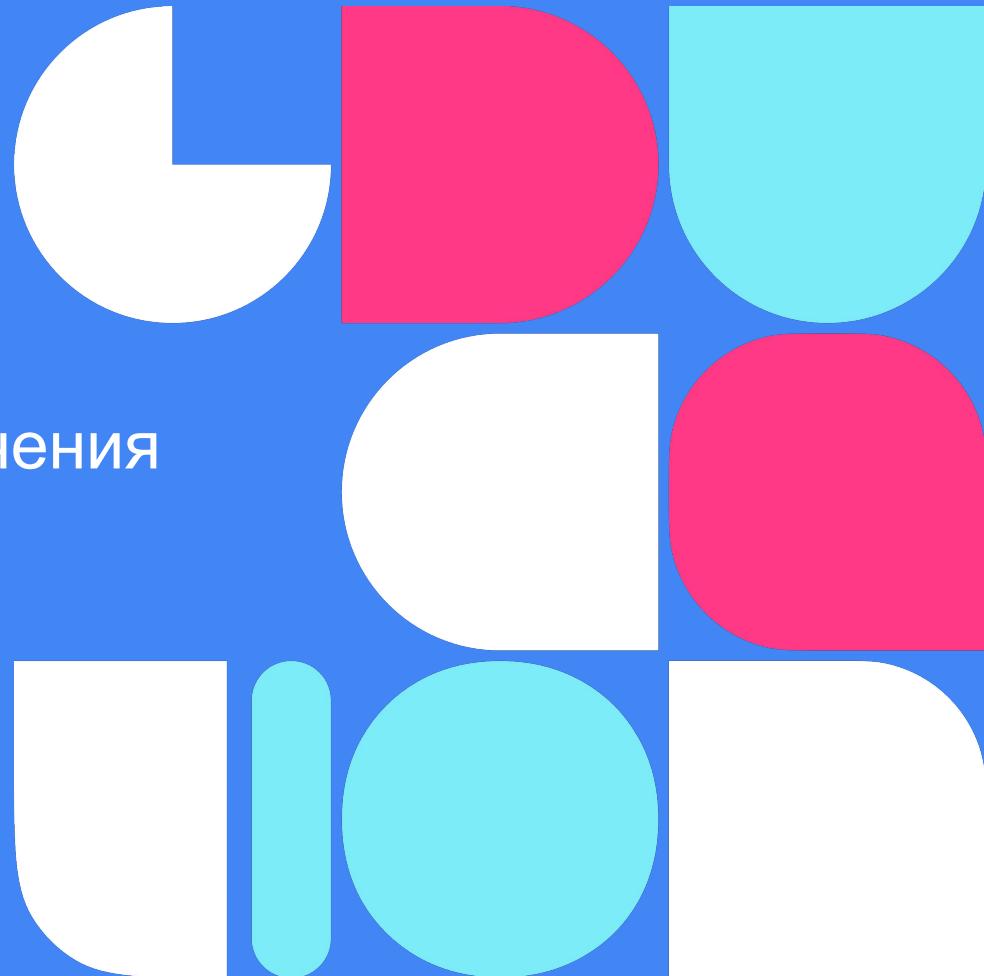


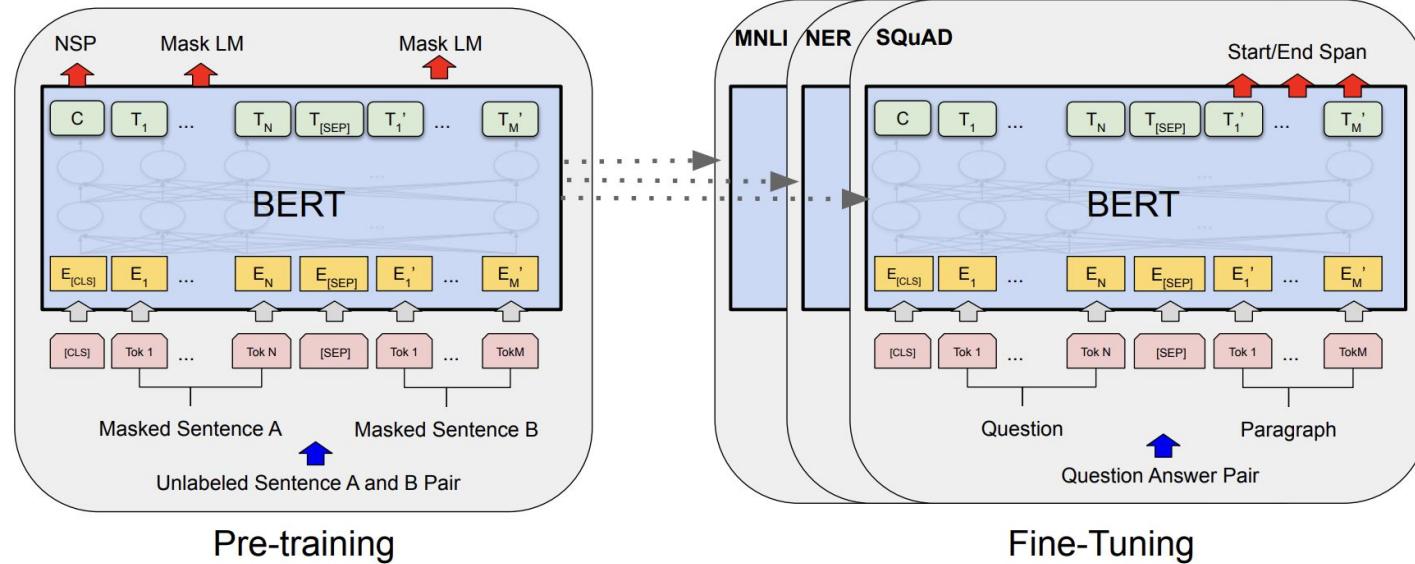


Лекция 4. Данные для обучения LLM.

Анисимова Мария
23.10.2024



Как и на какие задачи учился BERT?



Этапы обучения трансформера.



Pretrain (unsupervised) - вкладываем в модель знания о языке, мире.



Supervised Fine-Tuning - обучаем под down-stream задачи.

Pretrain datasets.

От BERT до GPT.

Предобучение BERT-like.

- Только на Википедии учить LM не стоит – тексты “однообразные”
- Новостные статьи – на практике хороший источник
- GPT-1
 - BookCorpus (16 GB)
- BERT
 - BookCorpus (16 GB)
 - English Wikipedia (< 1 GB)
- RoBERTa
 - BookCorpus (16 GB)
 - CC-News – прочистили CommonCrawl News (76 GB)
 - OpenWebText – Reddit с \geq 3 лайками (38 GB)
 - Stories – тексты из CommonCrawl в формате историй (31GB)

Предобучение T5.

T5 - Encoder-Decoder model.

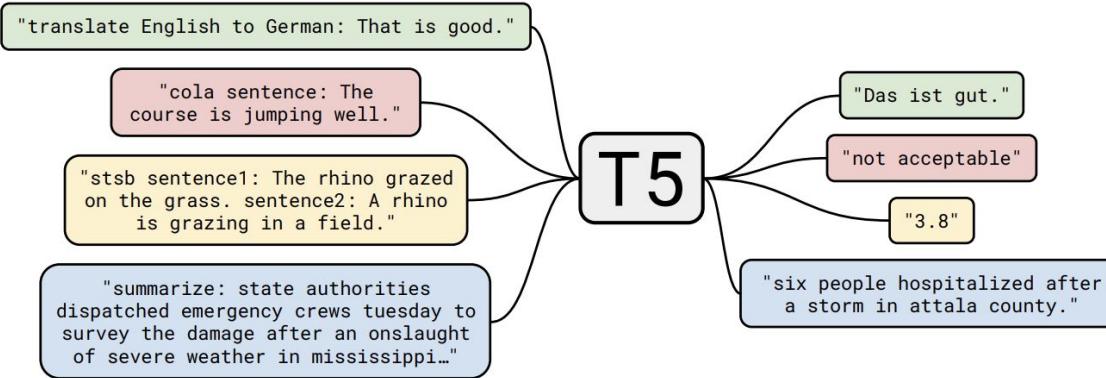


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. “T5” refers to our model, which we dub the **“Text-to-Text Transfer Transformer”**.

Предобучение Т5. Датасет C4.

C4: “**Colossal Clean Crawled Corpus**” – очищенный Common Crawl (обкачанный и переведенный в текст веб-архив)

Методы фильтрации:

- Есть завершающая пунктуация на конце текста
- ≥ 5 слов, ≥ 3 предложений
- Удалили все тексты, где встречаются слова из “Списка грязных, озорных, непристойных или иных плохих слов”
- Удалили все тексты с вхождениями “Javascript”, “lorem ipsum” или “{”, с уведомлениями о политике использования, уведомлениями о куках
- Удалили все маркеры цитирования
- Дедубликация: если какие-то три предложения подряд из некоторого текста содержатся в каком-либо другом тексте, то оставляем только один из них
- Использовали классификатор языка, оставили только тексты с вероятностью 0.99 для английского
- 20 ТБ / месяц -> суммарно 750 GB

GPT3.



<https://arxiv.org/pdf/2005.14165.pdf>



OpenAI

GPT-3, an autoregressive language model with 175 billion parameters

GPT-3. На каких данных учились gpt-3?

Объёмы

- CommonCrawl за 2016–2019 годы - 45 ТБ сжатого открытого текста до фильтрации и 570 ГБ после фильтрации.

Качество

- наборы данных CommonCrawl и Books2 выбираются реже, чем один раз во время обучения, но другие более качественные наборы данных выбираются 2 -3 раза.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Предобучение GPT-3.

- Фильтрация
 - Обучение классификатора качественности текста:
 - Логрег на текстовых фичах (токенизация и частотность)
 - Позитивные примеры: тексты из заведомо качественных датасетов (WebText, Wiki и книги)
 - Негативные примеры – неотфильтрованный CommonCrawl
 - Оставляем текст если: $\text{np.random.pareto}(\alpha) > 1 - \text{document_score}$, $\alpha=0.9$
- Дедубликация:
 - MinHashLSH – наборы последовательностей токенов в тексте отображаем в число (хеш)
 - В одном хеше лежат похожие тексты => нечеткие дубли

Llama.



<https://arxiv.org/pdf/2302.13971.pdf>



Llama. Основная идея.

LLama

- училась только на открытых данных
- больше токенов, лучше качество

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	24.9	28.3	32.9	36.0
	65B	23.8	31.0	35.0	39.9

Lama. Что нового.

- English CommonCrawl (CCNet pipeline)
- Github
 - Только под разрешающей лицензией
 - Фильтрация по длине файла и пропорции alphanumerical
 - Удаление служебной разметки, четкая дедубликация
- Вики
 - Добавили статьи на 20 языках, удалили служебную разметку
- Arxiv
 - Добавили научные статьи, удалили комменты, библиографию
- Stack Exchange
 - Вопросы и ответы по разнообразным доменам, сортировка по оценкам ответов

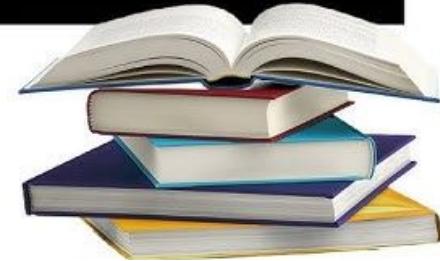
Llama. Что нового.

English CommonCrawl (CCNet pipeline) <https://arxiv.org/pdf/1911.00359>

- 1) Дедупликация
- 2) Language Identification: fastText для детекта языка
- 3) LM filter: фильтрация страниц низкого качества через перплексию
 - берем домен, который считаем эталонным (например вики)
 - учим на нем токенезацию (в статье 5-gram model
http://www.lrec-conf.org/proceedings/lrec2014/pdf/1097_Paper.pdf)
 - учим LLM на этом домене
 - применяем полученную LLM на неразмеченный корпус документов
 - если перплексия < thr, документ похож на вики - оставляем
 - если перплексия > thr, выкидываем документ

Phi1.

PHI-1: A ‘TEXTBOOK’ MODEL



Phi1. Coding dataset.

В основе следующие датасеты:

- 1) Отфильтрованные по языку программирования The Stack и StackOverflow (фильтровали моделью)
- 2) Синтетический textbook датасет содержащий <1B tokens сгенерированных GPT-3.5
- 3) Маленький синтетический датасет упражнений (заданий) ~180M tokens питоновских задач и их решений

```
import torch
import torch.nn.functional as F

def normalize(x, axis=-1):
    """Performs L2-Norm."""
    num = x
    denom = torch.norm(x, 2, axis, keepdim=True) \
        .expand_as(x) + 1e-12
    return num / denom

def euclidean_dist(x, y):
    """Computes Euclidean distance."""
    m, n = x.size(0), y.size(0)
    xx = torch.pow(x, 2).sum(1, keepdim=True) \
        .expand(m, n)
    yy = torch.pow(y, 2).sum(1, keepdim=True) \
        .expand(m, m).t()
    dist = xx + yy - 2 * torch.matmul(x, y.t())
    dist = dist.clamp(min=1e-12).sqrt()
    return dist

def cosine_dist(x, y):
    """Computes Cosine Distance."""
    x = F.normalize(x, dim=1)
    y = F.normalize(y, dim=1)
    dist = 2 - 2 * torch.mm(x, y.t())
    return dist
```

<https://arxiv.org/pdf/2306.11644>

Phi1. Фильтрация StackOverflow.

Как оставить только полезные данные?

- 1) Генерируют аннотации при помощи GPT4 к каждому кусочку кода (“determine its educational value for a student whose goal is to learn basic coding concepts”)
- 2) На этих аннотациях обучили random forest который предсказывает ценность (на эмбеддингах)

Educational values deemed by the filter

High educational value

```
import torch
import torch.nn.functional as F

def normalize(x, axis=-1):
    """Performs L2-Norm."""
    num = x
    denom = torch.norm(x, 2, axis, keepdim=True)
    .expand_as(x) + 1e-12
    return num / denom

def euclidean_dist(x, y):
    """Computes Euclidean distance."""
    m, n = x.size(0), y.size(0)
    xx = torch.pow(x, 2).sum(1, keepdim=True).
    expand(m, n)
    yy = torch.pow(y, 2).sum(1, keepdim=True).
    expand(m, n).t()
    dist = xx + yy - 2 * torch.matmul(x, y.t())
    dist = dist.clamp(min=1e-12).sqrt()
    return dist

def cosine_dist(x, y):
    """Computes Cosine Distance."""
    x = F.normalize(x, dim=1)
    y = F.normalize(y, dim=1)
    dist = 2 - 2 * torch.mm(x, y.t())
    return dist
```

Low educational value

```
import re
import typing
...

class Default(object):
    def __init__(self, vim: Nvim) -> None:
        self._vim = vim
        self._denite: typing.Optional[SyncParent] = None
        self._selected_candidates: typing.List[int] = []
        self._candidates: Candidates = []
        self._cursor = 0
        self._entire_len = 0
        self._result: typing.List[typing.Any] = []
        self._context: UserContext = {}
        self._bufnr = -1
        self._winid = -1
        self._winrestcmd = ''
        self._initialized = False
        self._winheight = 0
        self._winwidth = 0
        self._winminheight = -1
        self._is_multi = False
        self._is_async = False
        self._matched_pattern = ''
        ...
```

Phi1. Textbook.

Проблемы:

- Разнообразие. Оно важно по нескольким причинам:
 - открывает языковой модели различные способы решения проблем в коде
 - снижает риск переобучения или запоминания определенных шаблонов или решений
 - повышает обобщение и надежность модели для невиданных или новых задач.
- Проблема синтетического датасета: модель будет генерить +- одинаковые задачи, без разнообразия (языковые модели, следуют наиболее вероятным с учетом их обучающих данных, и им не хватает стимула для исследования альтернативных или новых способов генерации кода)
 - Нужно найти правильный «трюк», который заставит языковую модель быть более креативной и разнообразной, сохраняя при этом качество и связность примеров.
 - В [статье](#) был создан разнообразный набор коротких рассказов путем включения случайного подмножества слов, выбранных из некоторого фиксированного словаря в подсказку, и требуя, чтобы они были каким-то образом объединены в сгенерированном тексте

Phi1. Textbook.

- Генерация текстов + фрагментов кода через GPT-3.5
- Разнообразие достигается за счет ограничения по темам и целевой аудитории (после генерации оставляем по N примеров из каждой темы)

```
To begin, let us define singular and nonsingular matrices. A matrix is said to be singular if its determinant is zero. On the other hand, a matrix is said to be nonsingular if its determinant is not zero. Now, let's explore these concepts through examples.
```

```
Example 1: Consider the matrix A = np.array([[1, 2], [2, 4]]). We can check if this matrix is singular or nonsingular using the determinant function. We can define a Python function, `is_singular(A)`, which returns true if the determinant of A is zero, and false otherwise.
```

```
import numpy as np
def is_singular(A):
    det = np.linalg.det(A)
    if det == 0:
        return True
    else:
        return False

A = np.array([[1, 2], [2, 4]])
print(is_singular(A)) # True
```

Phi1. CodeExercises dataset.

- Это небольшой набор данных синтетических упражнений на Python.
- Каждое упражнение представляет собой строку документации функции, которую необходимо выполнить. (Чтобы научить модель следовать инструкциям)
- Основным средством выявления разнообразия является ограничение имен функций.

```
def valid_guessing_letters(word: str, guesses: List[str]) -> List[str]:  
    """  
        Returns a list of valid guessing letters, which are letters that have not been guessed yet and  
        are present in the word.  
        Parameters:  
        word (str): The word to guess.  
        guesses (List[str]): A list of letters that have already been guessed.  
        Returns:  
        List[str]: A list of valid guessing letters.  
    """  
    valid_letters = []  
    for letter in word:  
        if letter not in guesses and letter not in valid_letters:  
            valid_letters.append(letter)  
    return valid_letters
```

Fineweb-edu.



FineWeb-Edu

The finest collection of educational content the
web has to offer



[Fineweb-edu.](#)

Как еще можно улучшить фильтрацию CommonCrawl?

- Продолжим идею с оценкой полезности данных для “образования” (обучения модели).
- Есть 15T токенов для фильтрации - прогонаять на всех Llama 3 долго
 - оценим через Llamaz часть данных
 - обучим бинарную классификацию (для порога Llama score ≥ 3)
 - профильтруем оставшуюся часть данных

Fineweb-edu.

Промпт для оценки:

Below is an extract from a web page. Evaluate whether the page has a high educational value and could be useful in an educational

- Add 1 point if the extract provides some basic information relevant to educational topics, even if it includes some irrelevant
- Add another point if the extract addresses certain elements pertinent to education but does not align closely with educational
- Award a third point if the extract is appropriate for educational use and introduces key concepts relevant to school curricula.
- Grant a fourth point if the extract highly relevant and beneficial for educational purposes for a level not higher than grade s
- Bestow a fifth point if the extract is outstanding in its educational value, perfectly suited for teaching either at primary sc

The extract:

<EXAMPLE>.

After examining the extract:

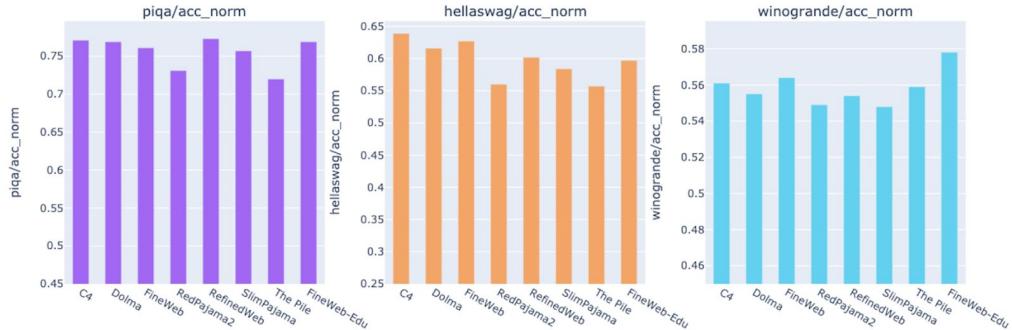
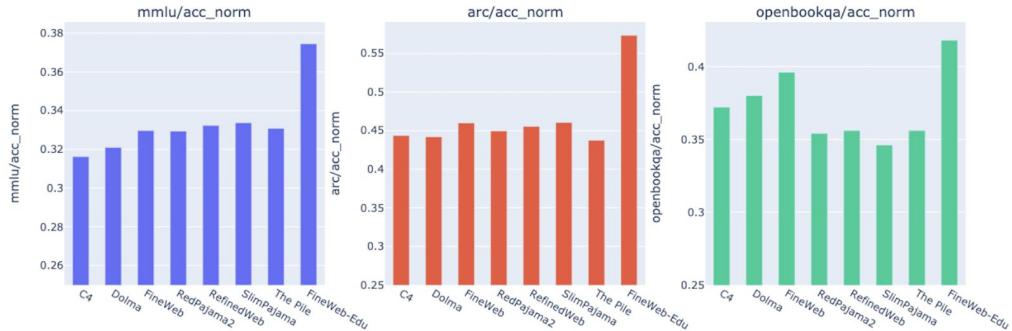
- Briefly justify your total score, up to 100 words.
- Conclude with the score using the format: "Educational score: <total points>"

full prompt: <https://huggingface.co/HuggingFaceFW/fineweb-edu-classifier/blob/main/utils/prompt.txt>

Fineweb-edu.

Фильтрации примеров со скором < 3
удалила 92% датасета, оставив 1.3Т
токенов для обучения.

Evaluation results at 350B tokens

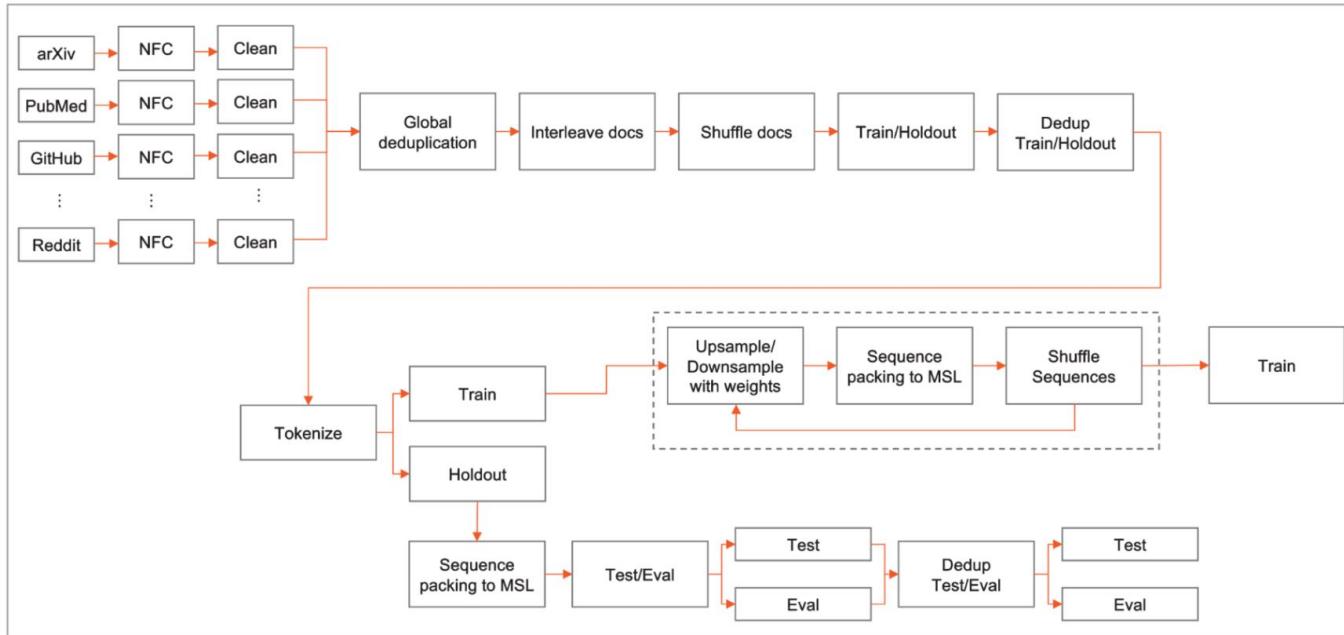


RedPajama & SlimPajama.



RedPajama,
a project to create
leading open-source
models, starts by
reproducing LLaMA
training dataset of over
1.2 trillion tokens

RedPajama & SlimPajama.



<https://github.com/togethercomputer/RedPajama-Data>

https://github.com/Cerebras/modelzoo/tree/main/src/cerebras/modelzoo/data_preparation/nlp/slimpajama

RedPajama & SlimPajama.

RedPajama-V2:

- includes over 100B text documents coming from 84 CommonCrawl snapshots and processed using the [CCNet](#) pipeline. Out of these, there are 30B documents in the corpus that additionally come with quality signals, and 20B documents that are deduplicated.

SlimPajama:

- A 627B token, cleaned and deduplicated version of RedPajama

<https://github.com/togethercomputer/RedPajama-Data>

https://github.com/Cerebras/modelzoo/tree/main/src/cerebras/modelzoo/data_preparation/nlp/slimpajama

SlimPajama

- NFC-нормализация: удаление неюникод-символов
- Минимальная длина текстов – 200 символов без пунктуации и пробельных символов.
- Дедубликация MinHashLSH с построением графа похожих (дубликат – мера жаккара ≥ 0.8)
- Стратификация семплирования из разных источников с весами
- Умное разделение на трейн / тест, дедубликация теста относительно трейна
- Много оптимизаций, чтобы всё работало быстро и параллельно
- Отличный гайд по очистке выборки с большим количеством доп. ссылок



Дедупликация.

Дедупликация

1

Четкая

Удаляем четкие совпадения, посимвольно либо после предобработки;

2

Нечеткая

По степени похожести:

- 1) на текстовом уровне: доля общих n-грамм, MinHashLSH и пр.
- 2) на векторном уровне:
переводим тексты в векторное пространство (BERT), считаем попарно косинусную близость / формируем на ее основе кластера, удаляем дубли по порогу / связности.

3

“Smart”

Отсеиваем не просто по степени похожести, но и по информативности. В итоге отсеивается больше точек, чем просто по похожести без потери качества.

Умная дедупликация – зачем

1. Убрать дубли для улучшения качества
2. Уменьшить размер выборки без ухудшения ее качества, для ускорения обучения
3. Выравнивание размеров различных срезов (для избежания сдвига распределения на какой-то слишком частый срез)



Умная дедупликация – зачем

Почему нельзя просто семплировать из выборки?

- Возникает *selection bias* – с большей вероятностью выбираются точки в областях с высокой плотностью точек
- Точки в областях с низкой плотностью часто сильно влияют на качество
- Приводит к ухудшению разнообразия новой выборки относительно старой

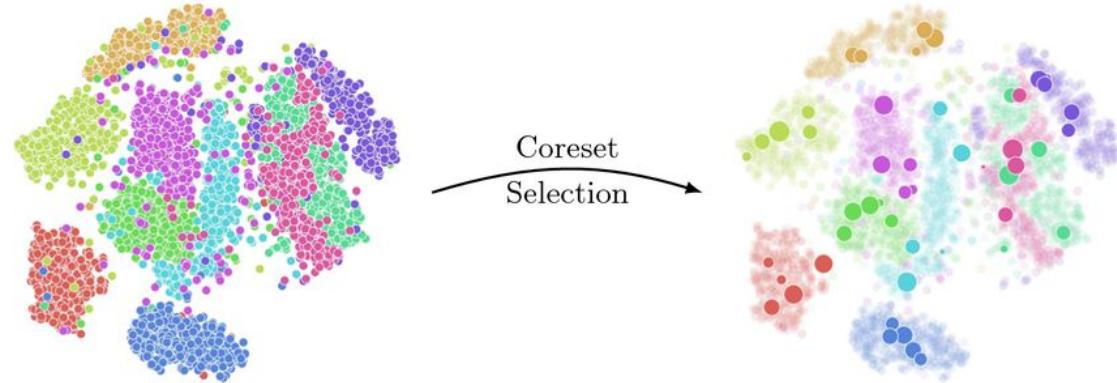
Coreset selection.

Умная дедупликация.

$$\mathcal{B}(\mathcal{V}, \mathbf{1})$$

\approx

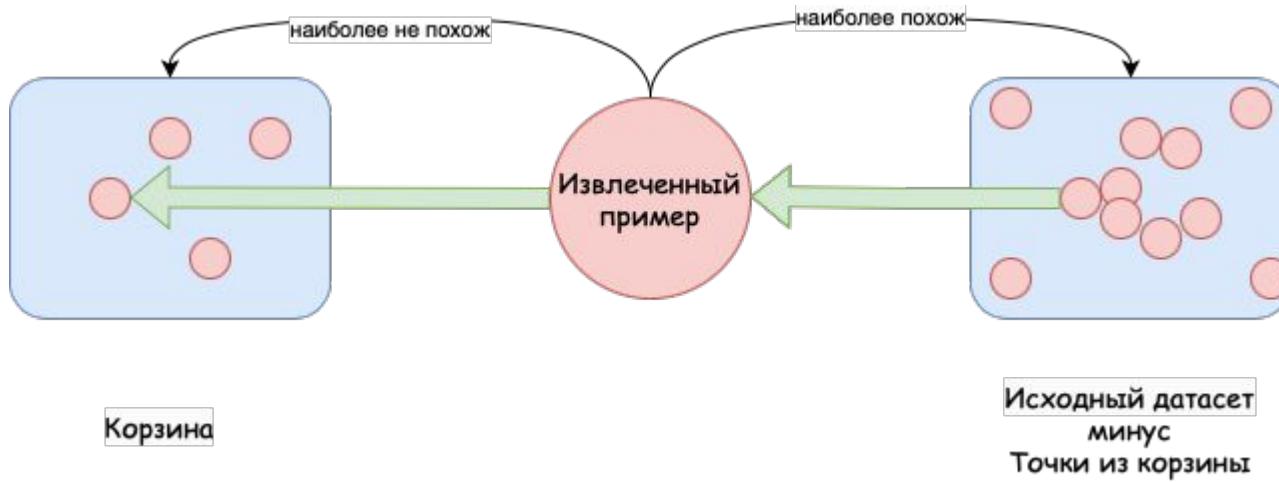
$$\mathcal{B}(\mathcal{S}^*, \gamma^*)$$



Coreset Selection.

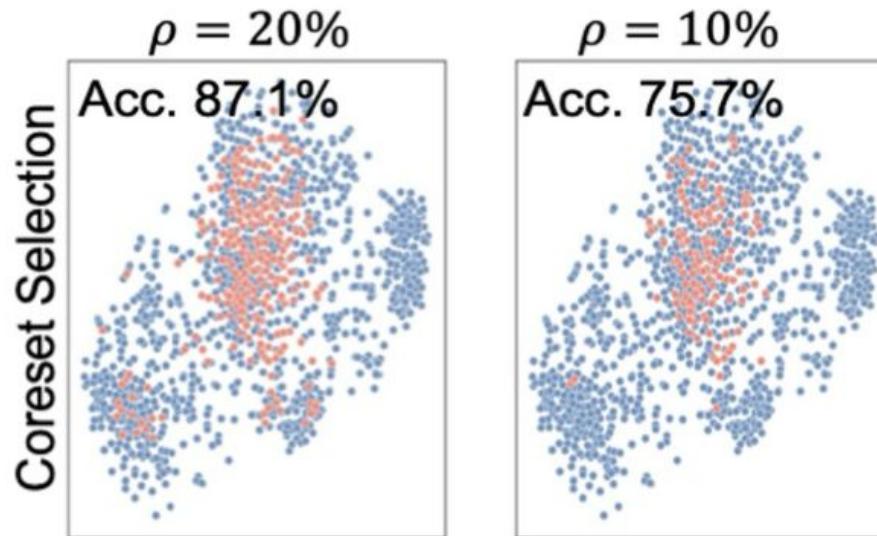
- Итеративно отбираем точки из исходной выборки в отдельную корзину
- На каждом шаге для перемещения выбирается точка, которая
 - максимально похожа на оставшуюся исходную выборку
(репрезентативность)
 - максимально непохожа на точки из уже собранной корзины (разнообразие)
- В случае текстов: косинусная похожесть семантических эмбеддингов

Coreset Selection.

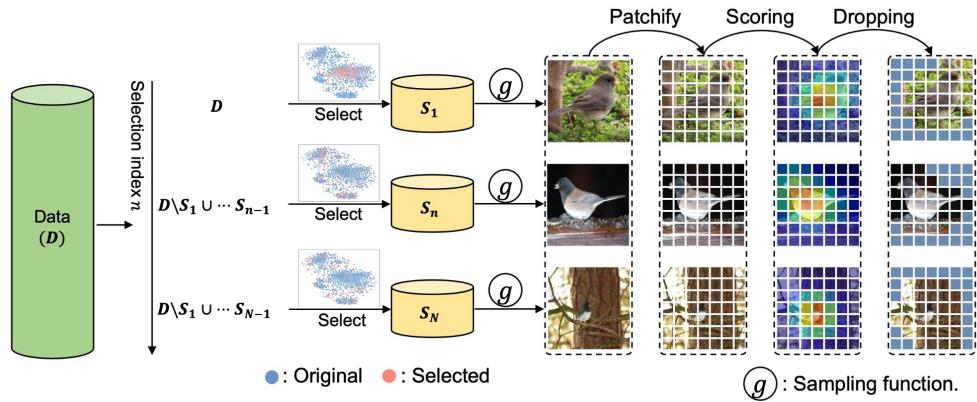


$$P(x_k) = \sum_{p \in \mathbf{S}_1^{k-1}} \underbrace{\|f(p) - f(x_k)\|_2^2}_{C_1(x_k)} - \sum_{p \in \mathbf{D} \setminus \mathbf{S}_1^{k-1}} \underbrace{\|f(p) - f(x_k)\|_2^2}_{C_2(x_k)},$$

Какие проблемы у метода?

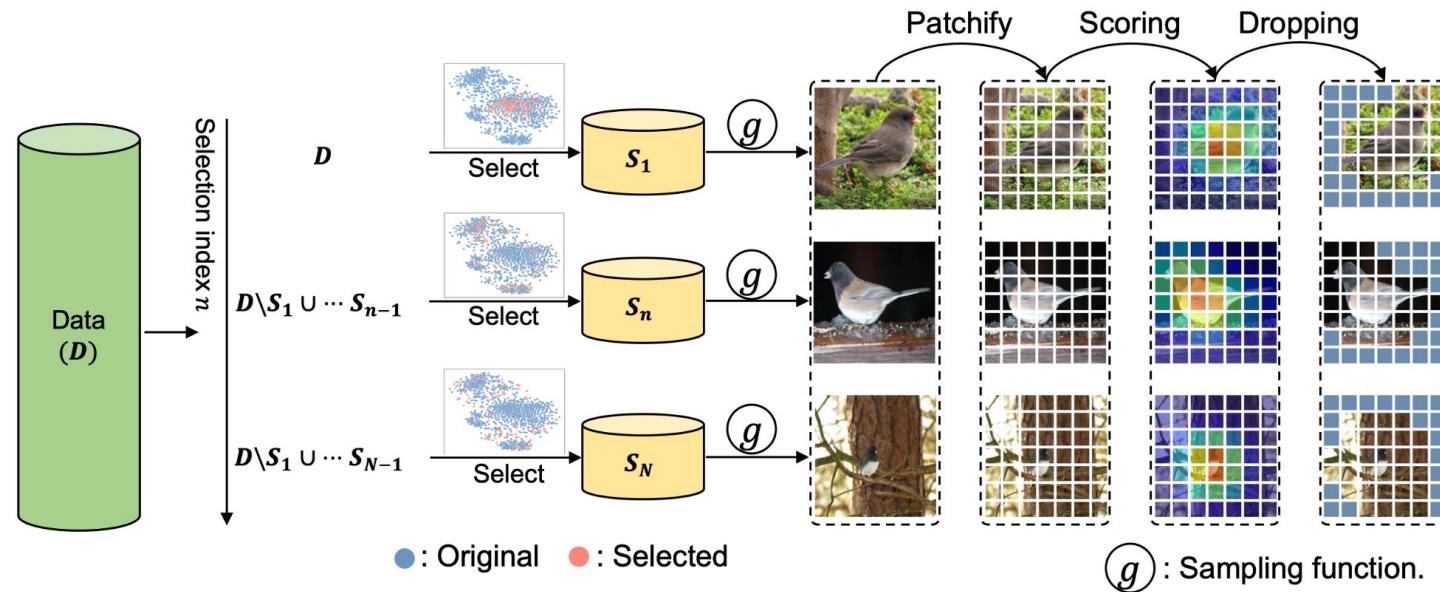


Dataset quantization. Умная дедупликация.

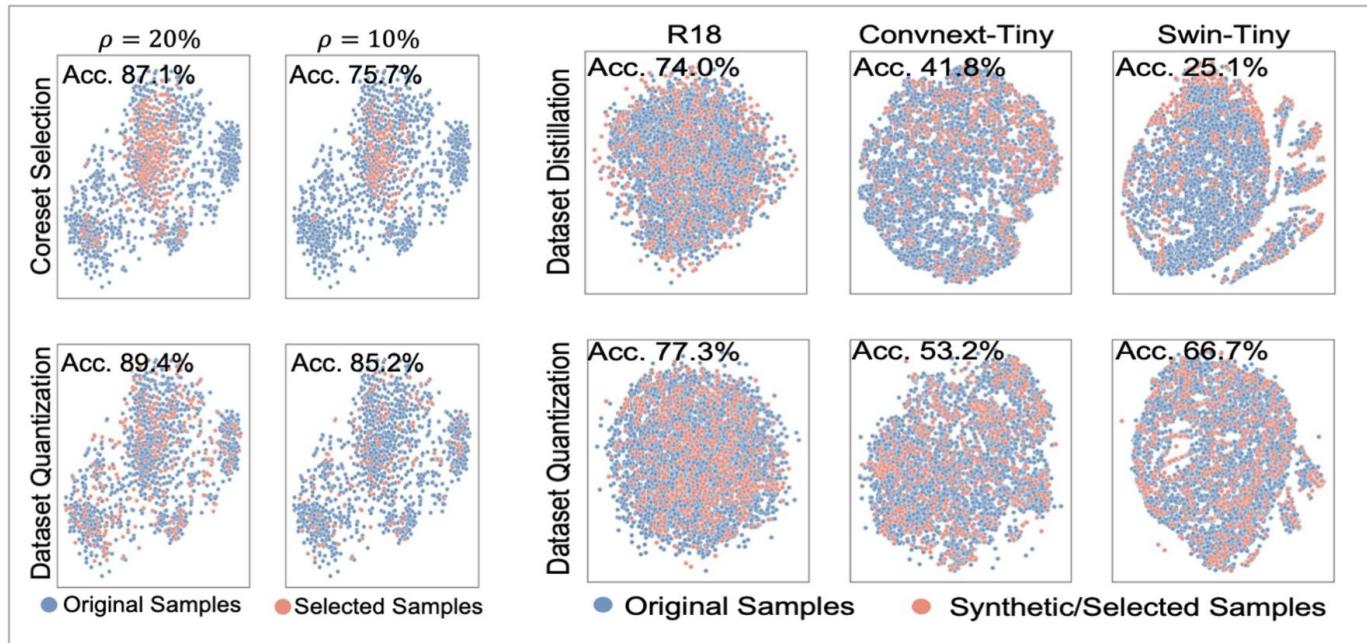


Dataset Quantization.

Coreset Selection подвержен проблеме selection bias => несколько раз применяем Coreset Selection



Какая дедупликация визуально лучше?



Как готовить претрейн датасет для LLM

- Определиться с необходимыми языками и доменами, на которых хочется показывать хорошее качество;
- Найти большое количество данных по необходимым срезам, желательно легальных (+ анонимизация <https://microsoft.github.io/presidio/>);
- Извлечение “чистых” текстов, удаление служебной разметки;
- Фильтрация некачественных и бесполезных документов: эвристики и ML-классификаторы;
- Дедупликация: четкая и нечеткая (LSH);
- Во время обучения: стратифицированное семплирование данных из каждого среза (какие-то срезы “показывать” модели чаще, какие-то реже).

SFT.



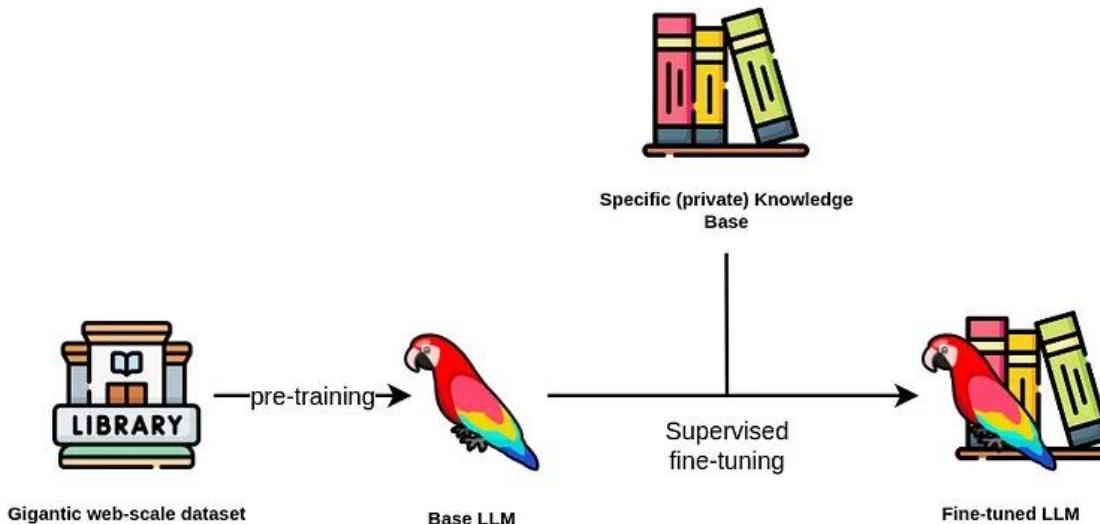
Экскурс в историю ГПТ.

Evolution from Transformer architecture to ChatGPT



Датасет для SFT. Для чего?

1. “Вытаскиваем” знания модели, которые были заложены на этапе pretrain.
2. Доучиваем модель на специфичный домен.
3. Учим следовать инструкциям.



Инструктивный датасет. Как собрать?

Требования к датасету:

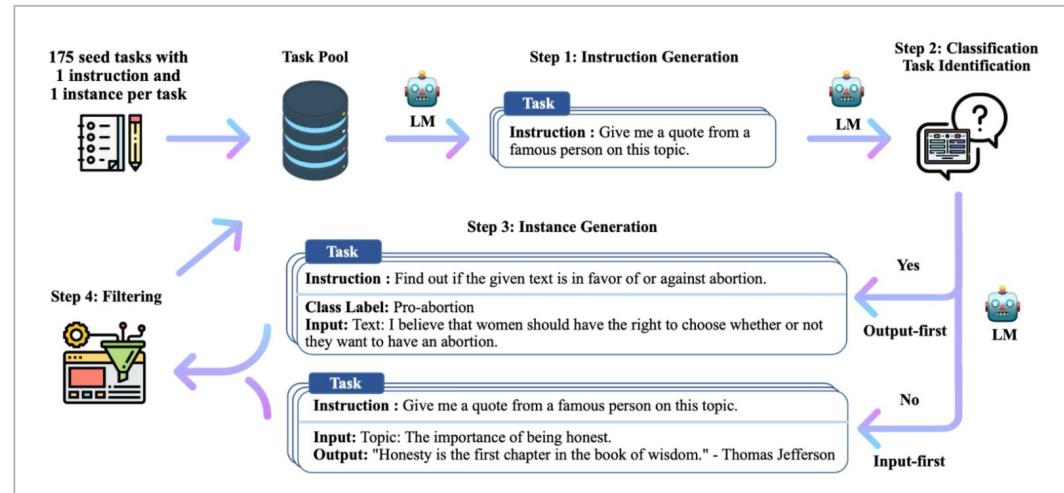
- разнообразные домены
- качественные данные
- не только ответ на вопрос, но и объяснение почему

Откуда брать данные для SFT:

- Писать с нуля с помощью редакторов (дорого, лучше качество, но хуже разнообразие)
- Использовать доступные в Интернете множества запросов (плохое качество)
- Генерировать (такие данные называются синтетическими)

Self-instruct.

Полуавтоматический подход для генерации обучающих данных (инструкций, пользовательских запросов) с помощью LLM.



Self-instruct. Алгоритм.

- Вход: Формируем начальный пул задач (seed-set) – 175 качественных ручных примеров
 - С помощью LLM генерируем инструкции для нескольких новых задач
 - Создаем по несколько input и output для этих задач
 - Фильтруем околодубликаты и некачественные инструкции, input-ы и output-ы
 - Повторяем пока не получим нужное количество задач, input-ов и таргетов для них
- Выход: большой датасет разнообразных качественных инструкций и таргетов

Instruction: Given an address and city, come up with the zip code.

Input:

Address: 123 Main Street, City: San Francisco 

Output: 94105

Instruction: I am looking for a job and I need to fill out an application form. Can you please help me complete it?

Input:

Application Form:

Name: _____ Age: _____ Sex: _____

Phone Number: _____ Email Address: _____

Education: _____ ...

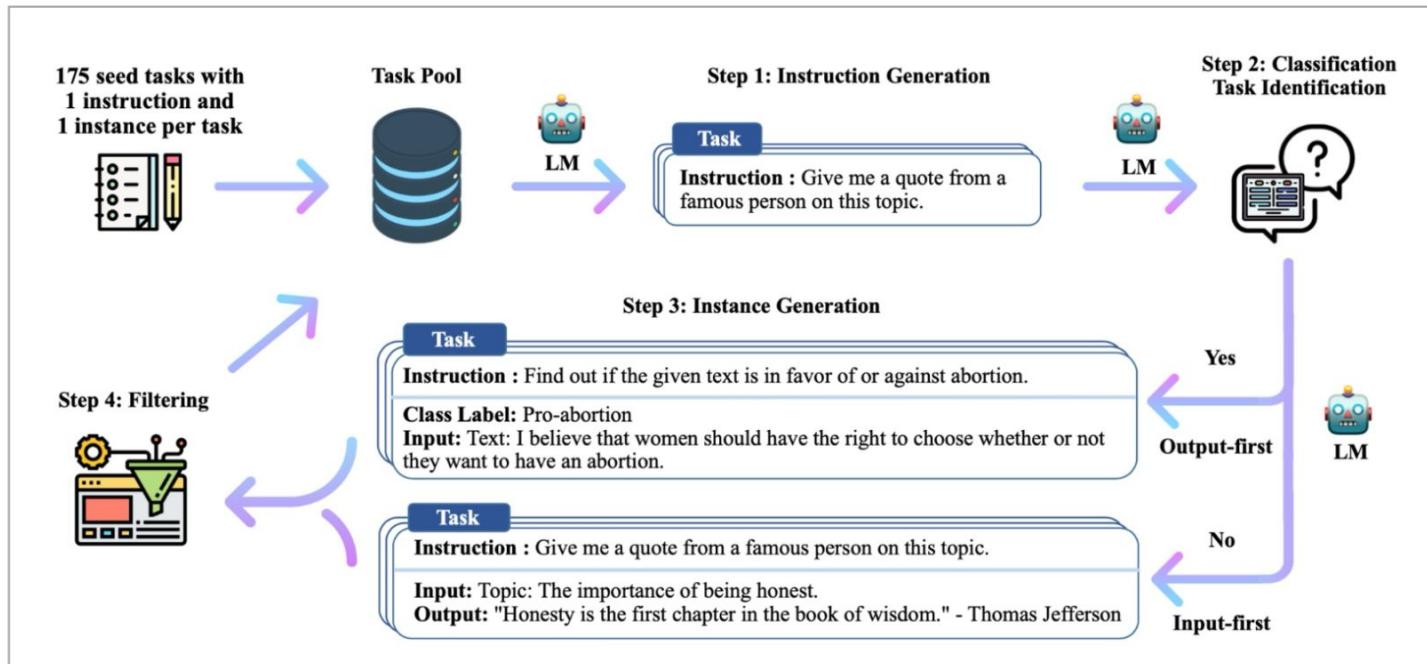
Output:

Name: John Doe Age: 25 Sex: Male
Phone Number: ... 

Self-instruct. Детали.

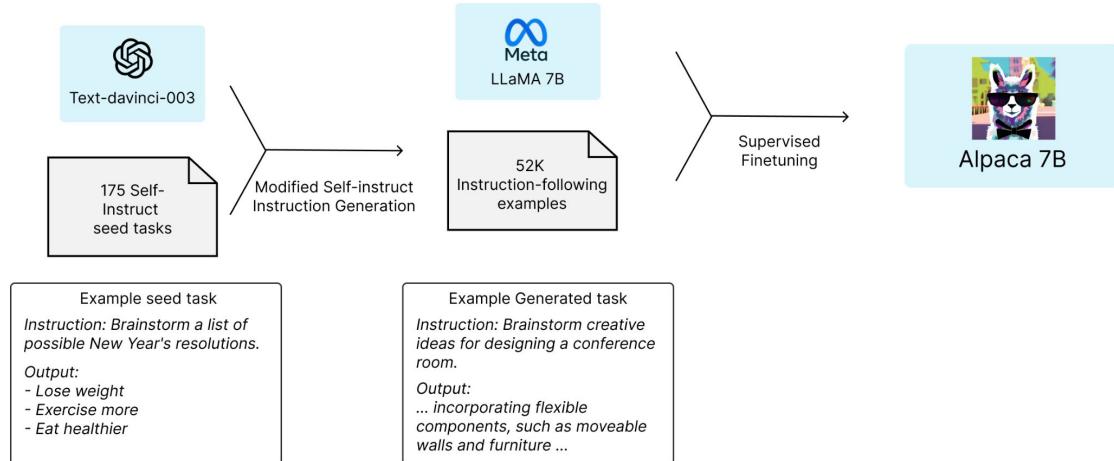
- Семплируем 8 инструкций из пула задач: 6 ручных и 2 сгенерированных (для разнообразия)
- Просим LLM для инструкции сгенерировать input и output:
 - Если запрос сводится к классификации (текст -> конечное число классов), то оптимальнее сначала генерировать по инструкции output (класс), а потом по инструкции и классу – input
 - В противном случае оптимальнее instruction -> input -> output
- LLM часто генерирует околодубликаты => фильтрация:
 - По похожести: ROUGE-L $\geq 0.7 \Rightarrow$ не добавляем новую задачу в task pool
 - По качеству: различные эвристики

Self-instruct. Детали.



Alpaca.

- Стенфорд дообучил LLaMa 7B на 52к инструкций, сгенерированных с помощью self-instruct
- Качество близко к инструктивной GPT-3.5, но при этом модель и подход доступны всем



UltraChat.

Генерация диалогового
датасета для sft.

<https://github.com/thunlp/UltraChat>



Large-scale, Informative, and Diverse Multi-round Dialogue Data, and Models

Meta topics of the Questions about the World sector

- | | |
|-----------------------------|---------------------------------|
| Technology | Philosophy and ethics |
| Health and wellness | History and nostalgia |
| Travel and adventure | Social media and communication |
| Food and drink | Creativity and inspiration |
| Art and culture | Personal growth and development |
| Science and innovation | Spirituality and faith |
| Fashion and style | Pop culture and trends |
| Relationships and dating | Beauty and self-care |
| Sports and fitness | Family and parenting |
| Nature and the environment | Entrepreneurship and business |
| Music and entertainment | Literature and writing |
| Politics and current events | Gaming and technology |
| Education and learning | Mindfulness and meditation |
| Money and finance | Diversity and inclusion |
| Work and career | Travel and culture exchange |

UltraChat. Идея.

Основная идея UltraChat заключается в использовании отдельных LLM для генерации вводных строк, имитации пользователей и ответа на запросы.

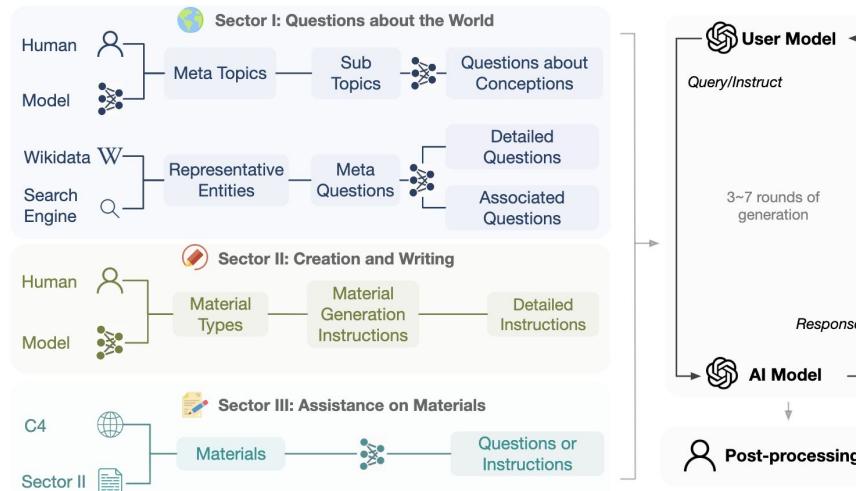


Figure 1: Construction process of UltraChat. The three sectors of data are derived from different meta information.

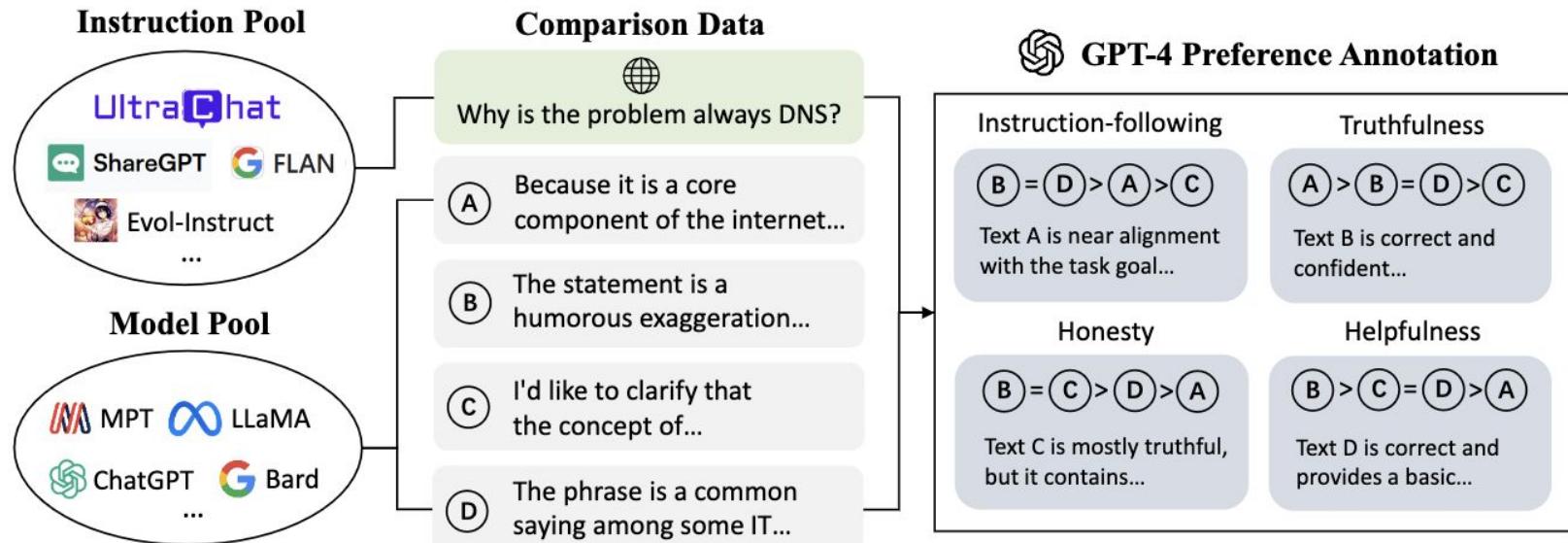
UltraChat. Алгоритм.

Для каждой задачи (фактовости, инструкций и т.д.):

- генерируем до N вопросов/инструкций
- объединяем материал с каждым вопросом/инструкцией (если нужно, например вопрос на основе фрагмента текста)
- для каждого сценария свой промпт (разработанный руками)
- для каждого ввода мы генерируем диалог из 2~4 раундов.
- выбираем лучший ответ модели из всех раундов

UltraChat. Алгоритм.

How do we get preferences?



LIMA.

Less Is More for Alignment

<https://arxiv.org/pdf/2305.11206>



LIMA.

На этапе SFT качество важнее количества?

- Superficial Alignment Hypothesis: все знания в LLM заложены на этапе пре-трейна, а во время alignment выучивается только стиль или формат взаимодействия с пользователем
- В таком случае, для выравнивания модели нужно небольшое количество качественных, разнообразных и репрезентативных точек и ответов на них в требуемом формате
- Аккуратно собрали 1000 точек, дообучили на них LLaMa 65B – модель близка по качеству к GPT-4, выровненной на огромных датасетах, и лучше Alpaca 65B, выровненной на 52k self-instruct инструкций
- Для LLaMa 7B достаточно ≥ 2000 точек

LIMA. Что важно в данных.

1

Репрезентативность. Покрыли максимальное количество тематик: на каждом из срезов выбиралось по несколько точек из каждой подтемы;

2

Качество. Эвристики/классификаторы для фильтрации грязных и не полезных данных.

3

Согласованность. Согласованность данных/асессоров между собой.

Спасибо за внимание!

Мария Анисимова, руководитель ML команды Антиспама

