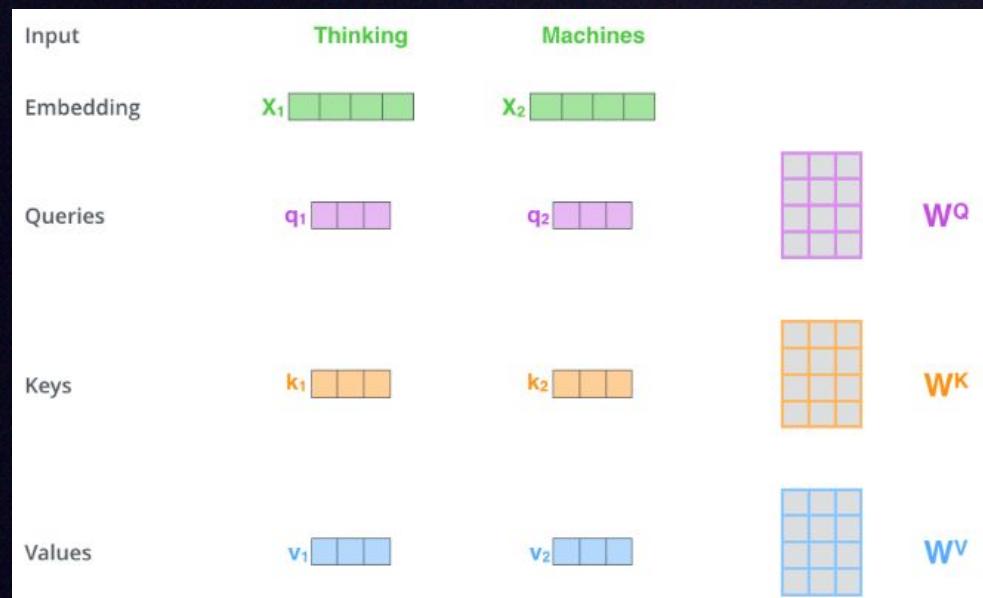
$$y_i = \sum_j w_{ij} v_j .$$



Self-attention шаг 1



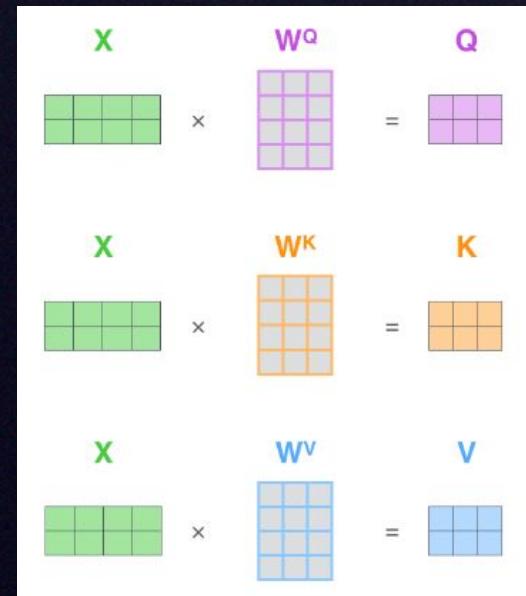
- Получаем проекции query, key, value каждого слова (x) в исходном предложении



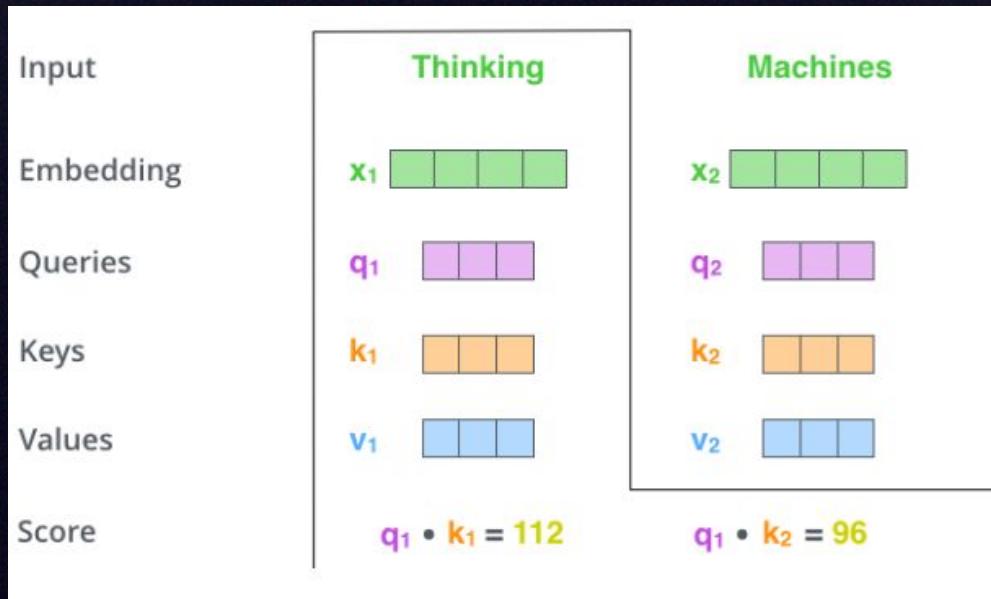
Self-attention K, Q, V



- Каждая строка в матрице X отвечает за слово входного предложения



Self-attention шаг 2

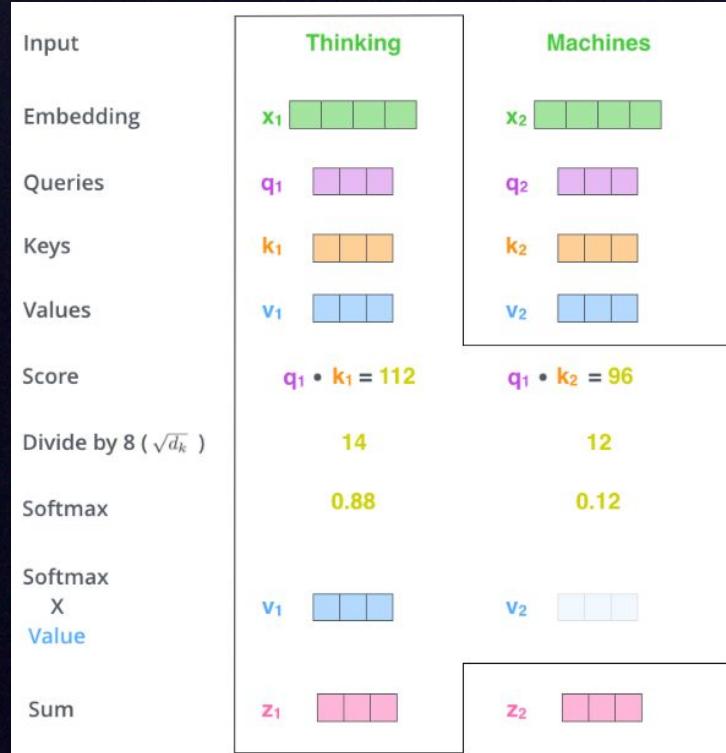


Self-attention шаг 3 и 4, нормализация и softmax



Input	Thinking		Machines	
Embedding	x_1	[4 green boxes]	x_2	[4 green boxes]
Queries	q_1	[3 purple boxes]	q_2	[3 purple boxes]
Keys	k_1	[3 orange boxes]	k_2	[3 orange boxes]
Values	v_1	[3 blue boxes]	v_2	[3 blue boxes]
Score	$q_1 \cdot k_1 = 112$		$q_1 \cdot k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

Self-attention взвешенная сумма значений слов



Self-attention в матричной форме



$$\text{softmax} \left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$

=

$$\mathbf{Z}$$

The diagram illustrates the matrix form of self-attention. It shows the calculation of attention weights from query \mathbf{Q} and key \mathbf{K}^T , and their application to value \mathbf{V} . Below, the result is labeled \mathbf{Z} .

Q K^T

V

softmax

$\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}$

\mathbf{V}

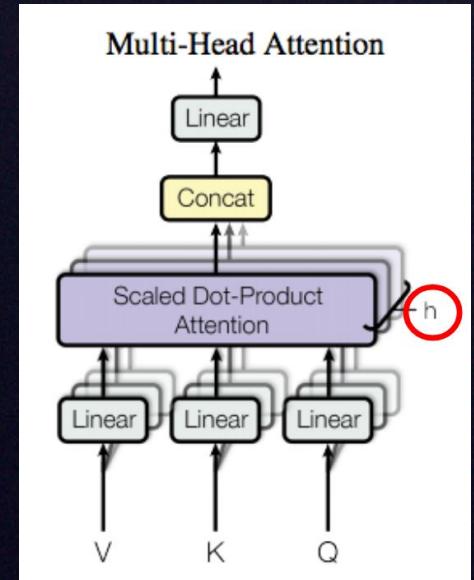
=

Z

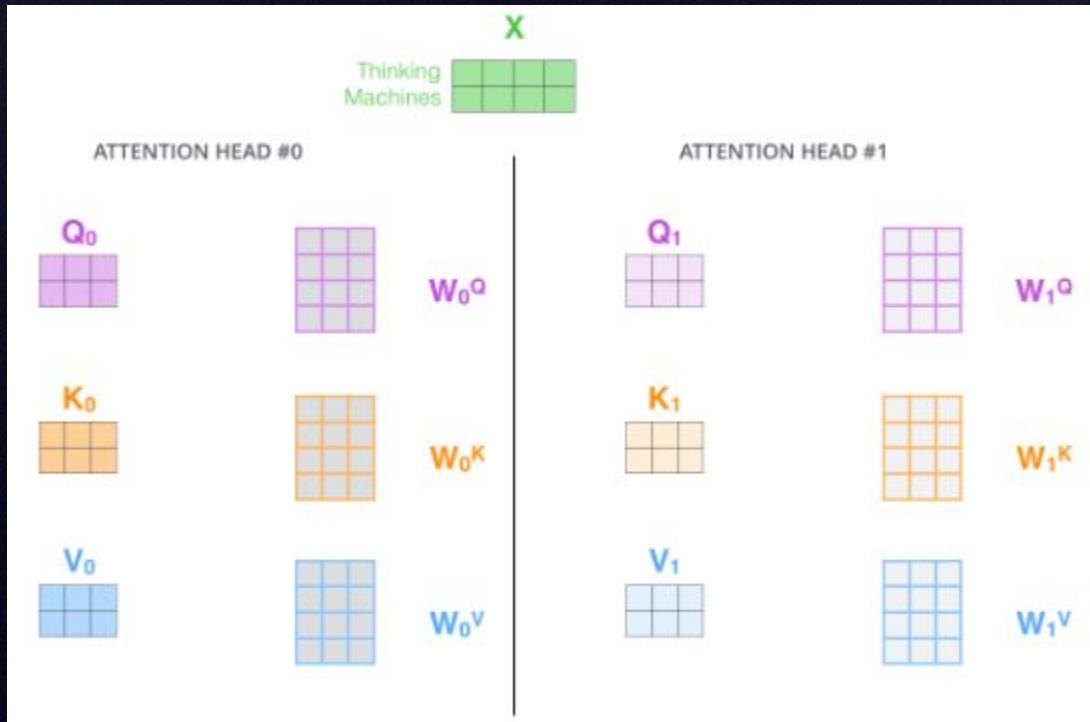
Что такое self-attention



- h - это количество параллельных “голов” само-внимания

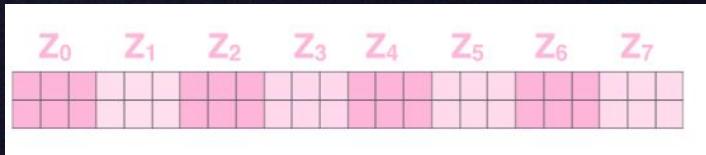


Многоголовое внимание

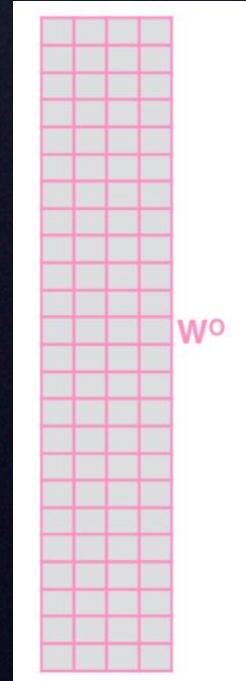


Многоголовое внимание

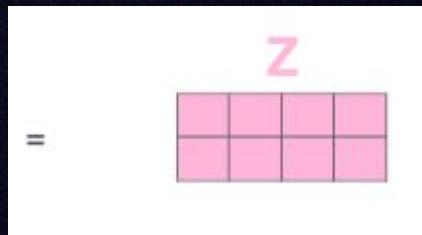
Производим конкатенацию



Перемножаем с
матрицей весов,
обученной
совместно



В результате получаем матрицу , которая захватывает информацию
со всех голов внимания. Можем отправить её в FFNN

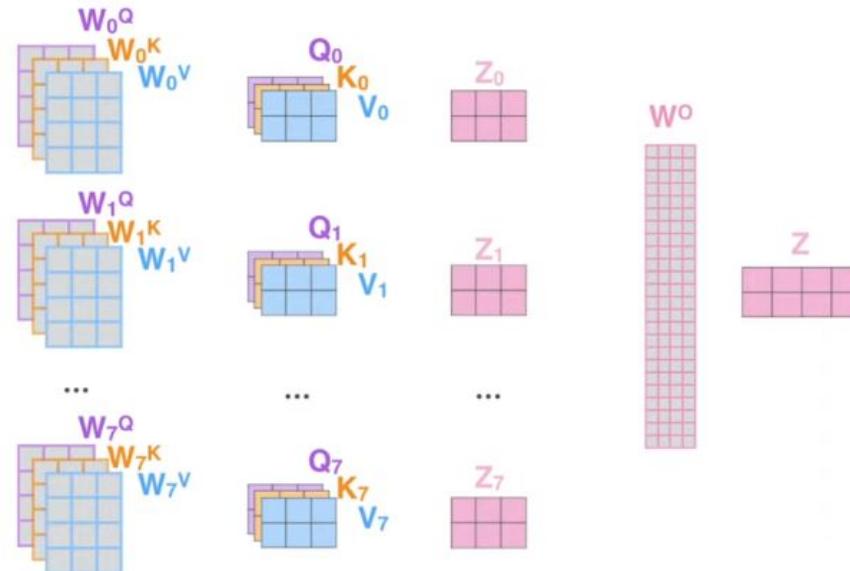


Полная картина



- 1) This is our input sentence* X
- 2) We embed each word* R
- 3) Split into 8 heads. We multiply X or R with weight matrices W_0^Q, W_0^K, W_0^V
- 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

Thinking Machines X



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

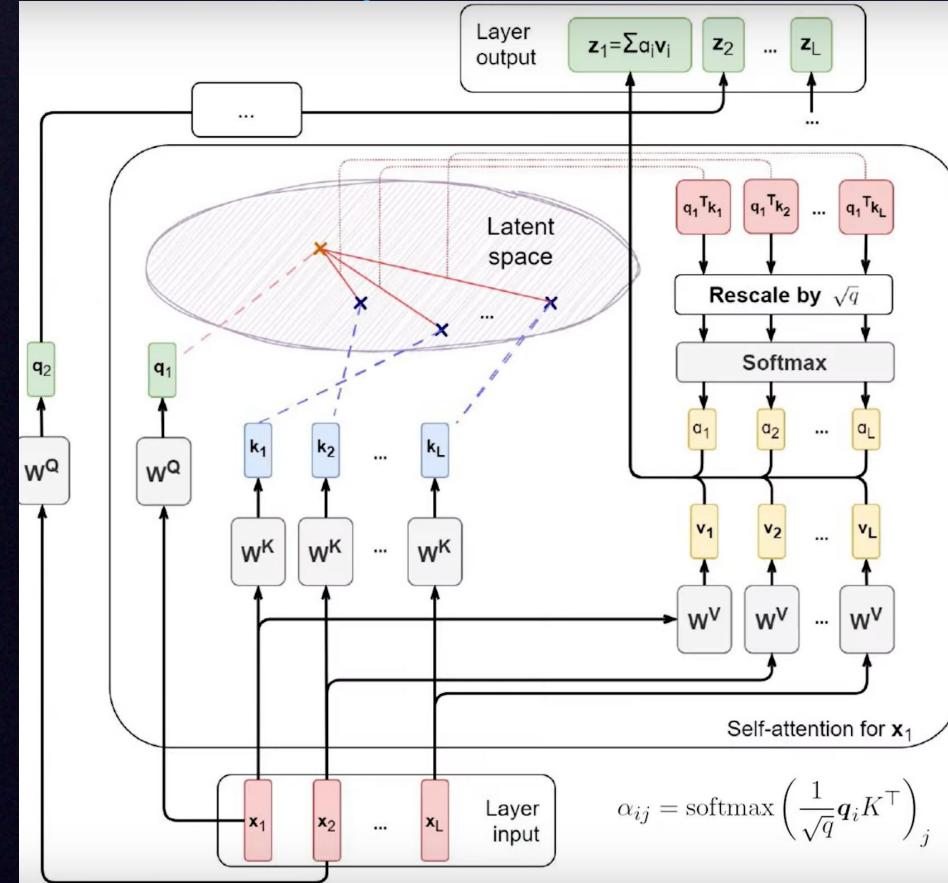
Информационный поиск

Ранжируем разные значения на основе того, насколько близки друг к другу запрос данного x и ключи всех документов

Но всё из одних векторов
 q^*k оценки близости

Затем получаем, на что хочет посмотреть x из V
Каждый x смотрит на каждый другой x сразу, но весов теперь квадратичное число.

Это мешает увеличению длины контекстного окна

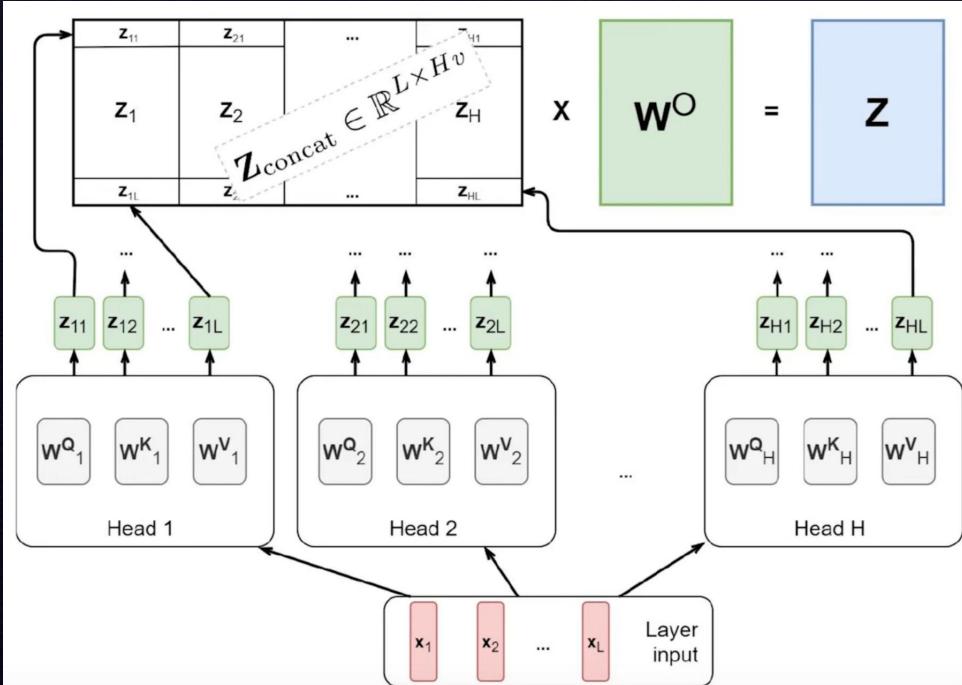


$$\alpha_{ij} = \text{softmax}\left(\frac{1}{\sqrt{q}} q_i K^\top\right)_j$$

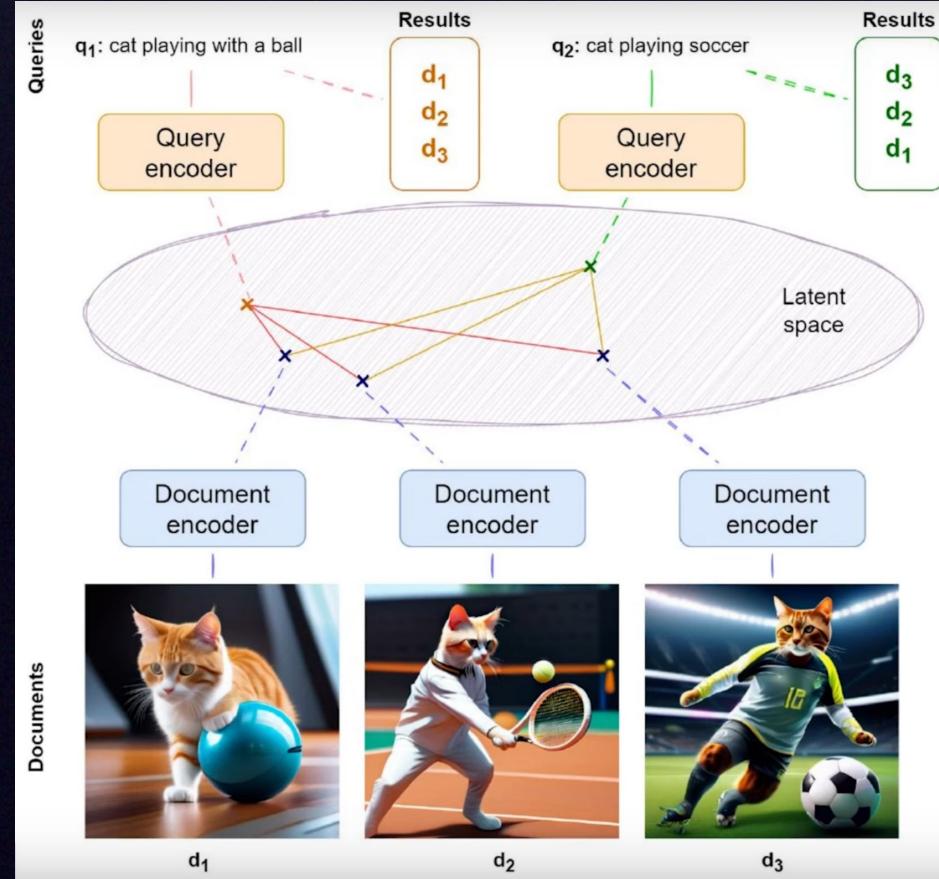
Информационный поиск

Смысл этих сложных преобразований в том, что мы имеем разные представления на других головах

Смотрим “под другим углом”



Аналогия



Визуализация внимания на 2 головы

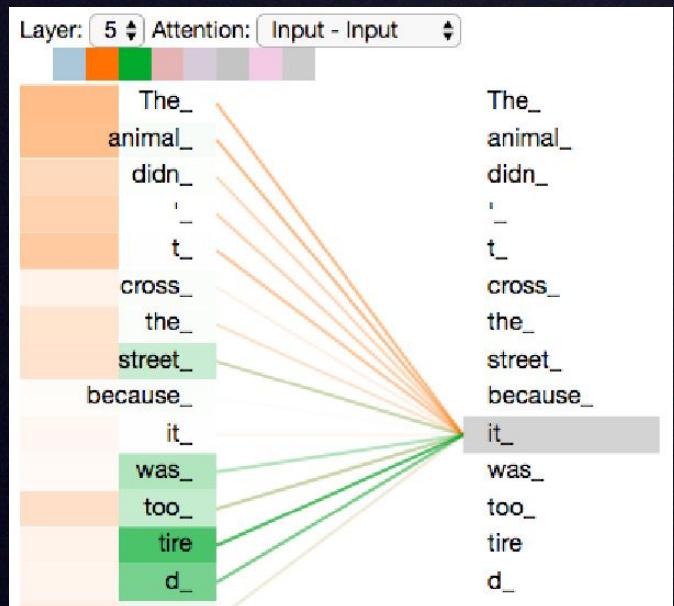


Когда мы кодируем слово “it”

Одна голова фокусируется на “the animal”

Другая на “tired”

В итоге получаем представление, что “it” связано и с объектом, и с его состоянием

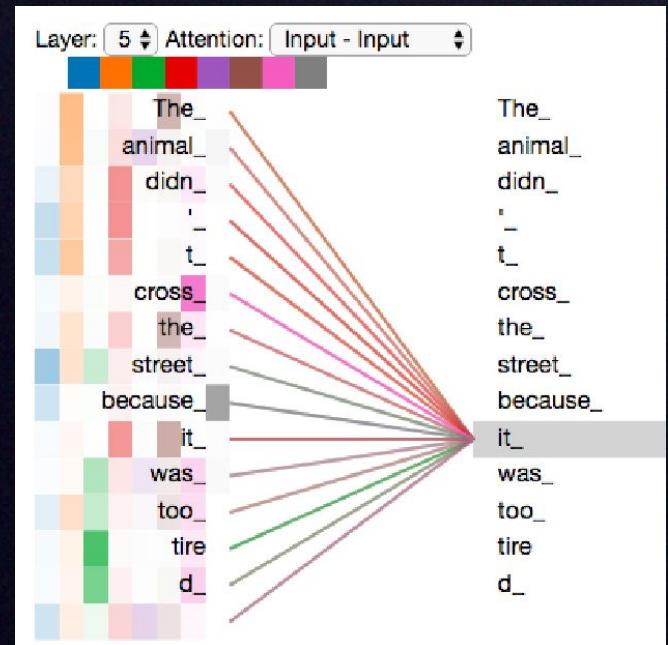


Визуализация внимания на 8 голов



Интерпретировать сложнее

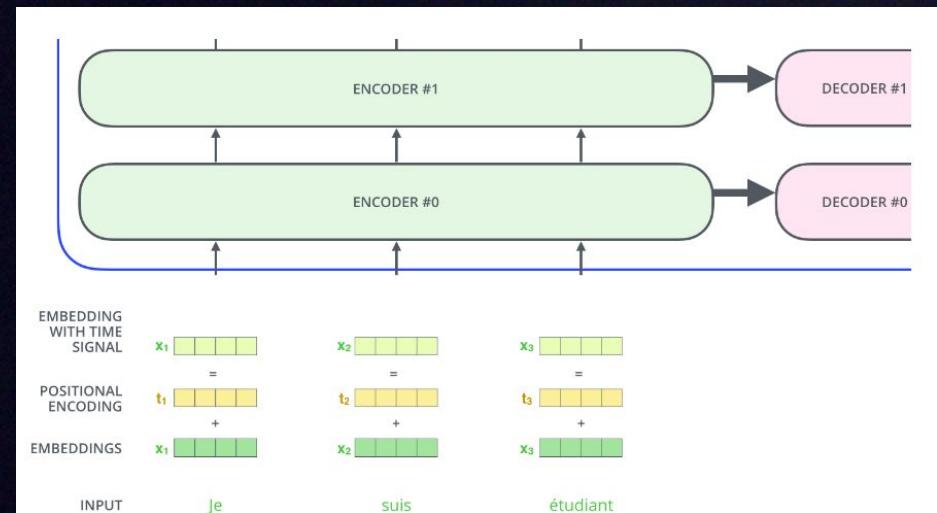
Слова могут быть связаны по грамматике, по эмоциям, по смыслу и т.д.



Позиционное кодирование



Убирая рекуррентную сеть, мы потеряли строгую поддержку
порядка слов

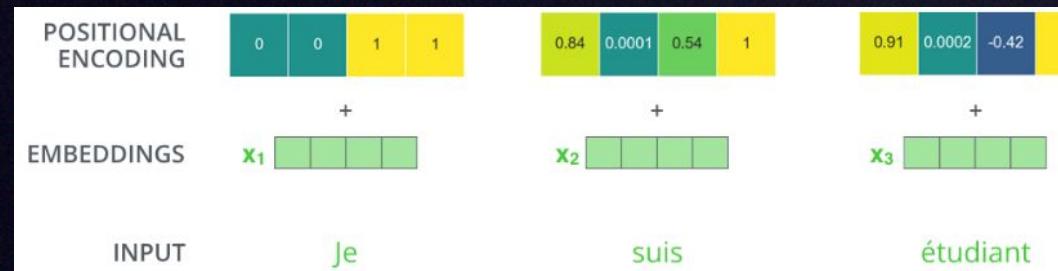


Позиционное кодирование



pos – позиция слова во входной последовательности, i - индекс размерности

Пример на dim=4



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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

Позиционное кодирование



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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

Почему мы так делаем?

Позиционное кодирование



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

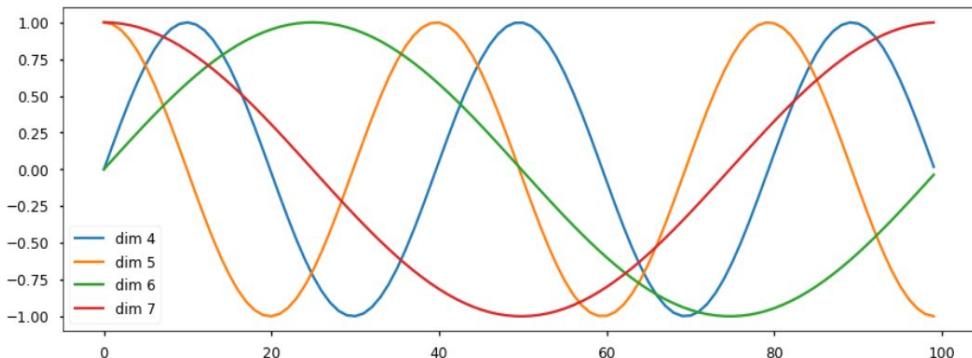
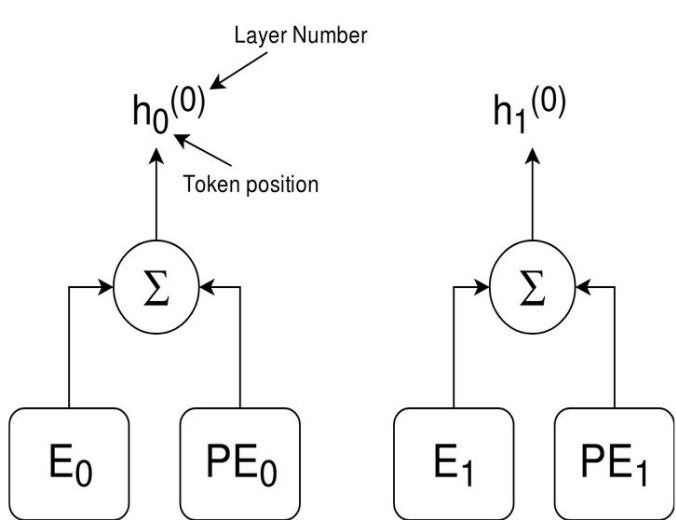
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

Почему мы так делаем?

По каждой размерности идет своя синусоида, идея в том, чтобы для каждого фиксированного k , наш $PE(pos+k)$ был бы линейной функцией от PE_{pos} , облегчая обучение того, как смотреть на относительные смещения.

“Растягиваем” синус для i , чтобы медленнее менять значение

Позиционное кодирование



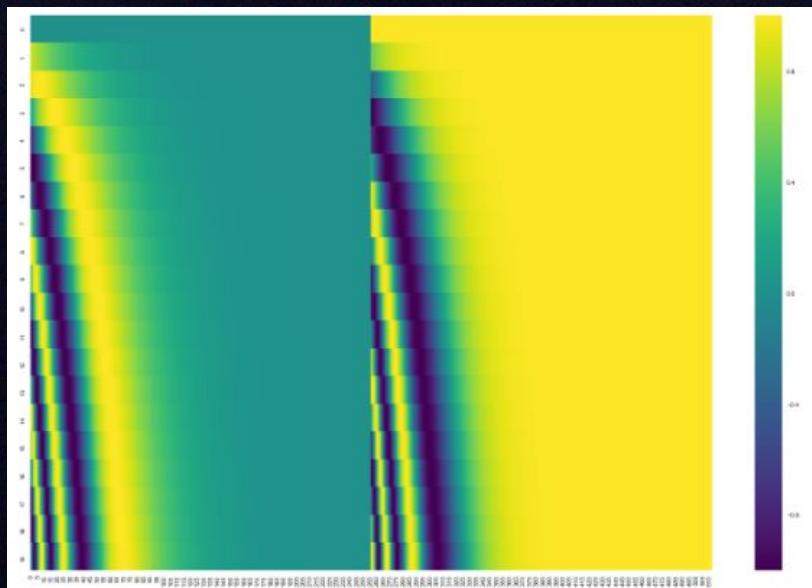
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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

Позиционное кодирование



Пример для 20 слов (строк) с размером эмбеддингов 512



RoPE (Su et al., 2021)



Позиция токена не добавляется к его эмбеддингу , а встраивается через поворот вектора в многомерном пространстве
информация о позиции вводится на уровне механизма внимания , модифицируя взаимодействие между query и key векторами

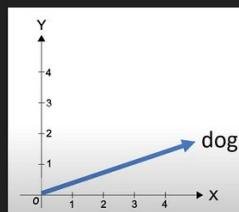


Fig 1.
(2,D) Position encoding
at position 1

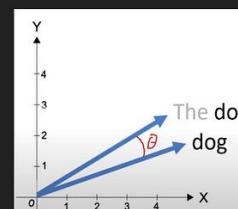


Fig 2.
(2,D) Position encoding
at position 2

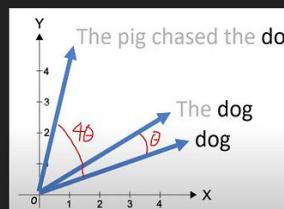


Fig 3.
(2,D) Position encoding
at position 5

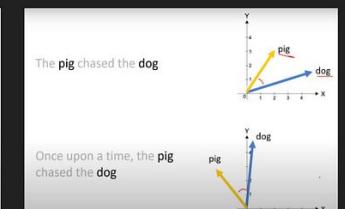


Fig 4.
(2,D) Consistent difference in
positional encoding of 2 words that
are two words apart, regardless of
sentence length

RoPE (Su et al., 2021)



Внутреннее скалярное произведение двух уже-повёрнутых векторов зависит только от разности углов ($\vartheta_i - \vartheta_\square$), т.е. от того, насколько далеко токены друг от друга.

Поэтому само внимание «видит» именно относительный сдвиг, даже если мы никогда явно его не сообщали.

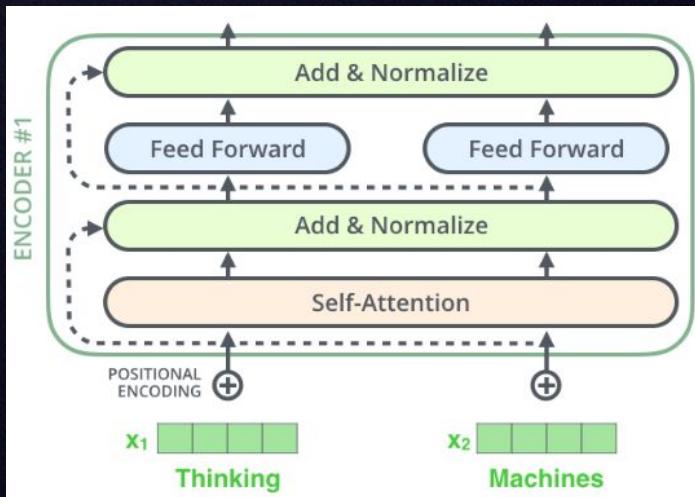
Теперь, увидев «мама моет раму», модель чувствует, что «мама» перед «моет», потому что её «стрелка-компас» повернулась чуть меньше.

Остаточная связь и нормализация



Каждый слой (self-attention, ffn) в каждом энкодере имеет residual connection и нормализацию

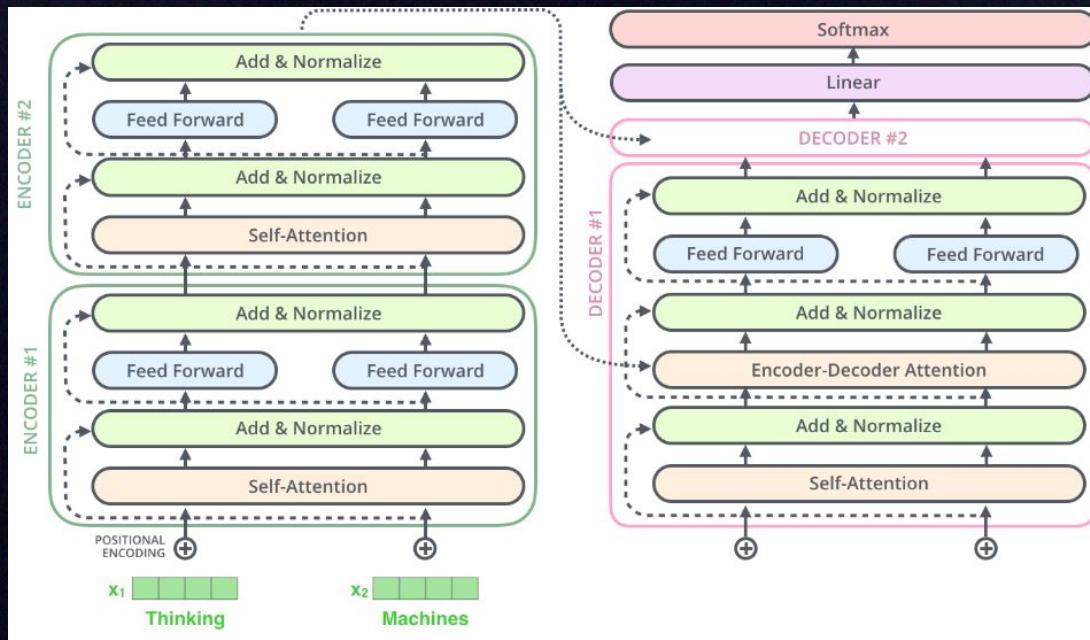
$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



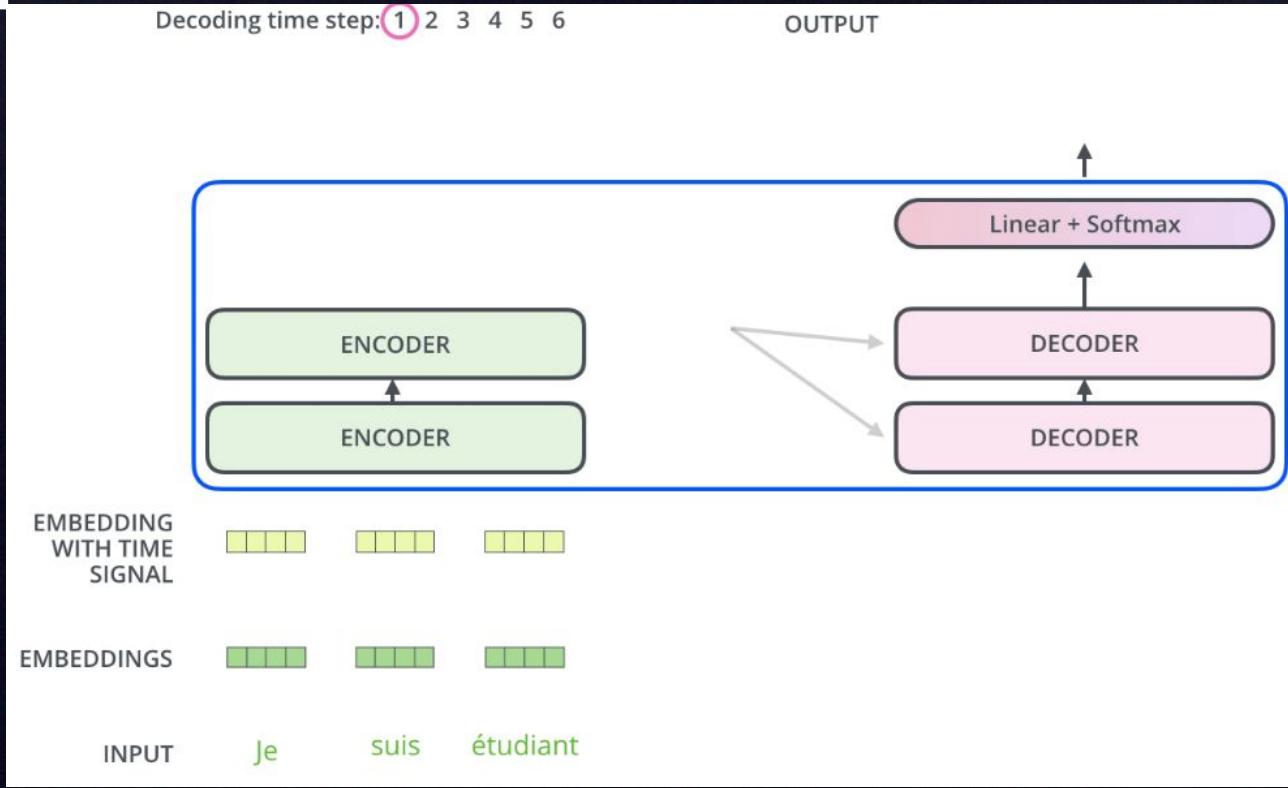
Остаточная связь и нормализация



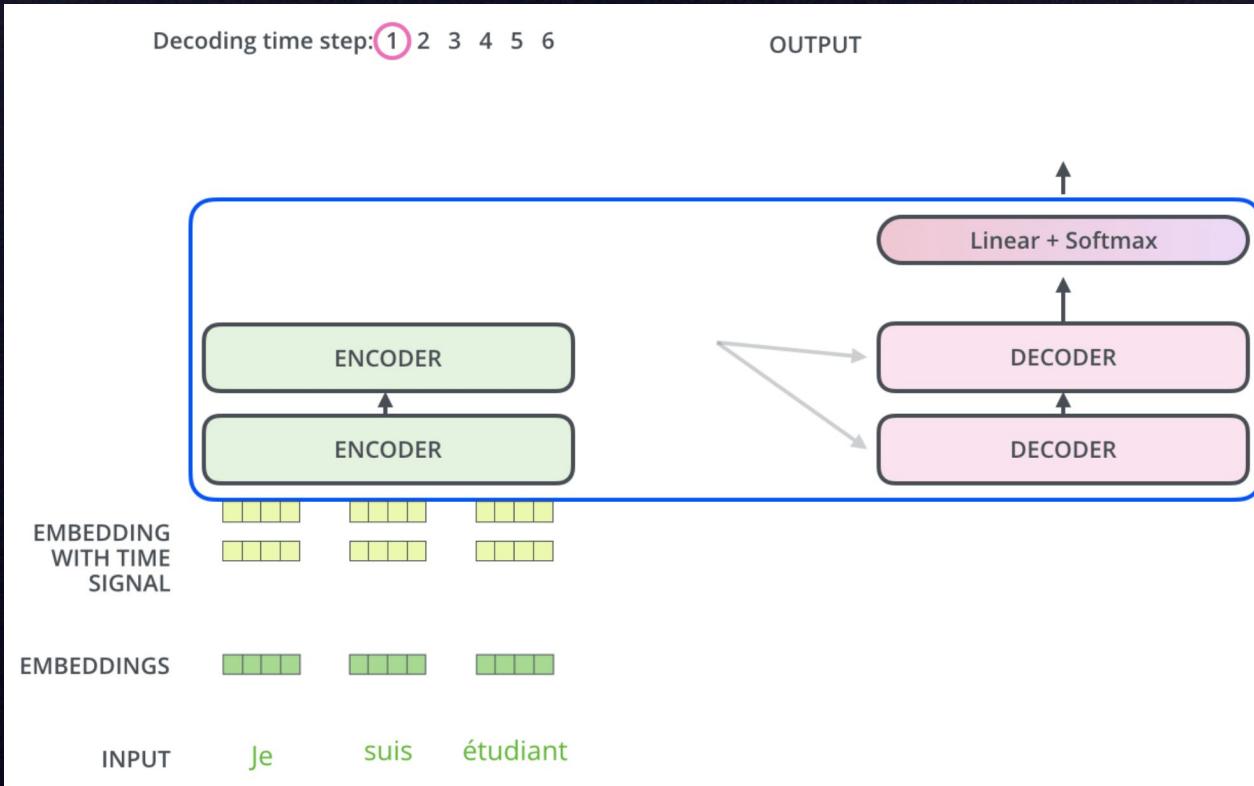
В декодере ситуация аналогичная



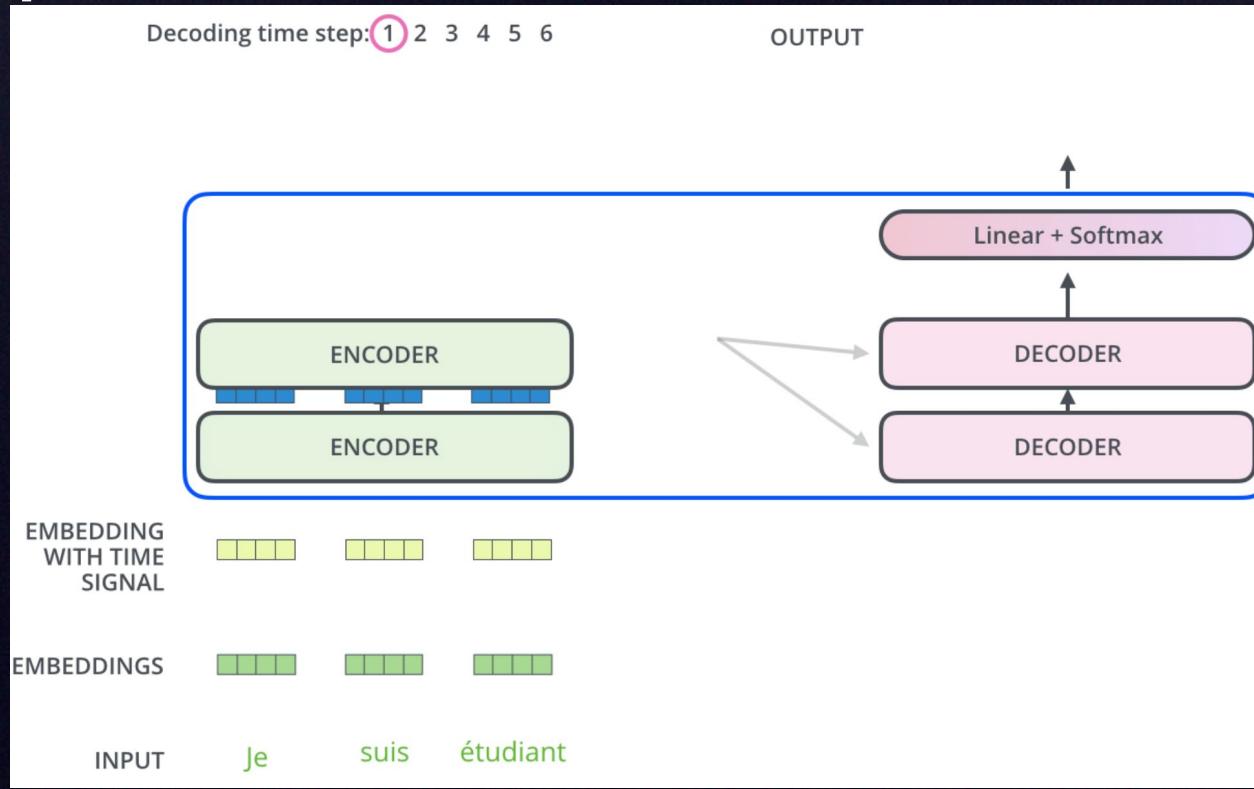
Декодирование



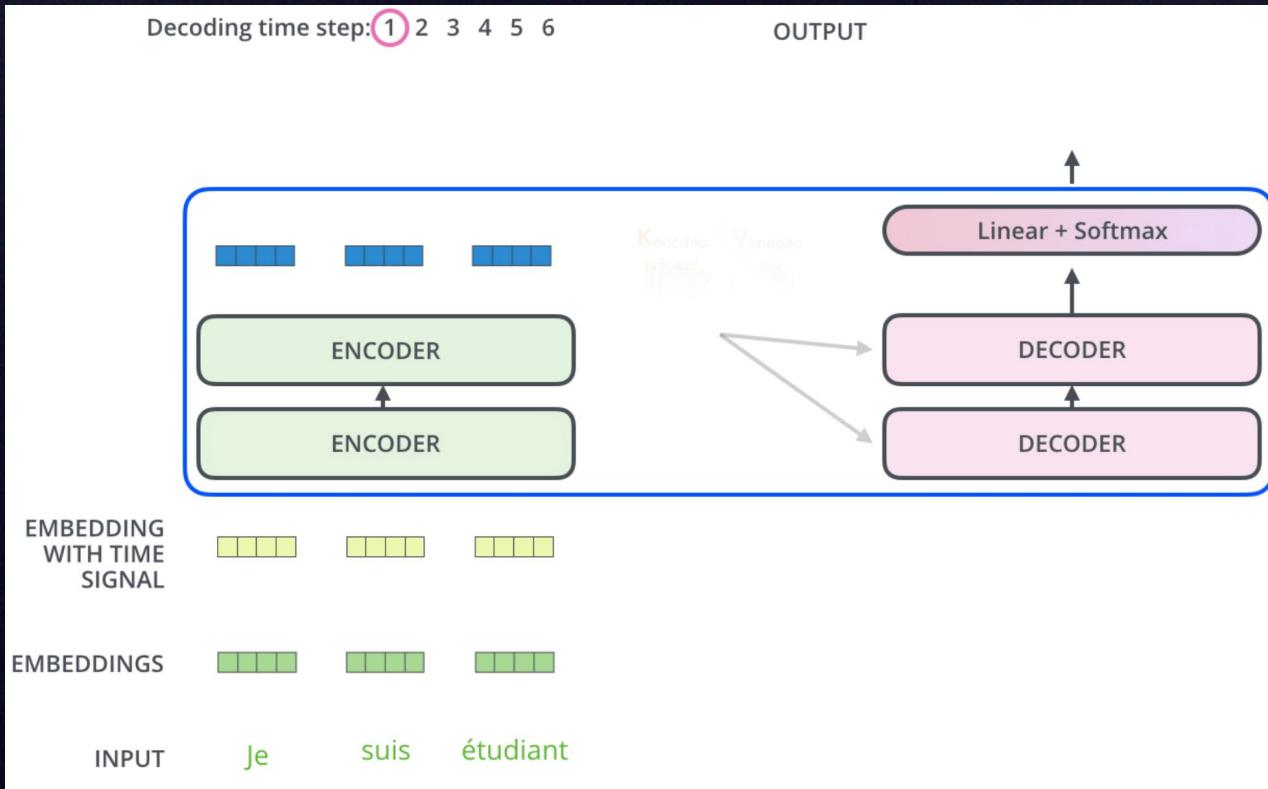
Декодирование



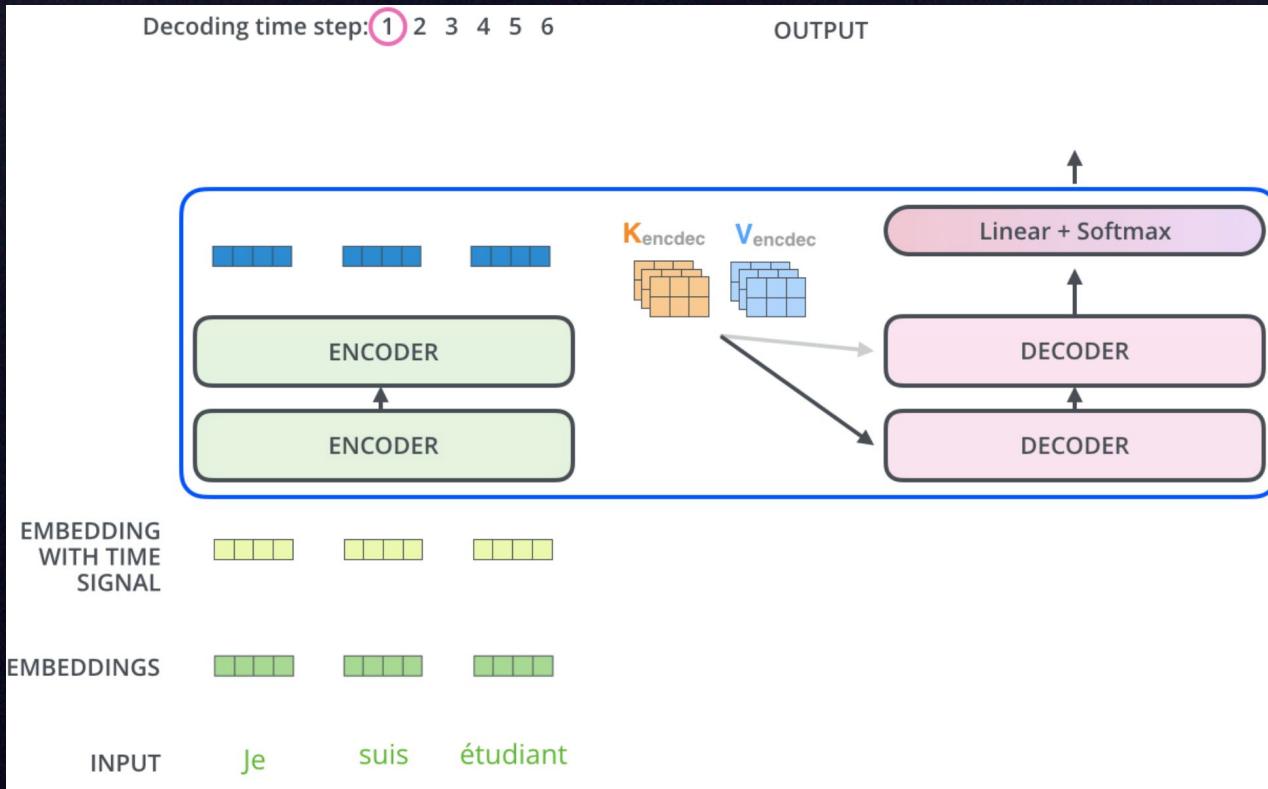
Декодирование



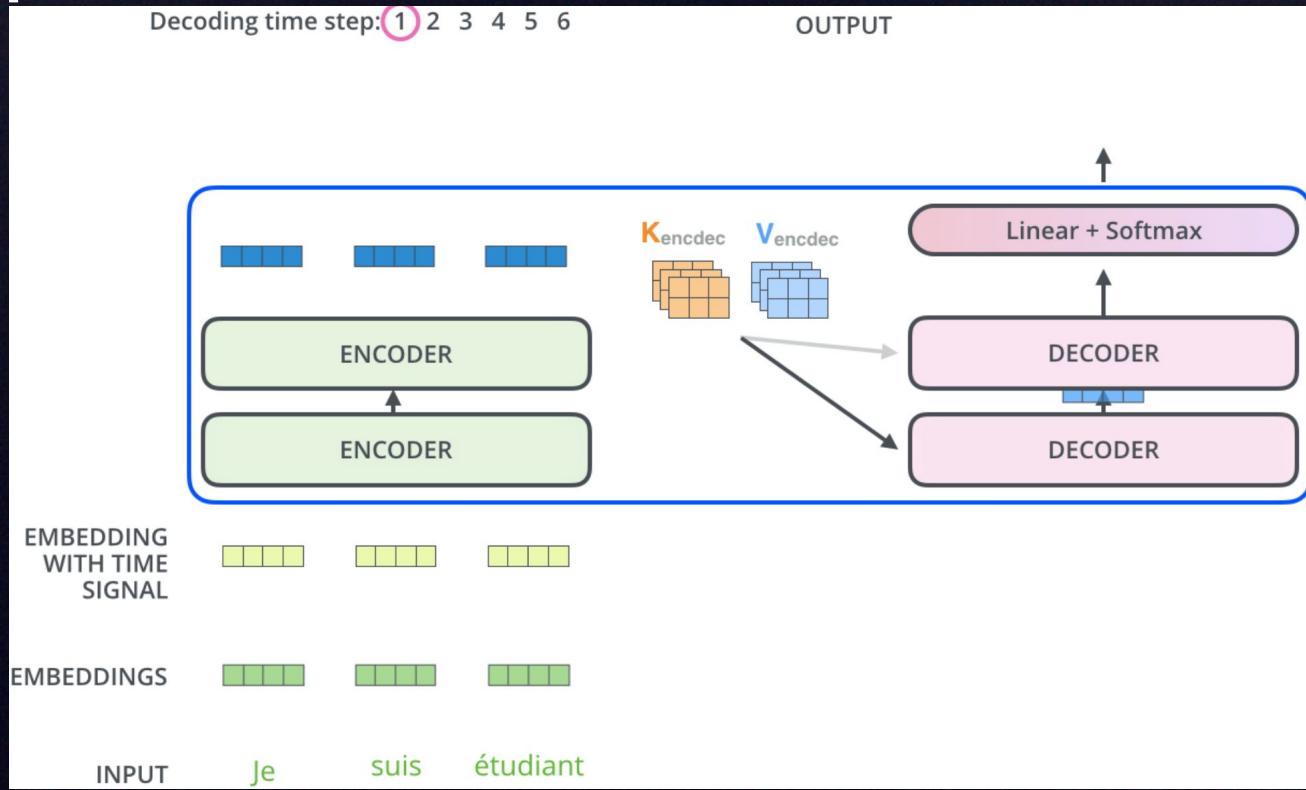
Декодирование



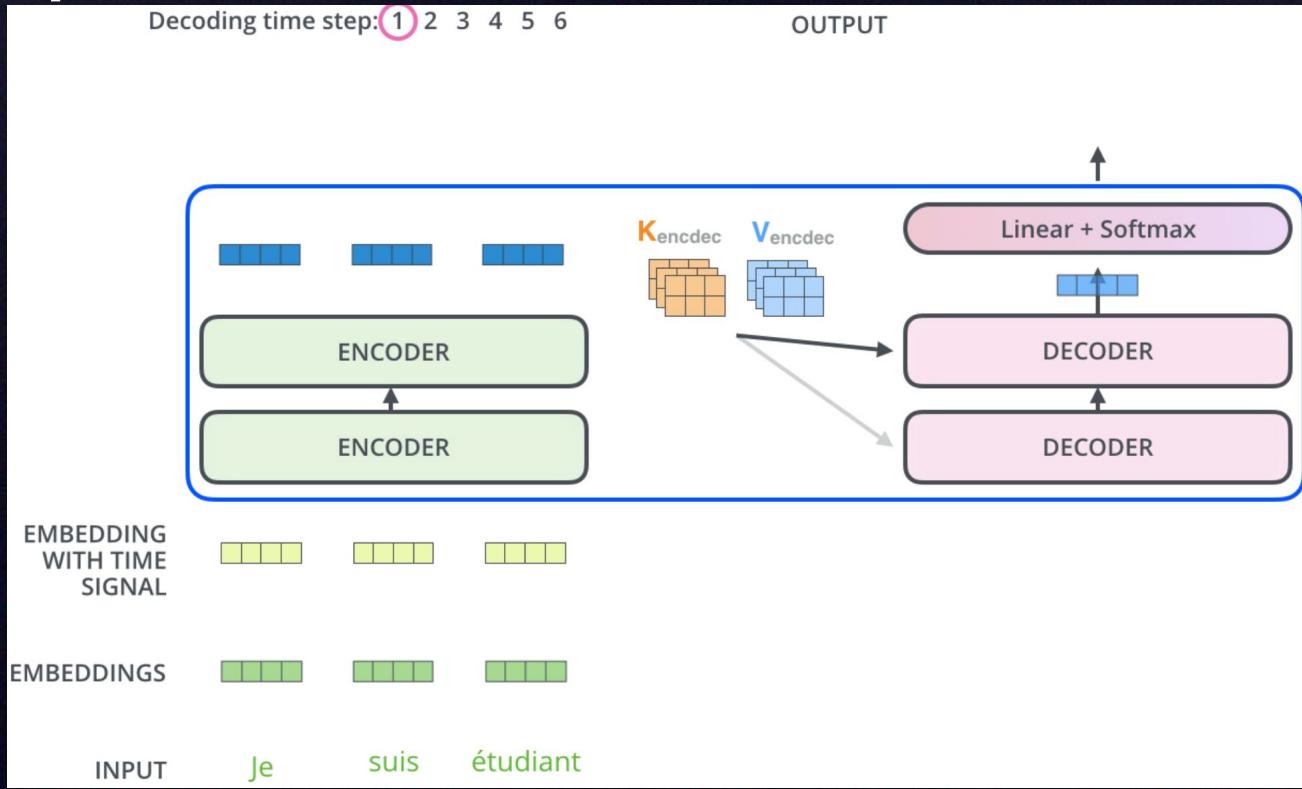
Декодирование



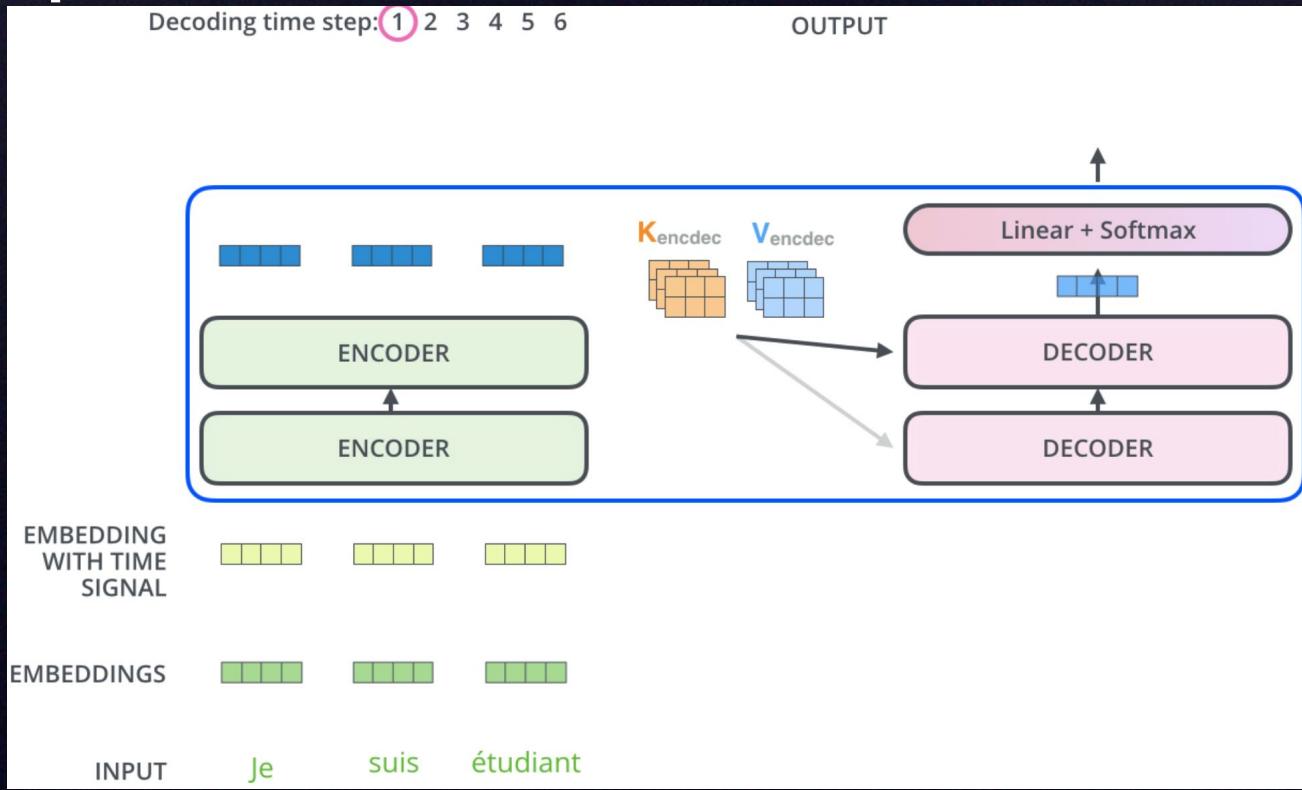
Декодирование



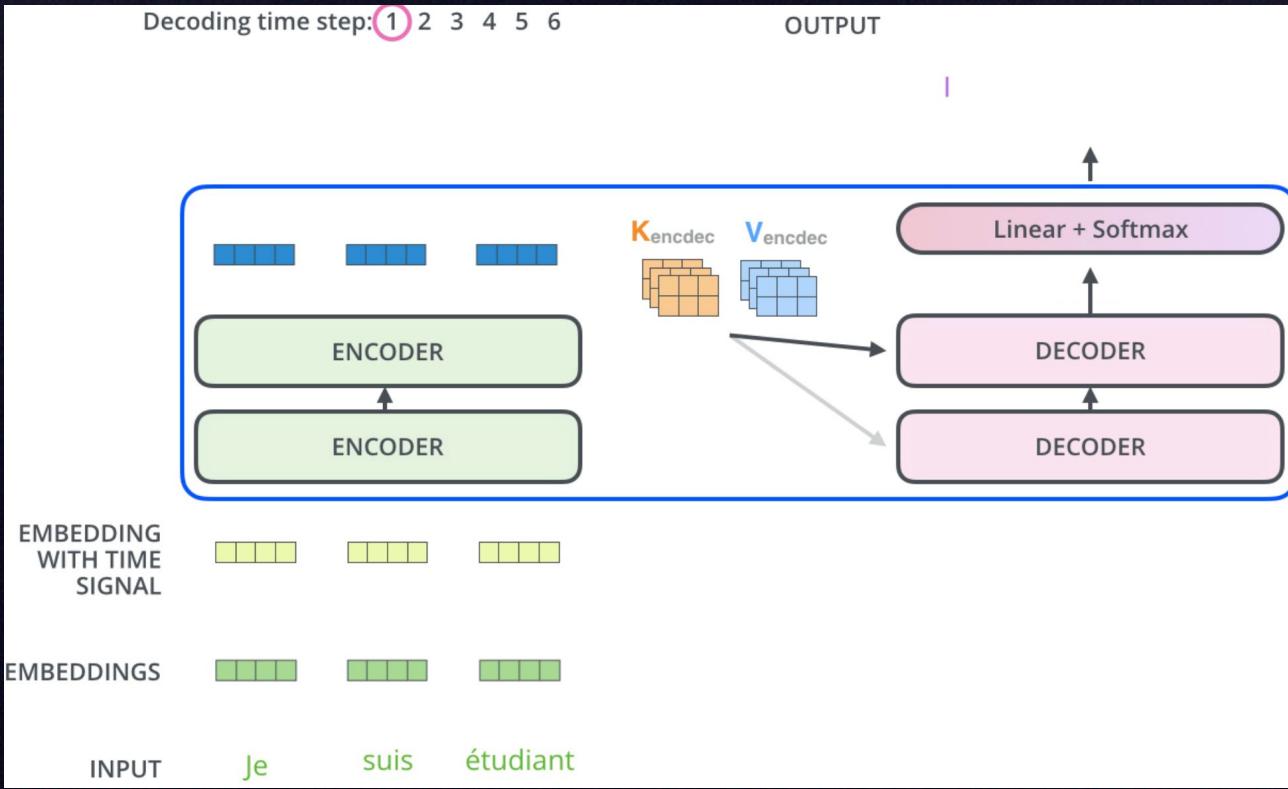
Декодирование



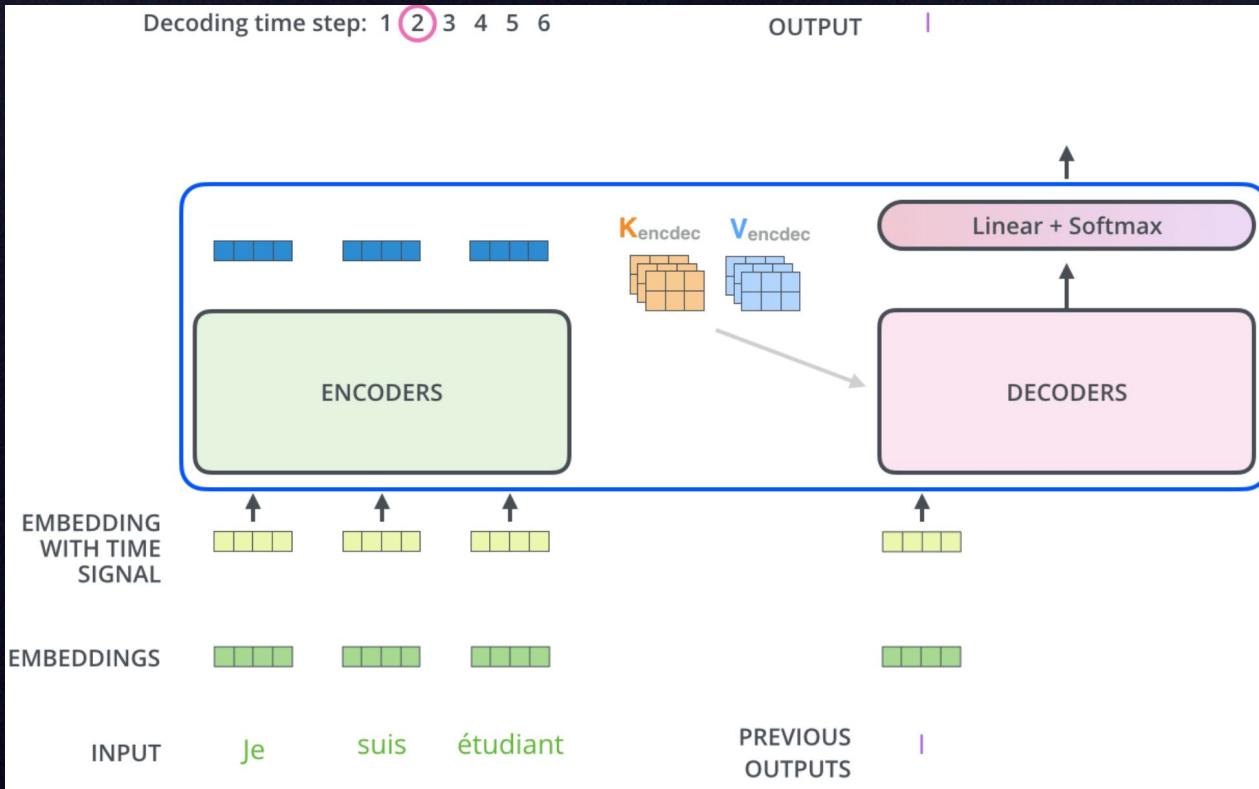
Декодирование



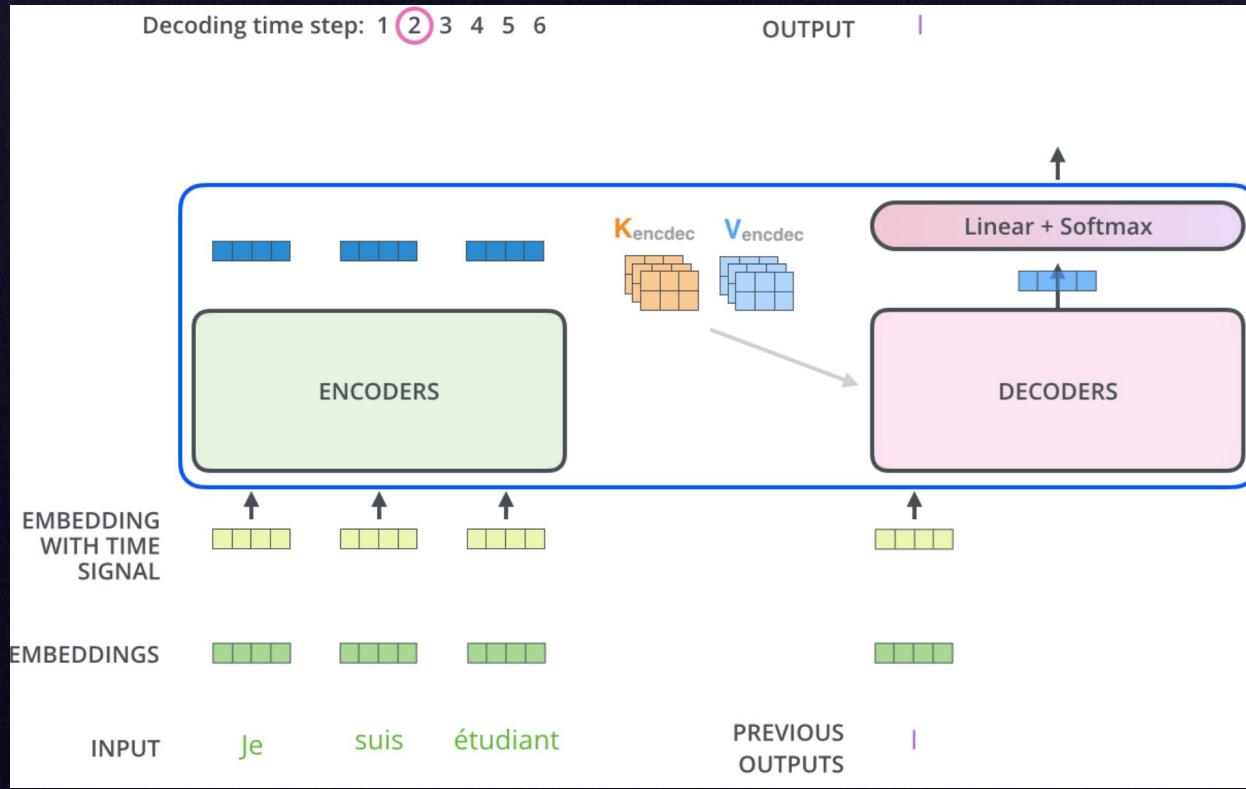
Декодирование



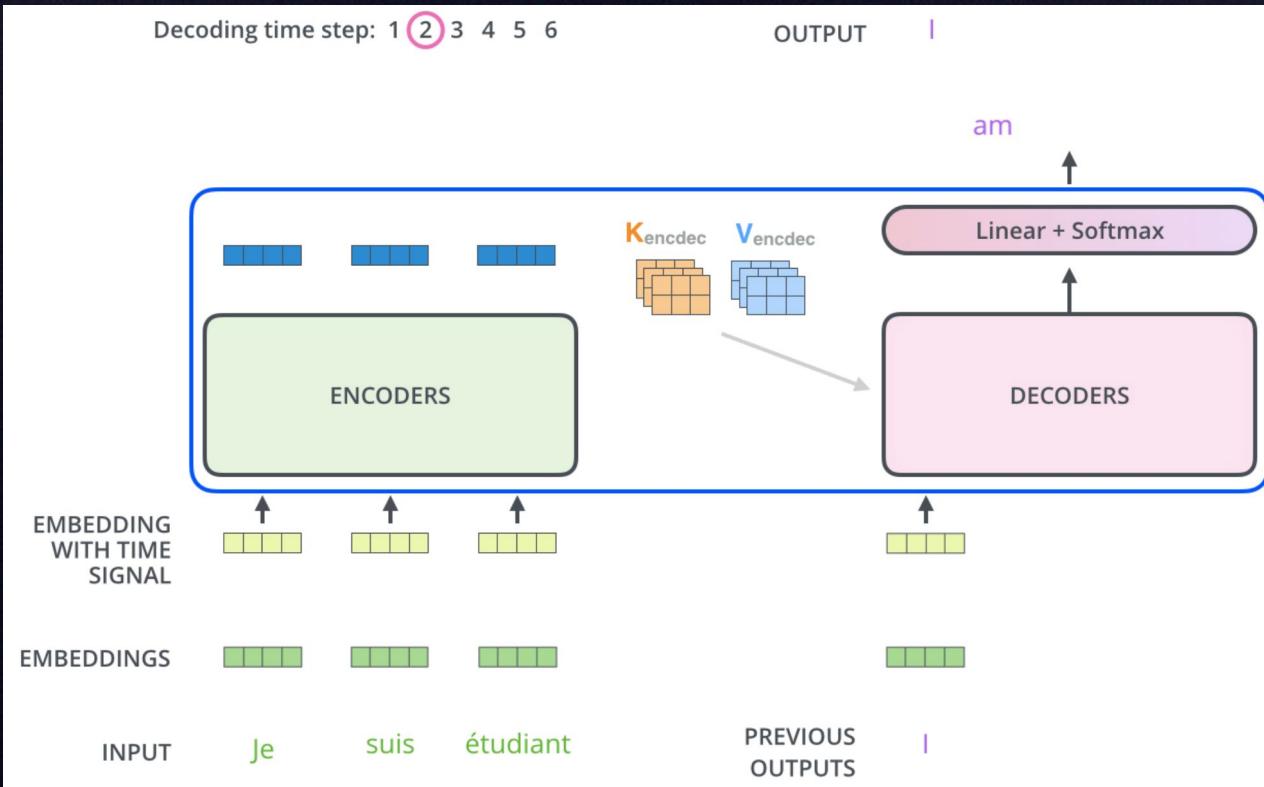
Декодирование



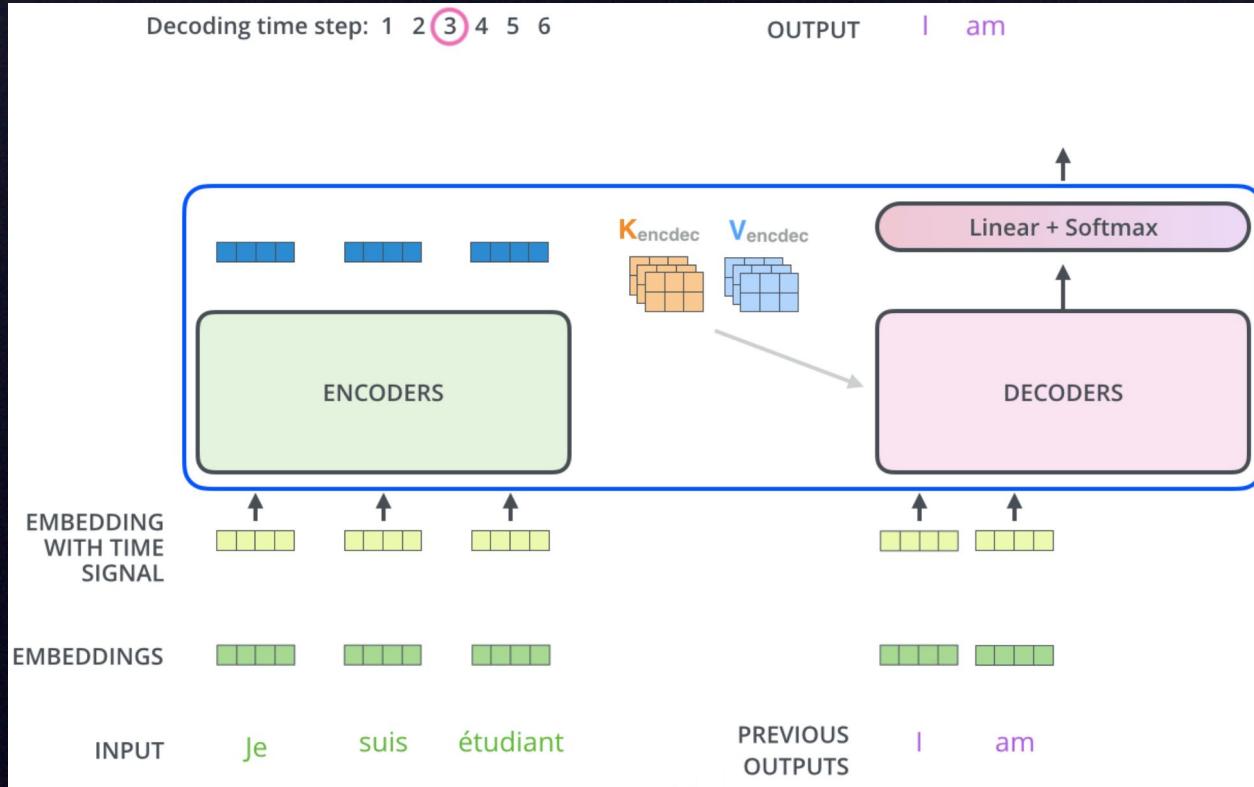
Декодирование



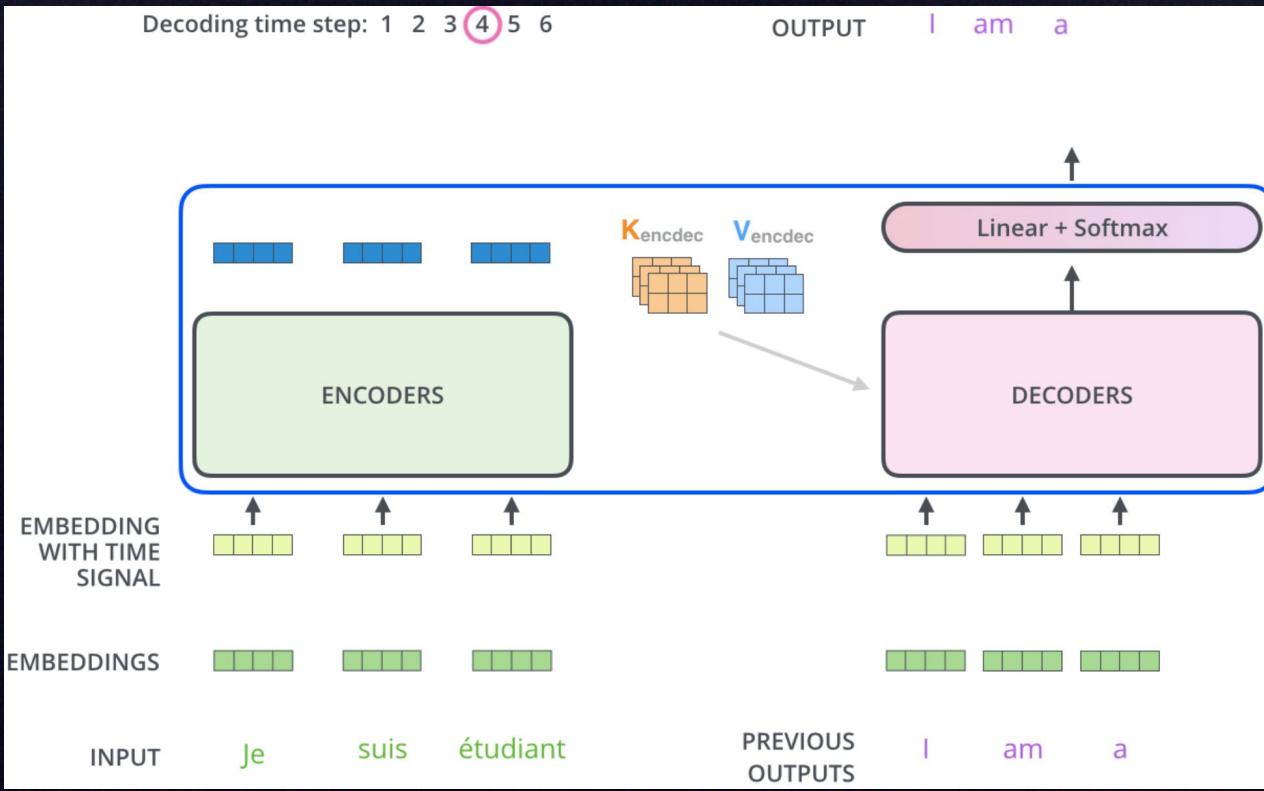
Декодирование



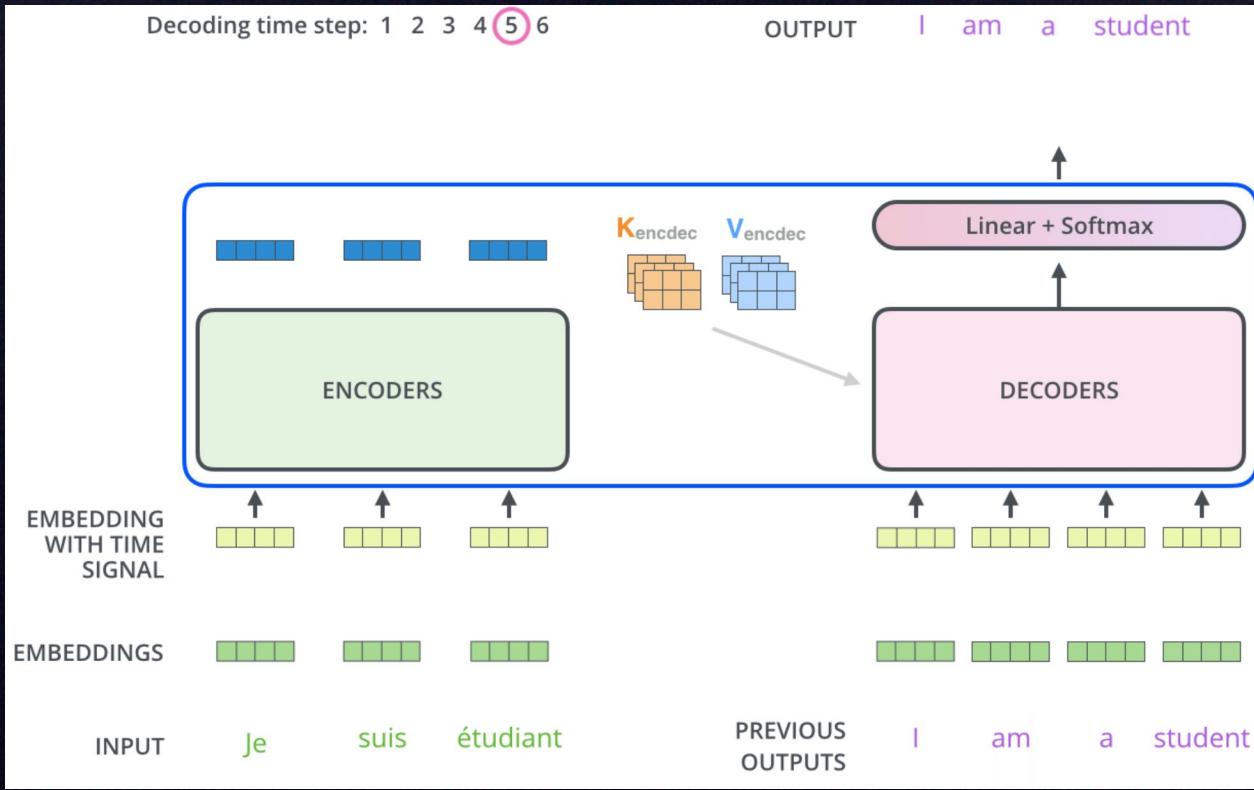
Декодирование



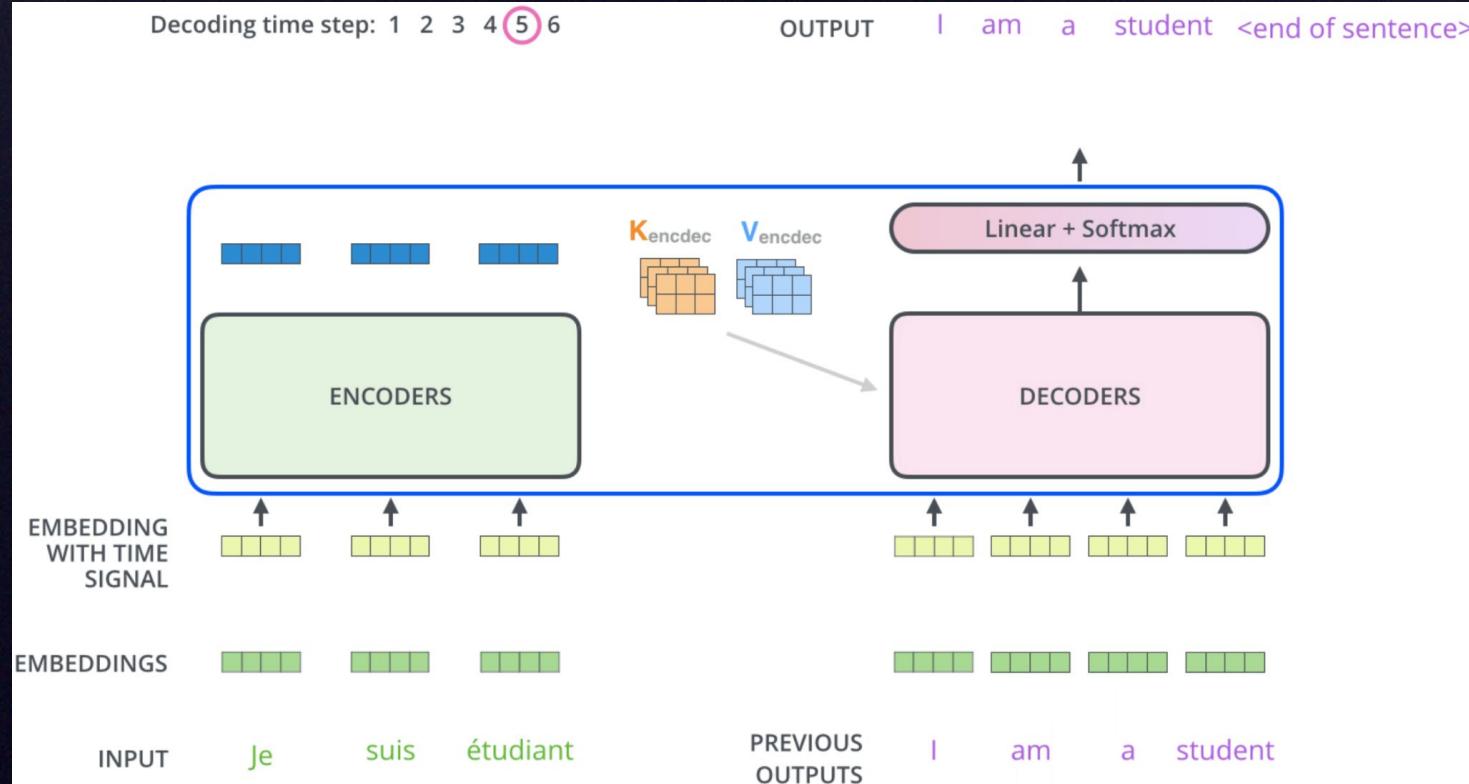
Декодирование



Декодирование



Декодирование

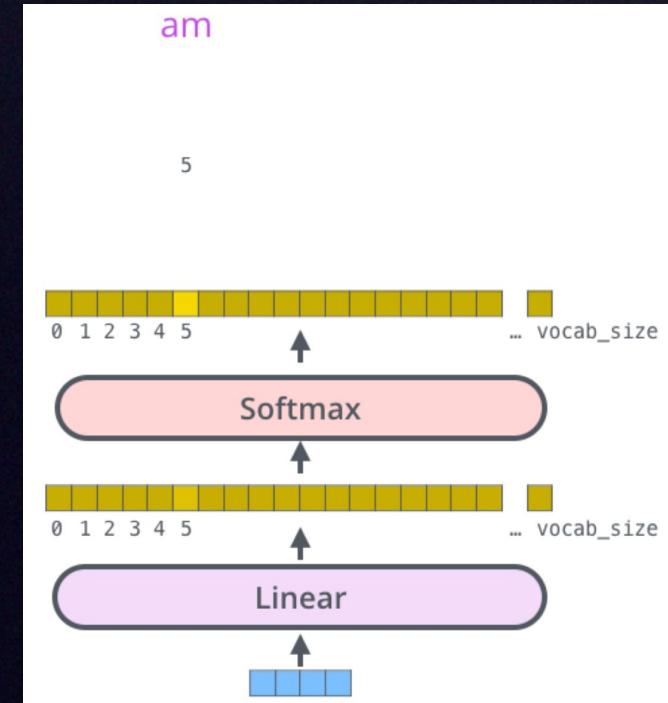


Последний слой

- Берем argmax индекс
- Сопоставляем индекс со словарем

Logprobs

Logits



Machine Translation: WMT-2014 BLEU



- >3x по скорости обучения

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	28.4	41.8

Machine Translation: WMT-2014 BLEU



- >3x по скорости обучения

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		$3.3 \cdot 10^{18}$
Transformer (big)	28.4	41.0		$2.3 \cdot 10^{19}$

Self-Attention



- Вычислительный бонус
- Сокращенные пути между словами
- Интерпретируемость

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)



- Идеи:
 - Masked language model (MLM): предсказывает слово под маской на основе контекста
 - Предсказание следующего слова
- BERT:
 - Fine-tune для SOTA на любой задаче
 - Побеждает SOTA на 11 NLP задачах

BERT



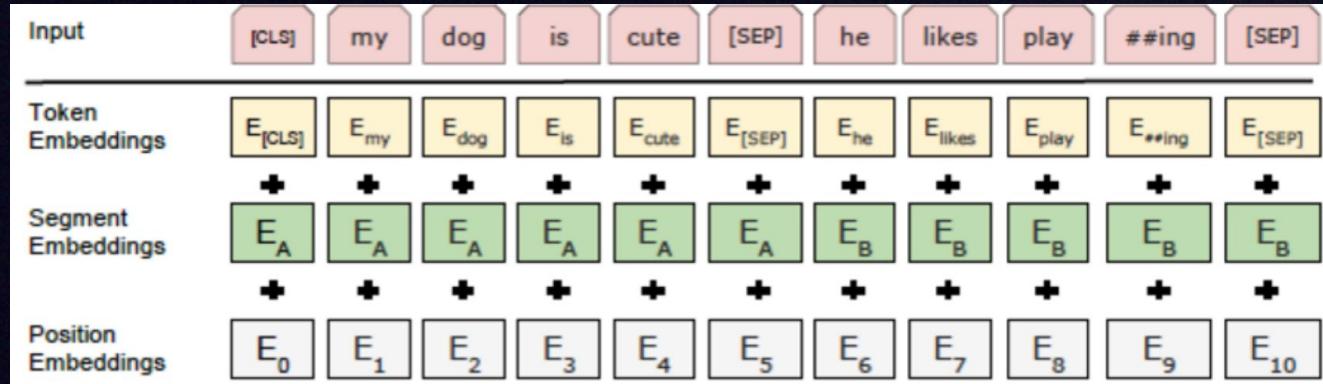
- BERT_{BASE}: L=12, H=768, A=12, Total Parameters=110M
 - (L : number of layers (Transformer blocks), H is the hidden size, A : the number of self-attention heads)
- BERT_{LARGE}: L=24, H=1024, A=16, Total Parameters=340M
- Фактически тот же трансформер, но теперь двунаправленный, при условии левого и правого контекста во всех слоях



BERT: входные представления



- На вход предобученные WordPiece эмбеддинги



Byte Pair Encoding, BPE (Sennrich et al., 2016)



- Слово = последовательность токенов (пока символов, изначально использовались unicode-символы)
- Словарь = все токены (на нулевой итерации — символы)
- Повторять пока не достигли ограничения на размер словаря
 - Назначаем новым токеном объединение двух существующих токенов, которое встречается чаще других пар в корпусе (имеется в виду: встречаются вместе).

«I_{</w>} like_{</w>} ke_{</w>}» → «I_{</w>}», «li», «##ke_{</w>}», «ke_{</w>}»

AABABCABBAABAC	ADDCDBADAC	EDCDBEAC
AA - 2		
AB - 4 AB = D	AD - 2 AD = E	
BA - 3	DD - 1	
BC - 1	DC - 1	
CA - 1	CD - 1	
BB - 1	DB - 1	
AC - 1	DA - 1	
	AC - 1	

Byte Pair Encoding, BPE (Sennrich et al., 2016)



- Автоматически собирает словарь по корпусу
- Не зависит от языка
- Unsupervised
- Баланс по длине
- Трюк: “е” и “е</w>” – разные символы

```
«I</w> like</w> ke</w>» → «I</w>», «li», «##ke</w>», «ke</w>»
```

BPE, “квазиморфы”



Пример токенизации:

Начинаем с исходной строки:

aaabdaaabac

Выделяем самые частотные биграммы:

ZabdZabac Z=aa

Затем снова находим самые частотные биграммы:

ZYdZYac Y=ab Z=aa

И снова, и снова:

XdXac X=ZY Y=ab Z=aa

*B
##есной
1890
года
В
##ар
##ша
##вский
и
Томск
##ий
университет
##ы
из
##бира
##ют
его
профессором*

WordPiece (Schuster et al., 2012) and SentencePiece (Kudo et al., 2012)



- Предложил Google
- Максимизируется правдоподобие вместо частоты для слияния токенов (language model log likelihood)
- SentencePiece применяется на сырых текстах без токенизации, пробел там рассматривается как специальный символ

WordPiece (Schuster et al., 2012) and SentencePiece (Kudo et al., 2012)



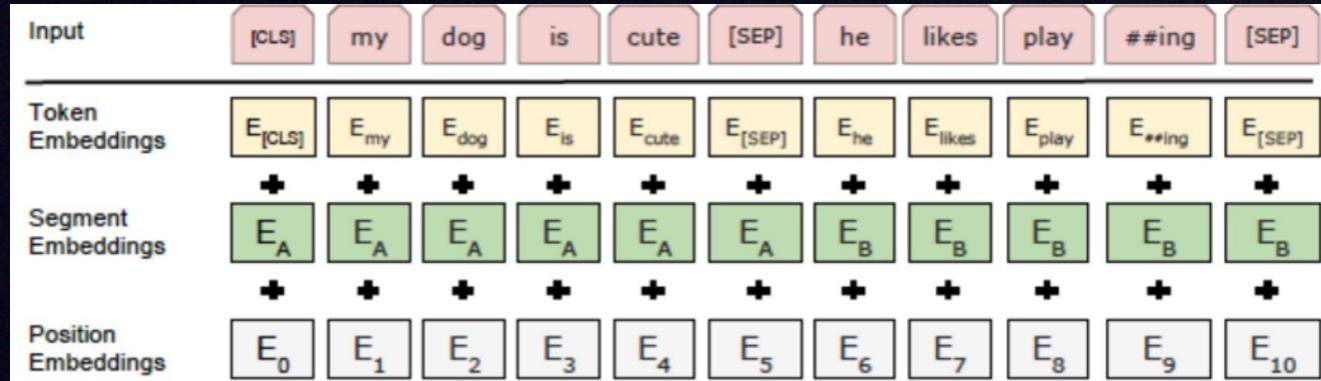
- Предложил Google
- Максимизируется правдоподобие вместо частоты для слияния токенов (language model log likelihood)
- SentencePiece применяется на сырых текстах без токенизации, пробел там рассматривается как специальный символ

```
# загружаем библиотеку
import sentencepiece as spm
# обучение из файла text.txt
spm.SentencePieceTrainer.Train('--input=test/text.txt --'
model_prefix=m --vocab_size=1000')
# загружаем обученную модель
sp = spm.SentencePieceProcessor()
sp.Load("m.model")
# теперь кодируем текст
sp.EncodeAsIds("I like ke")
```

BERT: входные представления



- На вход предобученные WordPiece эмбеддинги
- Обучаемые позиционные эмбеддинги (Gehring et al., 2017)
- Эмбеддинг предложения добавляется каждому токену в предложении
- Токен [CLS]
- Токен [SEP]



Задача 1: Masked Language Modeling



- 15% слов маскируется случайно
 - Задача: предсказать слово по левому и правому контексту
- Маскируются особым образом:
 - 80% отобранных заменяются на <MASK> токен *my cat is <mask>*
 - 10% случайным токеном *my cat is banana*
 - 10% не изменяются *my cat is black*

Задача 1: Masked Language Modeling



Use the output of the masked word's position to predict the masked word

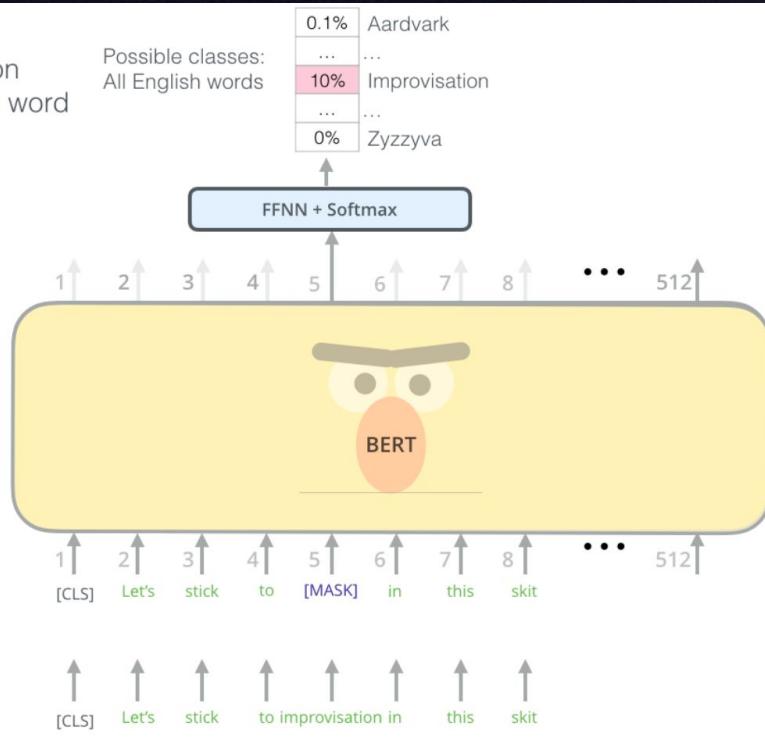
Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zzyzyva

FFNN + Softmax

Randomly mask
15% of tokens

Input



Задача 2: Next Sentence Prediction



- Мотивация:
 - Много задач основаны на понимании отношений между предложениями
 - Question Answering, Natural Language Inference
 - Языковое моделирование явно не учитывает это отношение
- Бинарная классификация, является ли предложение следующим

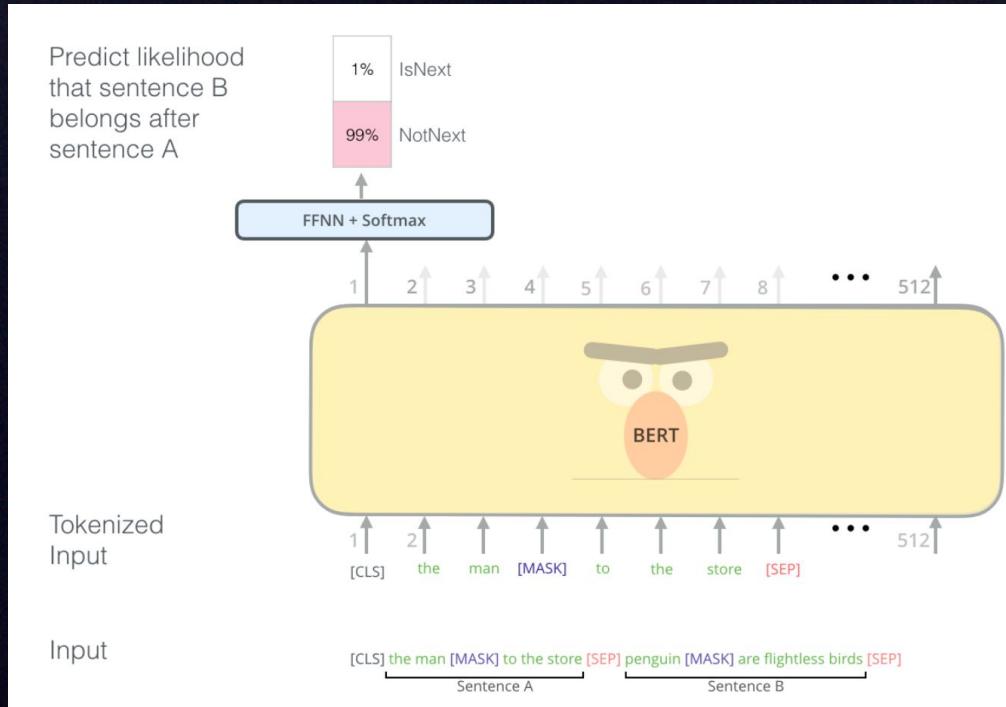
Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon
[MASK] milk [SEP]

Label = isNext

Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are
flight ##less birds [SEP]

Label = NotNext

Задача 2: Next Sentence Prediction



Generative Pretrained Transformers (Radford et al., 2018)



$$L_1(D) = \sum_i \log p(u_i | u_{i-k}, \dots, u_{i-1}; \theta);$$

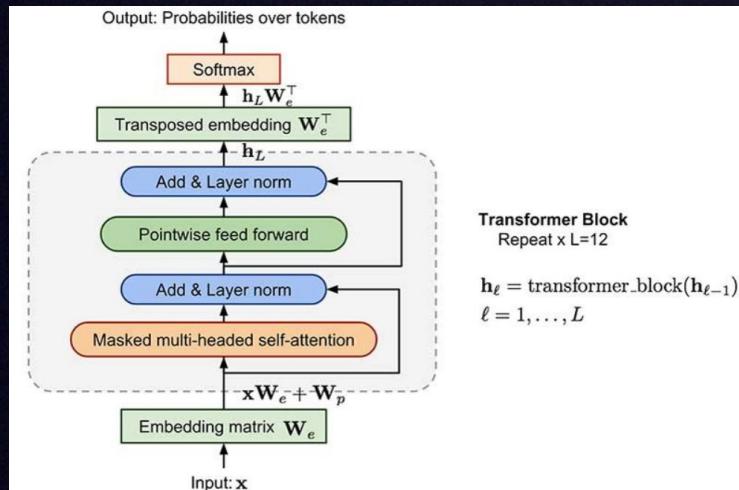
- Сначала обучаем языковую модель
- Добавляем линейный слой для каждой задачи и делаем fine-tune уже с учителем

$$L(C, D) = \sum_{(\mathbf{x}, y)} \log p(y | \mathbf{x}) + \lambda L_1(D).$$

Generative Pretrained Transformers



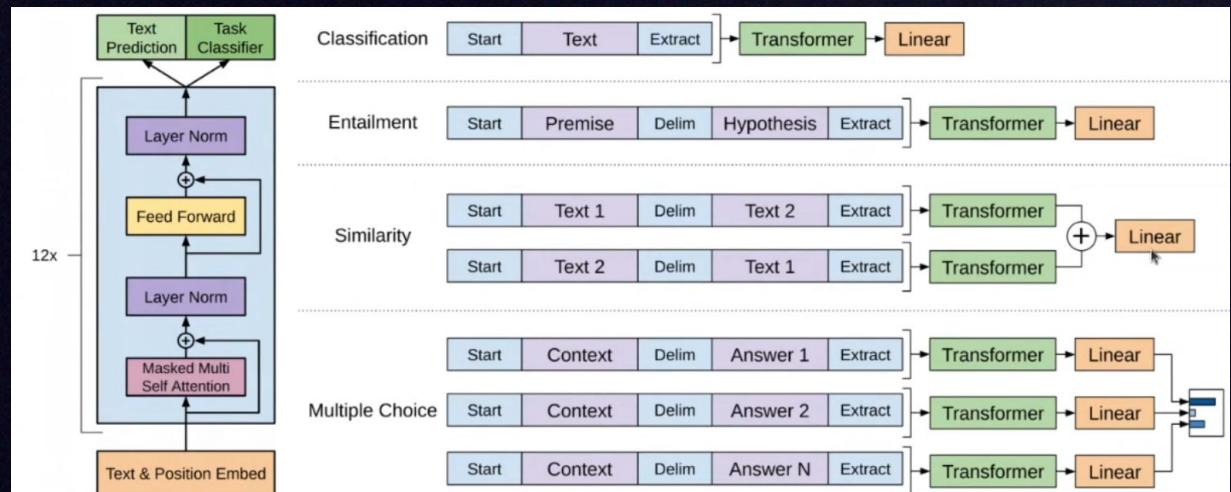
- Переиспользуем одну модель на множестве разных задач
- Хорошие результаты сразу для нескольких задач
- Огромный unsupervised корпус



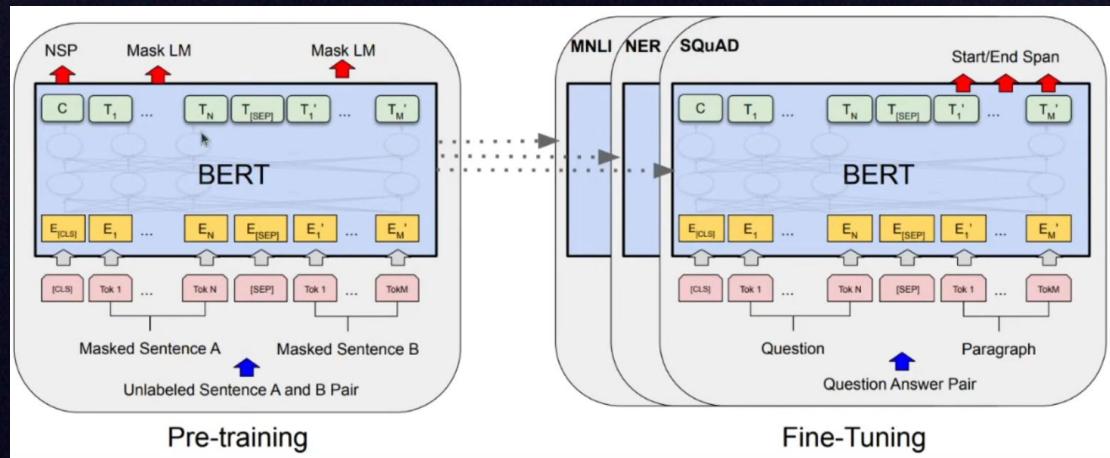
Generative Pretrained Transformers



- Переиспользуем одну модель на множестве разных задач
- Хорошие результаты сразу для нескольких задач
- Огромный unsupervised корпус



BERT fine-tuning

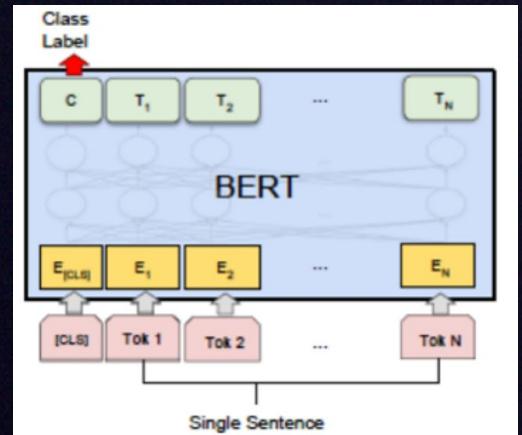


BERT fine-tuning



- Для классификации
 - Берем последний hidden state входного предложения (на позиции [CLS] токена)
 - Добавляем классификационный слой и softmax

$$P = \text{softmax}(CW^T)$$

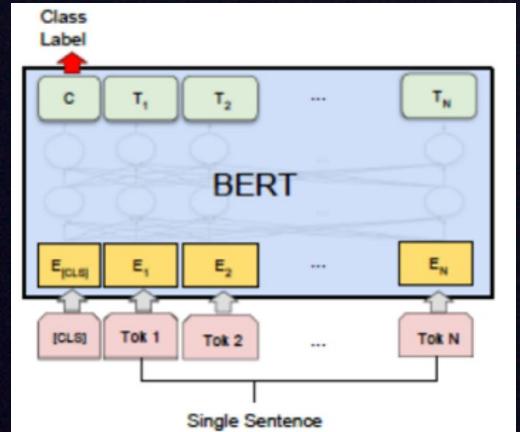


BERT fine-tuning



- Для классификации
 - Берем последний hidden state входного предложения (на позиции [CLS] токена)
 - Добавляем классификационный слой и softmax
 - Обучаем совместно
 - Меняем batch size, lr, # epochs

$$P = \text{softmax}(CW^T)$$



BERT fine-tuning



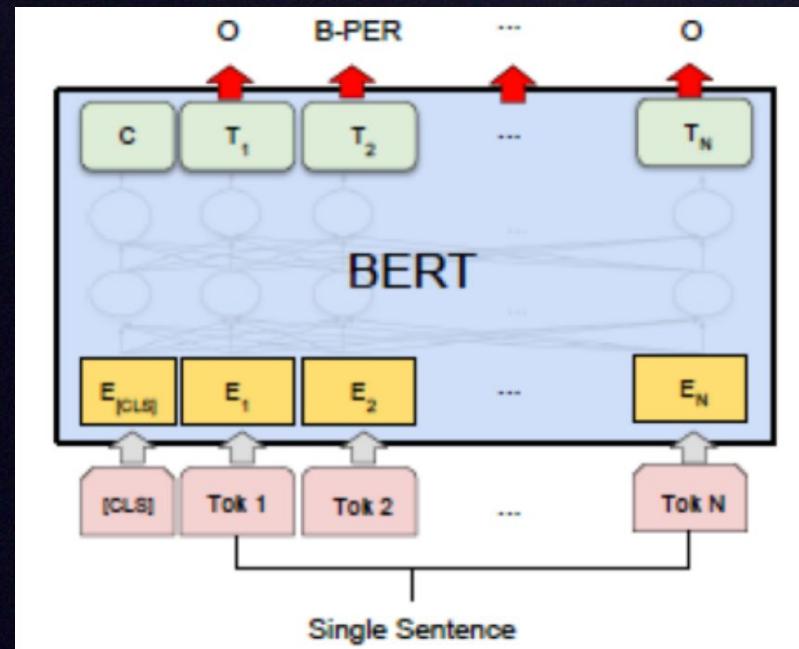
- Тэгирование (Named Entity Recognition)
 - Представление каждого токена отправляем в классификационный слой тэгов
 - Чтобы было совместимо с WordPiece, предикт только для первого токена слова

```
Jim    Hen    ##son was a puppet ##eer  
I-PER I-PER X      O   O O     X
```

BERT fine-tuning



- Тэгирование (Named Entity Recognition)



BERT fine-tuning

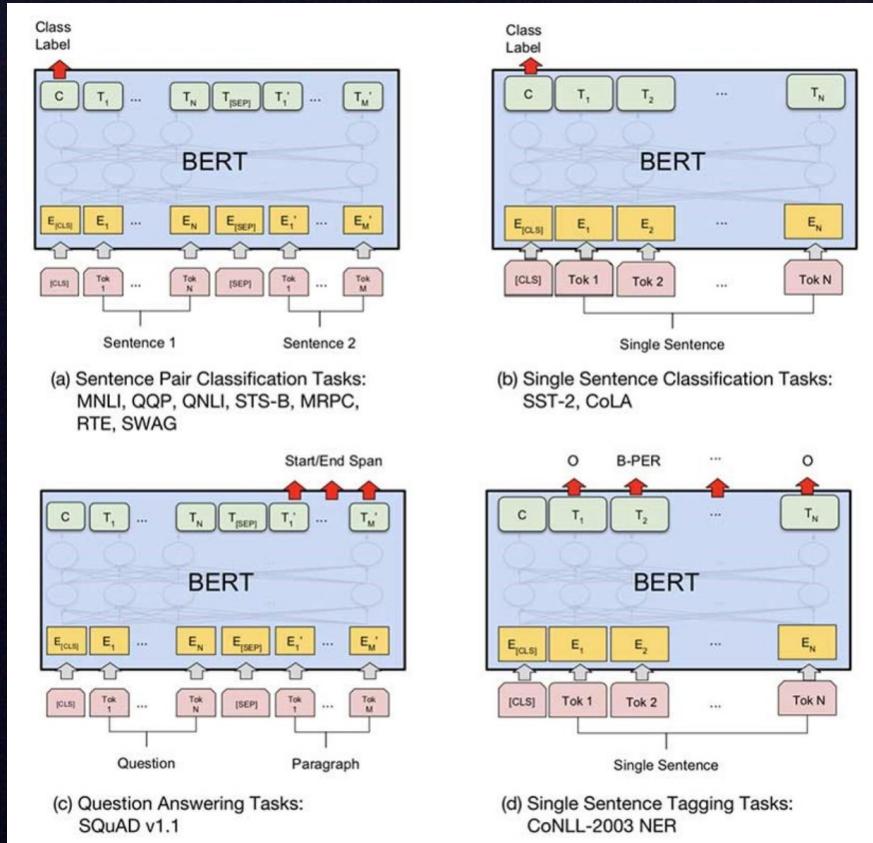


- Объединяем вопрос и параграф
- Рассчитываем вероятность того, что слово і будет началом ответа
- Log-likelihood для верных позиций начала и конца

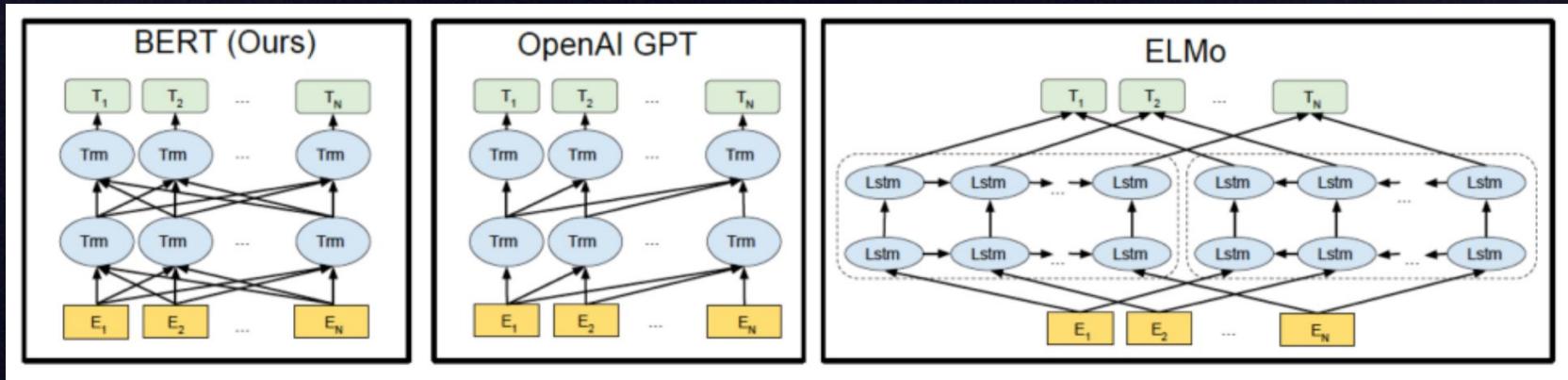
- Span-level task: SQuAD v1.1

- Input Question:
Where do water droplets collide with ice crystals to form precipitation?
- Input Paragraph:
.... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud.
- Output Answer:
within a cloud

BERT fine-tuning



Сравнение



Эксперименты

- GLUE (General Language Understanding Evaluation)
 - MNLI: Multi-Genre Natural Language Inference
 - QQP: Quora Question Pairs
 - QNLI: Question Natural Language Inference
 - SST-2: Stanford Sentiment Treebank
 - CoLA: Corpus of Linguistic Acceptability
 - STS-B: Semantic Textual Similarity Benchmark
 - MRPC: Microsoft Research Paraphrase Corpus
 - RTE: Recognizing Textual Entailment
 - WNLI: Winograd NLI



Эксперименты



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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); BERT_{BASE} = (L=12, H=768, A=12); BERT_{LARGE} = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from <https://gluebenchmark.com/leaderboard> and <https://blog.openai.com/language-unsupervised/>.

Text-to-Text Transfer Transformer T5

(Raffel et al., 2018)



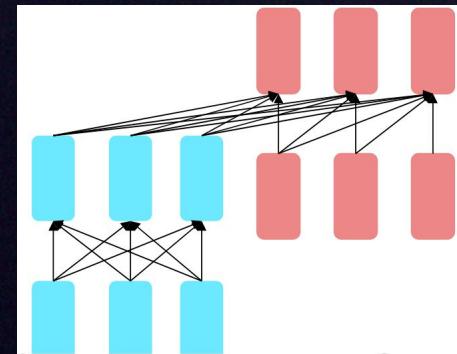
Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

- “Span corruption”
- Повреждаем фрагменты текста и заменяем заглушками 15%
- Декодируем заглушки

Targets

<X> for inviting <Y> last <Z>



Inputs

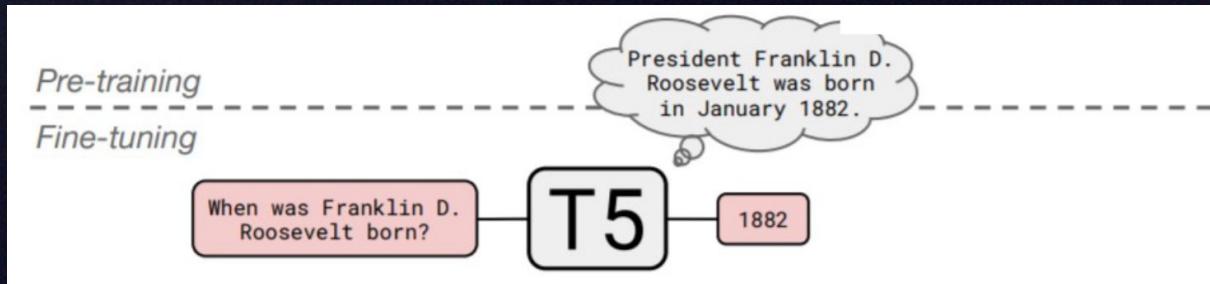
Thank you <X> me to your party <Y> week.

Text-to-Text Transfer Transformer T5



Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Text-to-Text Transfer Transformer T5



	NQ	WQ	TQA		220 million params
			dev	test	
Karpukhin et al. (2020)	41.5	42.4	57.9	—	
T5.1.1-Base	25.7	28.2	24.2	30.6	770 million params
T5.1.1-Large	27.3	29.5	28.5	37.2	3 billion params
T5.1.1-XL	29.5	32.4	36.0	45.1	11 billion params
T5.1.1-XXL	32.8	35.6	42.9	52.5	
T5.1.1-XXL + SSM	35.2	42.8	51.9	61.6	

GPT-2



- 1.5B параметров (x10 от GPT), почти везде SOTA в zero-shot
- Предобучение на 8 миллионах веб-страниц
- ВРЕ на UTF-8 для работы с любой unicode строкой
- Меньше версий слов dog? dog!

GPT-2



- Сдвинута нормализация на блоки
- Дополнительная нормализация после финального self-attention
- Скалирование весов $1/\sqrt{N}$ по количеству блоков с residual
- Инициализация с учетом глубины
- Словарь и окно контекста увеличены

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

GPT-2



Radford et al., 2019

Language Models are Unsupervised Multitask Learners

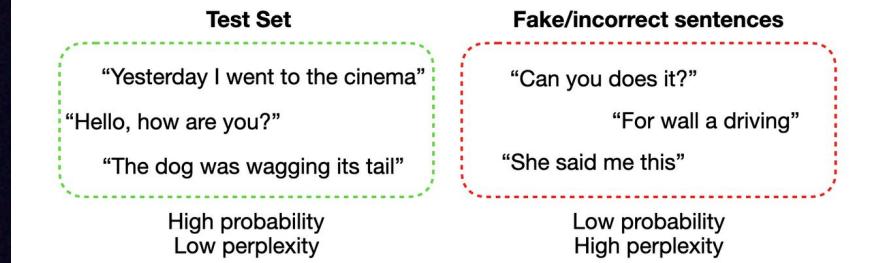
	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	21.8
117M	35.13	45.99	87.65	83.4	29.41	65.85	1.16	1.17	37.50	75.20
345M	15.60	55.48	92.35	87.1	22.76	47.33	1.01	1.06	26.37	55.72
762M	10.87	60.12	93.45	88.0	19.93	40.31	0.97	1.02	22.05	44.575
1542M	8.63	63.24	93.30	89.05	18.34	35.76	0.93	0.98	17.48	42.16

Перплексия



обратная вероятность тестового набора,
нормализованная по количеству слов

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$



Размер имеет значение

Самая большая T5 имеет 11B параметров

GPT-3 уже имела 175B параметров



Два способа взаимодействия:

- Работаем с предобученным распределением через промпт
- Дообучаем на задачу, которая нам важна

Но обо всём завтра

А для русского языка?



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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

Результаты на англоязычных датасетах

А для русского языка? BERT



Multilingual BERT

104 языка, разные алфавиты (не только латиница), 110 млн. параметров

Предобученная модель:

<https://huggingface.co/google-bert/bert-base-multilingual-cased>

Russian BERT

адаптирован на русских текстах в рамках проекта iPavlov

Предобученная модель:

<https://huggingface.co/DeepPavlov/rubert-base-cased>

Оценка



FactRuEval-2016

Распознавание именованных сущностей в русских текстах

Датасет

<https://github.com/dialogue-evaluation/factRuEval-2016>

122 текста для обучения

133 текста для тестирования

Оценка



Метод	F1 по ORG, PER и LOC
словари и эмбеддинги + SVM <i>Named Entity Recognition in Russian: the Power of Wiki-Based Approach</i>	74,67 %
эмбеддинги + SVM <i>Russian Named Entities Recognition and Classification Using Distributed Word and Phrase Representations</i>	87,88 %
ELMo + CRF	89,01 %

Оценка



	Precision	Recall	F1
RuBERT (frozen) + CRF	85,26%	83,47%	84,35%
RuBERT (tuned) + CRF	81,48%	91,77%	86,32%
RuBERT (frozen) + BiLSTM (64x2) + CRF	87,05%	86,93%	86,99%
RuBERT (tuned) + BiLSTM (64x2) + CRF	83,97%	90,42%	87,07%
<i>MultiBERT</i> (frozen) + CRF	82,83%	81,01%	81,91%
<i>MultiBERT</i> (tuned) + CRF	73,90%	82,78%	78,09%
<i>MultiBERT</i> (frozen) + BiLSTM (64x2) + CRF	69,47%	78,33%	73,64%
<i>MultiBERT</i> (tuned) + BiLSTM (64x2) + CRF	79,55%	88,28%	83,69%

https://github.com/bond005/factRuEval-2016/tree/master/elmo_lstm

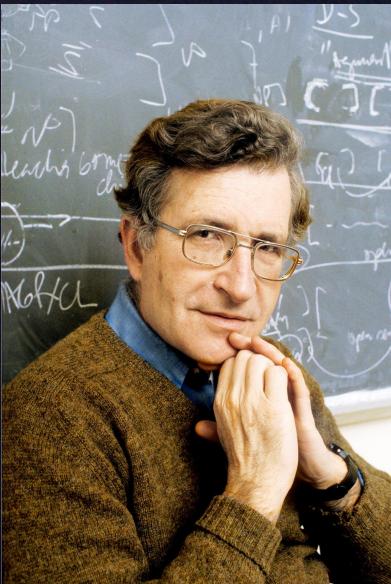
BERT и его друзья для классификации текстов



	FastText-Bow	FastText-IDF	SkipThoughts	Bert (multilingual)
SST binary (accuracy, %)	62,30%	58,33%	81,89%	98,94%
SST fine-grained (accuracy, %)	86,63%	86,61%	86,98%	86,46%
MRPC (accuracy, %)	99,78%		99,93%	99,98%
Readability classifier (accuracy, %)	30,97%	23,48%	40,25%	38,38%
Tag classifier (accuracy, %)	31,41%	26,78%	25,02%	20,28%
Poems classifier (accuracy, %)	37,74%	36,12%	35,58%	37,47%
Proza classifier (accuracy, %)	44,53%		45,55%	43,77%
translated TREC (accuracy, %)	20,80%	19,60%	35,00%	49,80%
translated STS Benchmark (Pearson)	0,537460	0,461164	0,650095	0,551022
translated SICK-Relatedness (Pearson)	0,705766	0,672332	0,705707	0,628056

<https://github.com/comptechml/SentEvalRu>

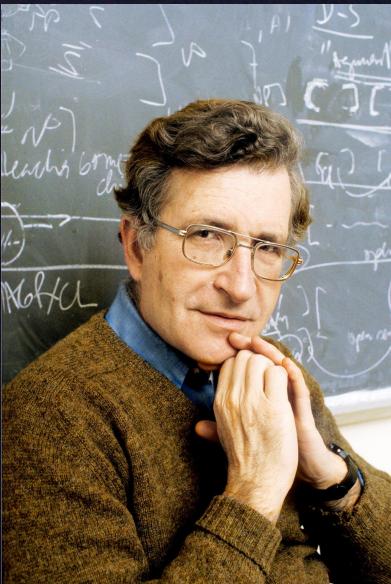
Мультиязычные модели для анализа текстов



Теория принципов и параметров
(Ноам Хомский и др.)

Всем языкам мира присуща универсальная грамматика:
общие принципы (для всех языков);
частные параметры (для каждого конкретного языка)

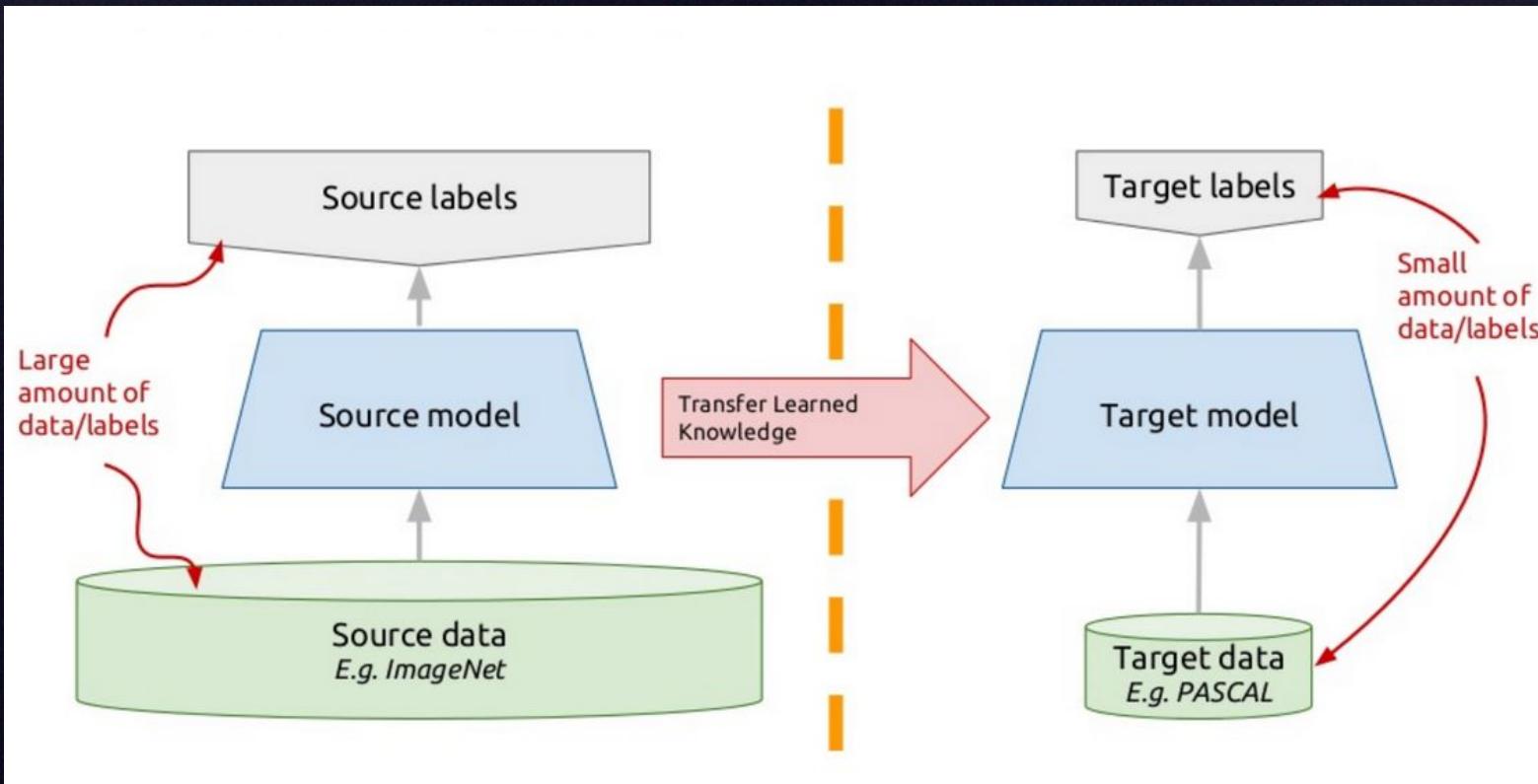
Мультиязычные модели для анализа текстов



Теория принципов и параметров
(Ноам Хомский и др.)

Всем языкам мира присуща универсальная грамматика:
общие принципы (для всех языков);
частные параметры (для каждого конкретного языка)

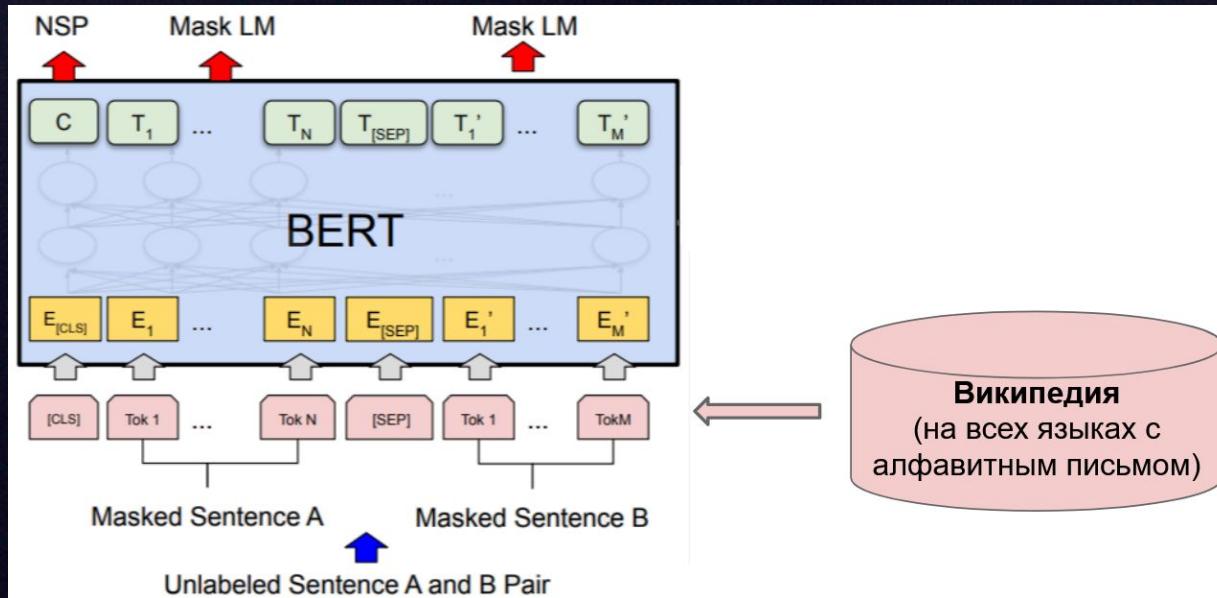
Перенос обучения



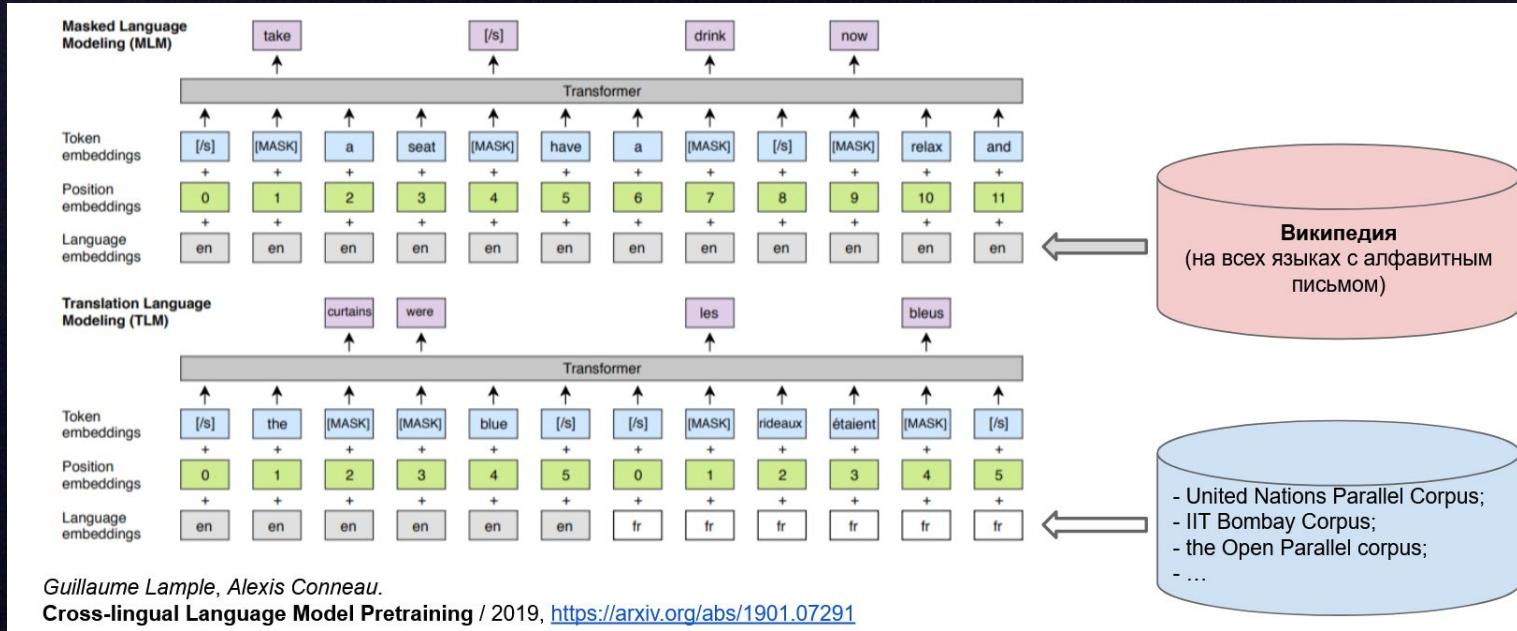
Перенос обучения



Мультиязычный BERT



XLM: не просто мульти, а кросс!



Multilingual Toxic Comments 2020



- Все размеченные тексты на английском языке с предыдущих соревнований
 - более 200 000 текстов в 2018 году
 - почти 2 000 000 текстов в 2019 году
- 8 000 размеченных текстов на испанском, итальянском и турецком для дополнительного обучения
- 30 932 текстов на испанском, итальянском и турецком для финального тестирования
- 32 880 текстов на французском, русском и португальском - тоже для финального тестирования!

<https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification>

Multilingual Toxic Comments 2020



Алгоритм	ROC-AUC на приватной части тестсета
mBERT	0,8756
XLM-RoBERTa (base)	0,9030
XLM-RoBERTa (large)	0,9289

Что видит BERT/XLM в тексте?



<https://huggingface.co/exbert>

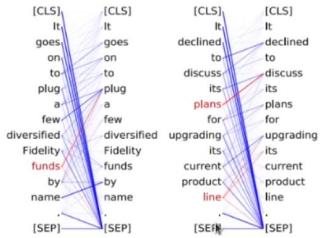
Explorable Transformers:
трансформеры, которые можно
анализировать

ЧТО ВИДИТ XLM В ТЕКСТЕ?



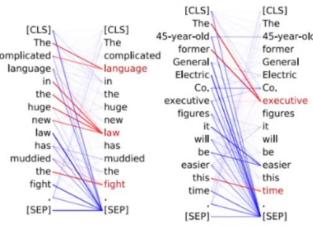
Head 8-10

- Direct objects attend to their verbs
- 86.8% accuracy at the dobj relation



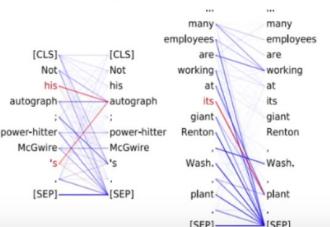
Head 8-11

- Noun modifiers (e.g., determiners) attend to their noun
- 94.3% accuracy at the det relation



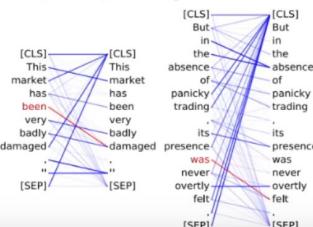
Head 7-6

- Possessive pronouns and apostrophes attend to the head of the corresponding NP
- 80.5% accuracy at the poss relation



Head 4-10

- Passive auxiliary verbs attend to the verb they modify
- 82.5% accuracy at the auxpass relation



Head 1-1 Attends broadly

... found in taiwan
[SEP] the wingspan is 24 mm [SEP]

Head 3-1 Attends to next token

... found in taiwan [SEP]

Head 8-7 Attends to [SEP]

... found in taiwan [SEP]

Head 11-6 Attends to periods

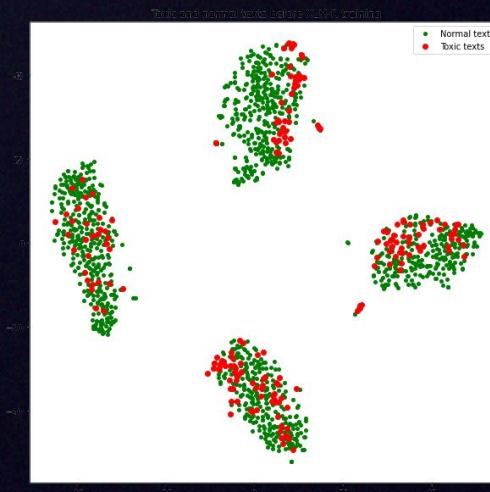
... found in taiwan [SEP] the wingspan is 24 mm [SEP]

Видны ли “общие принципы” Хомского?

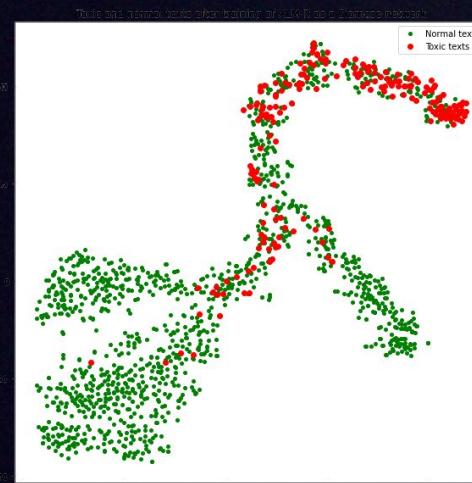


Проекции семантических векторов для нормальных и токсичных текстов на английском, испанском, итальянском и турецком с помощью “большой” языковой модели XLM-RoBERTa

Без дообучения на английских текстах



После дообучения



Спасибо за внимание!

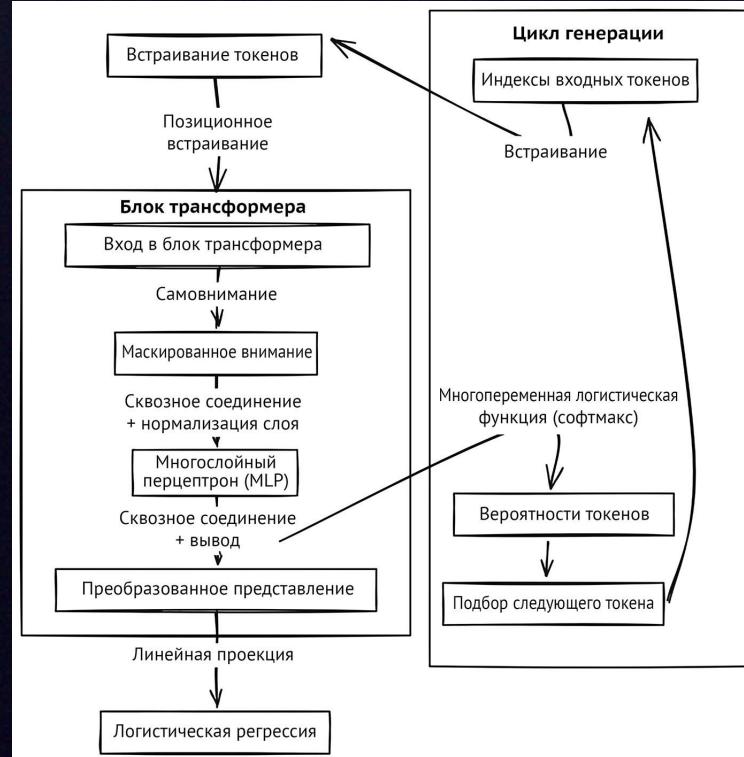


👉 Дерунец Роман



N★ Новосибирский
государственный
университет
***НАСТОЯЩАЯ НАУКА**

Ещё раз



Внимание



```
class Attention(nn.Module):
    def __init__(self, n_embed, context_length):
        super().__init__()
        # Полносвязный слой для вычисления матриц запросов (Q), ключей (K) и значений (V)
        self.qkv = nn.Linear(n_embed, n_embed * 3, bias=False)

        # Создаем нижнетреугольную матрицу для каузальной маскировки
        # (запрещает доступ к будущим токенам)
        self.tril = torch.tril(torch.ones(context_length, context_length))

    def forward(self, x):
        B, T, C = x.shape # B = размер пакета, T = длина последовательности, C = размерность
        # Разделяем выход слоя qkv на три матрицы: запросы (q), ключи (k) и значения (v)
        q, k, v = self.qkv(x).chunk(3, dim=-1)

        # Вычисляем коэффициенты внимания через скалярное произведение запросов и ключей
        attn = (q @ k.transpose(-2, -1)) / C**0.5

        # Накладываем каузальную маску, чтобы модель не смотрела на будущие токены
        attn = attn.masked_fill(self.tril[:T, :T] == 0, float('-inf'))

        # Применяем softmax к коэффициентам внимания и вычисляем взвешенные значения
        return (F.softmax(attn, dim=-1) @ v)
```

Внимание



```
class TransformerBlock(nn.Module):
    def __init__(self, n_embed, context_length):
        super().__init__()
        # Инициализируем механизм внимания (Attention)
        self.attn = Attention(n_embed, context_length)

        # Инициализируем многослойный перцептрон
        self.mlp = MLP(n_embed)

        # Слой нормализации для повышения стабильности обучения
        self.ln = nn.LayerNorm(n_embed)

    def forward(self, x):
        # Применяем LayerNorm и механизм внимания, затем добавляем остаточную связь
        x = x + self.attn(self.ln(x))

        # Применяем LayerNorm и MLP, снова добавляем остаточную связь
        return x + self.mlp(self.ln(x))
```

Некоторые проблемы

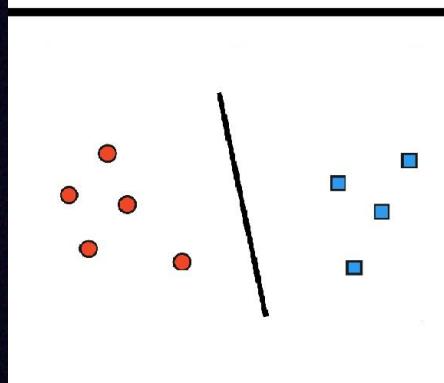
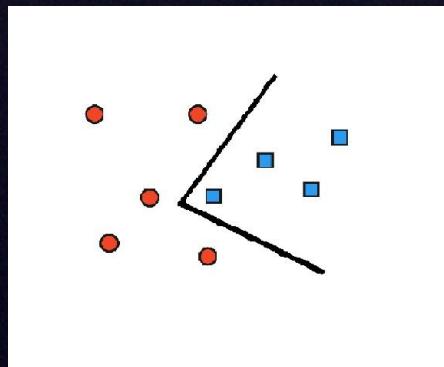


В Китае LOC отпраздновали 170-летие публикации
"Коммунистического манифеста MISC"

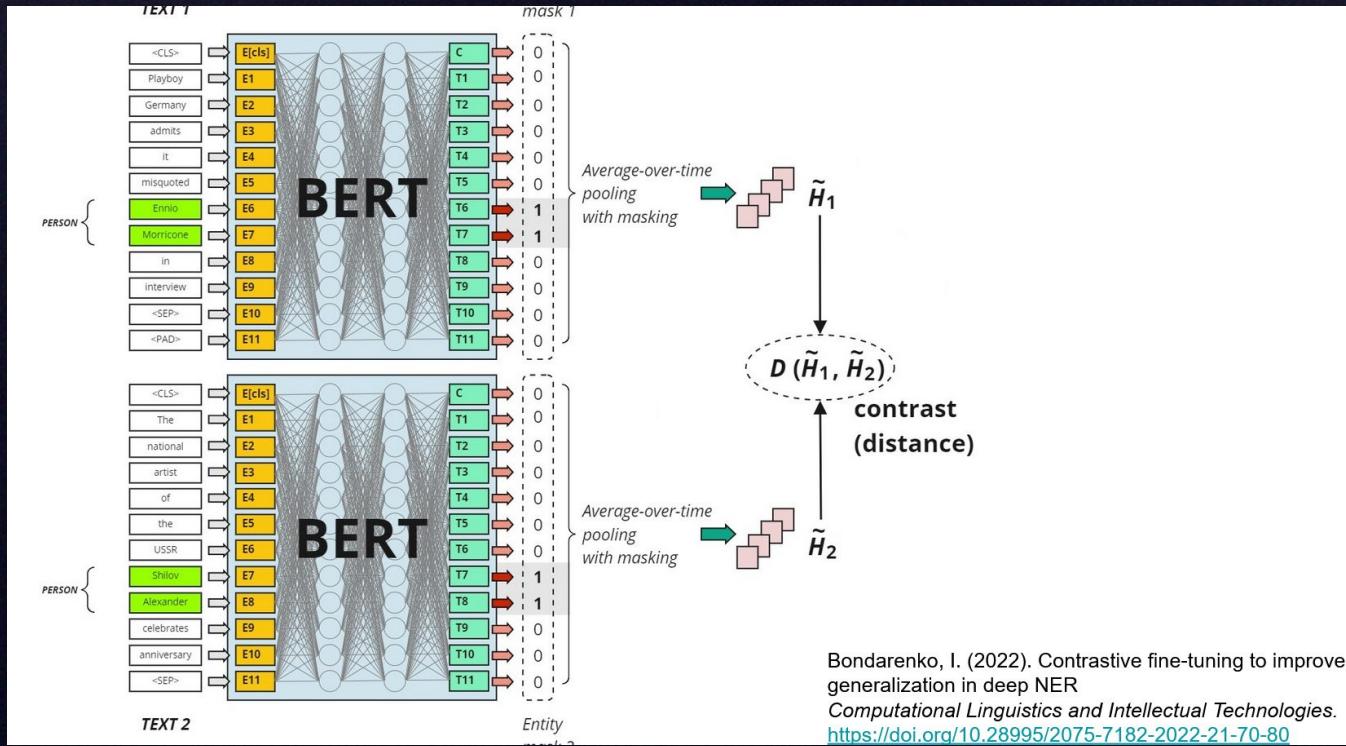
В Китае LOC отпраздновали 170-летие публикации
Коммун MISC истического манифеста

В Китае LOC отпраздновали 170-летие публикации
Коммун MISC истического манифест MISC а"

Некоторые проблемы

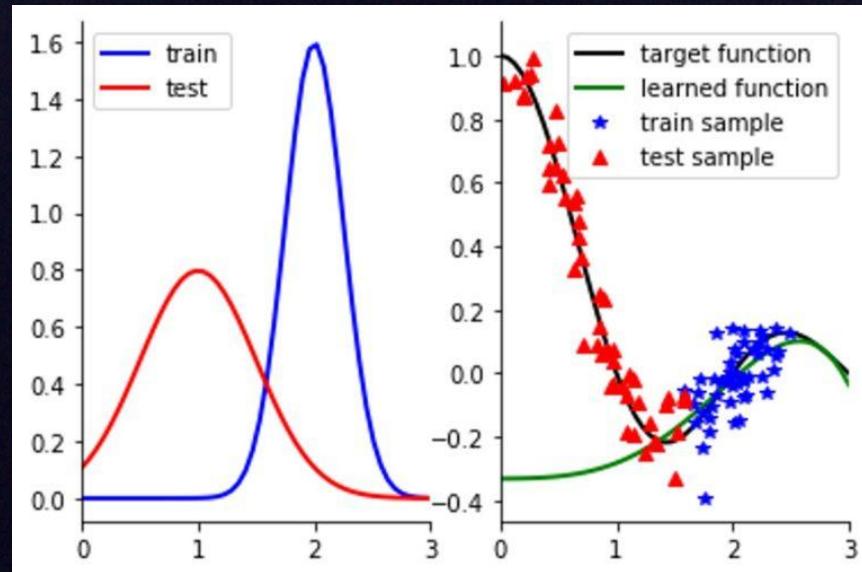


Сиамская сеть



Bondarenko, I. (2022). Contrastive fine-tuning to improve generalization in deep NER
Computational Linguistics and Intellectual Technologies.
<https://doi.org/10.28995/2075-7182-2022-21-70-80>

Сдвиг данных



Как обучаться и распознавать при сдвиге данных?



- **Инвариантность**

- модель должна находить инвариантные корреляции между признаками и целевой переменной

- **Неопределённость**

- модель должна отказываться от принятия решения, когда не уверена



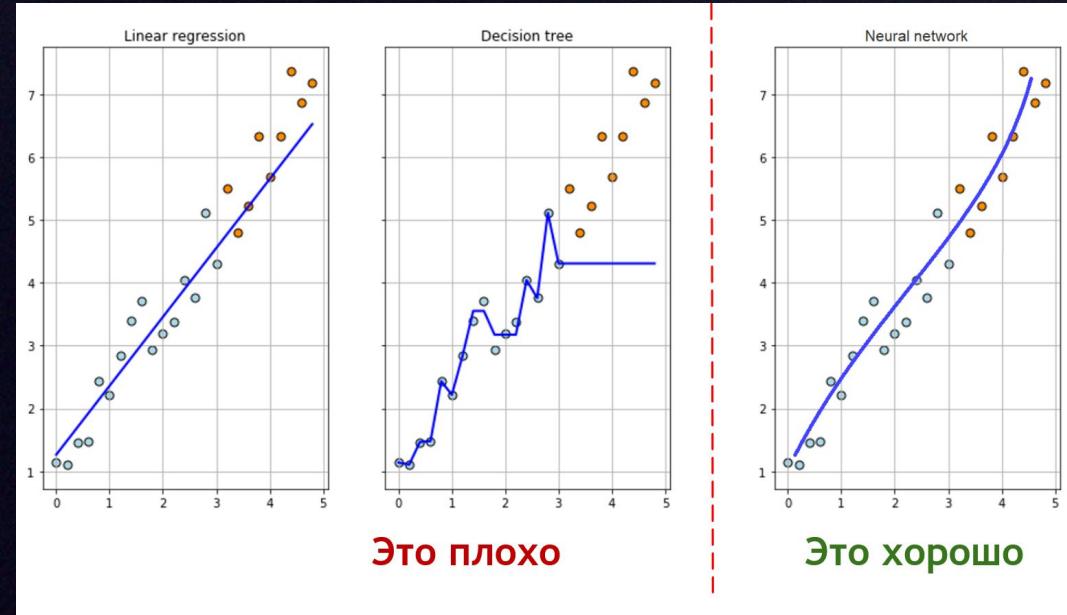
Как обучаться и распознавать при сдвиге данных?



**«Игрушечный»
пример на
синтетических
данных**

Голубые точки —
данные для обучения и
для *in*-тестирования.

Оранжевые точки —
данные для *out*-
тестирования.



Как обучаться и распознавать при сдвиге данных?



Общая неопределённость

=

Ошибки в данных

+

Ошибки в знаниях

Ансамбли из алгоритмов



Когда в процессе обсуждения участвуют многие, каждый может внести свою лепту добродетели и благоразумия... один понимает одну деталь, другой — другую, и все вместе понимают все.

Аристотель

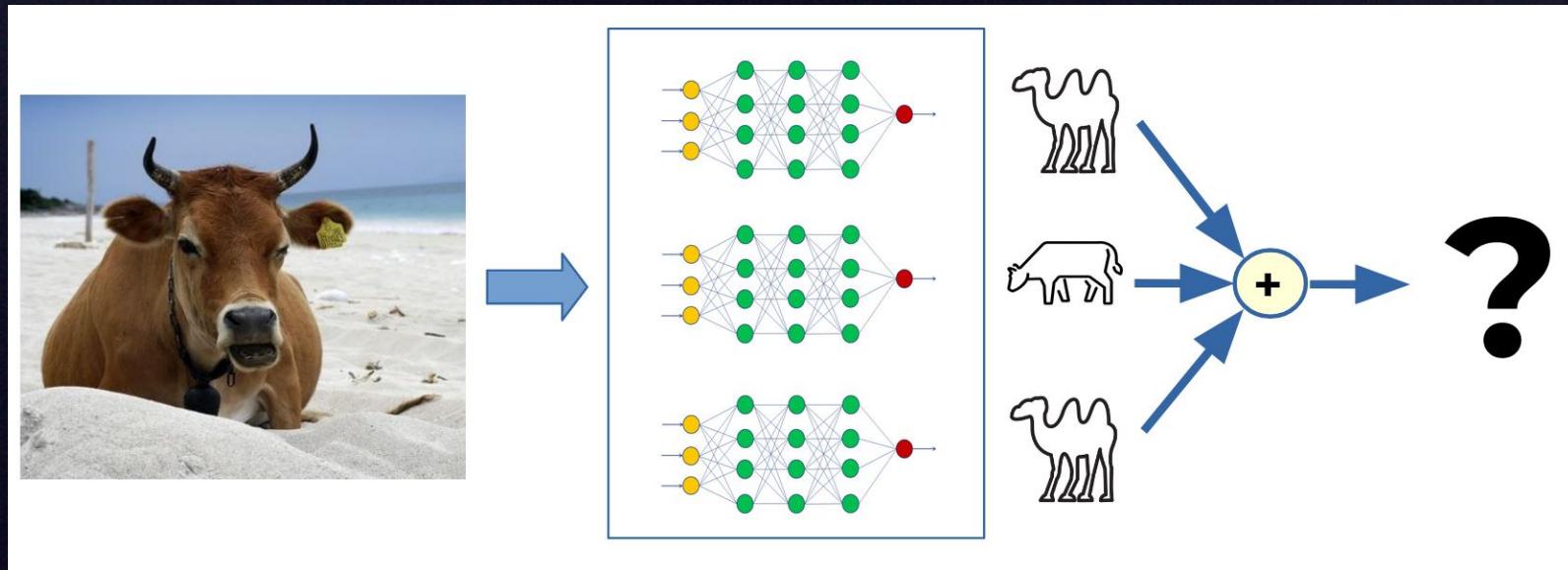
Ансамбли из алгоритмов



Энтропия решения ансамбля алгоритмов $H(A|X)$ может уменьшаться при увеличении числа алгоритмов, участвующих в ансамбле в системе

$$H(A|X_1, \dots, X_N) \leq H(A|X_1, \dots, X_{N-1})$$

Ансамбли из алгоритмов



Ансамбли из алгоритмов



Shifts Challenge

WEATHER

MACHINE TRANSLATION

VEHICLE MOTION PREDICTION

FAQ



Leaderboard

[ALL SUBMISSIONS](#) / [MY SUBMISSIONS](#)

RANK	TEAMNAME/ USERNAME	METHOD NAME	SCORE (R-AUC, MSE)	RMSE	MAE	AUC-F1	F1 @ 05%	ROC-AUC (%)	DATE SUBMITTED
1	bond005	SNN Ens U MT N...	1.141	1.979	1.379	0.557	0.679	71.556	7/11/2021
2	bond005	SNN Ens U MT N...	1.142	1.937	1.369	0.554	0.679	72.108	4/11/2021
3	bond005	SNN Ens U MT	1.153	1.929	1.371	0.548	0.675	68.992	29/10/2021
4	CabbeanWeather	Steel box v2	1.158	1.936	1.382	0.541	0.666	74.478	6/11/2021
5	KDDI Research	more seed ens	1.159	1.928	1.374	0.538	0.67	76.158	8/11/2021
6	CabbeanWeather	Steel box v1	1.161	1.936	1.382	0.54	0.666	74.629	5/11/2021
7	CabbeanWeather	Purple box v6	1.164	1.937	1.383	0.539	0.667	74.194	4/11/2021
8	KDDI Research	sub3	1.166	1.899	1.369	0.539	0.662	65.261	21/10/2021
9	CabbeanWeather	Purple box v5	1.168	1.942	1.386	0.537	0.665	67.966	3/11/2021
10	KDDI Research	seed ens	1.172	1.927	1.375	0.536	0.669	75.634	7/11/2021

← 1 / 8 →

<https://research.yandex.com/shifts/weather>

Ошибка



Ошибка - это:

1. “Недогенерация”

нейронная сеть - это эффективный инструмент а
далше?

2. Морфологические неточности

нейронные сеть - это эффективный инструмент
искусственного интеллекта

Галлюцинации



Галлюцинация - это:

1. “Зацикливание”

нейронная сеть - это эффективный это это это это

2. Ответ с правильным синтаксисом и морфологией, но бессмысленный

Вопрос: *Что такое нейронная сеть?*

Ответ: *Машины опорных векторов — семейство алгоритмов бинарной классификации, основанных на обучении с учителем*

Галлюцинации



Галлюцинация - это:

1. “Зацикливание”

нейронная сеть - это эффективный **это это это это**

2. Ответ с правильным синтаксисом и морфологией, но бессмысленный

Вопрос: *Что такое нейронная сеть?*

Ответ: *Машины опорных векторов — семейство алгоритмов бинарной классификации, основанных на обучении с учителем*

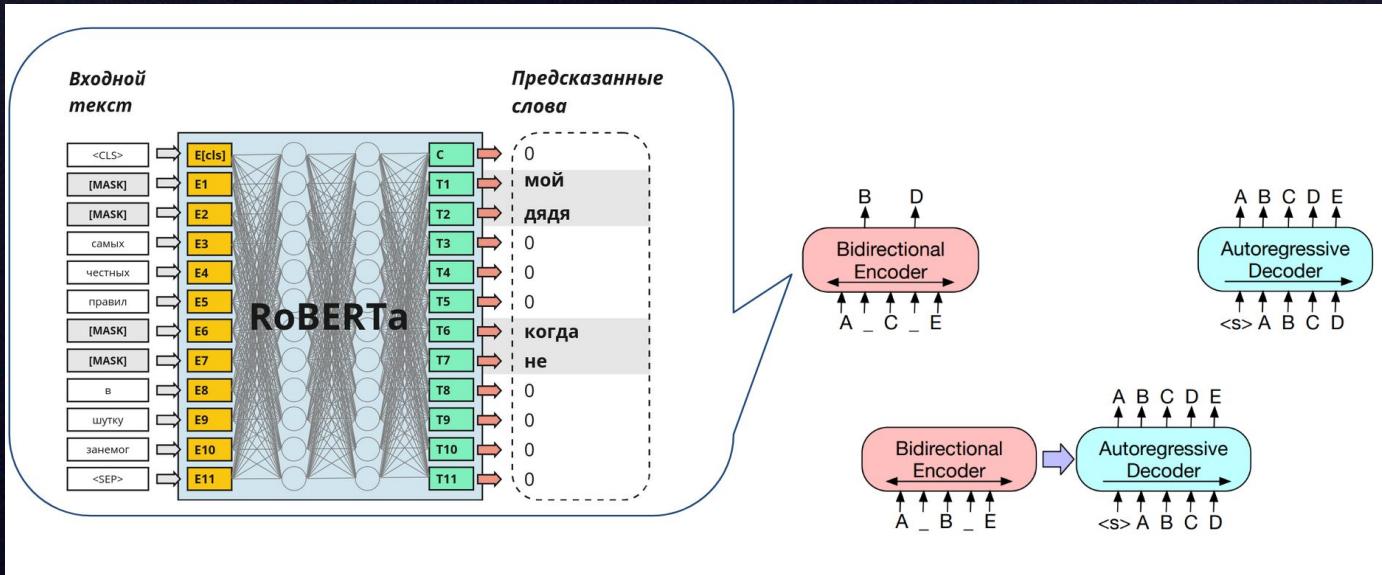
Галлюцинации



Две стадии обучения языковых моделей:

- предварительное обучение без учителя (*self-supervised pre-training*)
- точная настройка с учителем (*supervised fine-tuning*)

Предобучение



Тонкая настройка



Подожди, ты это серьёзно?



Excuse me, you're serious?