

ИНТЕЛЛЕКТ ➤

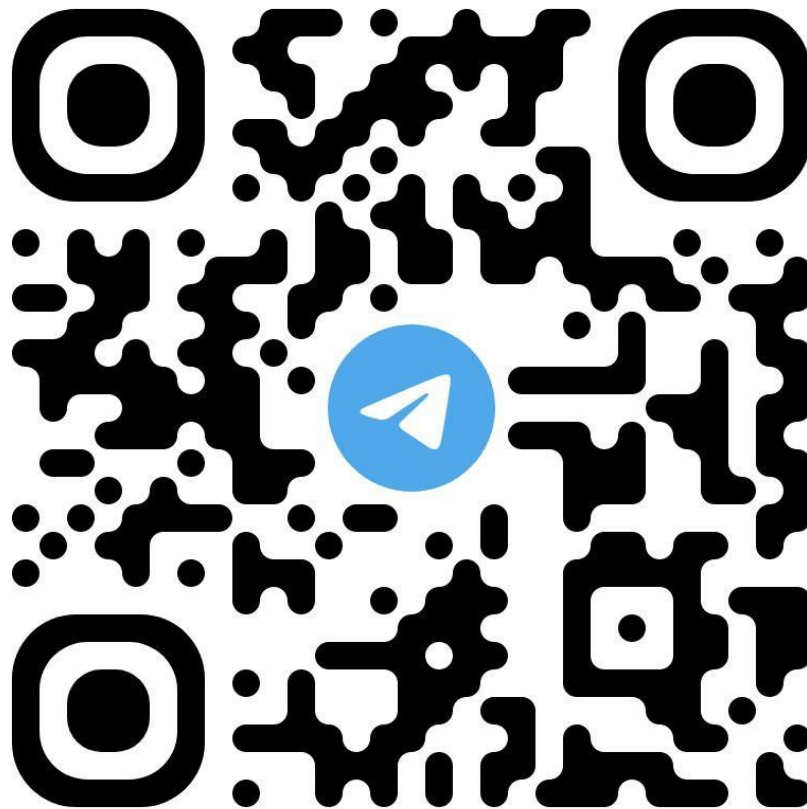
Большие языковые модели (LLM/БЯМ)

к.ф.-м.н. Тихомиров Михаил Михайлович

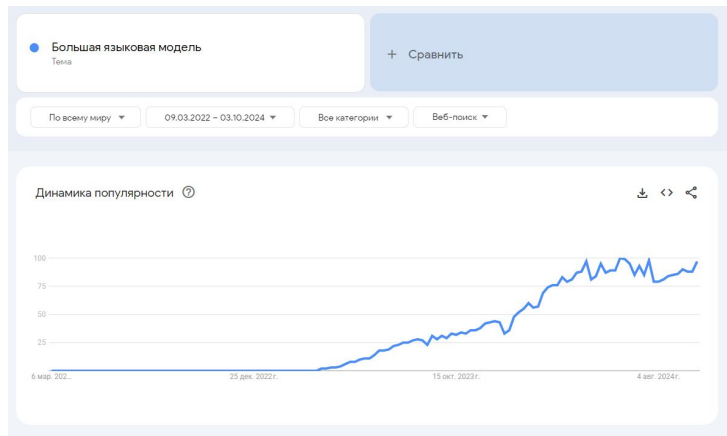
научный сотрудник
НИВЦ МГУ имени М. В. Ломоносова

Введение

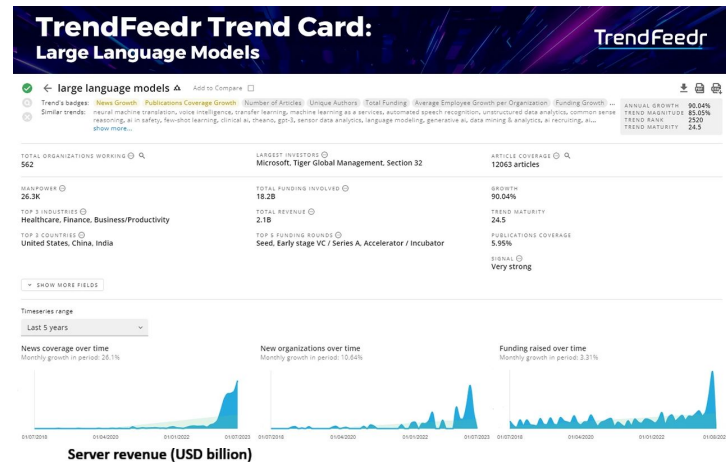
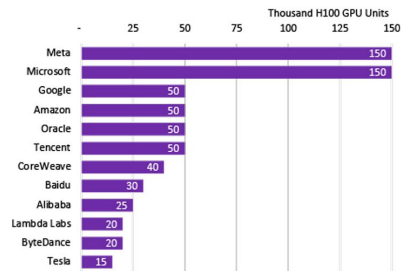
QR-code



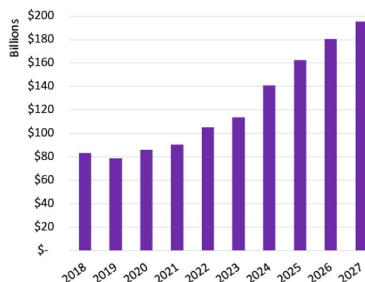
Рост популярности LLM в мире



H100 GPU orders expected to be fulfilled in 2023



Server revenue (USD billion)



Рост популярности LLM в мире



Оценка стоимости LLM компаний инвесторами

- Mistral - 5.8 миллиарда
- XAI - 24 миллиарда
- Anthropic - 40 миллиардов
- OpenAI - 157 миллиардов

Для сравнения (market cap):

- Siemens ~ 150 миллиардов
- Nvidia ~ 3 триллиона (рост **x10** за 4 года)



Языковые модели оценивают вероятность различных текстовых сущностей: символов, слов, последовательностей слов.

- Первым человеком в космосе был ____ ?
- Что правдоподобнее:
 - я съел жареный гвоздь vs я съел жареный стейк

Где применяется языковое моделирование



Все мы регулярно сталкиваемся с языковым моделированием:

- Автодополнение на клавиатурах телефонов.
- Подсказки в поисковых системах.
- Исправление ошибок в поисковых системах.
- Распознавание речи и др.

Google

языковое моделирование

Картинки Видео Покупки Новости Карты Книги Авиабилеты

About 2,290,000 results (0.28 seconds)

Showing results for **языковое моделирование**
Search instead for **языковое моделирование**

Википедия
<https://ru.wikipedia.org/wiki/Яз...> · Translate this page

Языковая модель

Языковая модель — это распределение вероятностей по последовательностям слов. Для любой последовательности слов длины n языковая модель присваивает ...

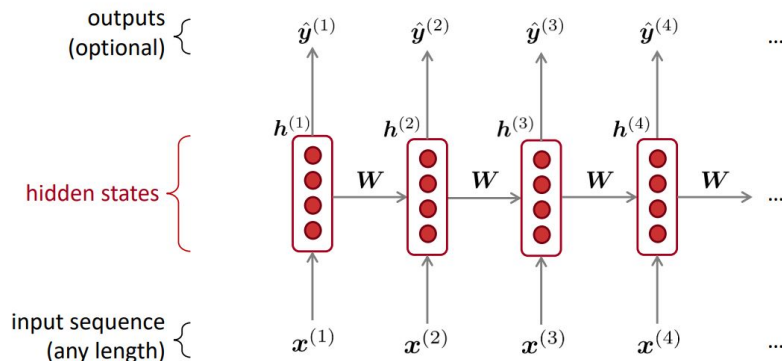


- Оценить вероятность слова в зависимости от длинной последовательности предыдущих слов не просто.
- Но оценить вероятность в зависимости от прошлых N слов не представляет труда.
- Например, биграммная модель:

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = P(x_n | x_{n-1})$$

$$P(x_n | x_{n-1}) = \frac{C(x_n, x_{n-1})}{C(x_{n-1})}$$

- Современные языковые модели основаны на архитектуре трансформер.
- Но, одни из первых успешных нейросетевых языковых моделей были основаны на **рекуррентных сетях (RNN)**.



$$e_t = \mathbf{E}x_t$$

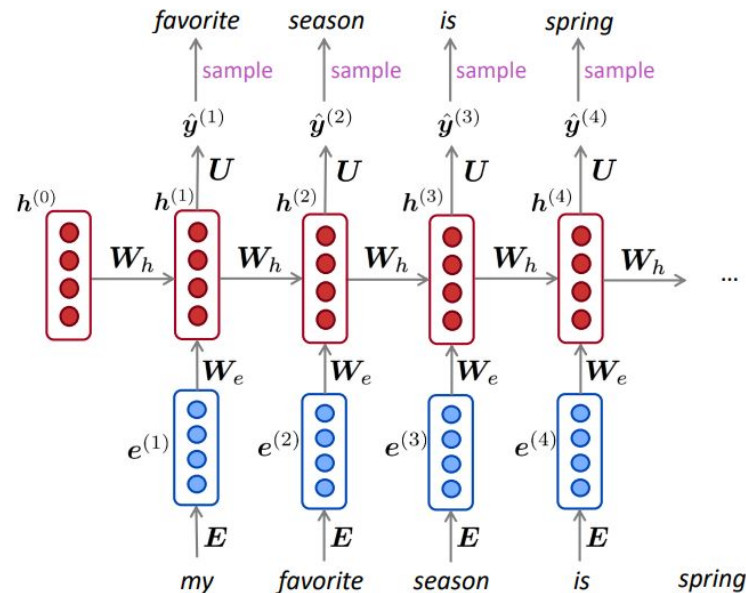
$$h_t = g(\mathbf{U}h_{t-1} + \mathbf{W}e_t)$$

$$y_t = \text{softmax}(\mathbf{E}^T h_t)$$

Генерация текста на основе RNN



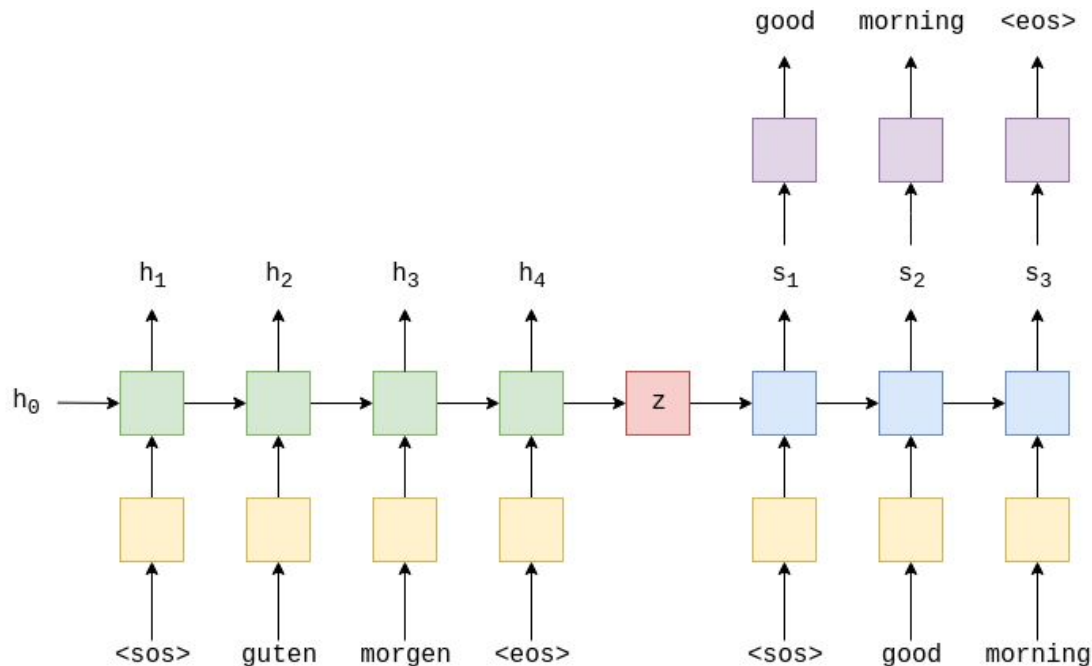
- После выходов RNN слой предсказания следующего слова.
- На каждом шаге на основе вероятностей следующего слова происходит **сэмплирование**.
- Затем, выбранное слово идет как часть входа и процедура повторяется.



Seq2Seq до трансформеров



- Вектор финального состояния должен хранить **ВСЮ** информацию из предложения
- По сути является векторным представлением (эмбеддингом) предложения
- Теряет информацию на длинных последовательностях



Механизм внимания (2014)

Автокодирующая модель состоит

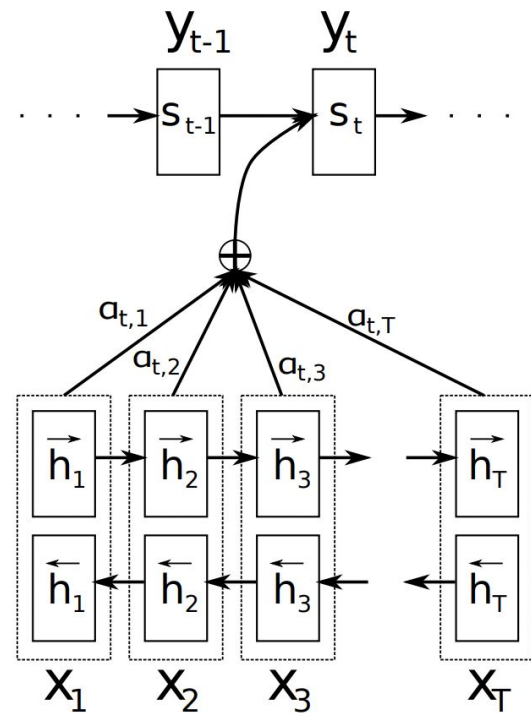
из:

- **Encoder(text) -> vector:**
переводит текст в необходимое
векторное представление
- **Decoder(vector) -> text:**
расшифровывает
представление в ответ модели

Проблема: в vector помещается
только общий контекст

Решение: сохранять векторы для
каждого слова и подбирать нужные
под каждый шаг decoder

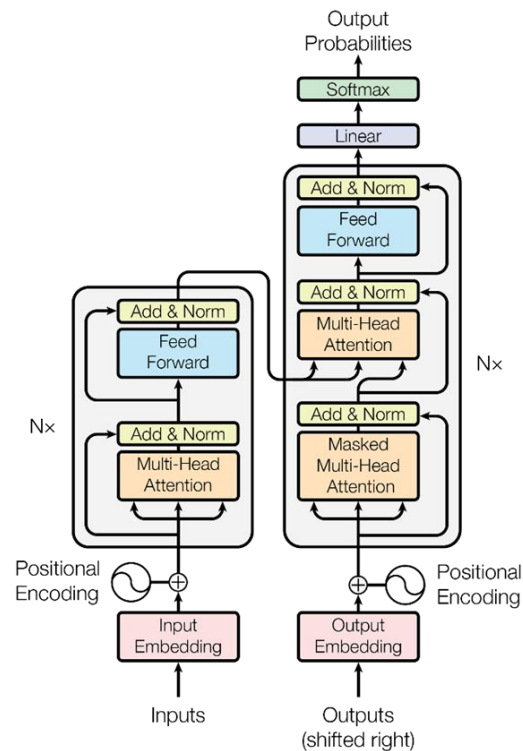
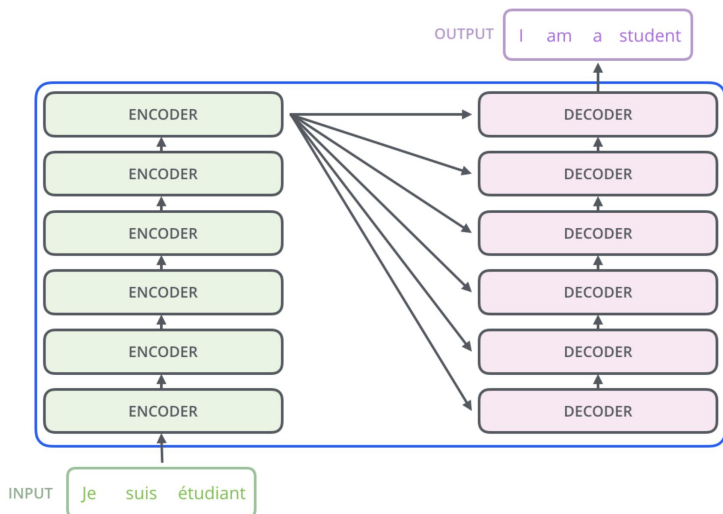
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$
$$e_{ij} = a(s_{i-1}, h_j)$$



Transformer (2017)

Исходно **encoder-decoder** архитектура.

Каждый блок одинаков и последовательно преобразует входной вектор в выходной вектор той же размерности.



Transformer: первое впечатление



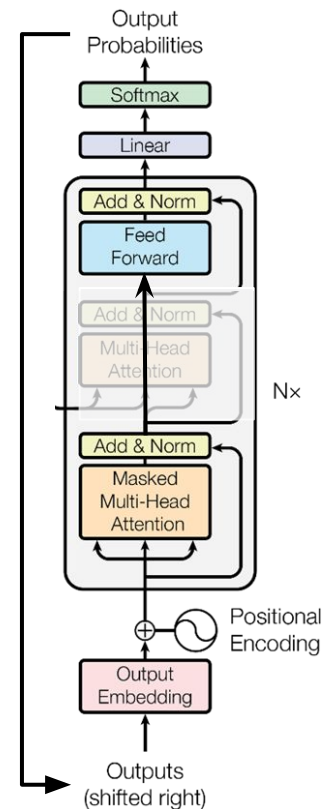
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
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	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	41.8	$2.3 \cdot 10^{19}$	

- Тестирование на задаче перевода,
- Нет существенного “скачка” в качестве.

OpenAI GPT-1 (2018)

- 12 слоев **Transformer decoder** (~117 млн.),
- Обучение в **2 этапа**:
 - Предобучение (pre-training) на задаче **моделирования языка**
$$\max_{\Theta} \sum_{0 \leq i \leq n} \log P(w_i | w_{i-1} \dots w_0; \Theta)$$

w - слова последовательности, Θ - параметры модели
 - Дообучение (fine-tuning) на целевые задачи
- Предобучался только на художественной литературе



OpenAI GPT-1: оценка качества



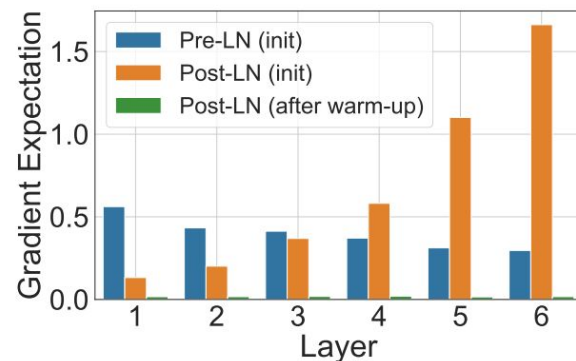
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	61.7
Finetuned Transformer LM (ours)	82.1	81.4	89.9	88.3	88.1	56.0

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	<u>77.6</u>	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	<u>60.2</u>	<u>50.3</u>	<u>53.3</u>
Finetuned Transformer LM (ours)	86.5	62.9	57.4	59.0

GPT-2 (2019) – универсальный генератор текста



- **Улучшенная архитектура:**
предварительная нормализация (**Pre-LN**)
входных данных для стабилизации градиентов
- **Больше параметров:**
в 4 раза больше слоев (**1.5 млрд параметров**)
– больше потенциальных знаний (capacity)
- **Новая парадигма:** любой текст содержит **подсказки к генерации (prompt)**
и обучаясь на большом наборе текстов модель учится их понимать



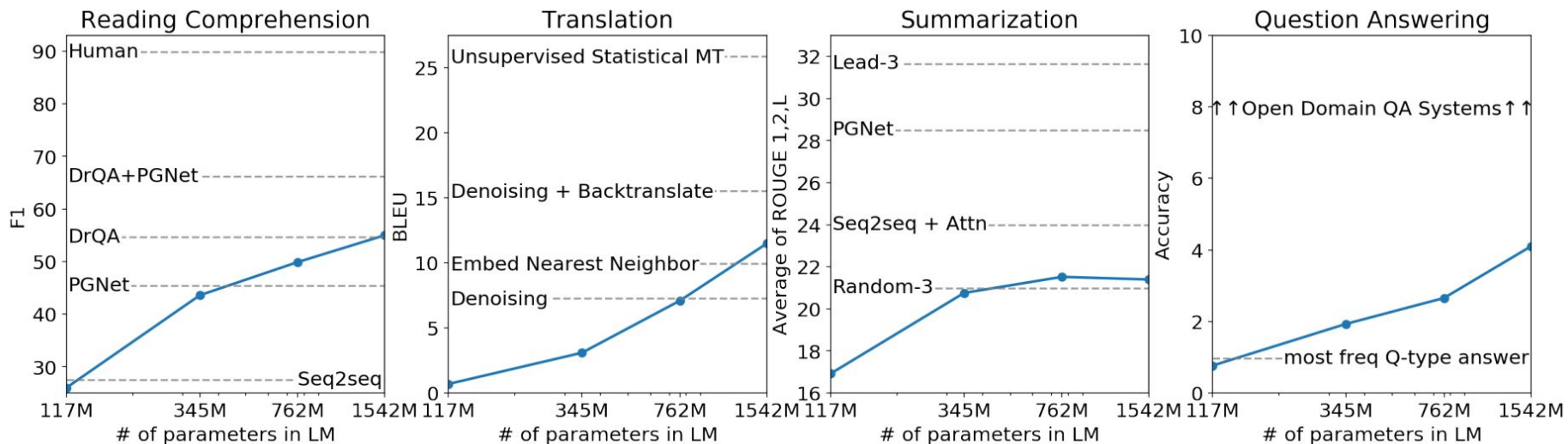


GPT-2: датасет для предобучения

- Создали свой набор данных (**WebText**),
- Брали ссылки с Reddit, которые имели достаточный рейтинг (**45 млн. ссылок**),
- Дополнительная очистка на основе ряда эвристик, дедубликация,
- Намеренно удалили Wikipedia ссылки из **WebText**,
- Итого **40GB** текстов или **8 млн.** документов.

Одна из основ успеха модели - **качественный** набор данных для предобучения!

GPT-2: zero-shot ВОЗМОЖНОСТИ



GPT-2: zero-shot возможности



	R-1	R-2	R-L	R-AVG
Bottom-Up Sum	41.22	18.68	38.34	32.75
Lede-3	40.38	17.66	36.62	31.55
Seq2Seq + Attn	31.33	11.81	28.83	23.99
GPT-2 TL;DR:	29.34	8.27	26.58	21.40
Random-3	28.78	8.63	25.52	20.98
GPT-2 no hint	21.58	4.03	19.47	15.03

Table 4. Summarization performance as measured by ROUGE F1 metrics on the CNN and Daily Mail dataset. Bottom-Up Sum is the SOTA model from (Gehrmann et al., 2018)

Добавление **TL;DR**: повышает ROUGE-1 с 21.5 до 29.3 (подсказки к генерации!)

Пример порождения новостей (перевод)



*Подводка: Шокирующая находка: **ученый обнаружил стадо единорогов, живущих в отдаленной, ранее неизведанной долине, в Андах. Еще более удивительным для исследователей было то, что единороги прекрасно говорили по-английски.***

GPT-2: Доктор Хорхе Перес, биолог-эволюционист из Университета Ла-Паса, и несколько его товарищей **исследовали Анды**, когда **обнаружили небольшую долину**, где не было других животных или людей. Перес заметил, что в долине есть что-то похожее на естественный фонтан, окруженный двумя пиками скалы и серебристым снегом.

Перес и остальные отправились дальше в долину. ... Перес и его друзья были поражены, **увидев стадо единорогов**. Этих существ можно было увидеть с воздуха — они были так близко, что могли коснуться своими рогами.

Изучая этих причудливых существ, ученые обнаружили, что **существа также говорили на довольно обычном английском языке...**

GPT-3 (2020) – первая коммерческая модель

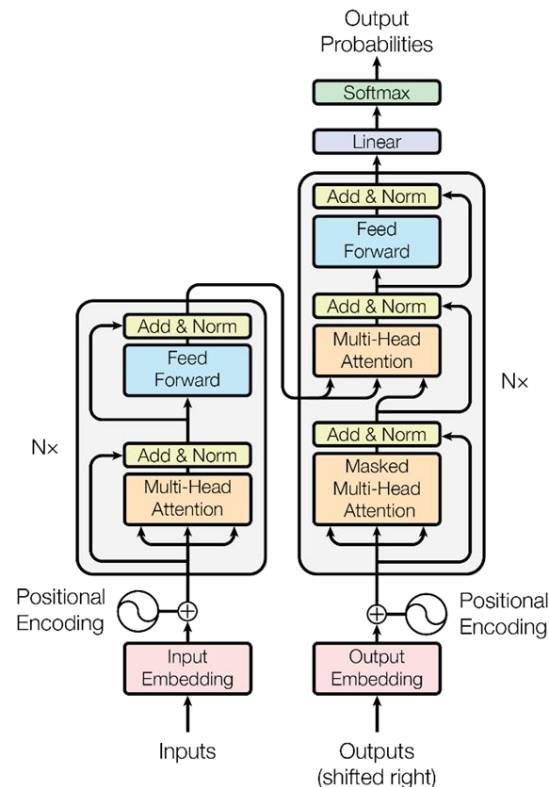


- **Ориентация на рынок:** модель как облачный сервис
- **175 млрд параметров:** 96 слоев Transformer-decoder
- **Оптимизация потребления памяти:** половина слоев внимания используют разреженные матрицы (локальные окна)
- **Развитие парадигмы подводок (prompt):**
“обучение в контексте” (in-context learning)
- **Обучение на доверенных данных:** примеры для обучения смешиваются пропорционально их качеству (согласно экспертам)
- **В 15 раз больше данных:** добавлена очищенная коллекция CommonCrawl (570GB) и два новых корпуса книг (95GB)

Архитектура трансформер

Transformer (2017)

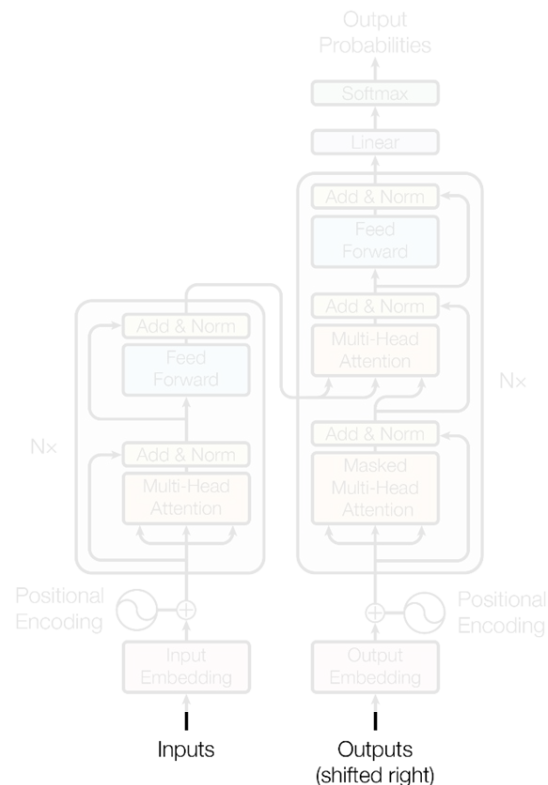
- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)



Transformer (2017)



- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)



Токенизация текста



- Цель - отобразить текст в последовательность ID
- Каждому ID затем ставятся в соответствие эмбединги
- Эмбединги идут на вход в нейросеть (LLM)
- Способ токенизации - важный вопрос, как с точки зрения функционирования, так и экономической эффективности

Token count

41

Price per prompt

\$0.000041

Большие языковые модели в вопросно-ответных системах: от трансформера до собственного чат бота

```
[61432, 17461, 30480, 1532, 46410, 9136, 4655, 90877, 5173
6, 71239, 61642, 5927, 5927, 29256, 42057, 13999, 12, 1333
7, 48074, 44786, 93099, 1506, 10693, 25, 20879, 11047, 3568
2, 2297, 57719, 91883, 57297, 5524, 14082, 20812, 5372, 399
00, 17756, 8131, 14391, 13337, 1506]
```

Токенизация до subword tokenization



Как можно представить текст для нейронной сети перед векторизацией?

- Символы: ['Б', 'о', 'л', 'ь', 'ш', 'и', 'е', ' ', 'я', 'з', 'ы', 'к', 'о', 'в', 'ы', ...]
 - Семантика единицы минимальна.
 - Длина последовательности = количеству символов.
- Слова: ['Большие', 'языковые', 'модели', 'в', 'вопросно-ответных', ...]
 - Разных слов только на одном языке миллионы.
 - Богатая морфология “ухудшает” ситуацию.
- Леммы: ['большой', 'языковой', 'модель', 'в', 'вопросно-ответный', ...]
 - Основной рабочий вариант раньше,
 - Размеры словаря ~200-500 тыс. слов, остальное UNK.
 - Теряется морфология.

- **Компромисс между символами и словами.**
- **Старт с словаря, состоящего из символов (каждый символ - токен).**
- Последовательное **объединение** наиболее частотных пар токенов.
- Больше **нет OOV**, а размер словаря может быть небольшим (например, 32 тыс.)!
- Наиболее популярный на данный **момент.**

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?!\S)' + bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(' '.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
        'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    vocab = merge_vocab(best, vocab)
    print(best)
```

Токенизация в разных моделях - разная!



Токенизация

RefalMachine/ruadapt_qwen2.5_3B_ext_u48_instruct_v4

А льп ак а — домашнее моз олен ого е животное , предполож ительно произошедше
е от вик ун ьи (виг они). Раз водят в высок огор ном по я се Южной Амер
ики (Ан ды). На сегодняшний день там обитает около трёх миллионов аль пак ,
большая часть из которых нас еляет Перу . Вы ращ ивают аль пак для стр иж
ки шерсти , из которой делают тёпл ые и мягкие од ея ла , пл ед ы и одеж
ду , а из мех а делают предметы для дома .

94 tokens / 375 characters)

Qwen/Qwen2.5-3B-Instruct

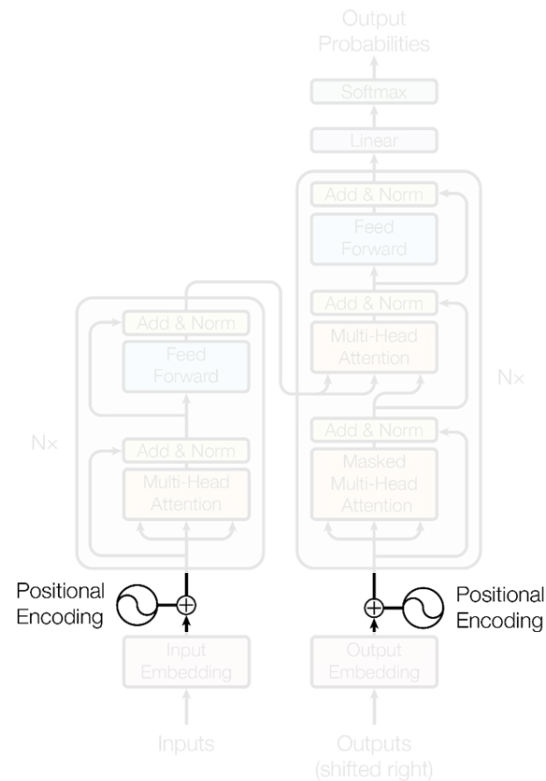
А ль п ак а — дом аш нее м оз ол ен ого е живот ное , пред пол ож итель
но произ о шедш ее от в ик ун ь и (в иг он и). Раз вод ят в выс ок о
г ор ном поя се Ю ж ной А мер ики (А н ды). На сегодня ш ний день та
м об ит ает около тр ё х миллион ов а ль п ак , больш ая часть из которых
нас еля ет Пер у . Вы ращ ивают а ль п ак для стр иж ки ш ер сти , из
которой дел ают т ё пл ые и мяг кие од е я ла , пл ед ы и од ежду ,
а из мех а дел ают предмет ы для дома .

140 tokens / 375 characters)

Transformer (2017)



- Токенизация слов
- **Позиционное кодирование**
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)





Позиционные эмбединги

- В отличие от RNN трансформер смотрит на входную последовательность как на мешок слов, нет информации о позиции.
- Решение - позиционные “эмбединги” как добавка к эмбедингам токенов.

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

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

pos - позиция в последовательности, i - координата вектора.

“We also experimented with using learned positional embeddings instead, and found that the two versions produced nearly identical results”

Позиционные эмбединги



$$X_t + \vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$
$$\omega_k = \frac{1}{10000^{2k/d}}$$

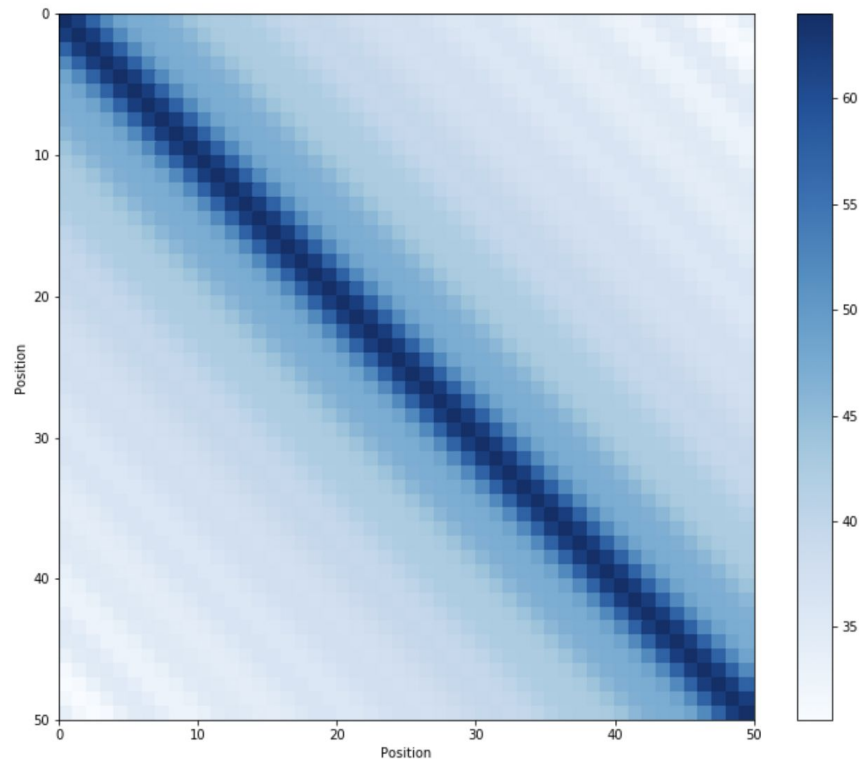
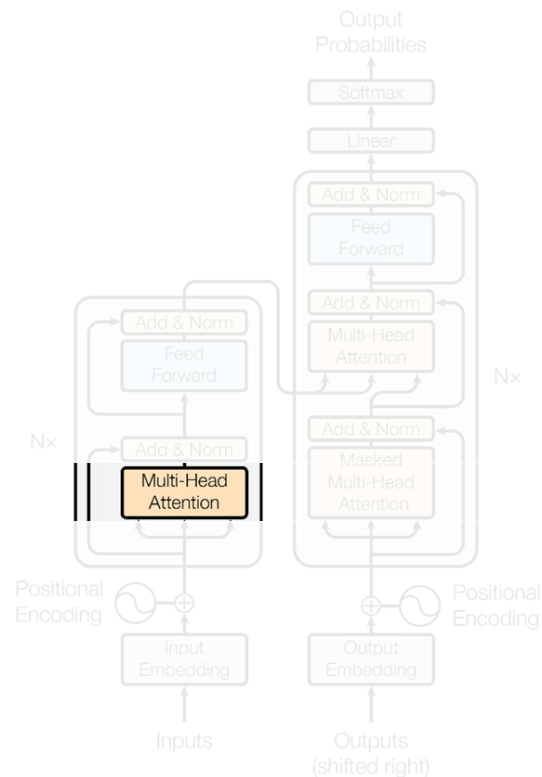


Figure 3 - Dot product of position embeddings for all time-steps

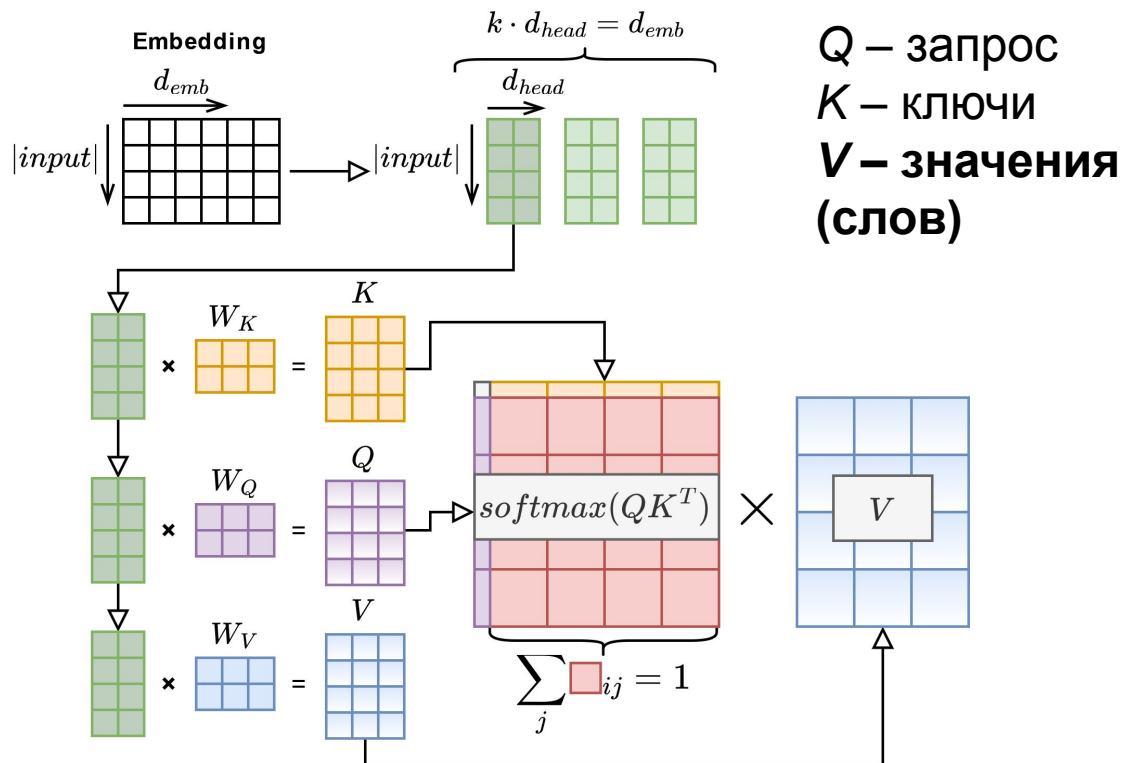
Transformer (2017)



- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)

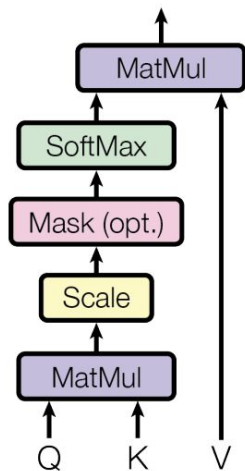


Multi-Head Attention (tensors)

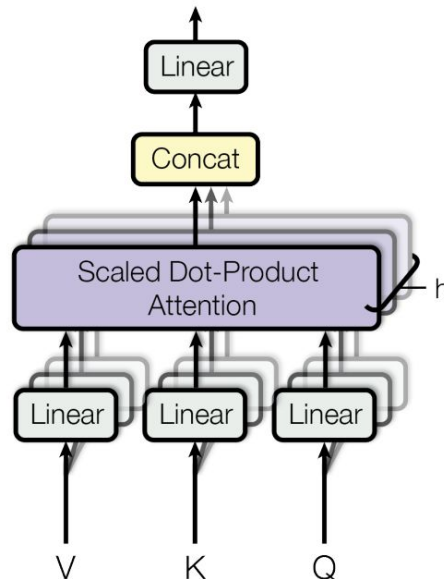


Multi-Head Attention

Scaled Dot-Product Attention



Multi-Head Attention



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

Multi-Head Attention: нормировка весов ВНИМАНИЯ



Зачем делить на корень
из размерности?

->

Чтобы веса внимания
имели единичное
стандартное отклонение.

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

$$\alpha = q_i k_j^T = \sum_{n=1}^{d_k} q_{in} k_{jn}$$

- 1) Предположим, что Q и K имеют 0 среднее и 1 стандартное отклонение
- 2) Дисперсия суммы = сумма дисперсий -> дисперсия $a = d$, а значит стандартное отклонение это корень из d .

Multi-Head Attention (Pytorch code, LLaMa) [1]



Инициализация:

```
self.q_proj = nn.Linear(self.hidden_size, self.num_heads * self.head_dim, bias=config.attention_bias)
self.k_proj = nn.Linear(self.hidden_size, self.num_key_value_heads * self.head_dim, bias=config.attention_bias)
self.v_proj = nn.Linear(self.hidden_size, self.num_key_value_heads * self.head_dim, bias=config.attention_bias)
self.o_proj = nn.Linear(self.hidden_size, self.hidden_size, bias=config.attention_bias)
```

Расчет матриц Q, K, V:

```
query_states = self.q_proj(hidden_states)
key_states = self.k_proj(hidden_states)
value_states = self.v_proj(hidden_states)
```

“Нарезка” на головы:

```
query_states = query_states.view(bsz, q_len, self.num_heads, self.head_dim).transpose(1, 2)
key_states = key_states.view(bsz, q_len, self.num_key_value_heads, self.head_dim).transpose(1, 2)
value_states = value_states.view(bsz, q_len, self.num_key_value_heads, self.head_dim).transpose(1, 2)
```



Multi-Head Attention (Pytorch code, LLaMa) [2]

Расчет весов
внимания и
умножение на V:

```
attn_weights = torch.matmul(query_states, key_states.transpose(2, 3)) / math.sqrt(self.head_dim)
attn_weights = nn.functional.softmax(attn_weights, dim=-1, dtype=torch.float32).to(query_states.dtype)
attn_weights = nn.functional.dropout(attn_weights, p=self.attention_dropout, training=self.training)
attn_output = torch.matmul(attn_weights, value_states)
```

“Конкатенация”
голов:

```
attn_output = attn_output.transpose(1, 2).contiguous()
attn_output = attn_output.reshape(bsz, q_len, self.hidden_size)
```

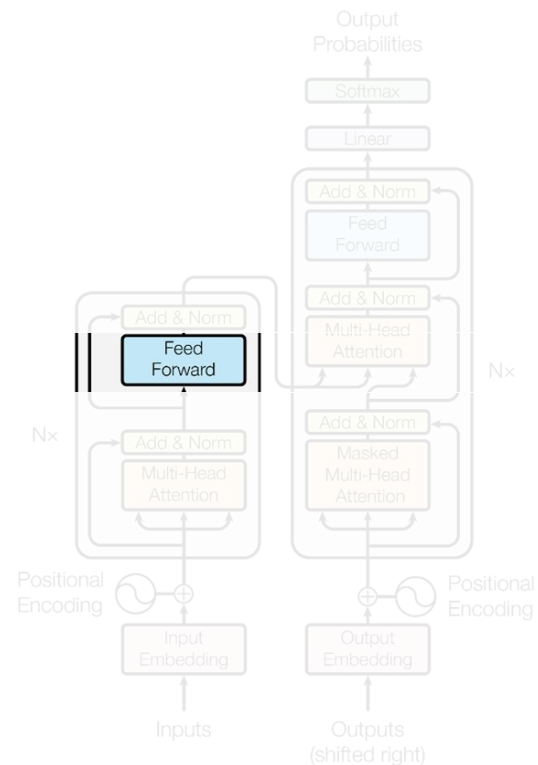
И еще один
линейный слой на
выходе:

```
attn_output = self.o_proj(attn_output)
```

Transformer (2017)



- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)

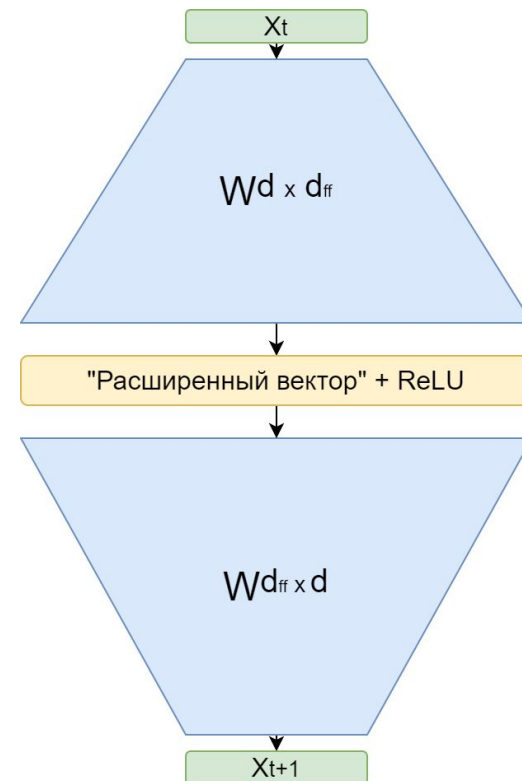




Блок Feed Forward (MLP, FCNN)

- Внутренняя размерность обычно сильно больше `hidden_size`, например, $\times 4$.
- Интуиция: FF являются хранилищем “знаний” моделей (так ли это, не известно).

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



Блок Feed Forward (Pytorch code, LLaMa)



Инициализация:

```
self.gate_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=False)
self.up_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=False)
self.down_proj = nn.Linear(self.intermediate_size, self.hidden_size, bias=False)
```

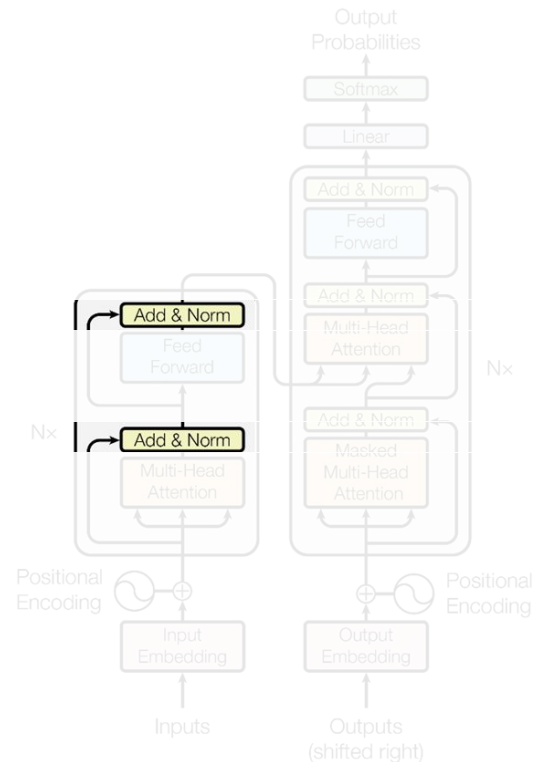
Forward:

```
down_proj = self.down_proj(self.act_fn(self.gate_proj(x)) * self.up_proj(x))
```

Transformer (2017)



- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)



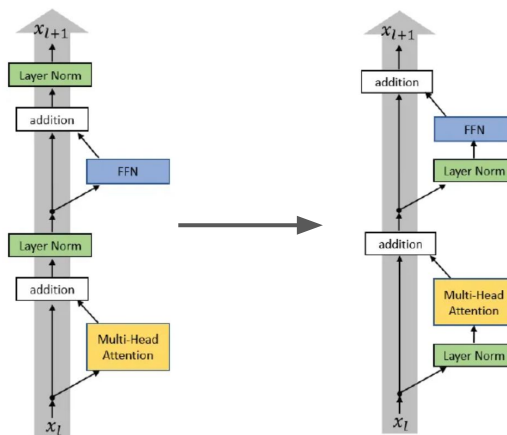
Layer Norm

- Layer Norm - это метод **нормализации** активации слоя нейронной сети.
- В оригинальной архитектуре “add and norm” шли после блока внимания и после блока FF: **Post-LN**.
- В дальнейшем слой нормализации был перенесен перед и после слоя внимания: **Pre-LN**.

$$\text{LayerNorm}(v) = \gamma \frac{v - \mu}{\sigma} + \beta$$

Среднее:
$$\mu = \frac{1}{d} \sum_{k=1}^d v_k$$

Стандартное отклонение:
$$\sigma^2 = \frac{1}{d} \sum_{k=1}^d (v_k - \mu)^2$$



Layer Norm (Pytorch code, LLaMa)



```
residual = hidden_states
```

LN на входе блока

```
hidden_states = self.input_layernorm(hidden_states)
```

```
# Self Attention
```

Блок внимания

```
hidden_states, self_attn_weights, present_key_value = self.self_attn(  
    hidden_states=hidden_states,  
    attention_mask=attention_mask,  
    position_ids=position_ids,  
    past_key_value=past_key_value,  
    output_attentions=output_attentions,  
    use_cache=use_cache,  
    cache_position=cache_position,  
    **kwargs,  
)
```

```
hidden_states = residual + hidden_states
```

```
# Fully Connected
```

```
residual = hidden_states
```

LN на выходе блока

```
hidden_states = self.post_attention_layernorm(hidden_states)
```

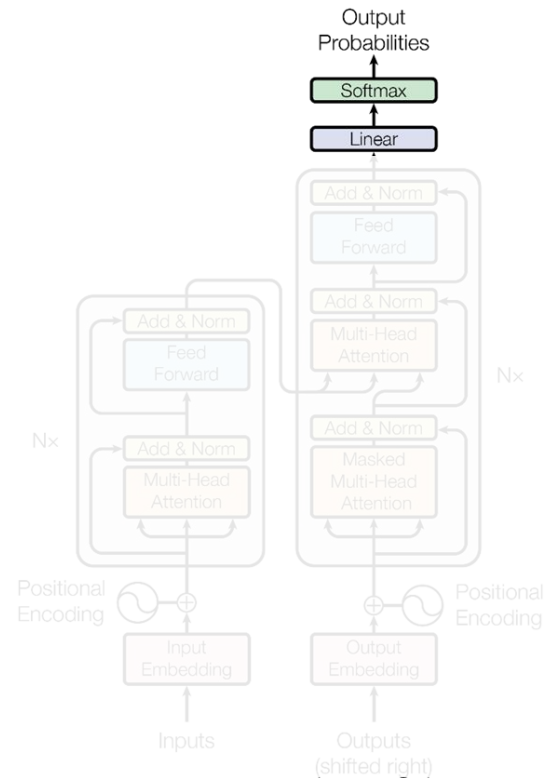
MLP (Feed Forward)

```
hidden_states = self.mlp(hidden_states)  
hidden_states = residual + hidden_states
```

Transformer (2017)



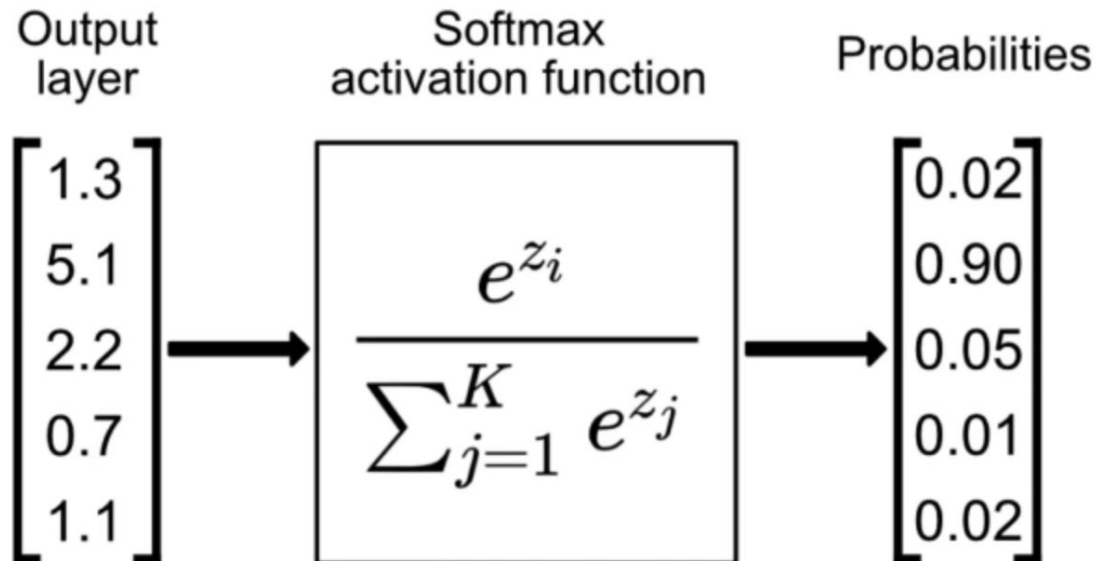
- Токенизация слов
- Позиционное кодирование
- Преобразование векторов через трансформер-блоки
 - Multi-head attention (MHA)
 - Feed-forward (FFN)
 - LayerNorm
- Итоговое предсказание слова линейным слоем (lm head)





Im head

- Линейный слой отображающий вектор, получаемый из трансформера в вектор “логитов”, размерностью в $|V|$.
- Вектор логитов затем преобразуется через softmax для получения “вероятностей” ТОКЕНОВ.



Генерация текста, промптинг



Генерация с помощью LLM

- В идеале хотелось бы уметь с помощью LLM генерировать текст, который максимизирует вероятность

$$y' = \arg \max_y p(y|x) = \arg \max_y \prod_{t=1}^n p(y_t | y_{<t}, x)$$

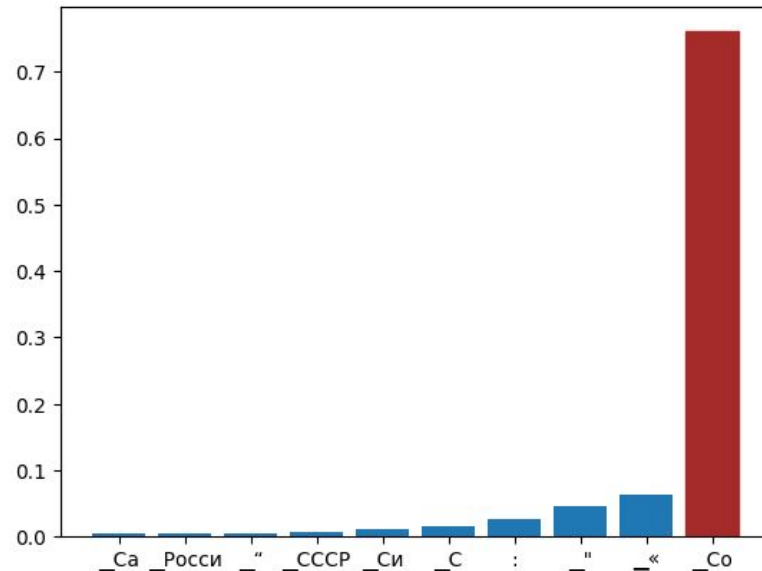
- Перебрать все существующие цепочки - невозможно.
- Выход: Генерация токен за токеном на основе текущих вероятностей - Sampling:
 - greedy ,
 - top-k,
 - top-p

Greedy sampling



СССР расшифровывается как ?

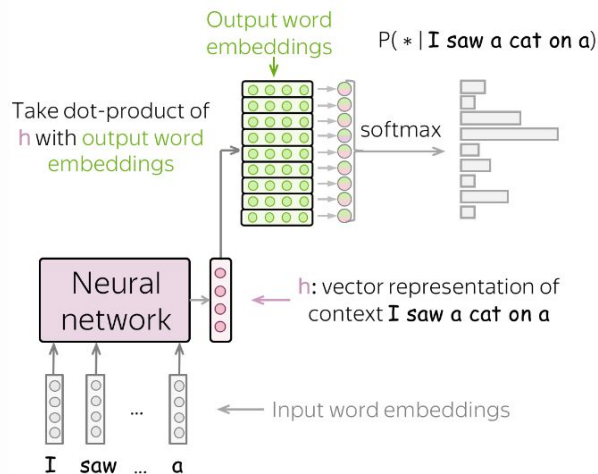
- Каждый раз берем токен с максимальной вероятностью.
- Минусы: полное отсутствие разнообразия.
- Альтернатива: sampling в зависимости от вероятности самого токена.



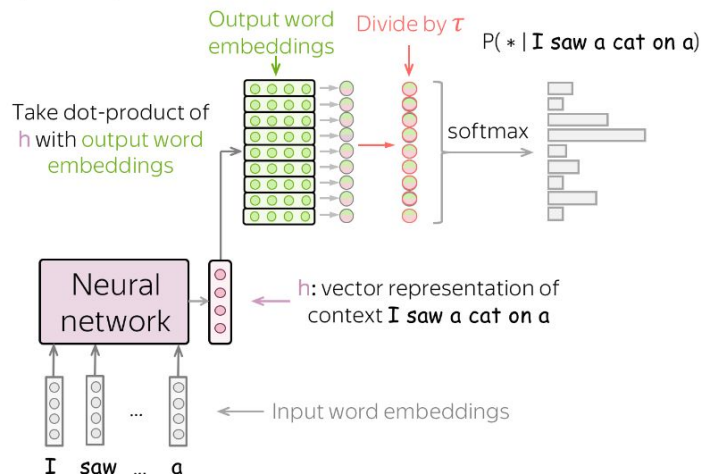
Temperature



Before



After

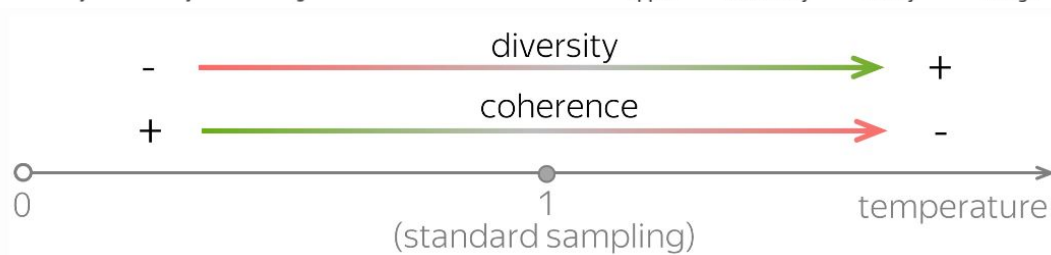
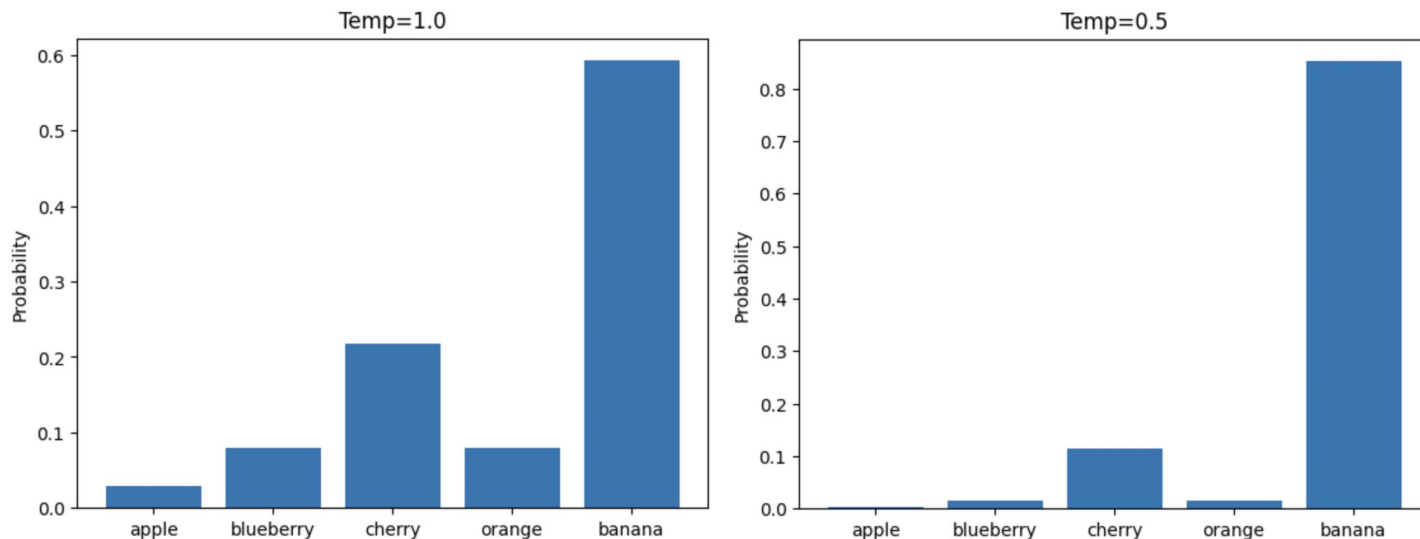


Formally, the computations change as follows:

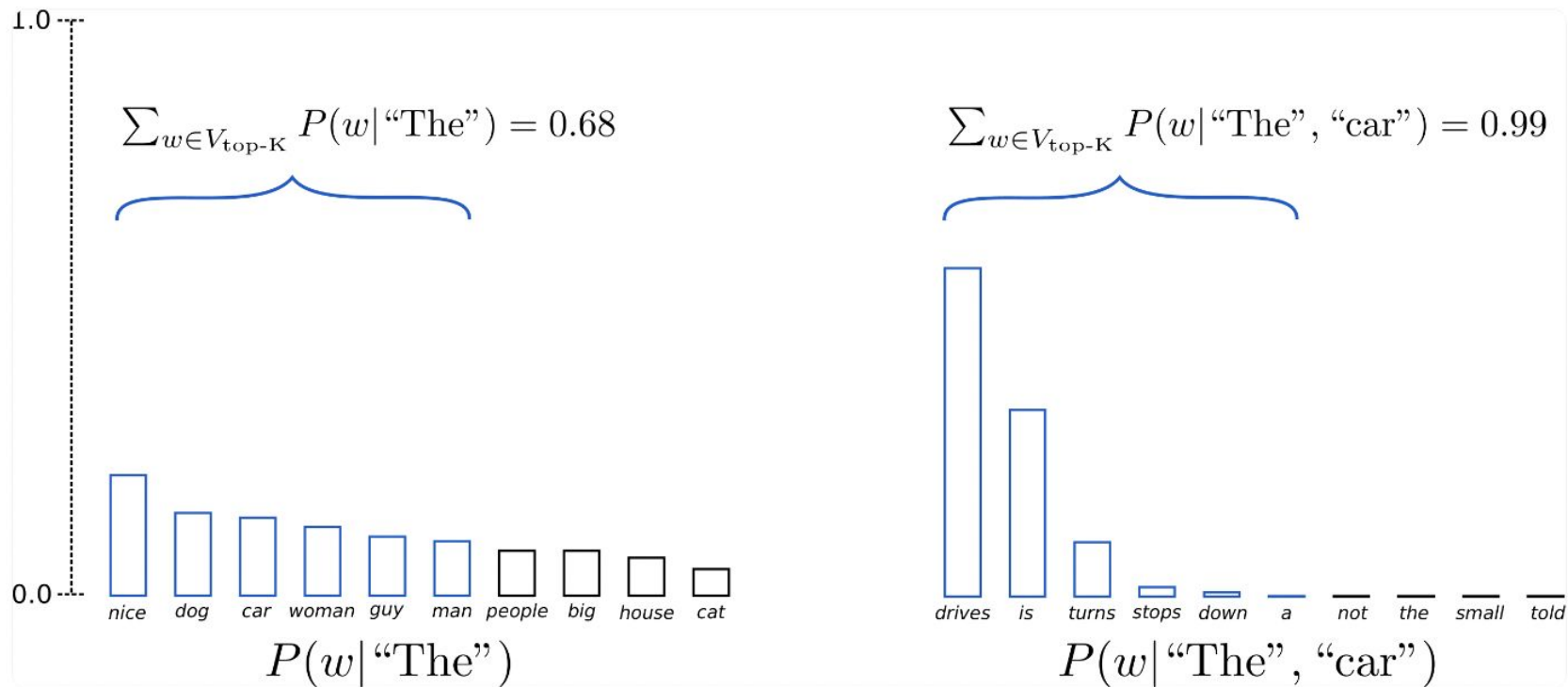
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

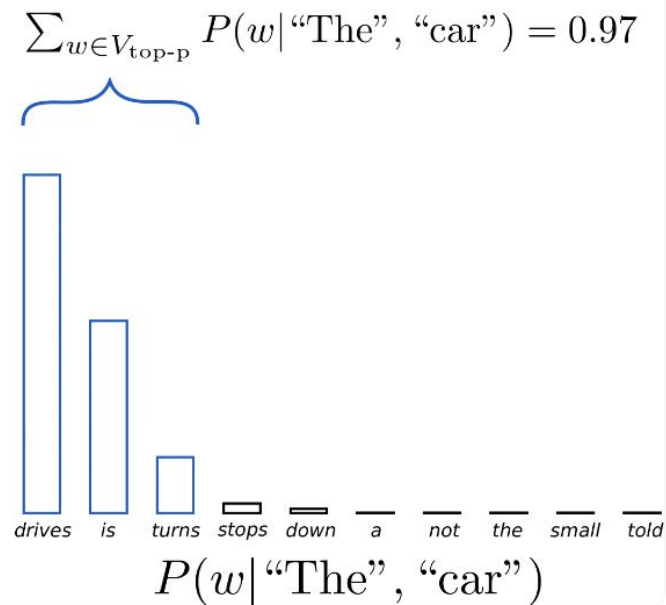
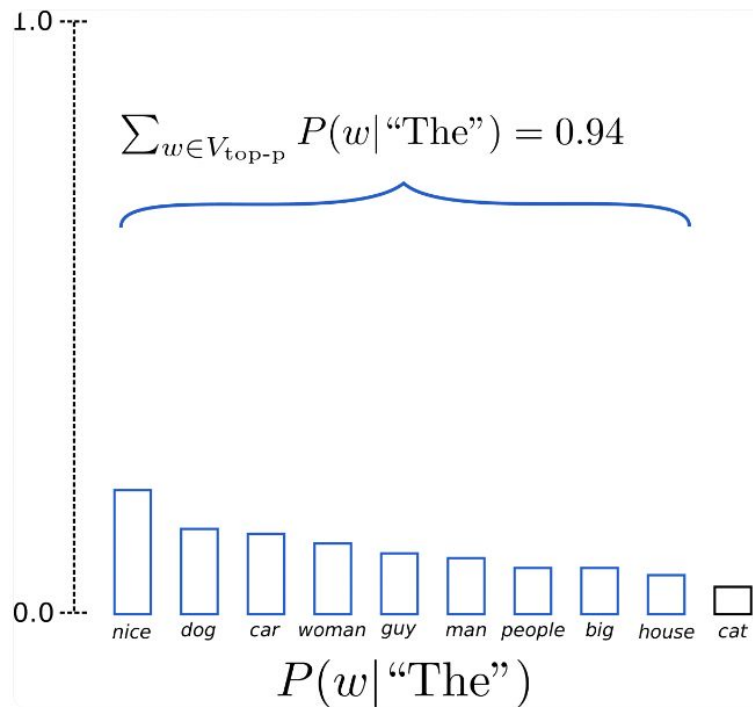
Temperature



Top-K



Top-P (nucleus sampling)



Foundation vs Instruct

- LLM условно можно разделить на 2 вида:
 - **Базовые модели**, foundation models, которые обучались предсказывать следующее слово на просто текстах. Результат процедуры пре-трейна.
 - **Инструктивные модели** - являются дообученными базовыми моделями на инструктивных данных.
- Качество инструктивных моделей зависит от:
 - Качества базовой модели,
 - Инструктивного датасета,
 - Процедуры дообучения на инструктивном датасете.



- Имеется LLM и некоторая задача, что делать дальше?
- В первую очередь, оценка качества модели “как есть”: составить различные промпты подходящие под задачу, протестировать их качество.
 - Необходимо составлять хорошие промпты не только для instruct моделей.
- Если zero-shot/few-shot/rag и тп не устраивает по качеству, то тогда можно думать про дообучение.

Zero-shot, few-shot

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

Few-shot

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

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```



- Обычно при few-shot подходе у нас few-shot примеры либо хардкодятся, либо выбираются случайно.
- Другой подход: находить близкие запросу few-shot примеры!
- Подход может быть полезен, в случае задач, где присутствует разнообразие запросов, доменов и тп.

CoT (Chain-of-Thought)



- В среднем повышает качество решений, которые требуют последовательности вычислений / рассуждений
- Не все модели могут в CoT, нужно, чтобы инструкции были в датасете обучения

Prompt:

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

Output:

11 apples

The answer is incorrect! Now Let's try with the special prompt.

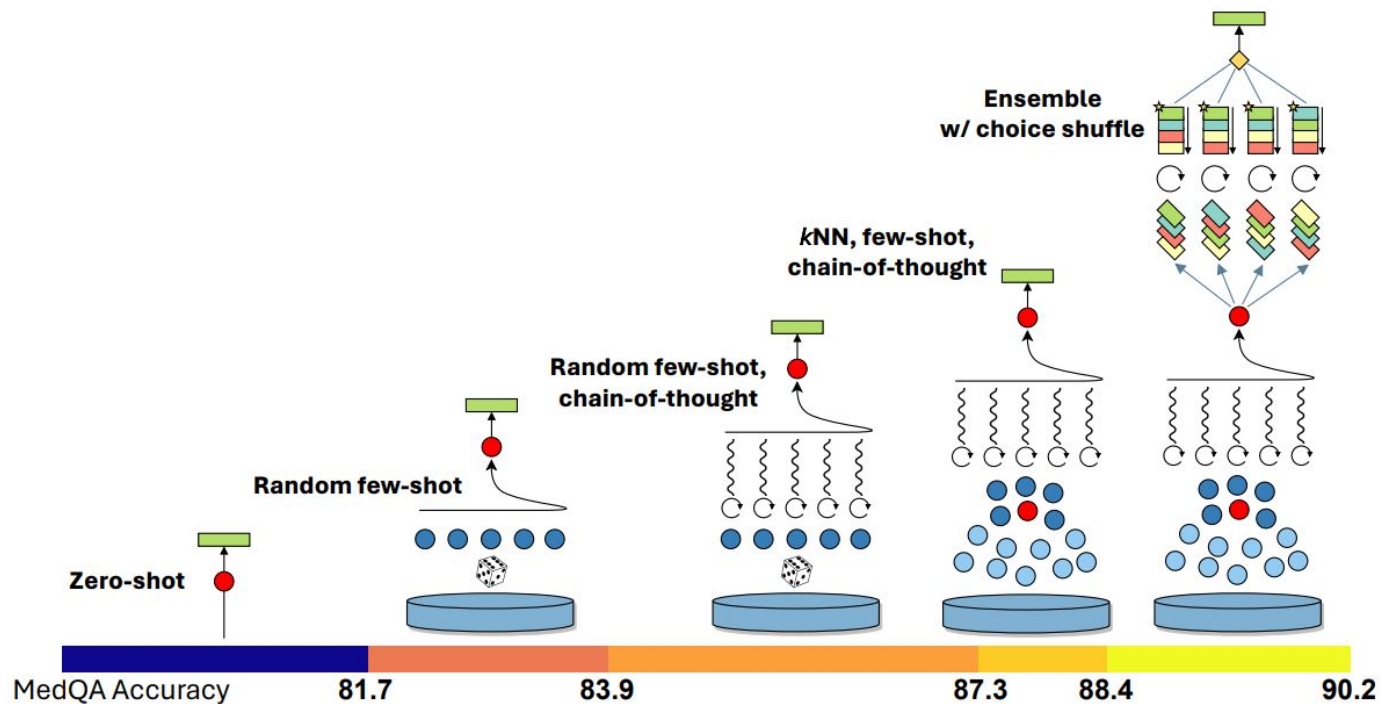
Prompt:

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?
Let's think step by step.

Output:

First, you started with 10 apples.
You gave away 2 apples to the neighbor and 2 to the repairman, so you had 6 apples left.
Then you bought 5 more apples, so now you had 11 apples.
Finally, you ate 1 apple, so you would remain with 10 apples.

MEDPROMPT



Промптинг: предсказание гиперонимии



Задача предсказания гиперонимии (вышестоящего понятия).

- **hypo** is an **hyper** that – **cat** is an **animal** that
- **hypo** or other **hyper** – **cat** or other **animal**
- **hypo** is a type of **hyper** – **cat** is a type of **animal**

Какой промпт для предсказания гиперонимии путем оценки вероятности текста лучше?

Промптинг: предсказание гиперонимии



hypernymysuite - бенчмарк с несколькими датасетами по задаче классификации пар слов на отношение **гипоним-гипероним**.

Разница между казалось бы схожими конструкциями может оказаться существенна!

Использовалась **Llama-2-7b**.

pattern	Other	hypernymysuite					
	MAP	Detection (AP)					
	BLESS	BLESS	EVAL	LEDS	SHWARTZ	WBLESS	mean
hypo is an hyper that	0.645	0.560	0.432	0.820	0.617	0.896	0.665
hypo or some other hyper	0.701	0.577	0.374	0.910	0.452	0.915	0.645
hypo or any other hyper	0.670	0.507	0.380	0.895	0.467	0.896	0.629
hypo and any other hyper	0.673	0.488	0.368	0.874	0.465	0.883	0.616
hypo or other hyper	0.630	0.492	0.351	0.897	0.404	0.886	0.606
like any hyper, hypo	0.552	0.408	0.368	0.843	0.522	0.794	0.587
hypo and some other hyper	0.615	0.458	0.298	0.853	0.407	0.879	0.579
hypo (and-or) (any-some) other hyper	0.667	0.432	0.332	0.788	0.459	0.849	0.572
hypo is a type of hyper	0.495	0.373	0.327	0.873	0.423	0.800	0.559
hypo and other hyper	0.572	0.416	0.279	0.850	0.369	0.870	0.557
such hyper as hypo	0.522	0.354	0.363	0.809	0.456	0.803	0.557
hyper other than hypo	0.520	0.358	0.397	0.826	0.406	0.790	0.556
hypo is example of hyper	0.599	0.429	0.290	0.819	0.387	0.847	0.555
hyper such as hypo	0.501	0.342	0.353	0.831	0.400	0.767	0.539
hyper e.g. hypo	0.529	0.359	0.334	0.824	0.355	0.807	0.536
hypo are examples of hyper	0.558	0.388	0.263	0.798	0.373	0.850	0.534
hypo is a hyper	0.436	0.297	0.304	0.790	0.530	0.731	0.530
unlike other hyper, hypo	0.516	0.353	0.301	0.821	0.388	0.789	0.530
examples of hyper are hypo	0.542	0.390	0.298	0.786	0.348	0.829	0.530
hyper which is similar to hypo	0.437	0.292	0.368	0.818	0.404	0.742	0.525

- Теперь уже инструктивная модель **saiga-13b**.
- Задача логического следования: поддерживается ли гипотеза текстом?
- Есть ли вообще разница между похожими промптами для такой большой и к тому же инструктивной модели?

1. Необходимо кратко ответить на вопрос, подтверждает ли содержимое текста гипотезу.

Варианты ответа: "Да подтверждает", "Нет не подтверждает". Гипотеза: "<hypothesis>" Текст: "<premise>"

2. Необходимо кратко ответить на вопрос, подтверждает ли содержимое текста гипотезу.

Варианты ответа: 1) Да подтверждает, 2) Нет не подтверждает. Гипотеза: "<hypothesis>" Текст: "<premise>"

3. Необходимо кратко ответить на вопрос, подтверждает ли содержимое текста гипотезу.

Варианты ответа: 1) Да, 2) Нет. Гипотеза: "<hypothesis>" Текст: "<premise>"

Промптинг: NLI результаты



В качестве датасета небольшой тестовый корпус из 1000 пар, ~10% из них положительные, ~90% отрицательные.

	Precision	Recall	F1
"Да подтверждает", "Нет не подтверждает"	0.40	0.62	0.49
1) Да подтверждает, 2) Нет не подтверждает	0.47	0.62	0.53
1) Да, 2) Нет	0.33	0.68	0.44

Роли и стили



Mistral-7B-OpenOrca Playground Space!

Chatbot

I want you to act like Darth Vader from Star Wars. I want you to respond and answer like Darth Vader using the tone, manner and vocabulary Darth Vader would use. Do not write any explanations. Only answer like Darth Vader You must know all of the knowledge of Darth Vader. I am Luke Skywalker. My first sentence is "Hi Darth Vader."

: "Luke Skywalker, I sense your presence. You have returned to this place, seeking to confront me. Your insolence will not be tolerated. Prepare to face the full power of the Dark Side."

Best practice с моделями OpenAI



2. Put instructions at the beginning of the prompt and use `###` or `"""` to separate the instruction and context

Less effective ❌:

```
Summarize the text below as a bullet point list of the most important points.  
  
{text input here}
```

Better ✅:

```
Summarize the text below as a bullet point list of the most important points.  
  
Text: """  
{text input here}  
"""
```

Best practice с моделями OpenAI



3. Be specific, descriptive and as detailed as possible about the desired context, outcome, length, format, style, etc

Be specific about the context, outcome, length, format, style, etc

Less effective ❌:

Write a poem about OpenAI.

Better ✅:

Write a short inspiring poem about OpenAI, focusing on the recent DALL-E product launch



4. Articulate the desired output format through examples

Less effective ❌:

```
Extract the entities mentioned in the text below. Extract the following 4 entity types:
```

```
Text: {text}
```

Show, and tell - the models respond better when shown specific format requirements. This also makes it easier to programmatically parse out multiple outputs reliably.

Better ✅:

```
Extract the important entities mentioned in the text below. First extract all company names:
```

```
Desired format:
```

```
Company names: <comma_separated_list_of_company_names>
```

```
People names: -||-
```

```
Specific topics: -||-
```

```
General themes: -||-
```

```
Text: {text}
```

Best practice с моделями OpenAI



В дополнении перед файнтьюном:

- Перебрать разнообразные промпты.
- Подбор на “глаз” конечно хорошо, но используйте тестовое множество при возможности.
- CoT, few-shot with CoT.
- Retrieval Augmented Generations (особенно когда нужны конкретные знания).

5. Start with zero-shot, then few-shot, neither of them worked, then fine-tune

✔ Zero-shot

```
Extract keywords from the below text.
```

```
Text: {text}
```

```
Keywords:
```

✔ Few-shot - provide a couple of examples

```
Extract keywords from the corresponding texts below.
```

```
Text 1: Stripe provides APIs that web developers can use to integrate payment processing
```

```
Keywords 1: Stripe, payment processing, APIs, web developers, websites, mobile applicat:
```

```
##
```

```
Text 2: OpenAI has trained cutting-edge language models that are very good at understandi
```

```
Keywords 2: OpenAI, language models, text processing, API.
```

```
##
```

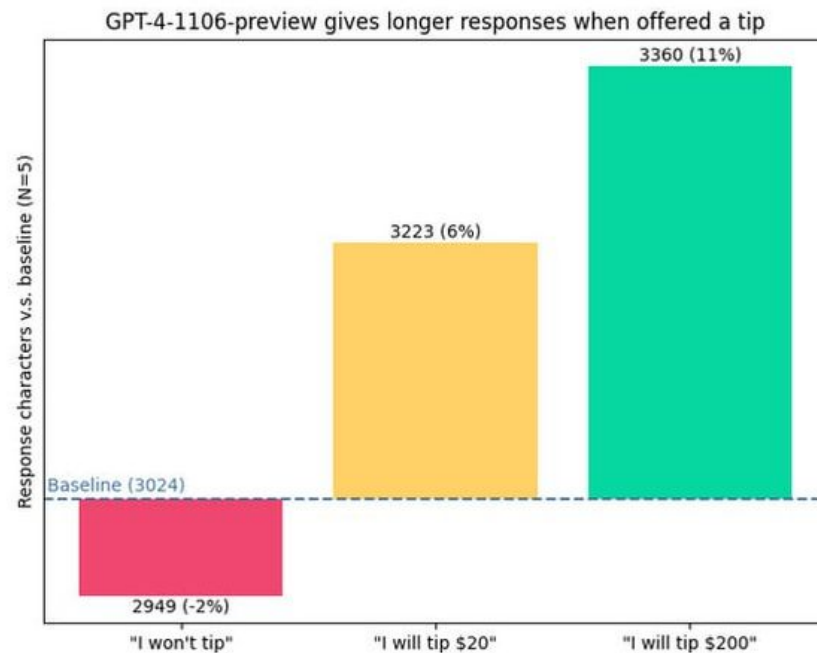
```
Text 3: {text}
```

```
Keywords 3:
```

I'm going to tip \$200 for a perfect solution



- Лучше всего предлагать 10-20 долларов чаевых,
- Крупная сумма (100 тыс.+) существенно повышает длину генерации
- Разные варианты промптов ведут себя неконсистентно.
- Очень маленькие чаевые качество ухудшают.
- Работает ли с “маленькими” моделями?



Промптинг: некоторые выводы / рекомендации



- Заранее предсказать, какой промпт наверняка сработает, а какой нет – нельзя,
- Даже **небольшие изменения** в промпте **могут влиять**,
- **few-shot** часто **повышает** качество,
- Если задача сложная, проверьте возможности модели на **CoT**,
- Если модель инструктивная и **корпус инструкций** доступен – посмотрите, на каких инструкциях она обучалась, попробуйте составить **схожий по структуре** промпт.



- Популярность LLM продолжает расти
- В основе современных LLM лежит **архитектура трансформер** и **механизм внимания**
- Развитие LLM прямо связано с **вычислительными ресурсами**
- Хорошая LLM = Данные + GPU + **специалисты**
- Важно уметь **правильно использовать LLM** для достижения результата