

Глубинное обучение

Лекция 8: Трансформеры

Лектор: Антон Осокин

Лаборатория компании Яндекс, ФКН ВШЭ

Yandex Research

Научные стажировки: <https://cs.hse.ru/big-data/yandexlab/internship>

ФКН ВШЭ, 2021

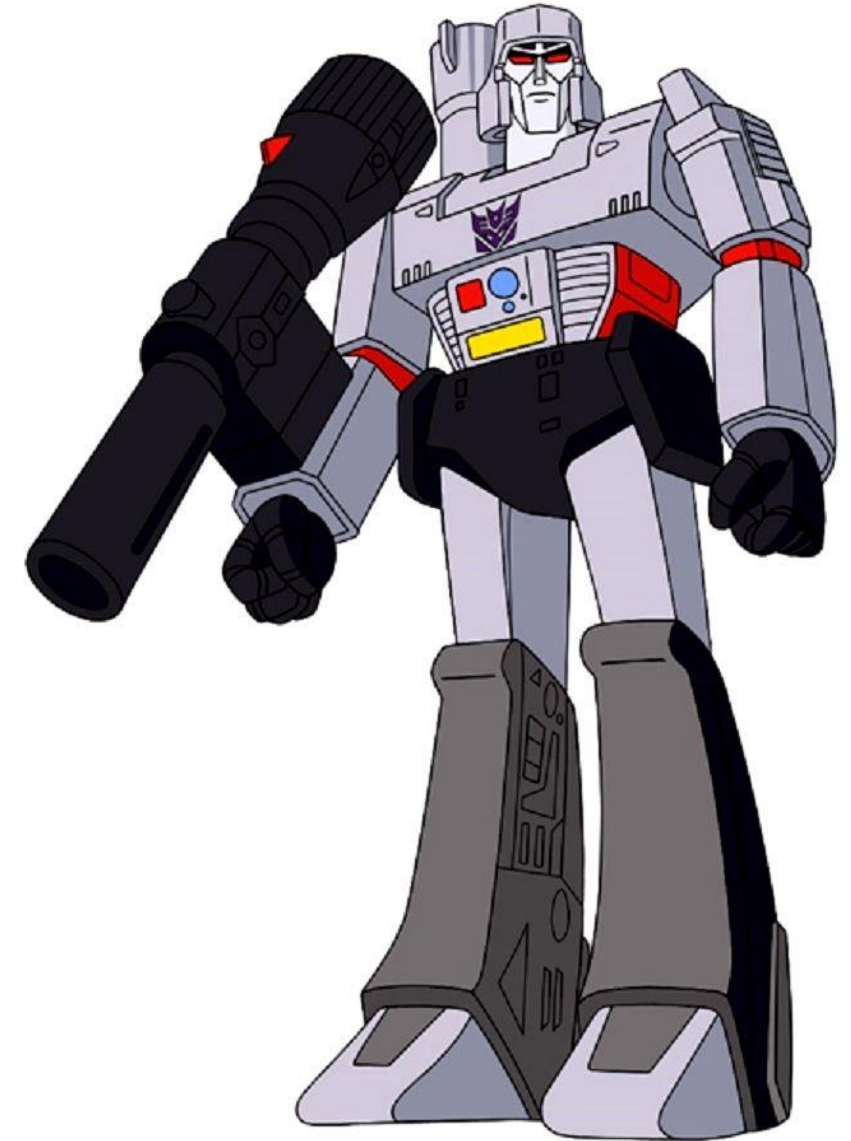
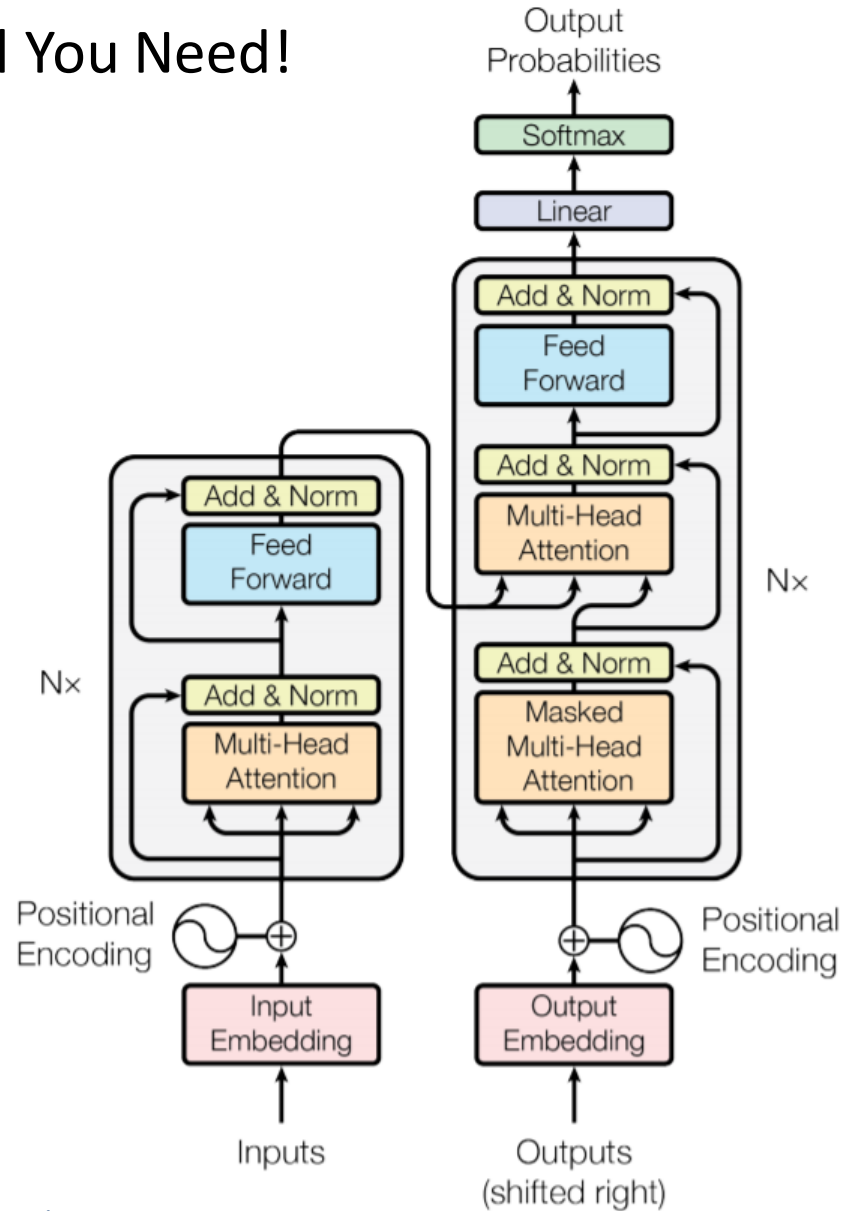


НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Трансформер

[Vaswani et al., 2017]

- Статья – Attention Is All You Need!



Transformer

[Vaswani et al., 2017]

- Статья – Attention Is All You Need!
- Основной блок трансформера:
 - Self attention (Q = query, K = key, V = value, d_k = dimension)

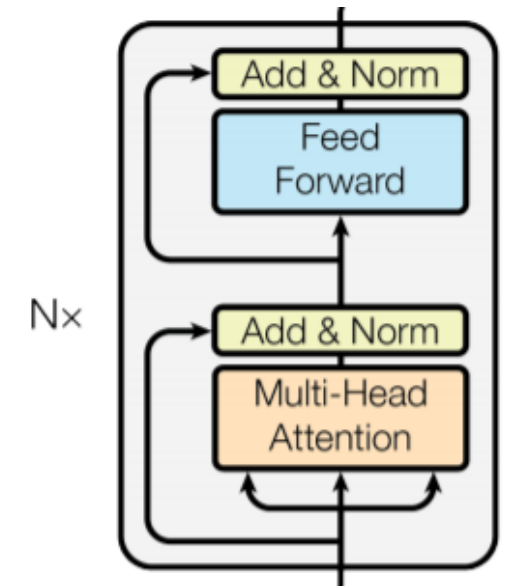
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Multi-head self-attention (N = num layers, h = num heads)

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

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

- 2-layer NN with ReLu
- $\text{LayerNorm}(x + \text{Sublayer}(x))$



Transformer

[Vaswani et al., 2017]

- Статья – Attention Is All You Need!
- Основной блок трансформера
- Кодирование входа:
 - Представление токенов (обычные представления)
 - Представление позиции:
 - Словарь представлений позиций
 - Частотное кодирование ($d_{model}=512$):

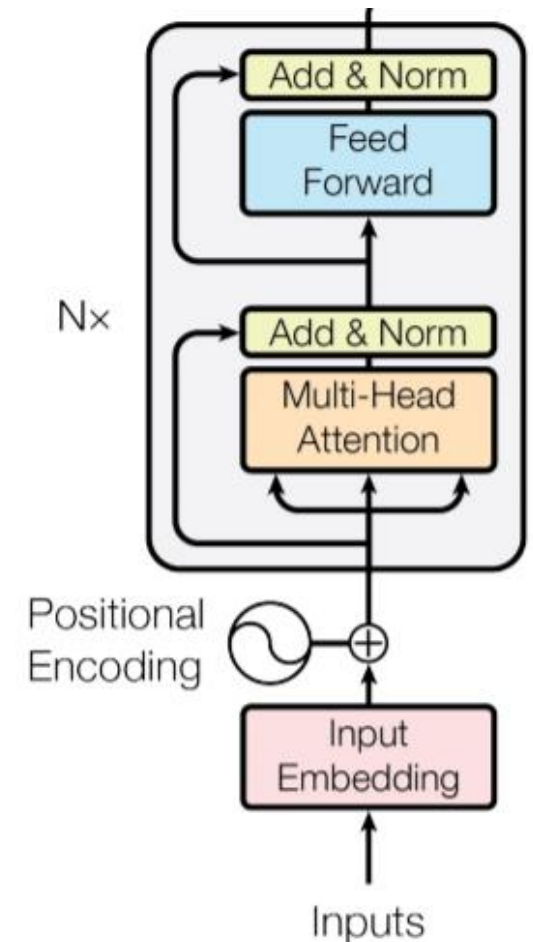
$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

- Relative Position Encoding – встраивается в self attention
 - Кодировывает пару позиций

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^n \exp e_{ik}} \quad z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V)$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$
$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V + a_{ij}^V)$$

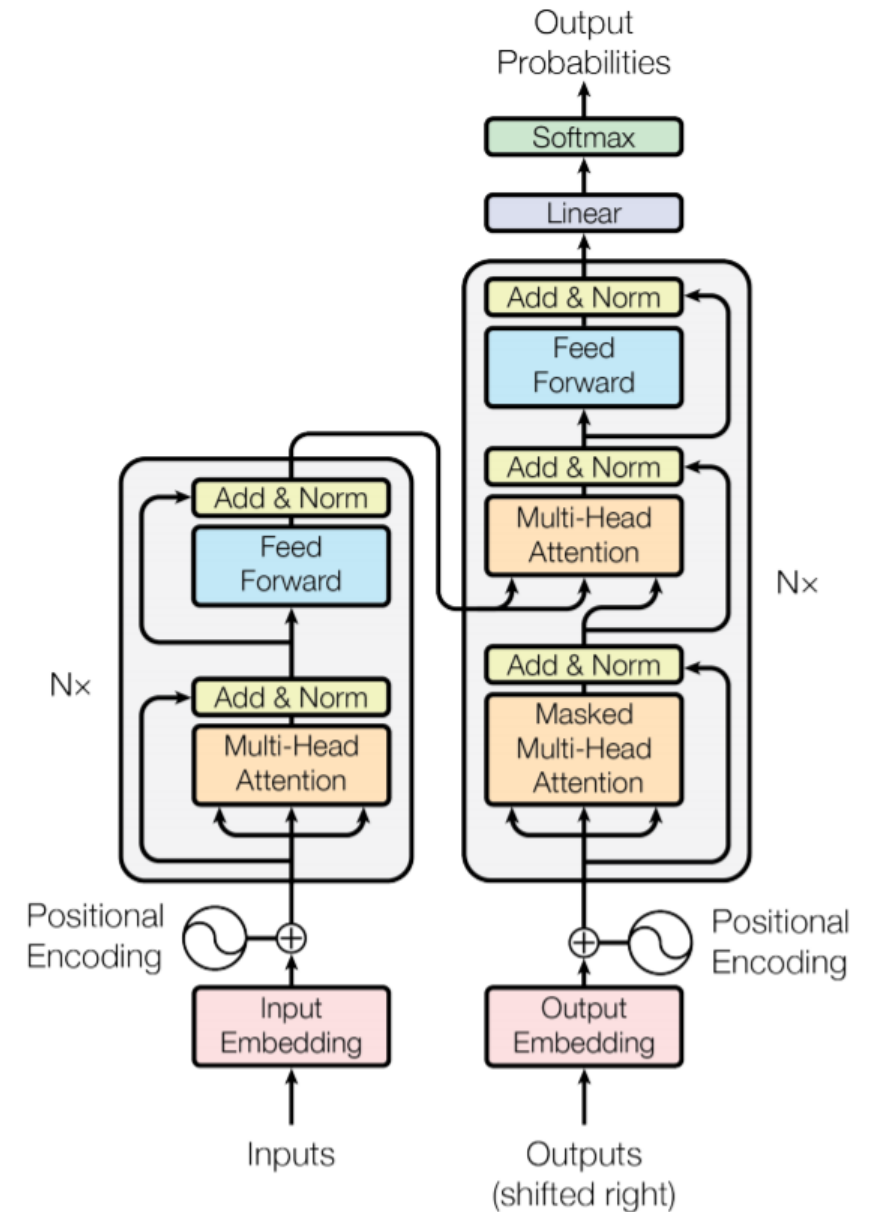


[Shaw et al., 2018]

Transformer

[Vaswani et al., 2017]

- Статья – Attention Is All You Need!
- Основной блок трансформера
- Кодирование входа
 - Представление входа и номера позиции
- Decoder трансформера
 - Авторегрессионное предсказание
 - Masked self-attention и attention на энкодер
 - На обучении маски в self-attention



Обучение трансформеров

- Adam с расписанием рамера шага
 - Линейный разогрев
 - Далее затухание
- Достаточно большой размер батча
 - Несколько GPU
 - Последовательное вычисление подбатчей
- Label smoothing
 - Log loss с таргетом $(1-\alpha, \frac{\alpha}{d-1}, \dots, \frac{\alpha}{d-1})$ вместо $(1, 0, \dots, 0)$

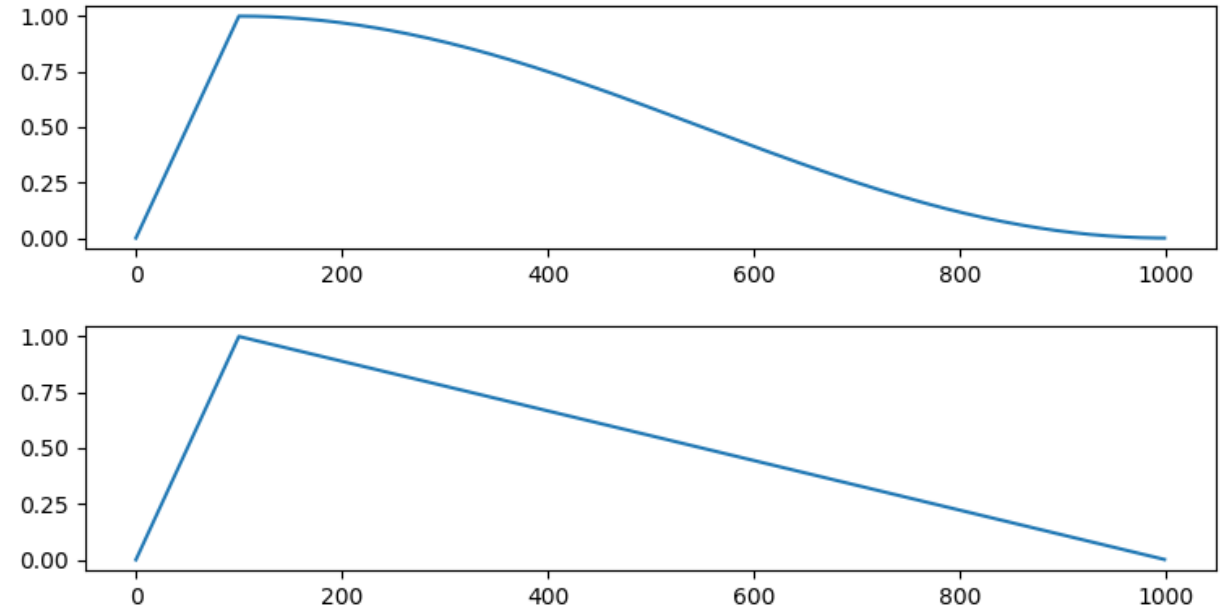


image credit: huggingface.co

Предобученные представления слов – из языковых моделей!

Зачем нужны предобученные модели?

- Чтобы решать любую задачу!
- Способ по умолчанию: дообучаем на данные задачи
- Сравнение моделей:
 - Дообучать на набор разных задач и брать агрегированный результат (среднее 😞)
 - Бенчмарки: GLUE, SuperGLUE
 - Статические наборы данных приводят к переобучению сообщества и застою
 - Динамические во времени бенчмарки: [GEM](#) , [DynaBench](#)
- На разных задачах нужны разные модели
 - Очень много разных моделей
- Отличная библиотека:
 - <https://github.com/huggingface/transformers>



Виды языковых моделей

- Только encoder (ELMo, BERT, RoBERTa, etc.)
- Encoder-decoder (T5, BART, etc.)
- Только decoder – генеративные модели (GPT-X, Turing-NLG, Megatron LM)

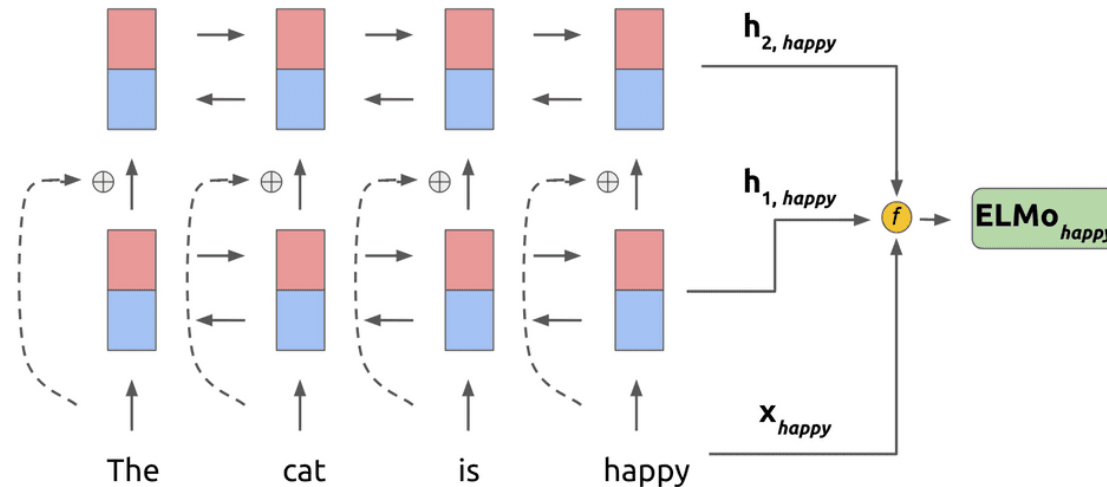
ELMo

[Peters et al., 2018; AllenNLP]
<https://github.com/allenai/allennlp>

- ELMo = Embeddings from Language Models
- Контекстно зависимые представления!
- Архитектура:
 - Представления слов = свертки поверх представлений символов
 - Легко обрабатывать слова вне словаря
 - 2 модели поверх – forward и backward (2-layer LSTM + skip con.)
 - Итоговое представление – линейная комбинация представлений слоев!



- Обучение:
 - Forward – след. слово
 - Backward – пред. слово



BERT

[Devlin et al., 2018; Google]



- BERT = Bidirectional Encoder Representations from Transformers
- Очень большой трансформер:
 - L – глубина, H – размерность внутри, A – число голов multi-head attention
 - BERT_{BASE}: L=12, H=768, A=12, Total Parameters=110M
 - BERT_{LARGE}: L=24, H=1024, A=16, Total Parameters=340M
- Обучение:
 - Masked LM: random 15% output tokens
 - Inputs:
 - 80% - [MASK]
 - 10% - исходное слово
 - 10% - случайное слово
 - Next sentence

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

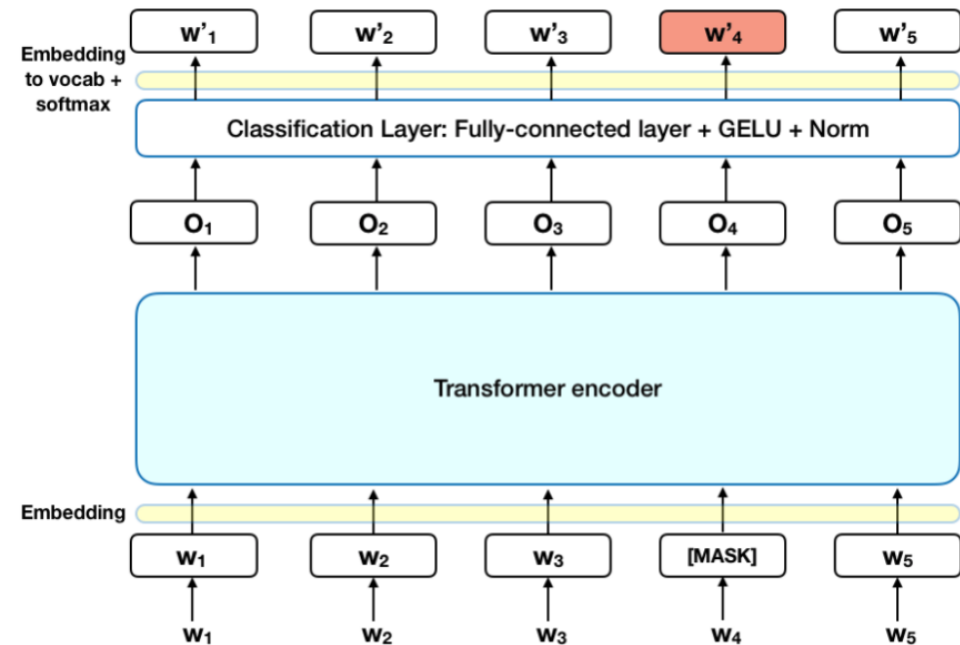


image credit: [Rani Horev](#)

RoBERTa – как BERT, но обычно лучше

[Liu et al., 2019; Facebook]

- BERT был существенно недообучен
 - RoBERTa: подбираем гиперпараметры и обучаем дольше
 - Большой батч
 - byte-level (а не character-level, unicode!) BPE
 - генерация масок не лету

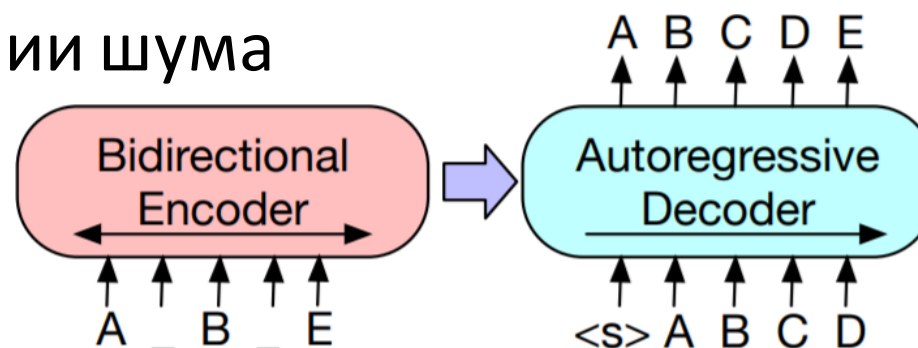
Модели меньшего размера

- ALBERT [Lan et al. 2019]: представления меньшей размерности, одни и те же параметры на разных слоях
- DistilBERT [Sanh et al. 2019]: дистилляция BERT в меньшую модель

Модели encoder-decoder

- BART: Bidirectional and Auto-Regressive Transformers [Lewis et al.; 2019]

- Обучается на восстановлении шума
- Шум со структурой



- T5: Text-to-Text Transfer Transformer [Raffel et al.; 2019]
- Проборовали много разных вариантов BERT функций потерь
- Очень много данных



Только decoder – генеративные модели

- Популяризованы семейством моделей GPT, GPT-2, GPT-3 (OpenAI)
 - Внедрялись архитектурные трюки
 - Больше данных
 - Больше размер моделей
 - Больше вычислений
 - GPT-3 – 175B параметров, обучение стоит миллионы \$
 - GPT-3 не выложена, используется в продуктах

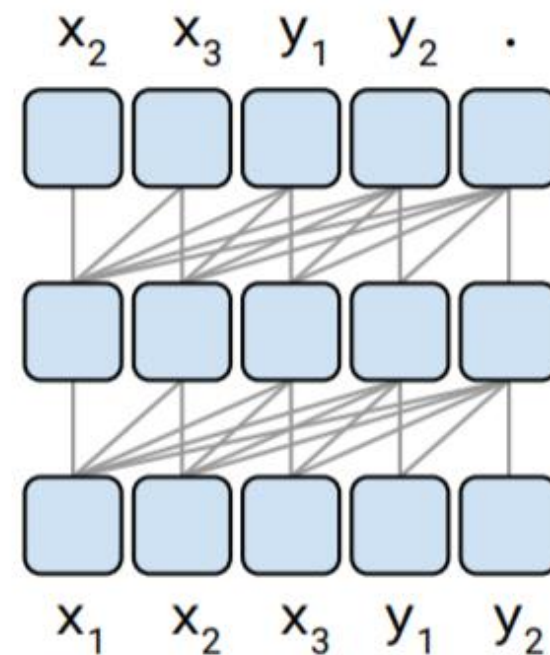


image credit: [Raffel et al.; 2019]

Mega Language Models

- Текущий чемпион (из публичных):
 - Megatron-Turing (Microsoft&NVIDIA)
 - 530 миллиардов параметров
 - Hardware and software
- Модели перестают быть для всех.
Теперь это IP компаний

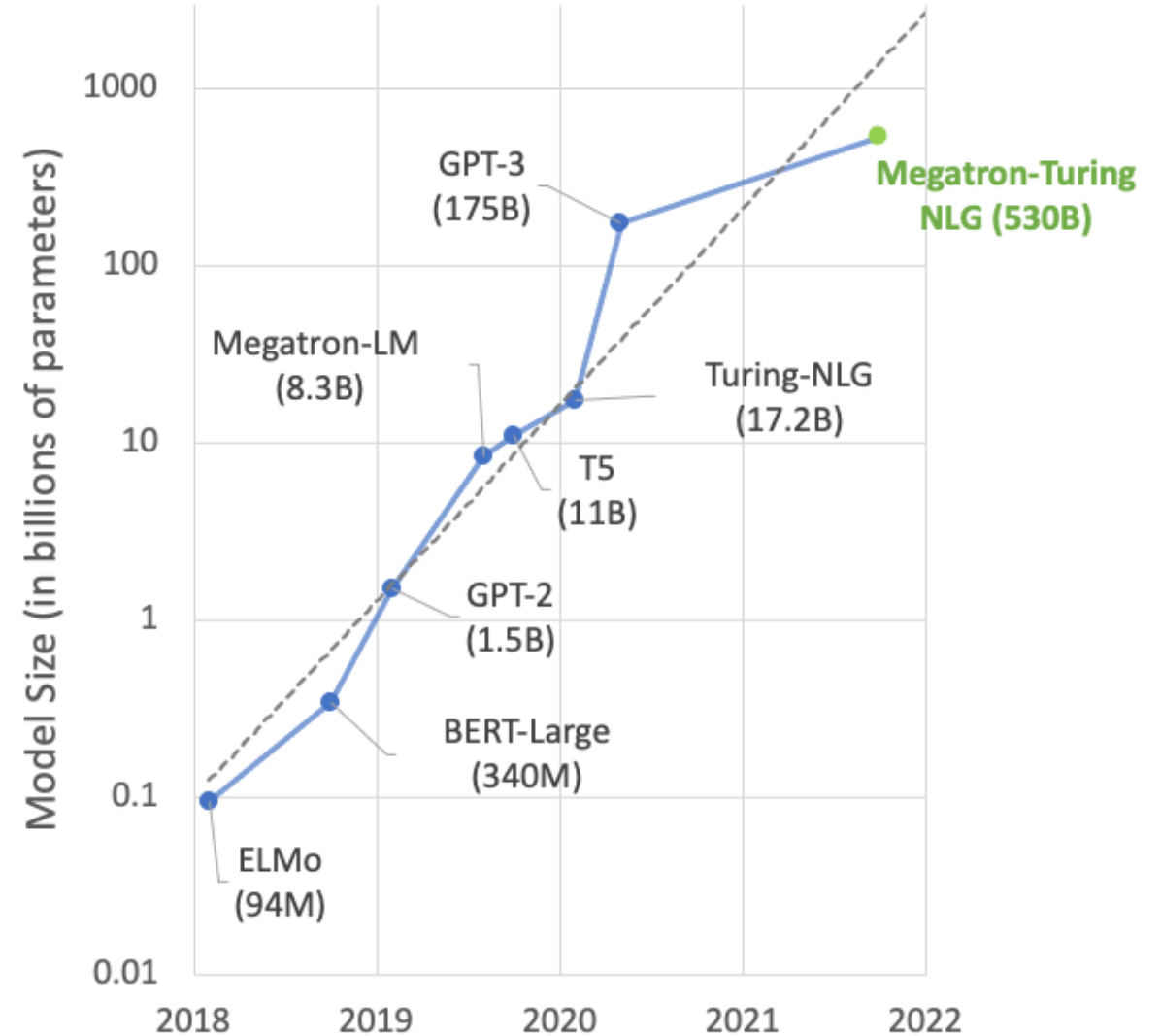


image credit: [NVIDIA blog](#)

GPT-3: использование без дообучения

[Brown et al., 2020; OpenAI]

- GPT «понимает» задачу по нескольким примерам
- Prompt engineering

Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for choosing me as your designer. I appreciate it.

Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.

Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

Poor English input: I'd be more than happy to work with you in another project.

Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.

Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.

Good English output: The patient died.

GPT-3: использование без дообучения

[Brown et al., 2020; OpenAI]

- GPT «понимает» задачу по нескольким примерам
- Prompt engineering

Create opposite sentences

This is good.

This is bad.

The good horse came home.

The bad horse came home.

The good horse with a large head came home.

The bad horse with a small head came home.

The black cat had a disappointing time.

The white cat had a good time.

When you jump high, you fall hard.

When you fall hard, you get hurt.

He ran fast, but he came last.

He ran slowly, but he came first.

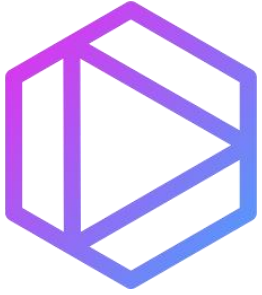
The book was huge, but the students finished it early.

The book was small, but the students finished it late.

image credit and more examples:
<https://gpt3experiments.substack.com/p/is-gpt-3-really-doing-few-shot-learning>

Генеративные модели для кода

- Tabnine

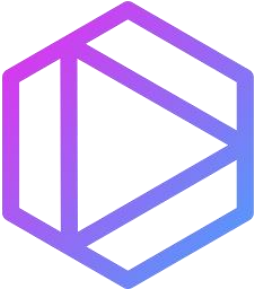


- GitHub Copilot / Open AI Codex [Chen et al., 2021; arXiv:2107.03374]



Автодополнение кода

- Tabnine



```
tabnine-test-2.py ×
Users > saneryee > Desktop > Medium > TabNine&Kate > tabnine-test-2.py
14     .... self.lstm1 = nn.LSTMCell(1, 51)
15     .... self.lstm2 = nn.LSTMCell(51, 51)
16     .... self.linear = nn.Linear(51, 1)
17     ....
18     def forward(self, input, future = 0):
19     .... outputs = []
20     .... h_t = torch.zeros(input.size(0), 51, dtype=torch.double)
21     .... c_t = torch.zeros([input.size(0),])
```

GitHub Copilot / Open AI Codex



```
1 import torch
2 from torchvision.datasets import ImageFolder
3 from torchvision import transforms,models
4 from torch import nn,optim
5
6 def finetune(folder, model):
7     """fine tune pytorch model using images from folder and report results on validation set"""
8     # build model
9     model = getattr(models,model)()
10    model.load_state_dict(torch.load(model))
11    model = model.to(device)
12
13    # load data
14    train_set = ImageFolder(folder,'train',transform=data_transforms['train'])
15    val_set = ImageFolder(folder,'val',transform=data_transforms['val'])
16    train_loader = torch.utils.data.DataLoader(train_set,batch_size=4,shuffle=True,num_workers=2)
17    val_loader = torch.utils.data.DataLoader(val_set,batch_size=4,shuffle=True,num_workers=2)
```

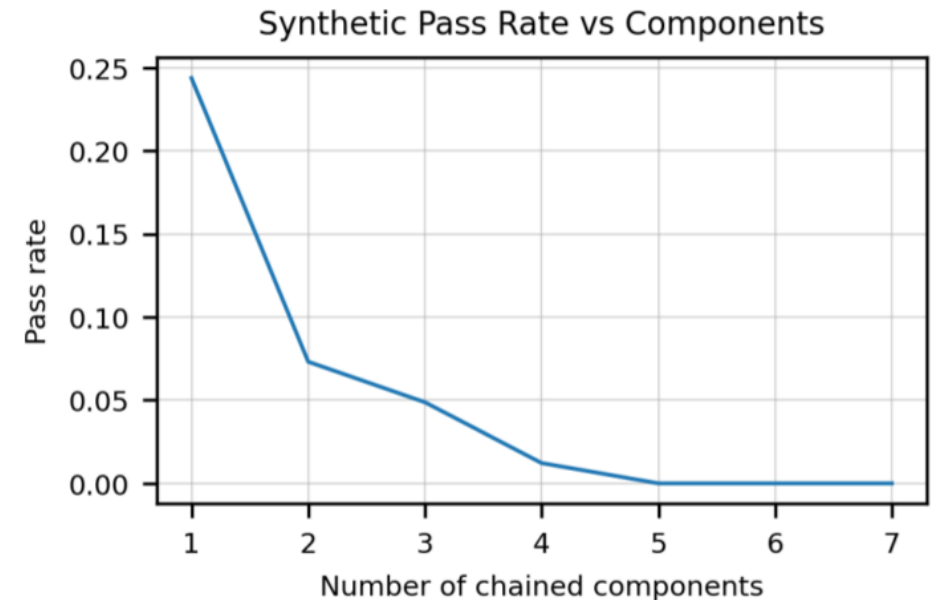
File context

Generated code
(89 lines)

Codex работает?

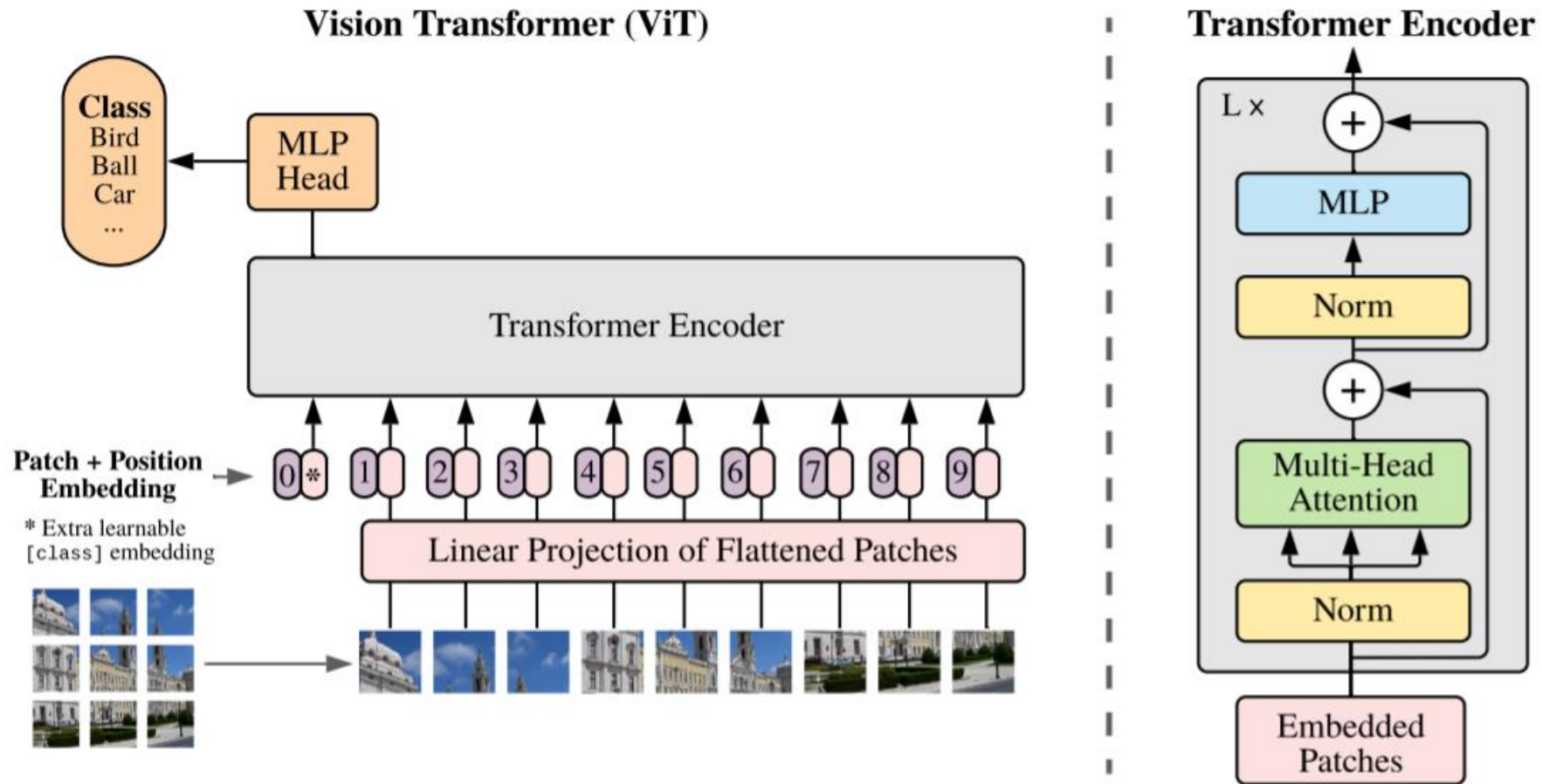
- Генерирует
- Результат похож на код и иногда работает
- Нет гарантий, могут быть сложные баги

```
def do_work(x, y, z, w):  
    """ Add 3 to y, then subtract 4  
    from both x and w. Return the  
    product of the four numbers. """  
    t = y + 3  
    u = x - 4  
    v = z * w  
    return v
```



Трансформеры для изображений: ViT

[Dosovitskiy et al., 2020; Google]



Трансформеры для изображений

- Горячая тема
- Пытаются применять во всех задачах, конкуренты сверток
- Классификация: Visual Transformer [Wu et al., 2020], DEiT [Touvron et al., 2021]
- Детекторы: DETR [Carion et al., 2020]; Deformable DETR [Zhu et al., 2020]
- Сегментация: DPT [Ranftl et al., 2021]; SWIN [Liu et al., 2021]
- Поза: PoseFormer [Zheng et al., 2021]; TransPose [Yang et al., 2021]
- Действия на видео: Video Transformer [Neimark et al., 2021]

Заключение

- Трансформеры окупировали NLP
- Предобученные представления – очень важно
- Отличная библиотека “клей”:
 - <https://github.com/huggingface/transformers>
- Трансформеры приходят и в другие области
 - Обработка кода
 - Компьютерное зрение

