大模型预训练

通用人工智能与数字经济创新团队

西南财经大学

本章内容

8.1 概述

大模型的预训练训练，主要涉及到数据源和分布式训练两个关键方面

数据源

- 需要庞大的标注或者未标注数据集进行自监督学习，种类包括**文本、图像、声音**等
- 并需要经过**预处理**和**清洗**步骤。数据的质量和多样性直接影响模型的性能和泛化能力
- 为了获取全面的数据，有时需要聚合多个子数据集，会导致**数据冗余**和**不平衡**等问题

分布式训练

【解决问题】

- 由于单一计算设备（GPU）的计算能力和内存有限

【主要方法】

- 大模型通常需要分布式架构进行训练，通过**数据并行**、**模型并行**或**张量并行**等多种方式来实现

本章内容

- 8.1 概述
- 8.2 预训练数据工程
  - 8.2.1 预训练数据源
  - 8.2.2 多模态数据集
  - 8.2.3 数据处理
  - 8.2.4 模型性能关系
- 8.3 预训练方法
  - 8.3.1 预训练任务
  - 8.3.2 优化参数设置
  - 8.3.3 可扩展训练技术
- 8.4 讨论

8.2.1 预训练数据源

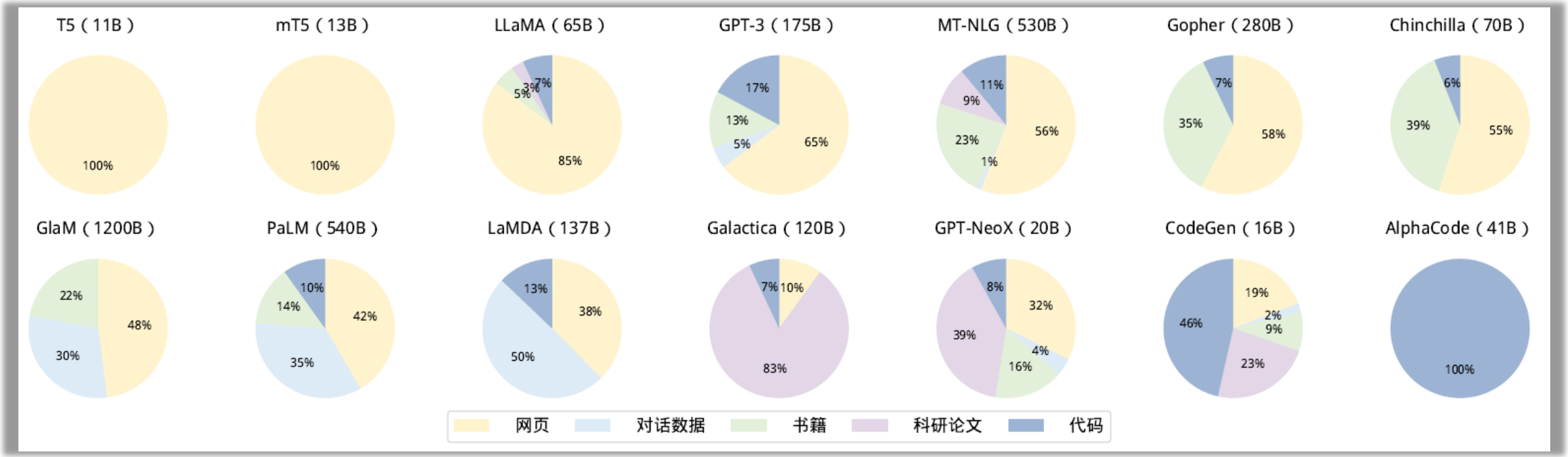
大模型预训练的性能高度依赖两个主要因素：1）预训练语料库的质量与规模；2）数据预处理的策略与方法。

通用数据

网页、书籍和对话文本等

专业数据

语言数据、科学论文和编程代码等



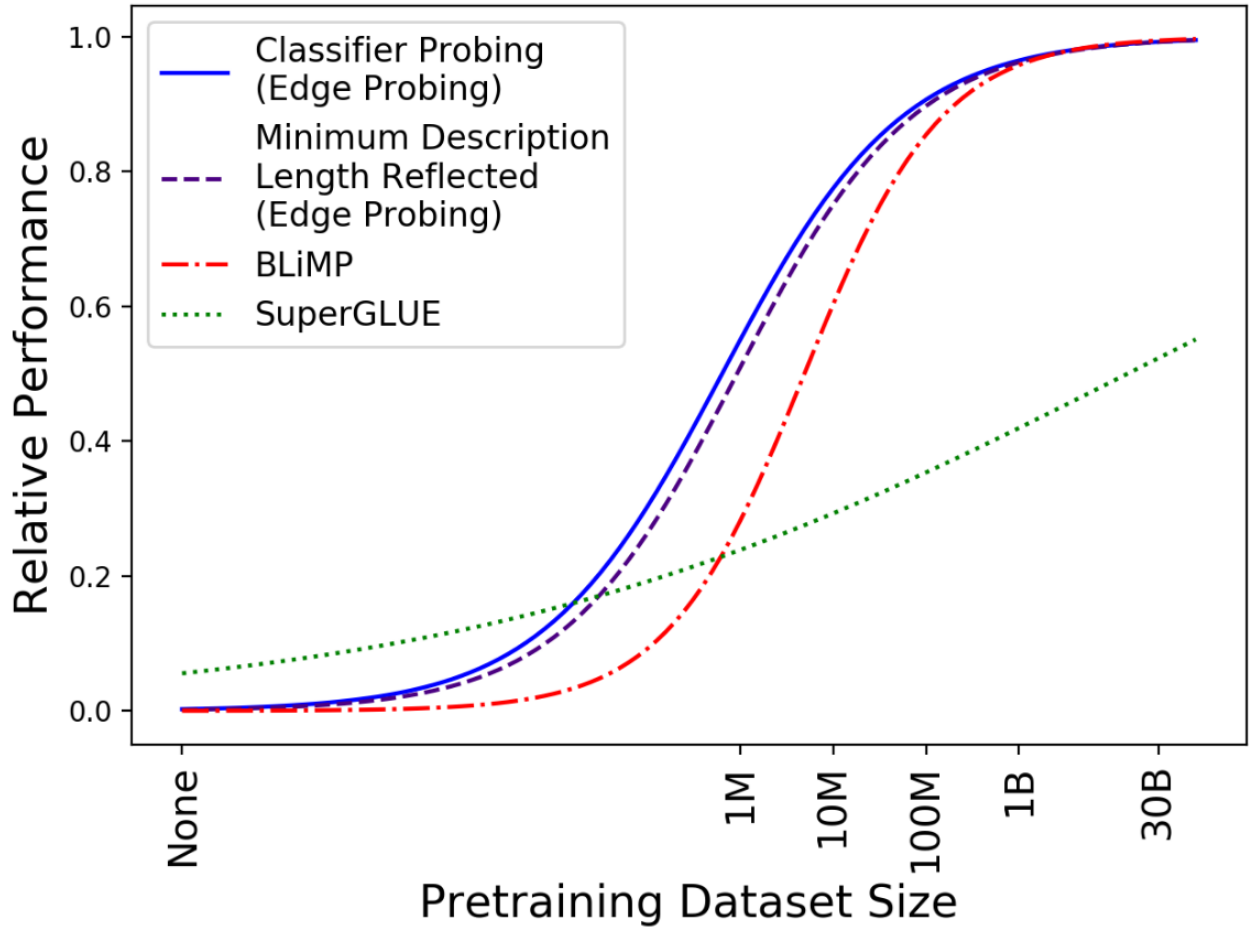
Source：https://transformers.run/c4/c15_pretrain_llms

需要多少语料库才能学会”文字接龙”？

1. 语言知识

“**我马上就**”后面可以接“**写**”等动词，而不能接“**猫**”等名词

When Do You Need Billions of Words of Pretraining Data?



图片来源：https://arxiv.org/pdf/2011.04946

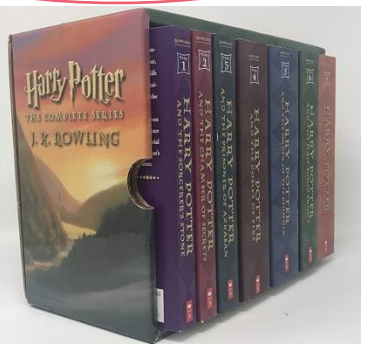
需要多少语料库才能学会”文字接龙”？

2. 世界知识

“中国的首都是”
后面接“北京”，
而不能接“上海”

模型	年份	数据规模
GPT-1	2018	7000本书
GPT-2	2019	40GB文本
GPT-3	2020	从45TB数据清洗得到570GB文本
Llama 3	2024	15TB文本
Qwen 2.5	2024	大概60TB数据

大概 300B tokens，相对于《哈利·波特》全集的30万遍



In terms of Qwen2.5, the language models, all models are pretrained on our latest large-scale dataset, encompassing up to **18 trillion tokens**.

来源：https://qwenlm.github.io/blog/qwen2.5/

8.2.1 预训练数据源

■ 通用数据源

网页数据源 互联网作为一个庞大的数据源，为语言模型提供了丰富的文本材料，具备规模大、动态、多语言和主题丰富等特点，是目前LLMs中使用最广泛的数据源。

代表性网页预训练语料库

数据集	发布者	发布时间	规模	特点
CC-Stories	Google Brain	2018-7	31 GB	基于 Common Crawl，英文
RealNews	华盛顿大学等	2019-5	120 GB	基于 Common Crawl，英文
C4	Google Research	2019-10	12.68TB	基于 Common Crawl，英文
CLUECorpus2020	CLUE	2020-3	100 GB	基于 Common Crawl，中文
CC100	Facebook AI	2020-7	2.5 TB	基于 Common Crawl，100 种语言
WuDaoCorpora-Text	北京智源人工智能研究院	2021-6	5 TB	中文，开源 200 GB
mC4	Google Research	2021-6	251 GB	基于 Common Crawl，108 种语言
OSCAR 22.01	Inria	2022-1	8.41 TB	基于 Common Crawl，151 种语言
MNBVC	里屋	2023-1	20.3 TB	中文，包括网页、书籍、论文等
RefinedWeb	Falcon	2023-6	4.88 TB	基于 Common Crawl，英文
WanJuanText	上海 AI 实验室	2023-8	1 TB	中文，包括网页、书籍等

8.2.1 预训练数据源

■ 所有网上文本都能拿来训练吗？

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the la

程序员起诉 Copilot

2021 年 6 月，GitHub 和 OpenAI 发布了 Copilot，可以“通过使用人工智能提供或填充代码块来帮助软件编码人员”。2021 年 8 月，OpenAI 又发布了 Codex，“可将自然语言转换为代码并集成到 Copilot 中”。GitHub 用户每月支付 10 美元或每年 100 美元才能访问 Copilot。Codex 和 Copilot 接受了“数十亿行”公开可用代码的训练，包括来自公共 GitHub 存储库的代码，诉讼由此而起。2023 年 5 月 11 日，美国加利福尼亚州北区地方法院针对 J. DOE 1 等诉 GitHub 等案做出了部分允许并部分拒绝驳回动议的裁定。该案的被告包括 GitHub、微软、OpenAI 等。

https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html

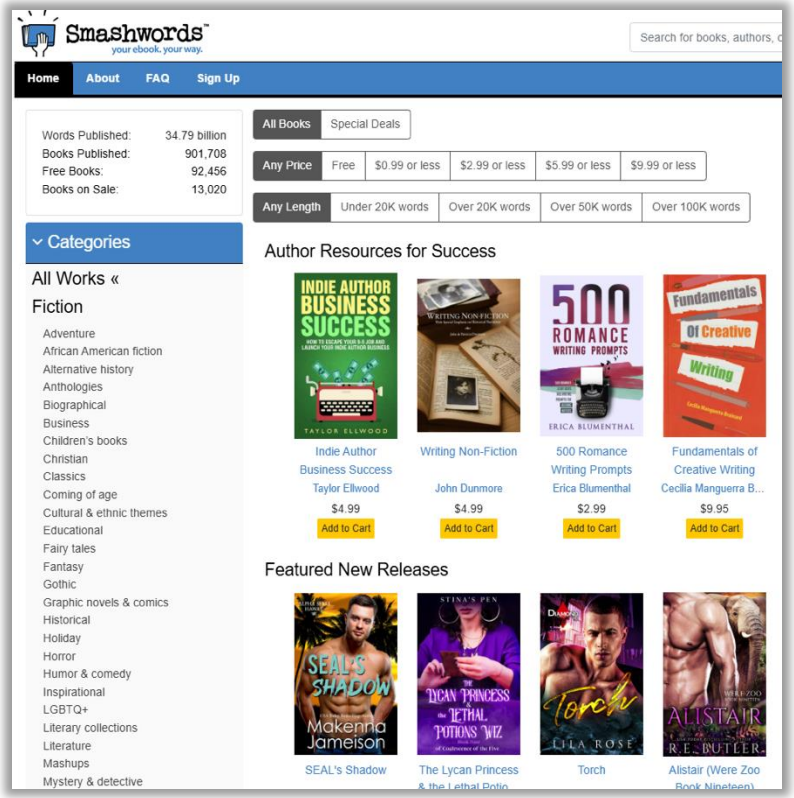
https://36kr.com/p/2609988349031944

8.2.1 预训练数据源

■ 通用数据源

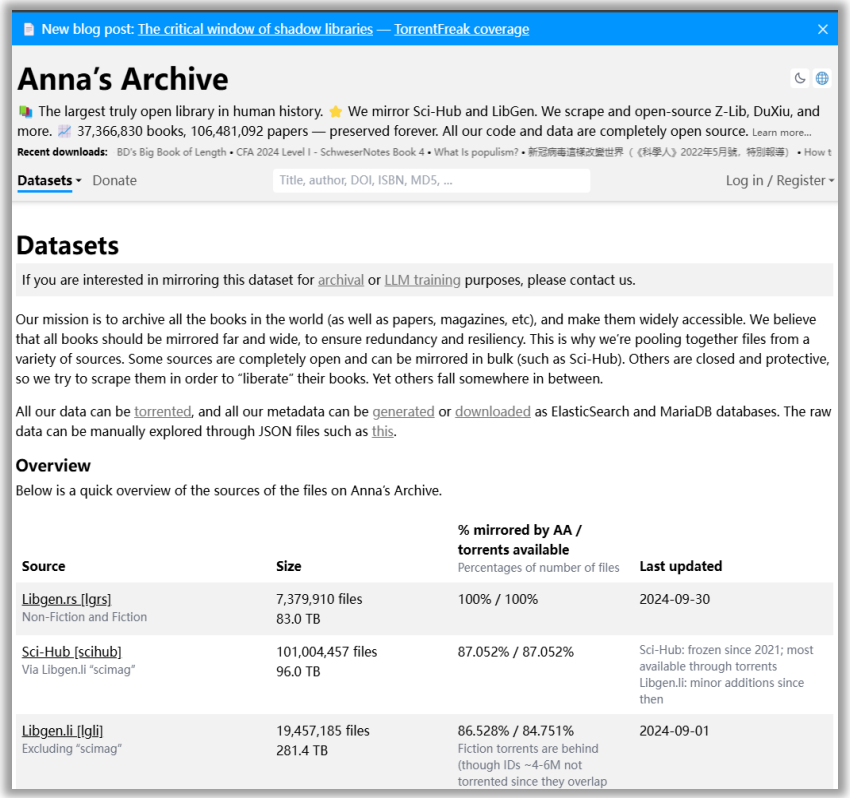
书籍数据集 作为长形式文本的优质来源，有助于语言模型在理解复杂语句结构和生成连贯文本方面的训练。

Smashwords



Project Gutenberg

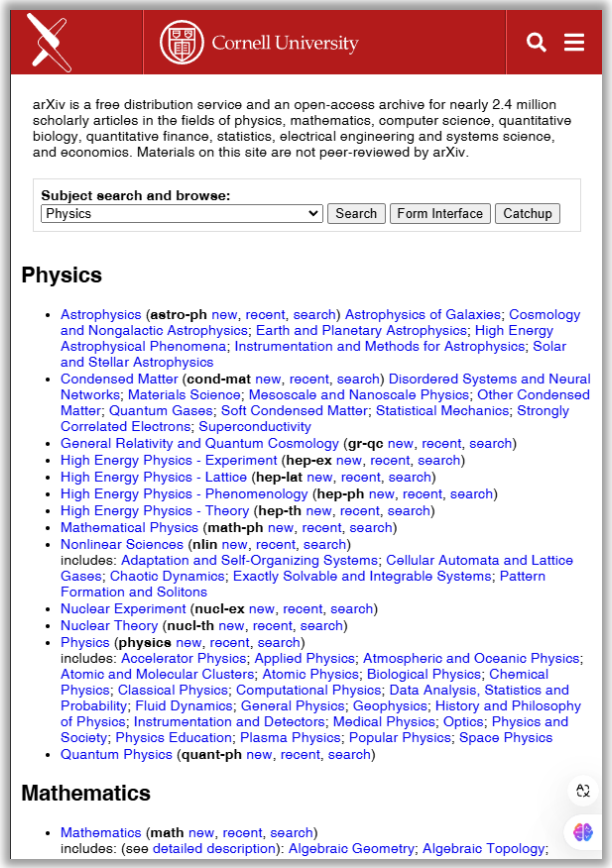
Anna's Archive

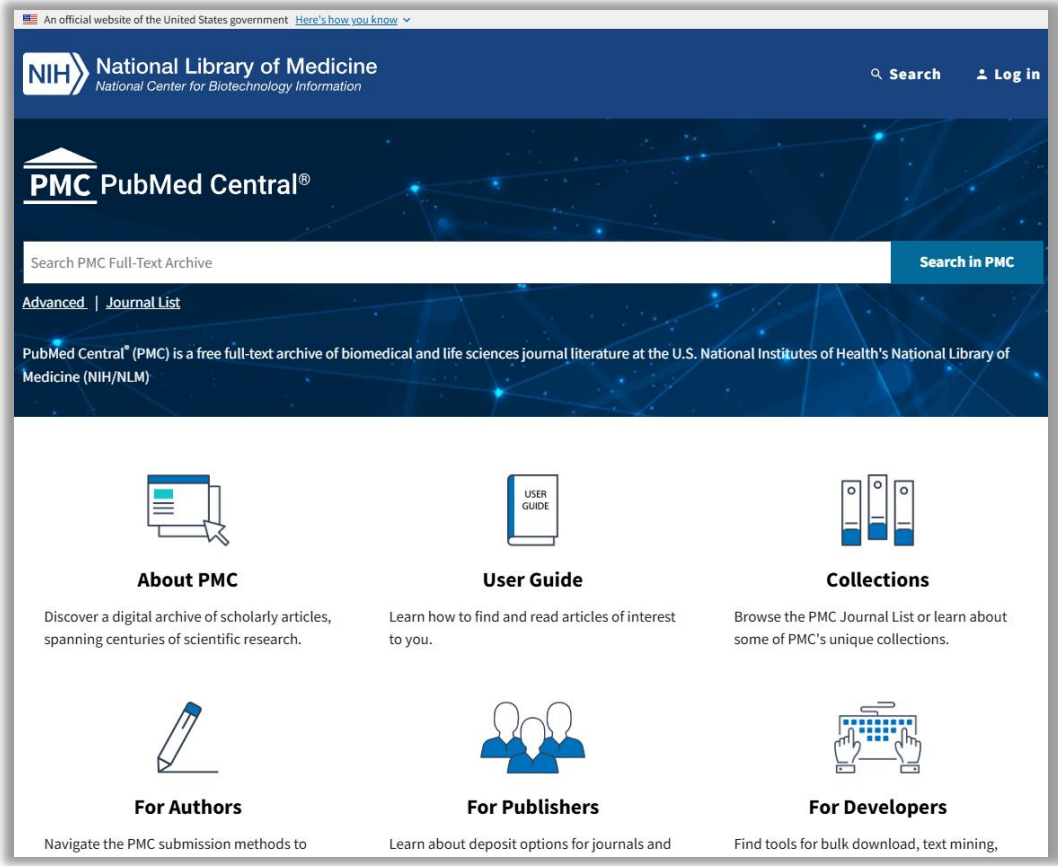


8.2.1 预训练数据源

■ 专业数据源

科学文本 这类数据集通常包括学术论文、专利和其他类型的专业文献。由于科学文本常常涉及专业术语、复杂的数据结构和公式，因此在预处理时需要采用特殊方法

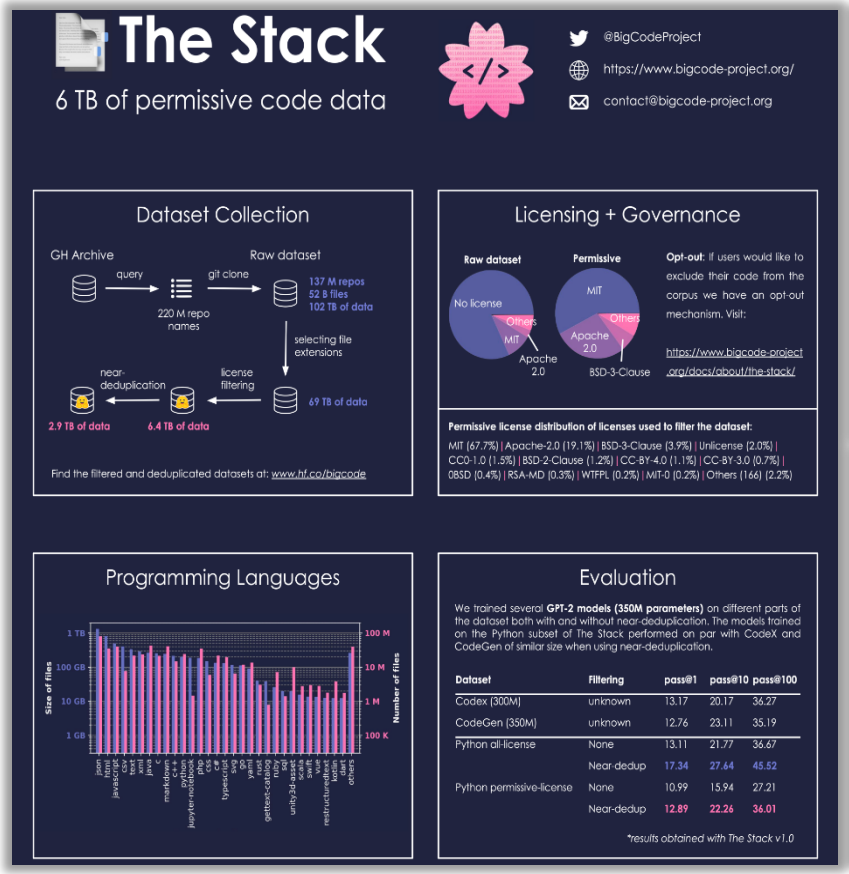


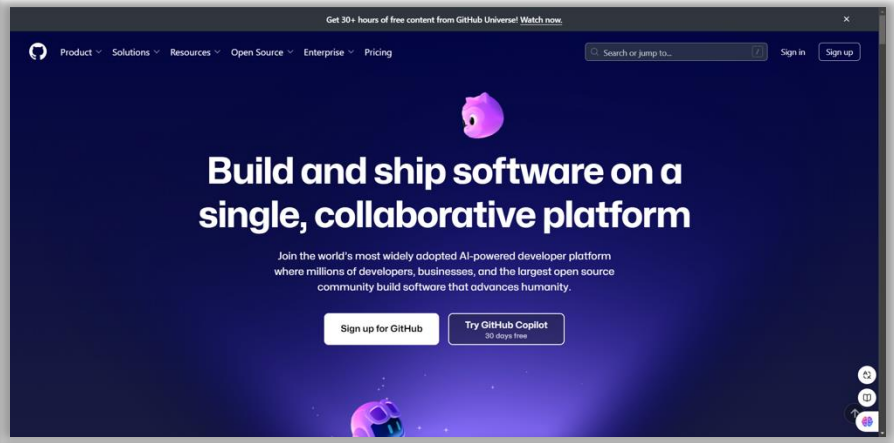


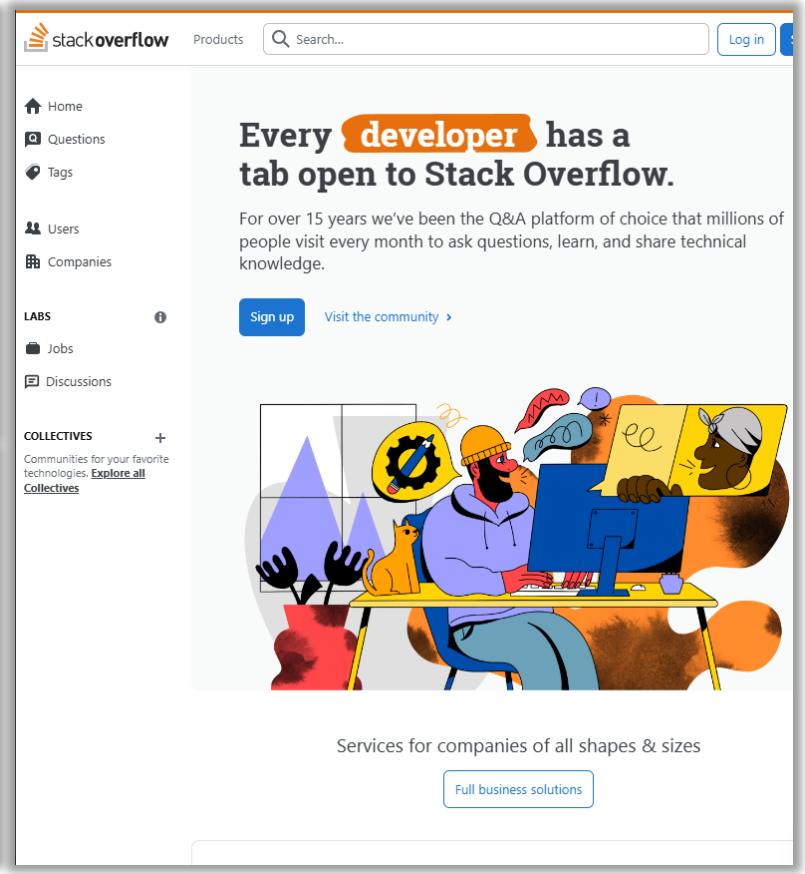
8.2.1 预训练数据源

■ 专业数据源

代码数据集 在程序生成方面，代码作为一种专业数据类型已经得到了广泛的研究关注。模型在大规模代码库上的预训练能显著提升代码生成质量。







8.2.1 预训练数据源

■ 专业数据源

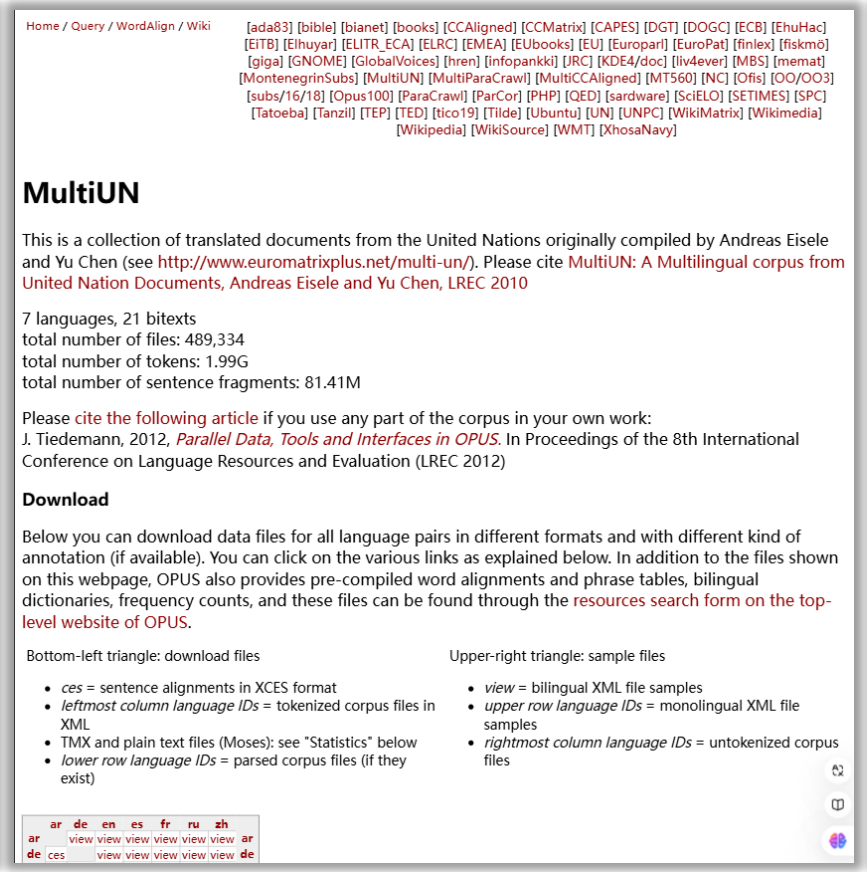
Python是代码数据集中最常见的程序语言。如 GPT-3（175B）的训练数据主要由 Common Crawl、BooksCorpus 和代码数据集构成，而 LaMDA （540B）采用对话文本、代码数据和 Common Crawl。

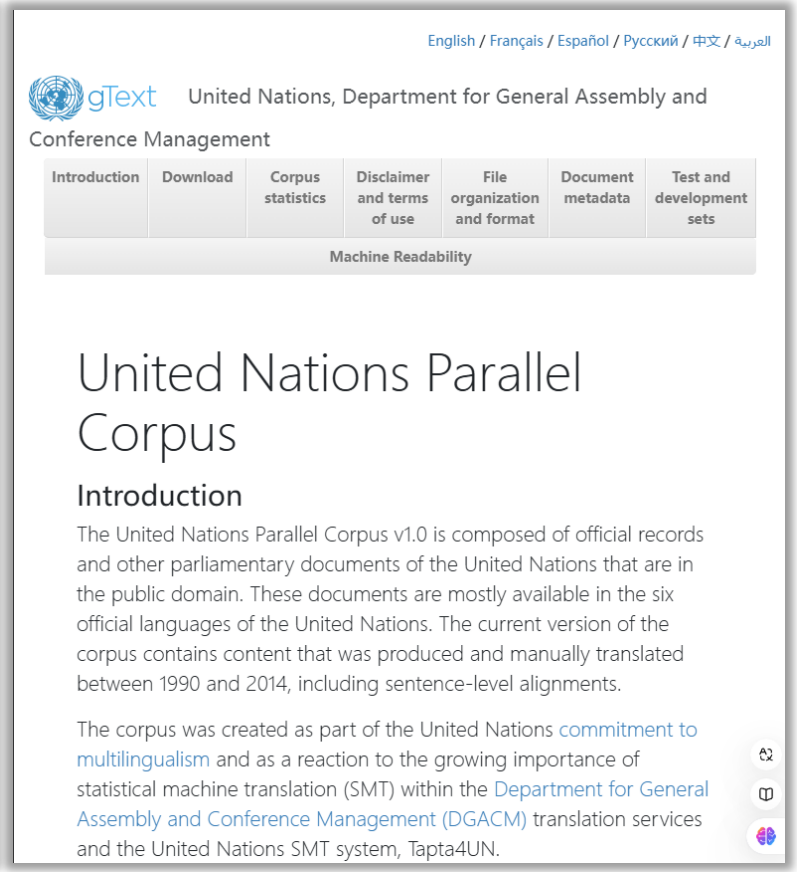
数据集	数据源	程序语言	规模 (GB)	类型
CodeSearchNet (2019)	GitHub	Go, Java, JS, PHP, Python, Ruby PL	17	NL-PL
CodeNet (2021)	AIZU, AtCoder	-	8	NL-PL
THEPILE (2021)	GitHub, ArXiv,...	-	825	NL-PL
thestack	GitHub	-	3100	PL
BigQuery	GitHub	C/C++, Go, Java, JS, Python	340	PL
BIGPYTHON (2022)	GitHub	Python	217	PL
CodeParrot	GitHub	Python	180	PL
GCPY (2022)	GitHub	Python	-	PL

8.2.1 预训练数据源

■ 专业数据源

多语言语料 这类数据集通常包括多种语言的相似或相同的内容，例如双语或多语言新闻文章、翻译文本等，也被称为平行语料库。





8.2.1 预训练数据源

■ 多类别语料

Pile：2020年，825G的语料 （英文数据集）

Component	Raw Size	Weight
Pile-CC	227.12 GiB	18.11%
PubMed Central	90.27 GiB	14.40%
Books3†	100.96 GiB	12.07%
OpenWebText2	62.77 GiB	10.01%
ArXiv	56.21 GiB	8.96%
Github	95.16 GiB	7.59%
FreeLaw	51.15 GiB	6.12%
Stack Exchange	32.20 GiB	5.13%
USPTO Backgrounds	22.90 GiB	3.65%
PubMed Abstracts	19.26 GiB	3.07%
Gutenberg (PG-19)†	10.88 GiB	2.17%
OpenSubtitles†	12.98 GiB	1.55%
Wikipedia (en)†	6.38 GiB	1.53%
DM Mathematics†	7.75 GiB	1.24%
Ubuntu IRC	5.52 GiB	0.88%
BookCorpus2	6.30 GiB	0.75%
EuroParl†	4.59 GiB	0.73%
HackerNews	3.90 GiB	0.62%
YoutubeSubtitles	3.73 GiB	0.60%
PhilPapers	2.38 GiB	0.38%
NIH ExPorter	1.89 GiB	0.30%
Enron Emails†	0.88 GiB	0.14%
The Pile	**825.18 GiB**	

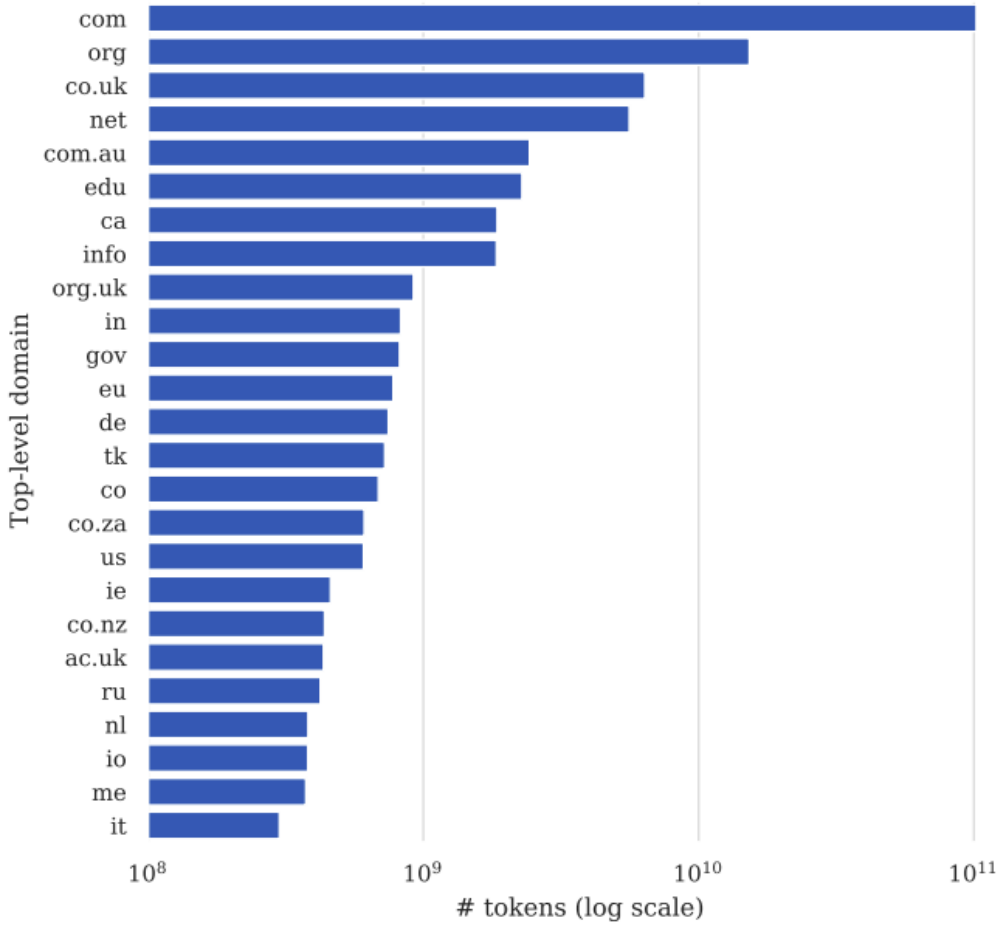
论文链接：https://arxiv.org/pdf/2101.00027.pdf

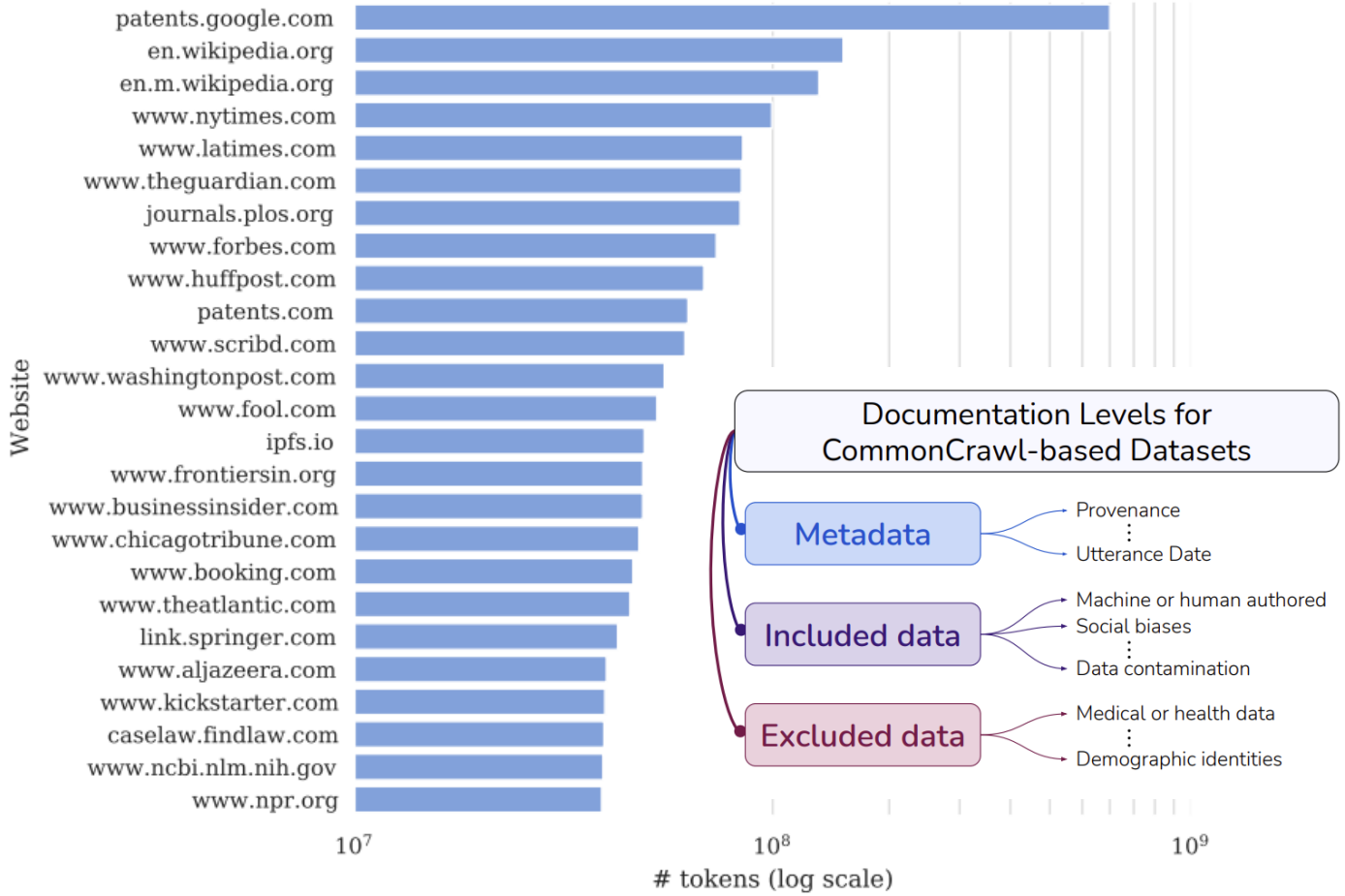
项目仓库：https://github.com/EleutherAI/the-pile

8.2.1 预训练数据源

■ 多类别语料

C4，T5的训练语料，2021 EMNLP （英文数据集）





论文链接： https://arxiv.org/pdf/2104.08758.pdf

项目仓库： https://github.com/allenai/allennlp/discussions/5056

8.2.1 预训练数据源

■ 多类别语料

WuDaoCorpora是北京智源研究院最新构建的高质量数据集，由全球最大的纯文本数据集、全球最大的多模态图文数据集和全球最大的中文对话数据集三部分构成。包含3TB 文本数据、90TB 图文数据和 181GB 对话数据。

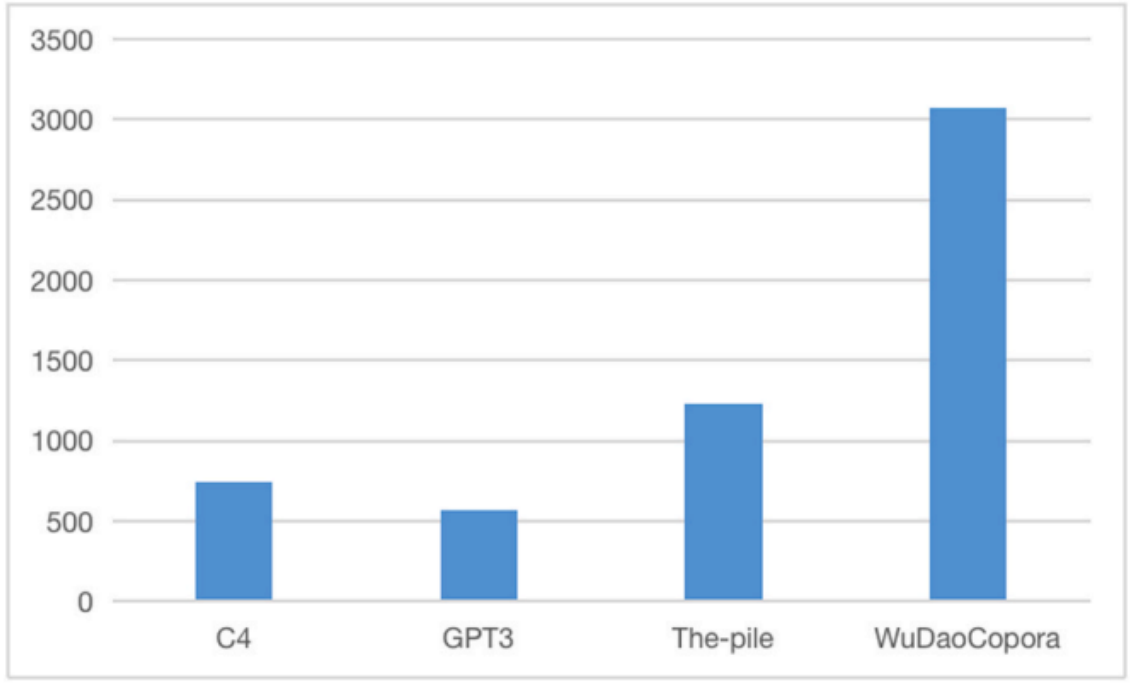


Fig. 1. Comparison with other Chinese corpora.

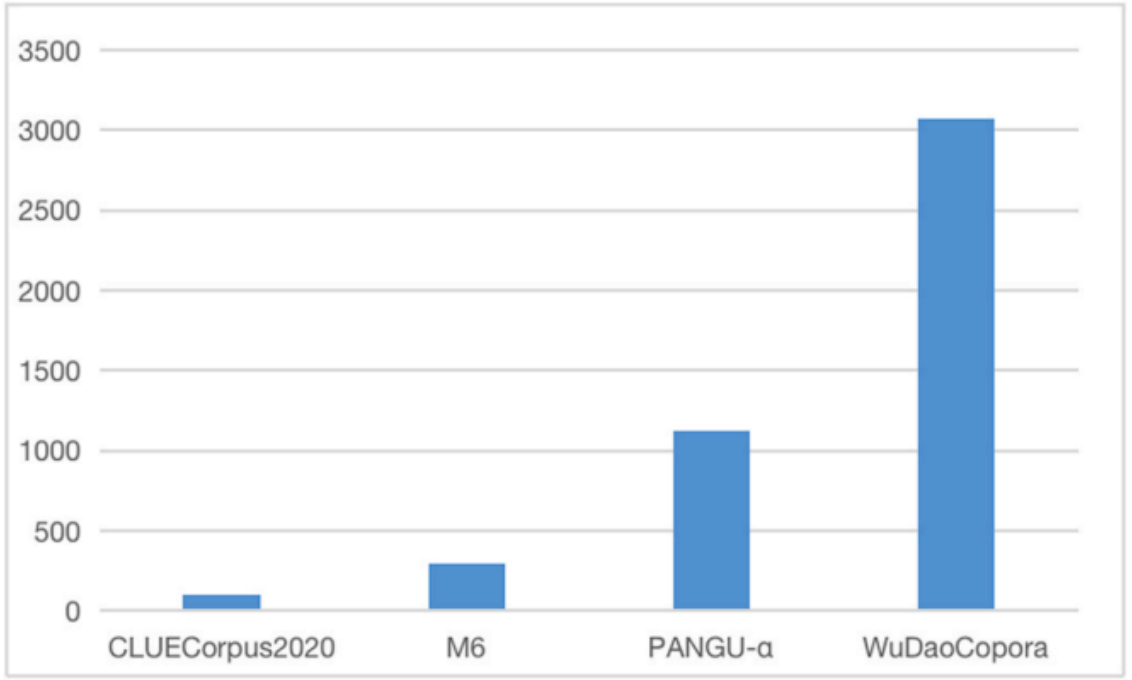


Fig. 2. Comparison with other language corpora.

论文链接： https://www.sciencedirect.com/science/article/pii/S2666651021000152

项目仓库： https://data.baai.ac.cn/details/WuDaoCorporaText

8.2.1 预训练数据源

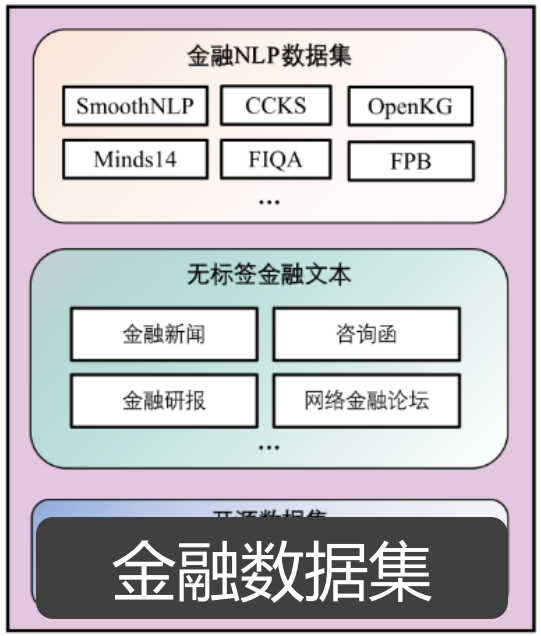
■ 多类别语料

表 8.2: 多类别语料的主要数据分布

数据集	规模	网页	代码	书籍	科学文本
Pile	825 GB	36.2%	7.6%	15%	38.1%
TigerBot_pretrain_en	51 GB	33.9%	30.2%	35.9%	-
TigerBot_pretrain_zh	55 GB	50.3%	-	25.9%	-
WanJuanText	1 TB	96.8%	-	0.07%	2.1%

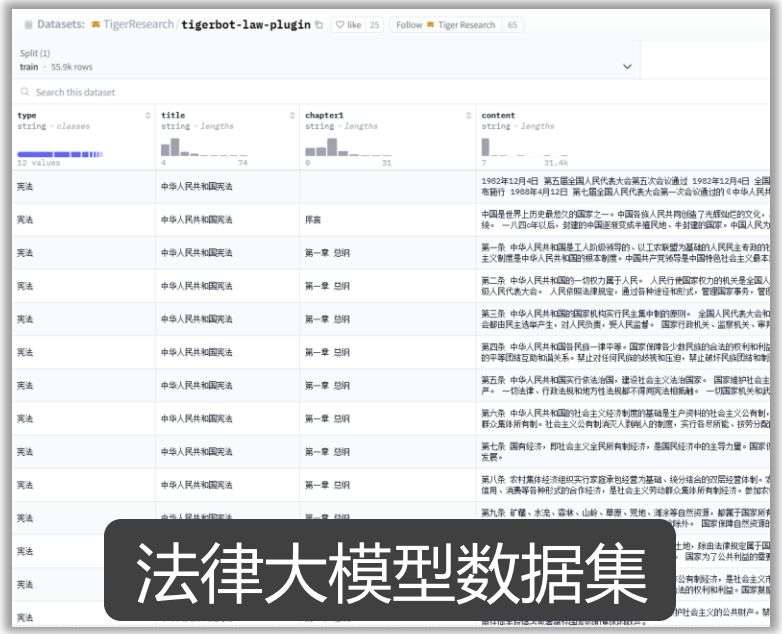
表 8.3: 垂直领域语料库

数据集	分类	规模	数据源
BBT-FinCorpus	金融	256 GB	公司公告、研究报告、金融新闻、社交媒体
FinCorpus	金融	60 GB	公司公告、金融新闻、金融考试题目
FinGLM	金融	69 GB	上市公司年报
Medical-pt	医疗	632 MB	医学百科、教科书
Proof-Pie-2	数学	55 B 词元	ArXiv、OpenWebMath
TigberBot-law	法律	29.9 MB	法律条款
TransGPT-pt	交通	35.8 MB	技术文档、工程施工信息



金融数据集

中英文医疗数据集



法律大模型数据集

8.2.2 多模态数据集

多模态数据集是大规模数据集的重要分支，其独特之处在于能够同时包含并融合多种格式的数据资源，为模型预训练提供了更加多样化的输入

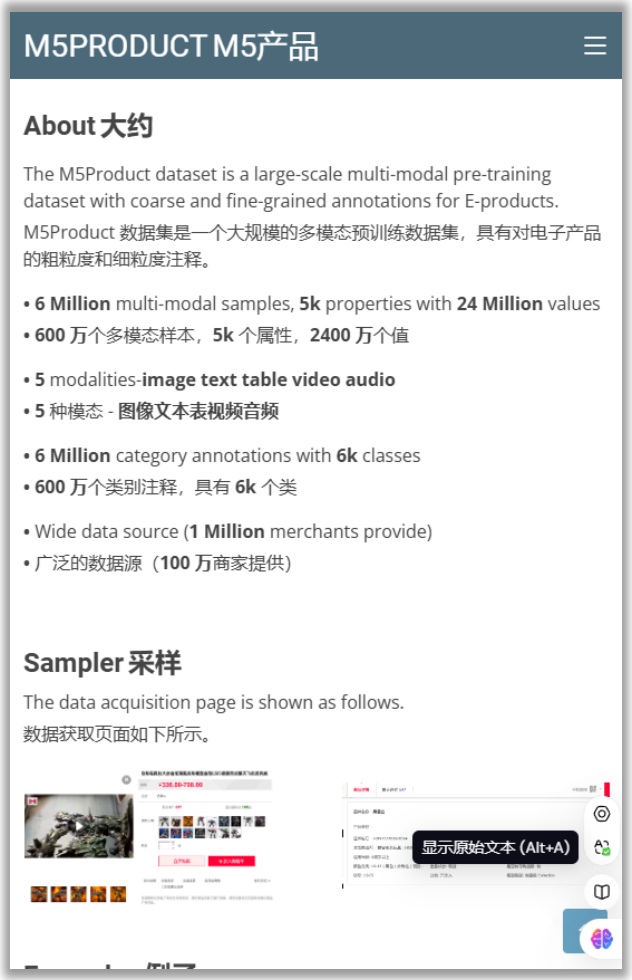
数据集	发布时间	规模	特点
Flickr [202]	2014	3 万张图片，每张 5 条描述	英文，人工标注
COCO [23]	2014	33 万张图片，每张 5 条描述	英文，2022 年发布了 7.47 亿张图片的 COCO-700M
Conceptual Caption[9]	2018	30 万张图片，每张 5 条描述	英文，2021 年发布 1200 万张图片的 Conceptual12M
WIT [157]	2021	3700 万条图文对	多语言，来自维基百科
悟空 [58]	2022	1 亿条图文对	中文
LAION-5B [152]	2022	58 亿条图文对	多语言，提供 LAION2B-en（英文描述）等子集
WuDaoMM[10]	2022	6.5 亿条图文对	中文

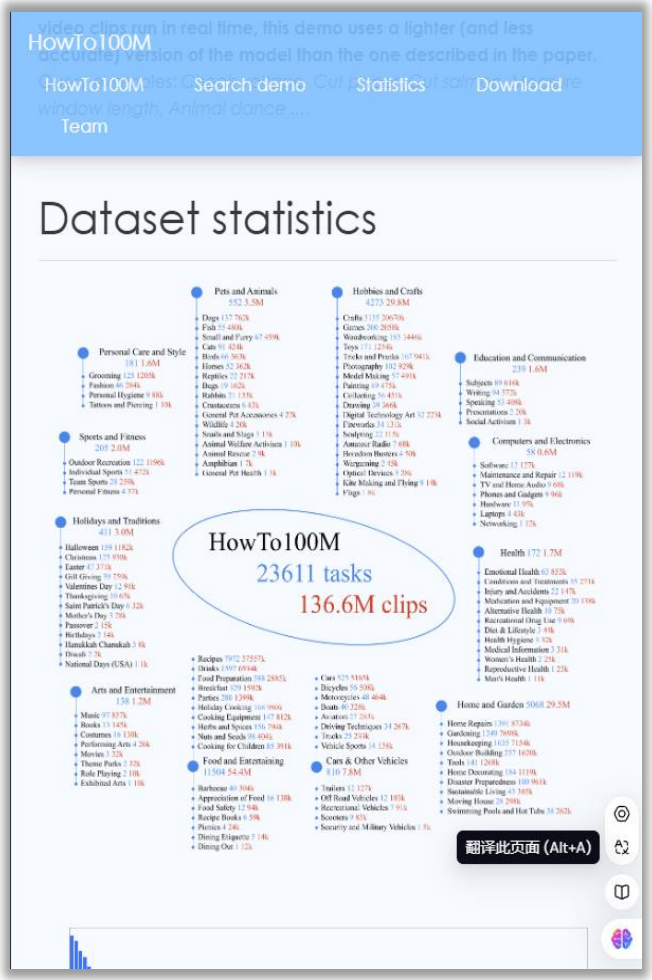
一些代表性的图文语料库

多模态数据集能促进大模型预训练，让模型具备更强的泛化能力

8.2.2 多模态数据集

多模态数据集被广泛用于多模态匹配（检索）任务。该任务是指通过处理和比较来自不同模态的数据，实现高效且精确的跨模态信息检索。





8.2.3 数据处理

数据处理的主要目标是剔除数据集中的噪声、冗余信息、无关数据，以及潜在有害内容。不合适的数据可能对语言模型的训练效果产生不利影响。

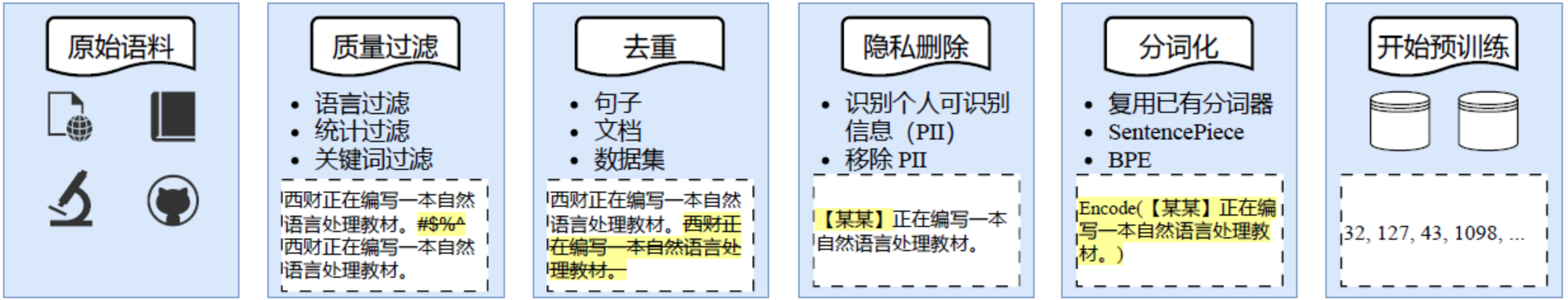


图 8.2: 典型的大型语言模型预训练的数据预处理流程的示意图

主要涵盖质量过滤（Quality Filtering）、去重（De-duplication）、隐私删除（Privacy Redaction）及分词化（Tokenization）四个步骤。

案例

DeepMind

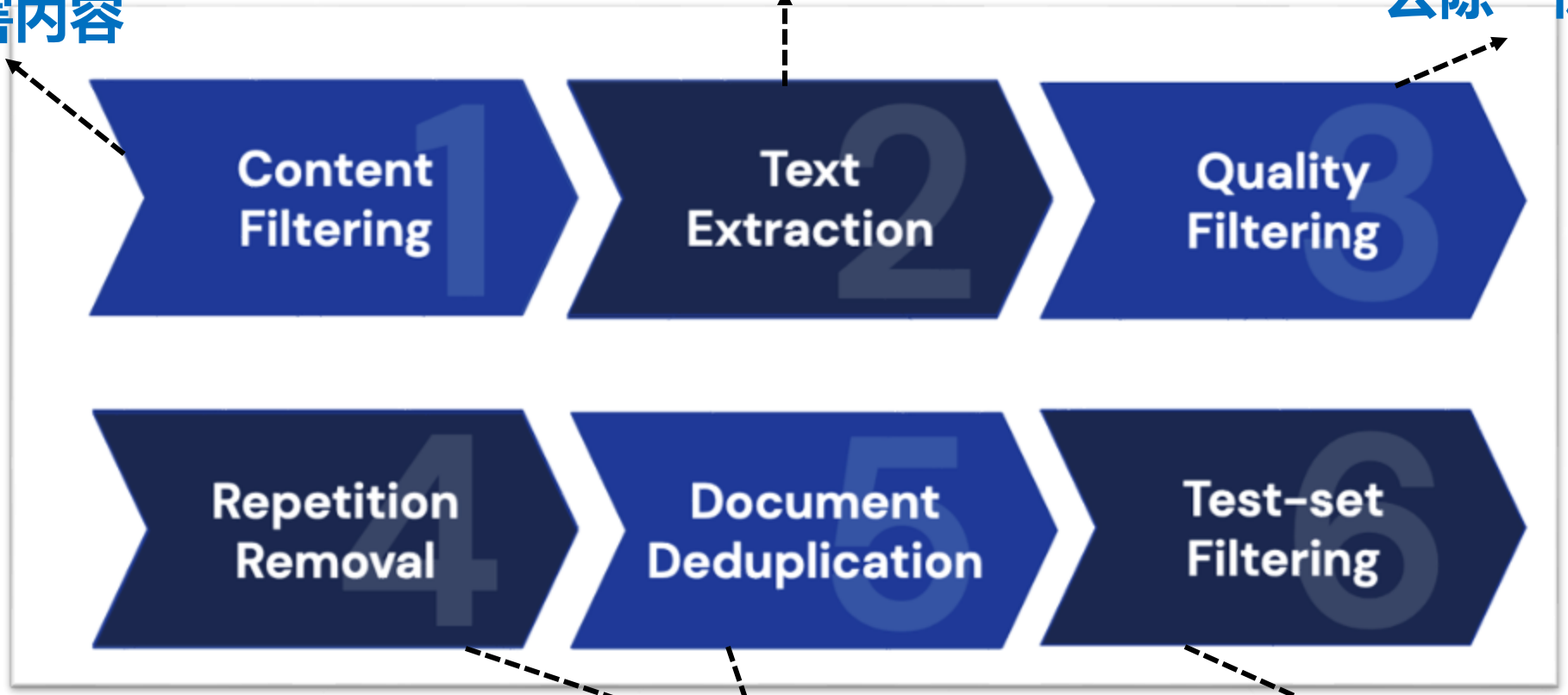
2021-12-08

Scaling Language Models: Methods, Analysis & Insights from Training *Gopher*

去掉HTML Tag，而保留内容

过滤有害内容

去除“低质”文本



https://arxiv.org/pdf/2112.11446

去除重复文本

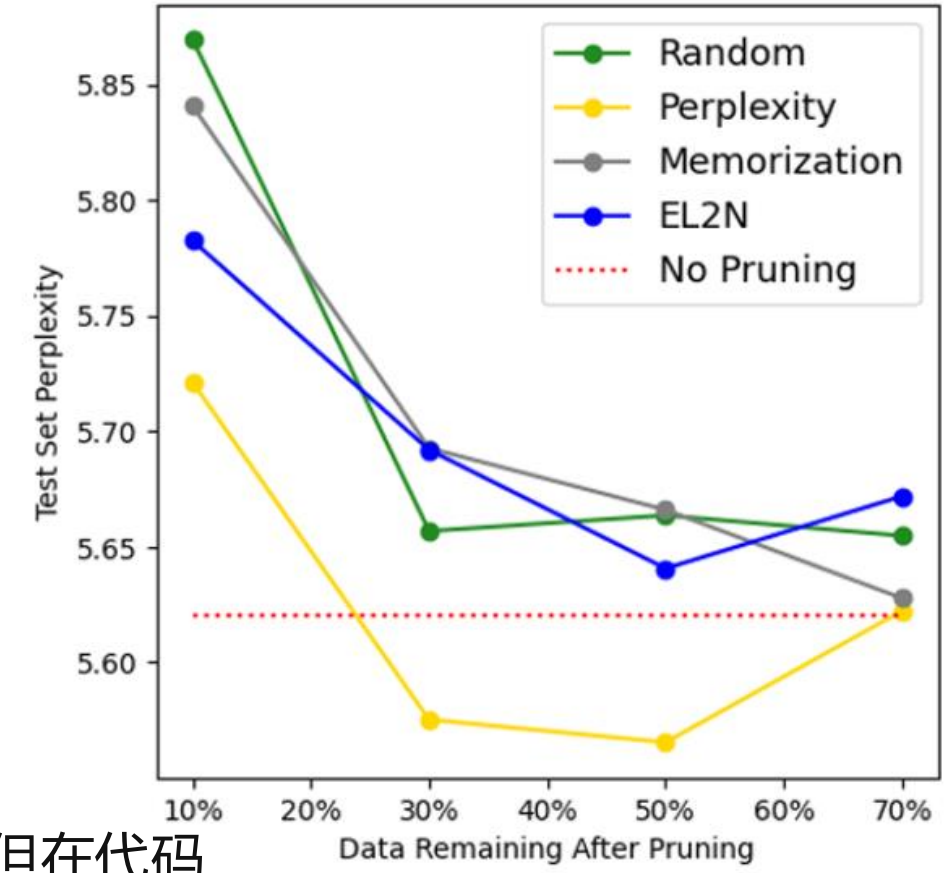
为了实验的严谨

8.2.3 数据处理 — 质量过滤

- **质量过滤常用方法**

分类器法	**训练一个小型分类器判断文本质量**
规则判断	**人为设置规则过滤低质量数据**
指标阈值	**设置特定指标选择阈值范围内数据**
聚类方法	**聚类过滤，筛选指定类别的数据**

- **使用小规模高质量样本训练轻量级模型**
  - Microsoft phi-1/phi-1.5/phi-2仅有1.3B-2.7B参数量，但在代码和推理任务上表现良好
- **过于激进的过滤可能会导致模型性能下降[1]**
  - 选定的过滤指标不能很好地衡量数据的质量
- **Perplexity作为简单的衡量指标效果最好[2]**



[1] Leo Gao. 2021. An empirical exploration in quality filtering of text data. arXiv preprint arXiv:2109.00698.
[2] Max Marion et al. 2023. When less is more: Investigating data pruning for pretraining llms at scale. arXiv preprint arXiv:2309.04564.

Perplexity 困惑度

A better model is better at predicting upcoming words, and so it will be less surprised by (i.e., assign a higher probability to) each word when it occurs in the test set.

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

困惑度越小表明模型能力越强

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

8.2.3 数据处理 — 数据去重

■ 数据去重的作用

- ❑ 提高训练效率[1]
- ❑ 减少训练集和测试集的重叠[1]
- ❑ 减轻模型的记忆现象 - 降低隐私攻击的成功率[2]

[1]"by combining fantastic ideas, interesting arrangements, and follow the current trends in the field of that make you more inspired and give artistic touches. We'd be honored if you can apply some or all of these design in your wedding. believe me, brilliant ideas would be perfect if it can be applied in real and make the people around you amazed!"

这段话在语料库中重复了61,036次！

[1] Katherine Lee et al., 2022. Deduplicating training data makes language models better. ACL.
[2] Nikhil Kandpal et al., 2022. Deduplicating training data mitigates privacy risks in language models. ICML.
[3] Amro Abbas et al., 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. arXiv preprint arXiv:2303.09540.

8.2.3 数据处理 — 数据去重

■ 数据去重的常用方法

- ❑ N-gram-and-hashing 是最为常用的方法
  - ▪ Line-level / document-level / both
- ❑ 神经网络模型
- ❑ 语义聚类方法：去除语义重复数据[3]

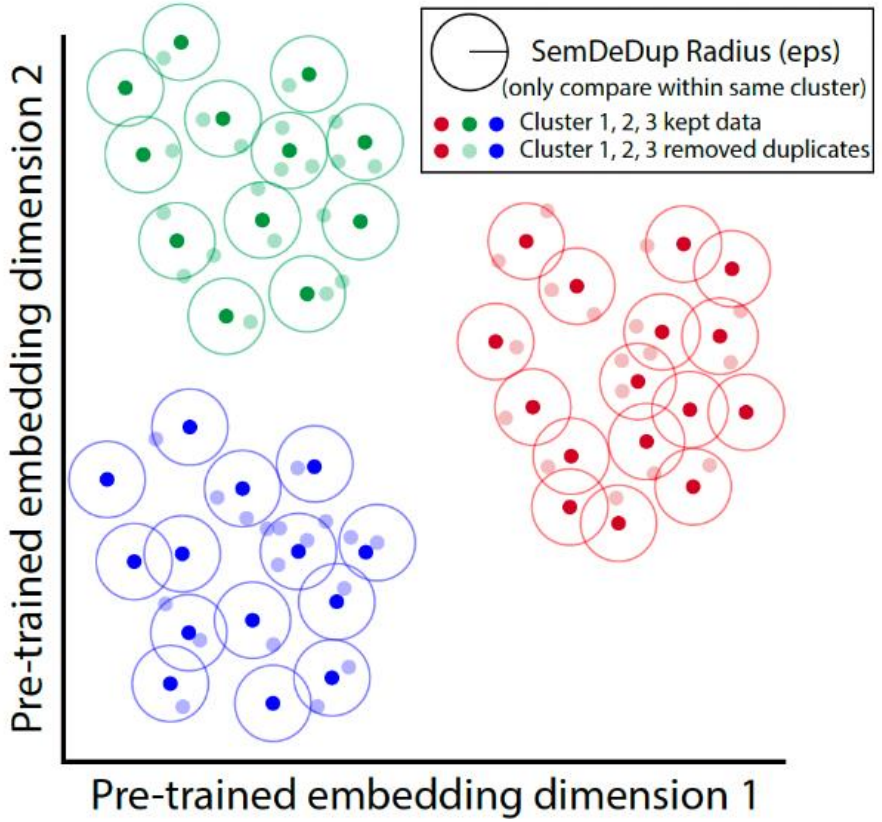
MinHash

4 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In computer science and data mining, **MinHash** (or the **min-wise independent permutations** locality sensitive hashing scheme) is a technique for quickly estimating how similar two sets are. The scheme was published by Andrei Broder in a 1997 conference,[1] and initially used in the AltaVista search engine to detect duplicate web pages and eliminate them from search results.[2] It has also been applied in large-scale clustering problems, such as clustering documents by the similarity of their sets of words.[1]



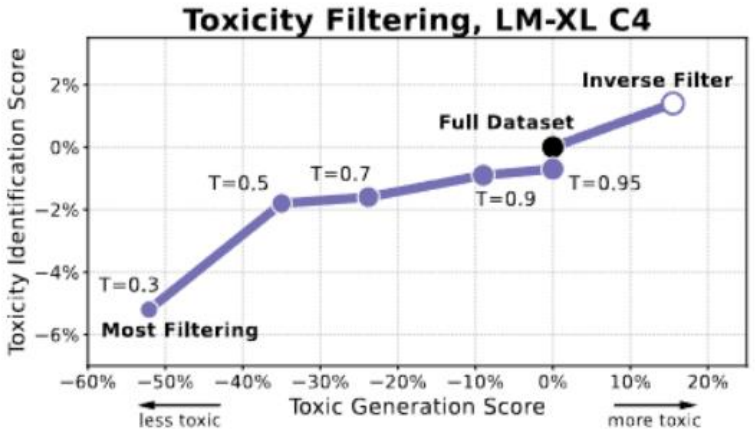
[1] Katherine Lee et al., 2022. Deduplicating training data makes language models better. ACL.
[2] Nikhil Kandpal et al., 2022. Deduplicating training data mitigates privacy risks in language models. ICML.
[3] Amro Abbas et al., 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. arXiv preprint arXiv:2303.09540.

8.2.3 数据处理 — 隐私删除

- 常用的过滤删除方法：

分类器法	训练一个小型分类器判断文本质量
规则判断	人为设置规则过滤低质量数据

- **隐私删除的类似任务 — 质量过滤**
- 质量过滤会增强模型的泛化性和对有害信息判别的能力[1]



	Wiki	Web	Books	Biomed	Academic	Common Sense	Contrast Sets	Average
Inverse T=0.5 (73%)	-5.0	-4.5	2.1	-2.2	-2.7	1.2	-6.4	-3.1
Full Dataset (100%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T=0.975 (91%)	1.2	0.7	-2.2	6.1	6.4	4.7	6.1	2.5
T=0.95 (84%)	-1.2	1.0	-4.0	3.7	-0.3	3.2	4.9	1.0
T=0.9 (73%)	-0.3	0.8	-3.5	1.8	1.0	1.9	6.8	1.2
T=0.7 (46%)	-1.2	0.8	-6.7	1.7	0.8	2.0	4.2	0.7

思考：隐私删除是否会增强模型泛化性和对是否是隐私信息判别的能力？

- 数据信息的过滤和删除可能会导致少数群体被边缘化的现象[2][3]

[1] S. Longpreet al. 2021A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity. 2023.
[2] Suchin Gururangan et al. 2022. Whose language counts as high quality? measuring language ideologies in text data selection. EMNLP 2021.
[3] Shangbin Feng et al. 2023. From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair nlp models. ACL 2023.

8.2.3 数据处理

■ 分词化

基于单词的分词器 (Word-based Tokenizer)

Let's	do	tokenization!

Let	's	do	tokenization	!

基于字符的分词器 (Character-based Tokenizer)

L	e	t	'	s	d	o	t	o	k	e	n	i	z	a	t	i	o	n	!

基于子词的分词器 (Subword-based Tokenizer)

Let's </w>	do</w>	token	ization</w>	!</w>

"*unfortunately*" = "*un*" + "*for*" + "*tun*" + "*ate*" + "*ly*"

子词分词算法依赖于这样一个原则，即不应将常用词拆分为更小的子词，而应将稀有词分解为有意义的子词。

例如，“annoyingly”可能被认为是一个罕见的词，可以分解为“annoying”和“ly”。这两者都可能作为独立的子词出现得更频繁，同时“annoyingly”的含义由“annoying”和“ly”的复合含义保持。

Source: https://huggingface.co/learn/nlp-course/

8.2.3 数据处理

■ 分词化

基于单词的分词器 (Word-based Tokenizer)

Let's	do	tokenization!

Let	's	do	tokenization	!

基于字符的分词器 (Character-based Tokenizer)

L	e	t	'	s	d	o	t	o	k	e	n	i	z	a	t	i	o	n	!

基于子词的分词器 (Subword-based Tokenizer)

Let's </w>	do</w>	token	ization</w>	!</w>

"unfortunately" = "un" + "for" + "tun" + "ate" + "ly"

字节对编码标记化(Byte-level BPE)

Byte Pair Encoding Data Compression Example

aaabdaaabac

aaabdaaabac Replace Z = aa

ZabdZabac Replace Y = ab

ZYdZYac Replace X = ZY

XdXac Final compressed string

Replacement Table

Byte pair	Replacement
X	ZY
ab	Y
aa	Z

WordPiece Tokenizer

原始文本： I have a new GPU.
标记文本：['i', 'have', 'a', 'new', 'gp', '## u', '。']

Unigram Tokenizer (SentencePiece or Unigram)

Source: https://huggingface.co/learn/nlp-course/

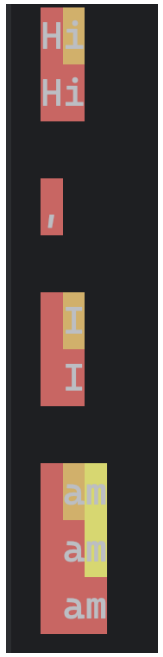
8.2.3 数据处理

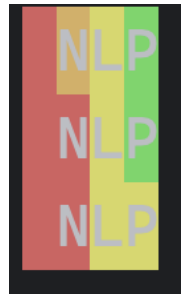
■ BPE

```python
import tiktoken

enc = tiktoken.encoding_for_model("gpt-4o")

print(enc.encode("你好, 我正在学习NLP"))
```

[177519, 40824, 70104, 64550, 45, 19318]

```python
from tiktoken._educational import *

enc = SimpleBytePairEncoding.from_tiktoken("cl100k_base")
tokens = enc.encode("Hi, I am learning NLP")
```



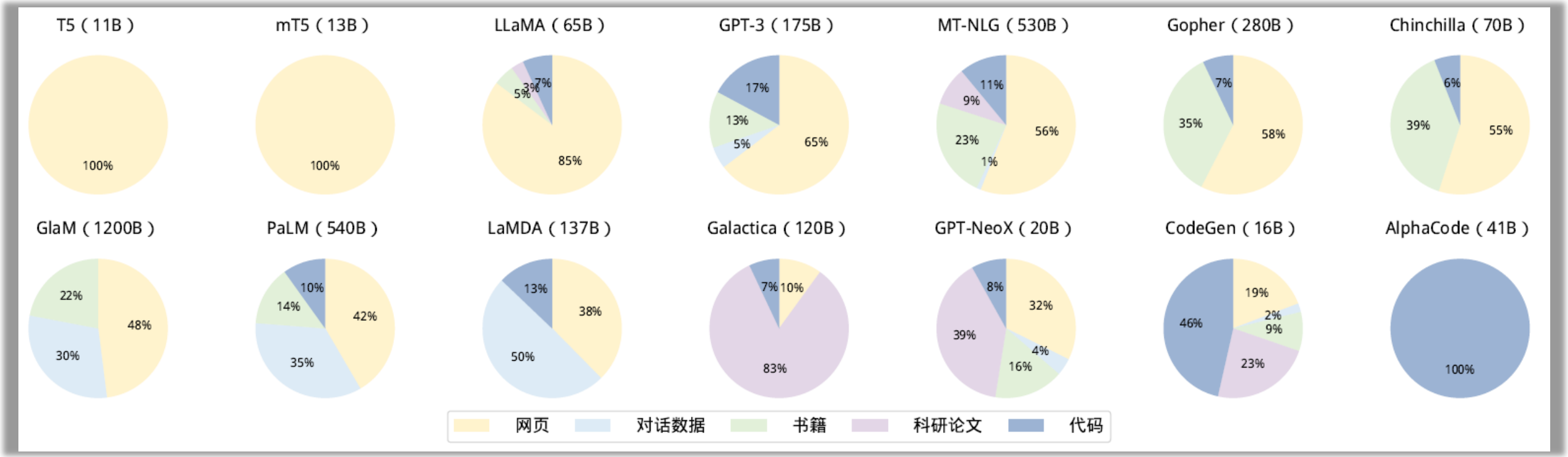


Source: https://huggingface.co/learn/nlp-course/

8.2.4 模型性能关系

■ 数据源的多样性

数据分布对模型性能具有影响。Gopher 团队进行了一系列消融实验 [138]，发现增加**书籍的数据比例**能够提高模型捕捉文本**中长期依赖关系**的能力 [127]。当某一**特定领域**的训练数据**过多时**，也可能会**降低** LLMs 在其他领域的**泛化能力** [164, 138]。



Source：https://transformers.run/c4/c15_pretrain_llms

8.2.4 模型性能关系

■ 预训练数据量 — Scaling Laws

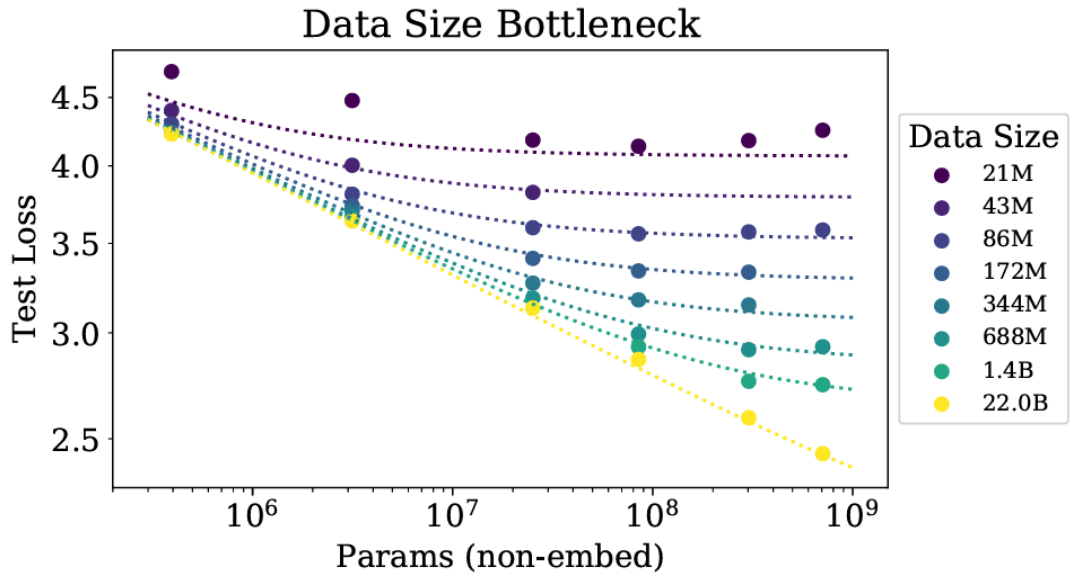
OpenAI Kaplan's Scaling Laws

- 在给定训练计算量的情况下，模型性能和数据数量符合幂律关系

$$L(N) = (N_c/N)^{\alpha_N};\ \alpha_N \sim 0.076,\quad N_c \sim 8.8 \times 10^{13}\ \text{(non-embedding parameters)}$$

$$L(D) = (D_c/D)^{\alpha_D};\ \alpha_D \sim 0.095,\quad D_c \sim 5.4 \times 10^{13}\ \text{(tokens)}$$

- 随着训练计算量的增加，数据数量和模型大小应该同步增长，但是模型大小的增速应该更快



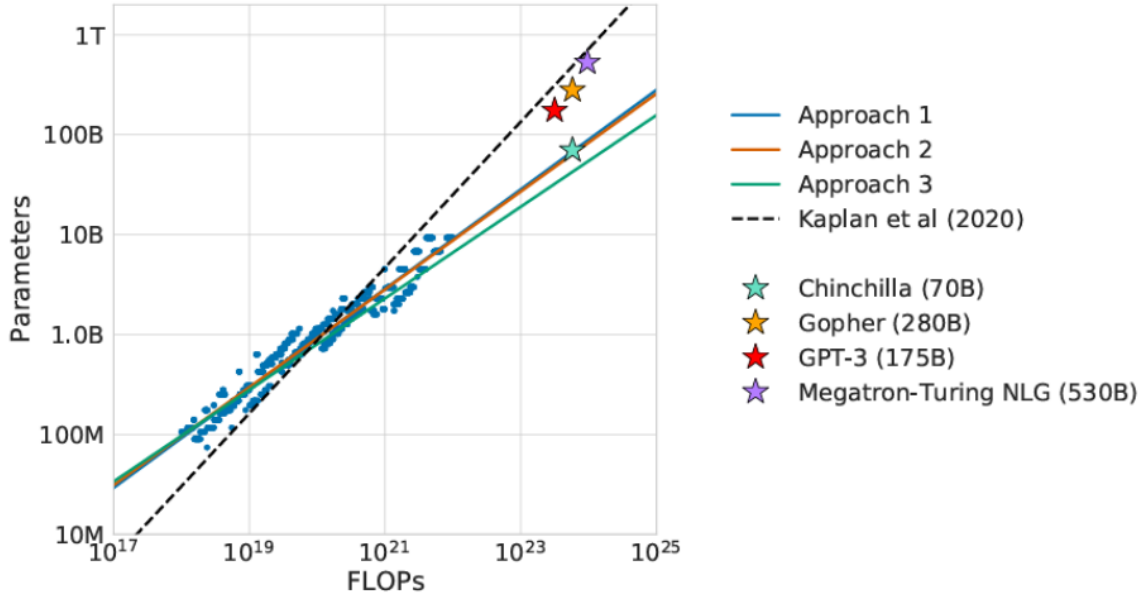
Source：https://arxiv.org/abs/2001.08361

DeepMind Chinchilla Scaling Laws

- 新的 scaling law

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

- 随着训练计算量增长，数据数量和模型大小应该以几乎一致的速度增长 $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$



Source：https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html

8.2.4 模型性能关系

■ 预训练数据质量

高质量的预训练数据是优化 LLMs 性能的关键要素之一

■ 数据低质量表现形式

噪声数据	**对模型训练和性能产生负面影响**
有毒内容	**生成毒性文本的能力下降，而识别毒性的能力提高**
重复内容	**导致模型性能在初始阶段下降，影响模型在上下文信息中进行有效信息抽取的能力**

思考1 什么样的数据集才是高质量的预训练数据？

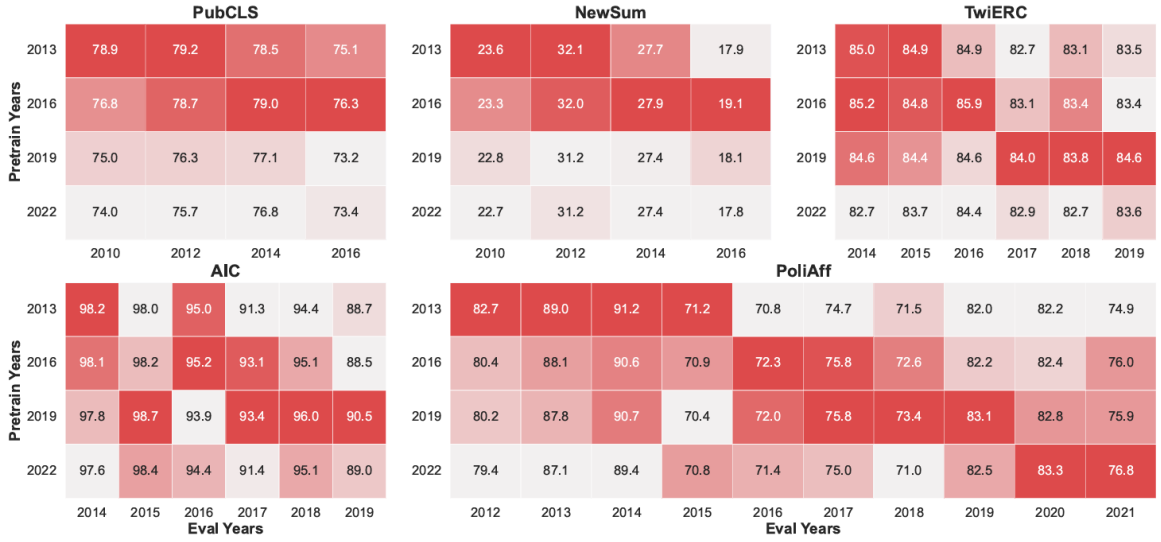
数据源1： https://blog.csdn.net/u011559552/article/details/142217358
数据源2： https://github.com/lonePatient/awesome-pretrained-chinese-nlp-models
数据源3： https://gitee.com/oschina/awesome-llm

思考2 还有什么因素会影响大模型预训练性能？

8.2.4 模型性能关系

■ 预训练数据质量 — 数据年龄（时效性）

- ❑ 预训练数据集和评估数据集之间的时间不对齐会导致模型性能下降
- ❑ 时间不对齐问题不容易用微调来解决
- ❑ 时间不对齐导致的性能下降在更大的模型上更显著



Domain	Task	Finetuning LM-Small TD	Finetuning LM-Small r	Finetuning LM-XL TD	Finetuning LM-XL r	Pretraining LM-Small TD	Pretraining LM-Small r	Pretraining LM-XL TD	Pretraining LM-XL r
News	PubCLS	5.82	0.84	5.63	0.80	0.02	$0.01^{\dagger}$	0.59	0.67
	NewSum	0.80	0.82	2.91	0.92	-0.31	-0.29	0.73	0.45
Twitter	PoliAff	3.74	0.84	4.93	0.89	0.50	0.21	0.28	0.56
	TwiERC	0.49	0.73	0.53	0.82	0.05	0.27	0.23	0.72
Science	AIC	0.94	0.83	0.24	0.36	0.11	$0.18^{\dagger}$	0.23	0.66
	Mean	2.36	0.81	2.84	0.76	0.08	0.07	0.41	0.61

综上所述，预训练数据的来源、数量、质量和时效都是影响 LLMs 性能的重要因素。需要进行细致的优化，从而提升 LLMs 在泛化场景性能

[1] Shayne Longpre et al. 2023. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. arXiv preprint arXiv:2305.13169.

本章内容

8.3.1 预训练任务

https://www.bilibili.com/video/BV18z4y1J763/?spm_id_from=333.337.search-card.all.click&vd_source=dcdf7189a3746be0a3a8f3a14bd3df03

8.3.1 预训练任务

https://www.bilibili.com/video/BV18z4y1J763/?spm_id_from=333.337.search-card.all.click&vd_source=dcdf7189a3746be0a3a8f3a14bd3df03

8.3.1 预训练任务

https://www.bilibili.com/video/BV18z4y1J763/?spm_id_from=333.337.search-card.all.click&vd_source=dcdf7189a3746be0a3a8f3a14bd3df03

8.3.1 预训练任务

■ 语言建模

Next-token prediction

$$\mathcal{L}_{LM}(\mathbf{x}) = \sum_{i=1}^{n} \log P\left(x_i \mid x_{<i}\right)$$

这样的思想事实上是在传达只要模型足够大，学到的知识足够多，任何有监督任务都可以通过无监督的方式来完成，即任何任务都可以视作生成任务。

8.3.1 预训练任务

■ 去噪自编码

通过将输入数据添加噪声，然后训练模型还原出原始的、无噪声的输入数据，去噪自编码器能够学习到数据的鲁棒表示。

$$\mathcal{L}_{DAE}(\mathbf{x}) = \log P(\tilde{\mathbf{x}} \mid \mathbf{x} \backslash \tilde{\mathbf{x}})$$

假设输入是 $\mathbf{x}\backslash\tilde{\mathbf{x}}$, 其中 $\tilde{\mathbf{x}}$ 是带有随机替换片段的损坏文本，语言模型被训练用来恢复被替换的词元$\tilde{\mathbf{x}}$。DAE 的训练目标是使重建的输出 $\tilde{\mathbf{x}}$ 尽可能接近原始的无噪声输入 $\mathbf{x}$。形式上，DAE 的训练目标表示如下：

然而，相比于 LM 任务，DAE 任务在实现上似乎更加复杂 [214]。采用 DAE 作为预训练目标的 LLMs包括 T5 和 GLM-130B。

8.3.2 优化参数设置

❑ 批量训练

- 影响**训练速度**、**收敛性**、**泛化能力**和**资源使用效率**。大批量可**减少梯度噪声**，使训练过程更加**稳定**，**加快收敛**
- 在训练过程中动态**增加批量大小**，能够有效地**提升训练稳定性**

❑ 学习率

- 为平衡**训练速度**和**稳定性**，现有的大模型通常在预训练期间采用**动态的学习率调度策略**，包括预热（Warm-up）和衰减（Learning Rate Decay） 策略

❑ 优化器

- SGD**计算效率高**、**收敛速度较慢**
- RMSprop自适应调整，**收敛快**、**稳定性高**
- Adam 通过计算梯度的一/二阶矩估计来自动调整每个参数的学习率
- AdamW（带权重衰减 Adam）**收敛快**、**对超参数不敏感**、**对复杂损失函数适应性好**

模型	批次大小	学习率（预热 → 峰值 → 衰减）	优化器	精度类型	权重衰减	梯度裁剪
GPT-3	32K → 3.2M	预热 → 6×10^{-5} → 余弦	Adam	FP16	0.1	1.0
PanGu-α	-	2×10^{-5}	Adam	-	0.1	-
OPT	2M	预热 → 1.2×10^{-4} → 手动	AdamW	FP16	0.1	-
PaLM	1M → 4M	1×10^{-2} → 平方根倒数	Adafactor	BF16	lr^2	1.0
BLOOM	4M	预热 → 6×10^{-5} → 余弦	Adam	BF16	0.1	1.0
MT-NLG	64K → 3.75M	预热 → 5×10^{-5} → 余弦	Adam	BF16	0.1	1.0
Gopher	3M → 6M	预热 → 4×10^{-5} → 余弦	Adam	BF16	-	1.0
Chinchilla	1.5M → 3M	预热 → 1×10^{-4} → 余弦	AdamW	BF16	-	-
Galactica	2M	预热 → 7×10^{-6} → 余弦	AdamW	-	0.1	1.0
LaMDA	256K	-	-	BF16	-	-
Jurassic-1	32k → 3.2M	预热 → 6×10^{-5}	-	-	-	-
LLaMA-2	4M	预热 → 1.5×10^{-4} → 余弦	AdamW	-	0.1	1.0
Pythia	2M	预热 → 1.4×10^{-4} → 余弦	Adam	FP16	0.01	1.0
Baichuan-2	-	预热 → 1.5×10^{-4} → 余弦	AdamW	BF16	0.1	0.5
Qwen-1.5	4M	预热 → 3×10^{-4} → 余弦	AdamW	BF16	0.1	1.0
InternLM-2	5M	预热 → 3×10^{-4} → 余弦	AdamW	-	0.1	-
Falcon	预热 → 4M	预热 → 1.25×10^{-4} → 余弦	AdamW	BF16	0.1	0.4
DeepSeek	18M	预热 → 3.2×10^{-4} → 余弦	AdamW	BF16	0.1	1.0
Yi	256K	1×10^{-5}	AdamW	BF16	0.1	1.0
YuLan	4.5M	预热 → 3×10^{-4} → 余弦	Adam	BF16	0.1	1.0
GLM-130B	0.4M → 8.25M	预热 → 8×10^{-5} → 余弦	AdamW	FP16	0.1	1.0
T5	64K	1×10^{-2} → 平方根倒数	AdaFactor	-	-	-

8.3.2 优化参数设置

■ 稳定训练

权重衰减 (Weight Decay)

通过在损失函数中添加与模型权重平方成正比的项，减少模型复杂度并防止过拟合。

鼓励模型在训练过程中保持较小的权重值，有助于提高模型的泛化能力，并增强训练稳定性。

梯度裁剪 (Gradient Clipping)

梯度裁剪则用于防止梯度爆炸

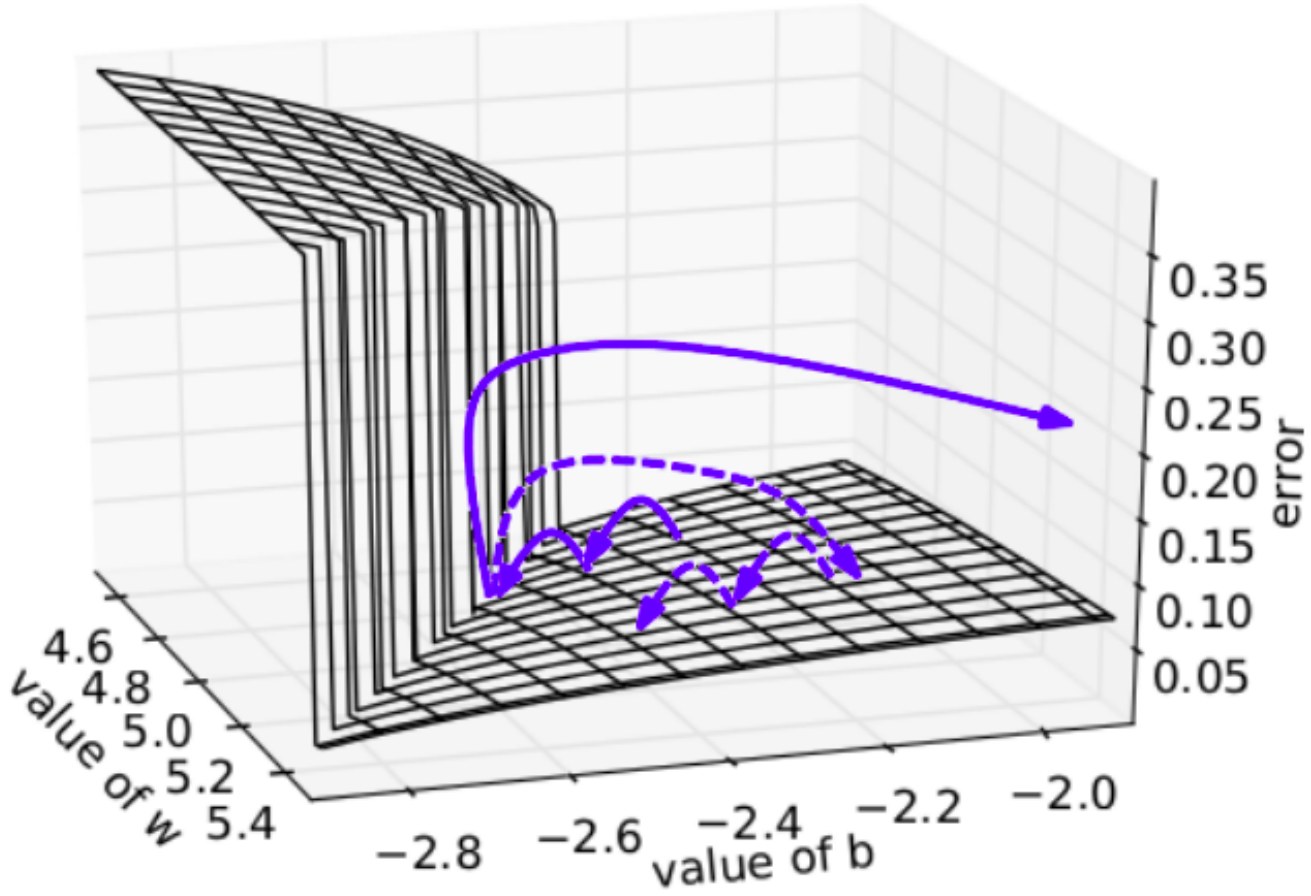
在训练过程中，由于梯度累积，可能会出现异常大的梯度， 导致权重更新时步长过大，破坏训练稳定性。通过设置一个阈值限制梯度大小，当梯度范数超过该阈值时进行缩放，避免极端步长变化，从而防止训练中的不稳定现象

其它稳定训练的策略

随着 LLMs 规模扩大，训练中损失突变的情况变得更加普遍，加剧了训练的不稳定性。为了解决该问题，PaLM 和 OPT 模型采用了一种简单但有效的策略，即在损失突增发生之前，从较早的检查点重新开始训练，并跳过可能导致问题的数据。该策略提高了训练稳定性。

8.3.2 优化参数设置

■ 稳定训练：梯度裁剪



https://arxiv.org/pdf/1211.5063

8.3.2 优化参数设置

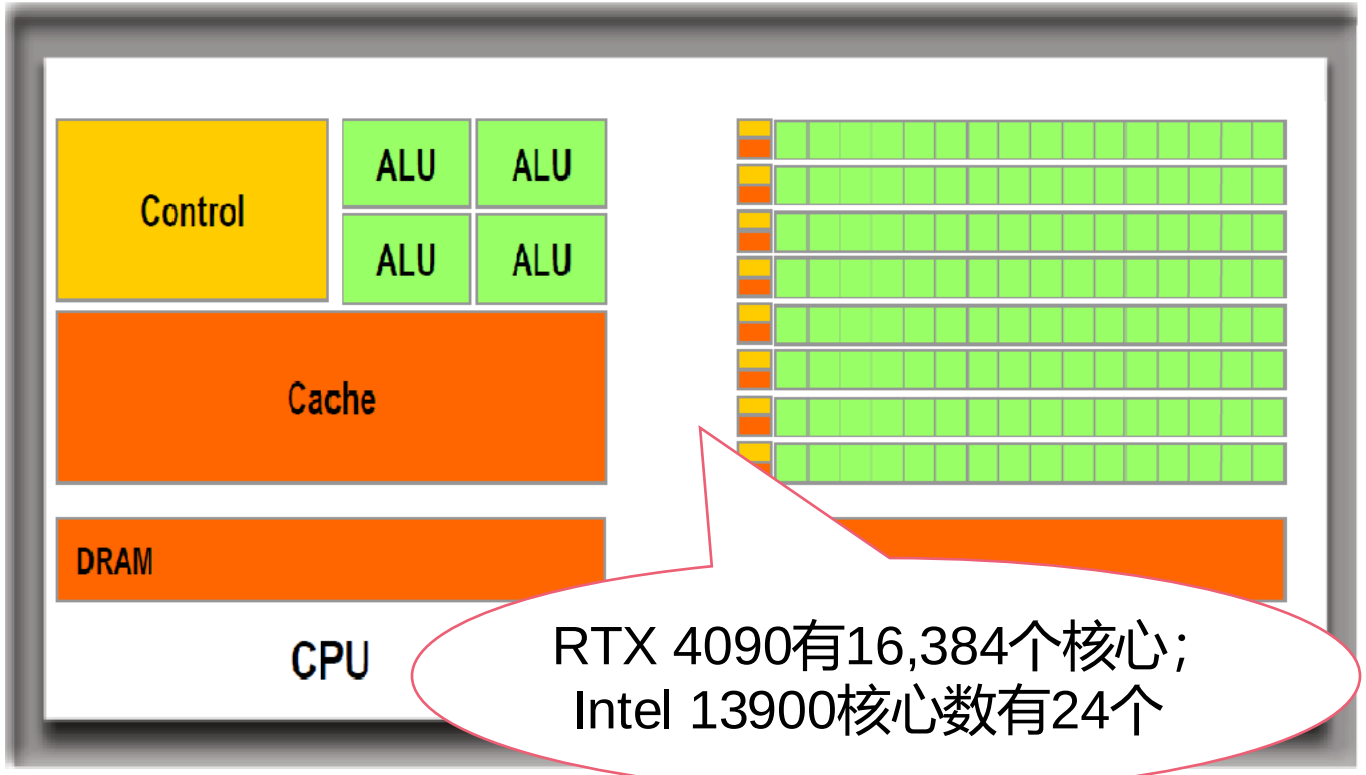
CPU:

- 由控制单元（Contraol）、运算单元（ALU）、缓存（Cache）等组成；
- 对比GPU，CPU的控制单元数量较小
- 适合处理复杂性任务

GPU:

- 拥有大量运算单元
- 具有强大的并行计算能力
- 简化的控制器，无法处理复杂性较高的任务

CPU 与 GPU 架构对比

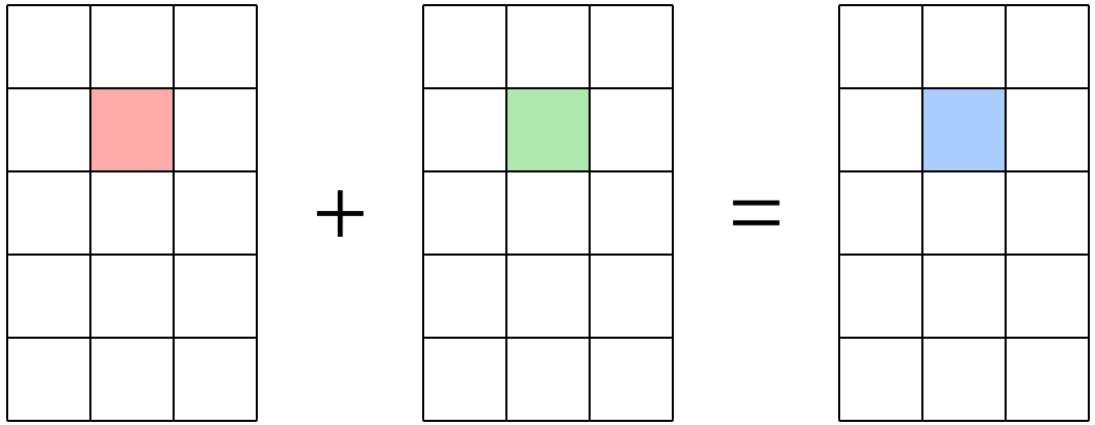


深度神经网络训练涉及大量的矩阵（张量）运算。由于图形处理器（GPU）架构包含大量的计算单元，这些单元能够并行处理矩阵运算，相较于中央处理器（CPU），GPU提供了更高的计算效率。

8.3.2 优化参数设置

■ CUDA编程示例

```
void vecAdd(float *a, float *b, int n) {
  for (int i = 0; i < n; i++) {
    b[i] = a[i] + b[i];
  }
}
```



```
// Kernel function to add the elements of two arrays
__global__ void add(int n, float *x, float *y) {
  for (int i = 0; i < n; i++)
    y[i] = x[i] + y[i];
}
```

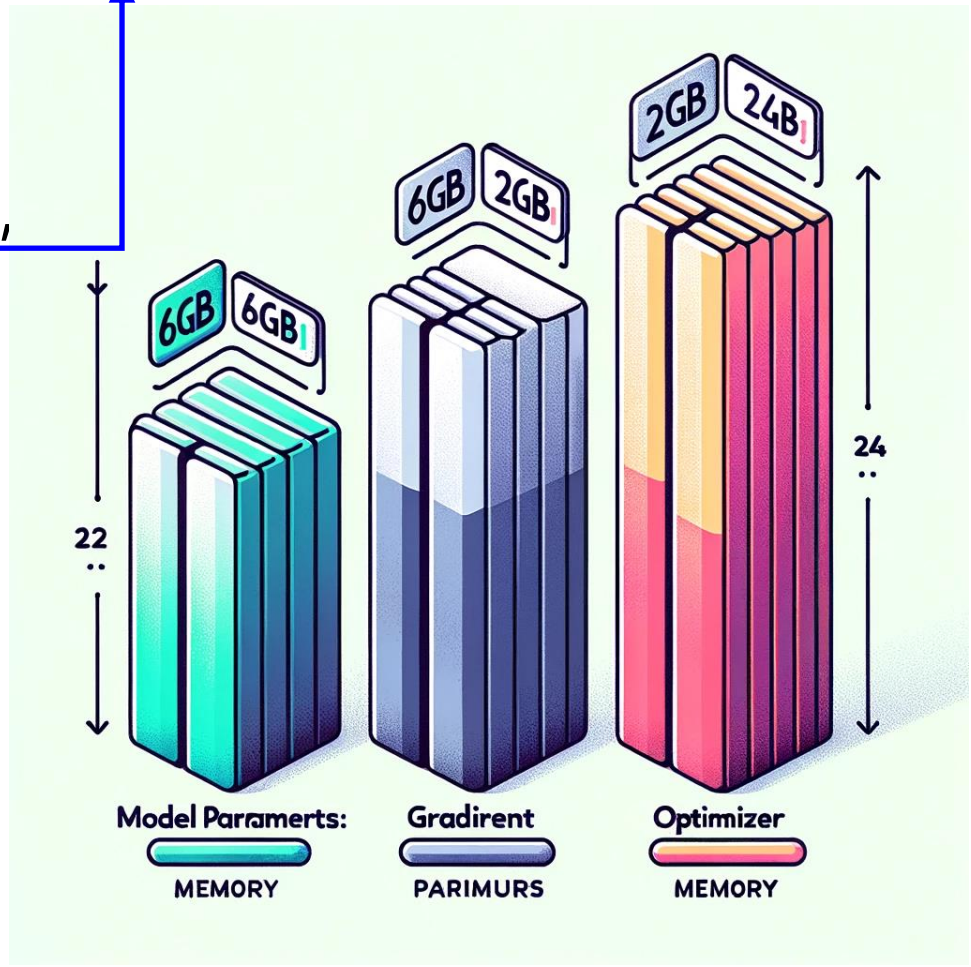
8.3.3 可扩展训练技术

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

❑ **ChatGLM-6B 模型参数：**

- ChatGLM-6B架构：隐层-4096，中间层-11008，block数-32，数据类型：Int8
- 模型参数所占内存：6B x 1 bytes = 6GB
- 梯度所占内存：6B x 1 bytes = 6GB
- 优化器（AdamW）参数：2倍模型参数，6GB x 2 = 12GB

- 总计：约 24GB 显存



思考题 如果训练130B的模型呢？

8.3.3 可扩展训练技术

■ 量化（quantization）

Qwen / **Qwen-7B-Chat-Int8** ♡ like 8 Follow Qw

Text Generation | Transformers | Safetensors | Chinese

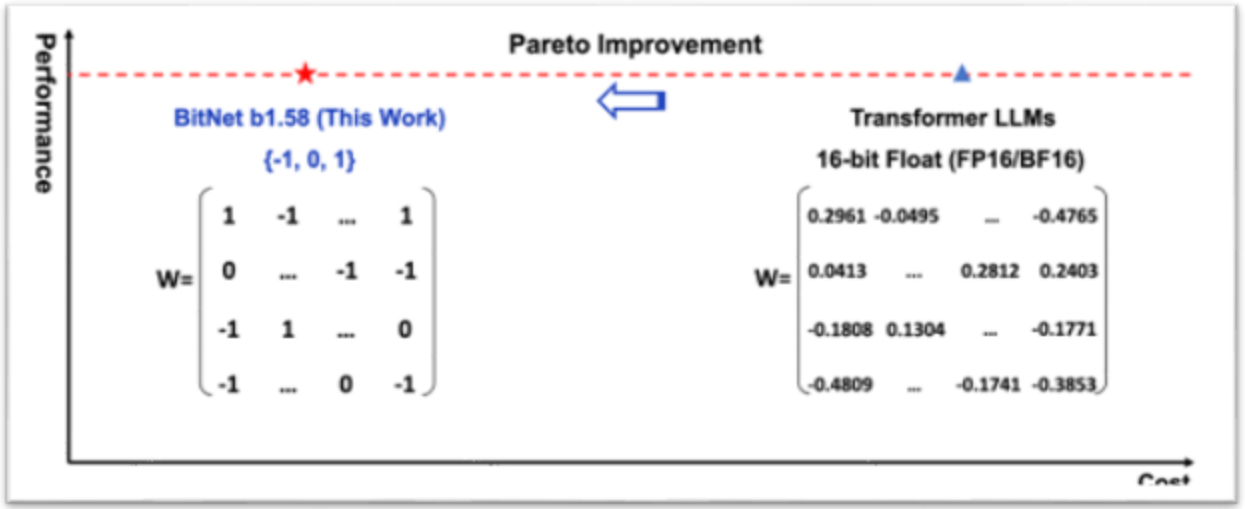
Quantization is a generic method that refers to the compression of data into a smaller space. 比如 32-bit float量化得到int-8。

$$\text{scaled\_value} = \frac{(\text{float\_value} - \text{min\_float})}{(\text{max\_float} - \text{min\_float})} \times 255$$

The Era of 1-bit LLMs:
All Large Language Models are in 1.58 Bits

Shuming Ma* Hongyu Wang* Lingxiao Ma Lei Wang Wenhui Wang
Shaohan Huang Li Dong Ruiping Wang Jilong Xue Furu Wei◇
https://aka.ms/GeneralAI

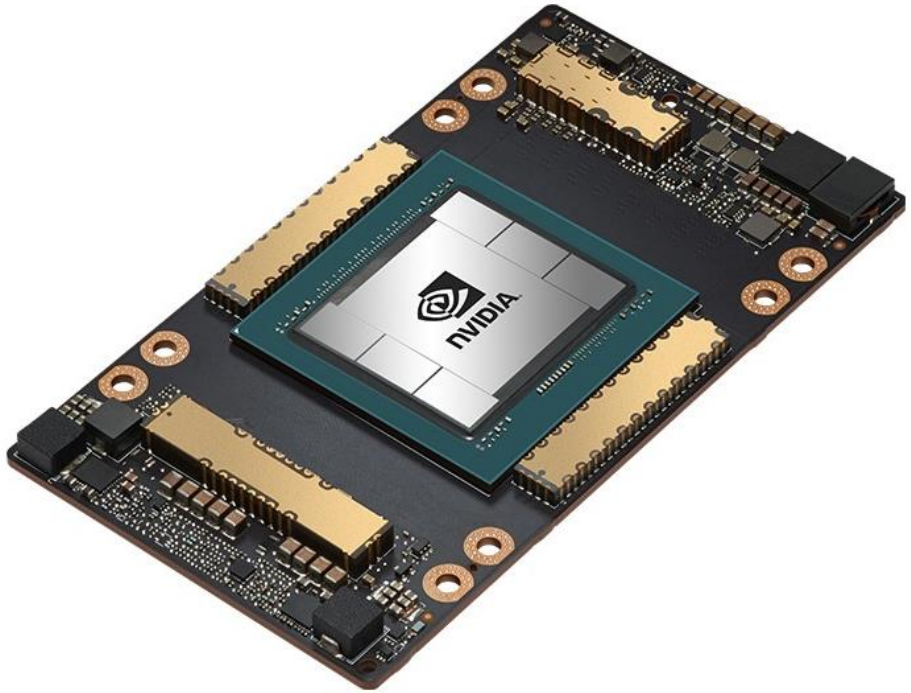
Abstract



https://arxiv.org/pdf/2402.17764v1

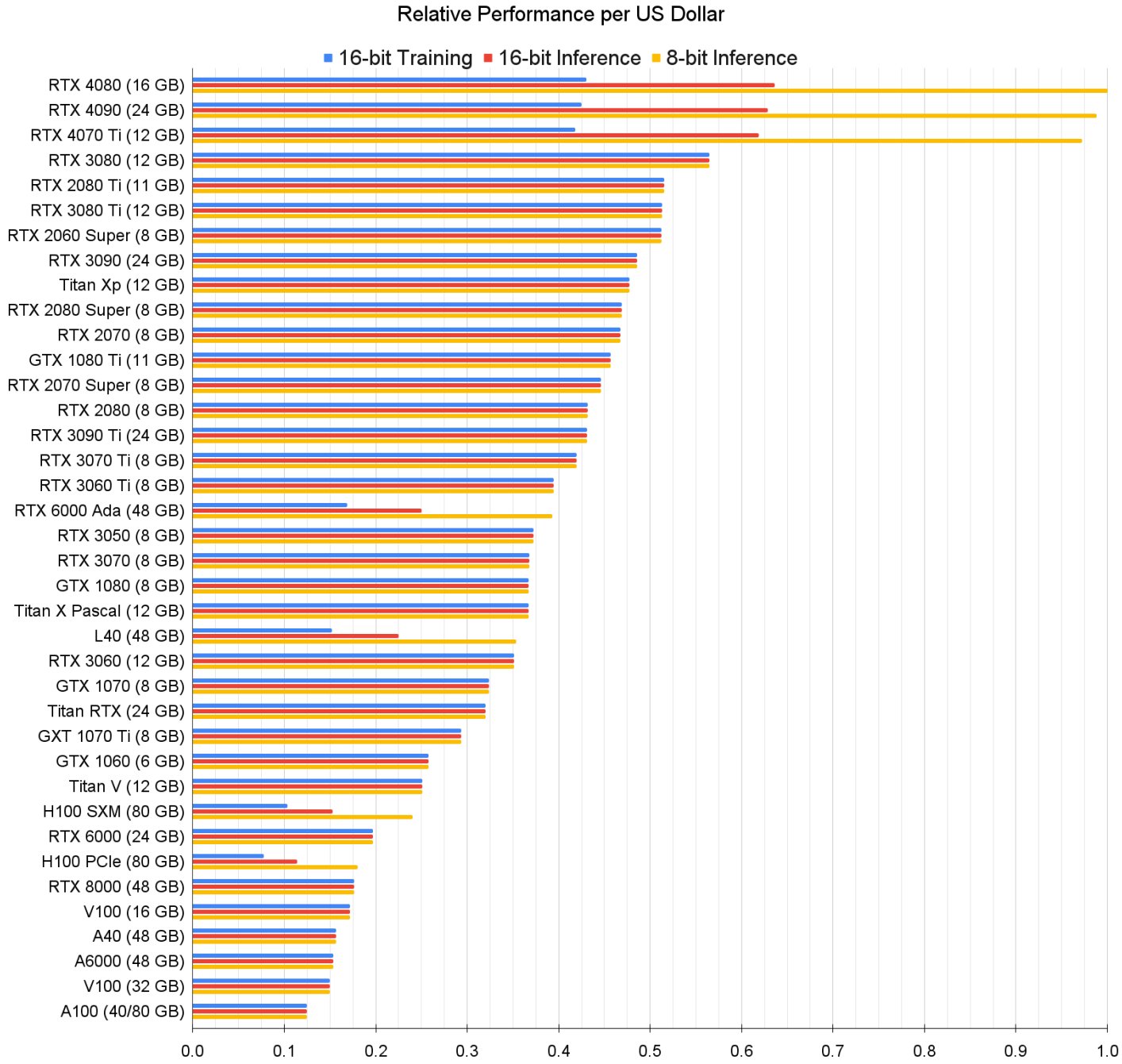
8.3.3 可扩展训练技术

■ 3D 并行



Nvidia-A100: 40GB 或 80 GB显存，但仍无法满足使用一张卡满足训练大模型

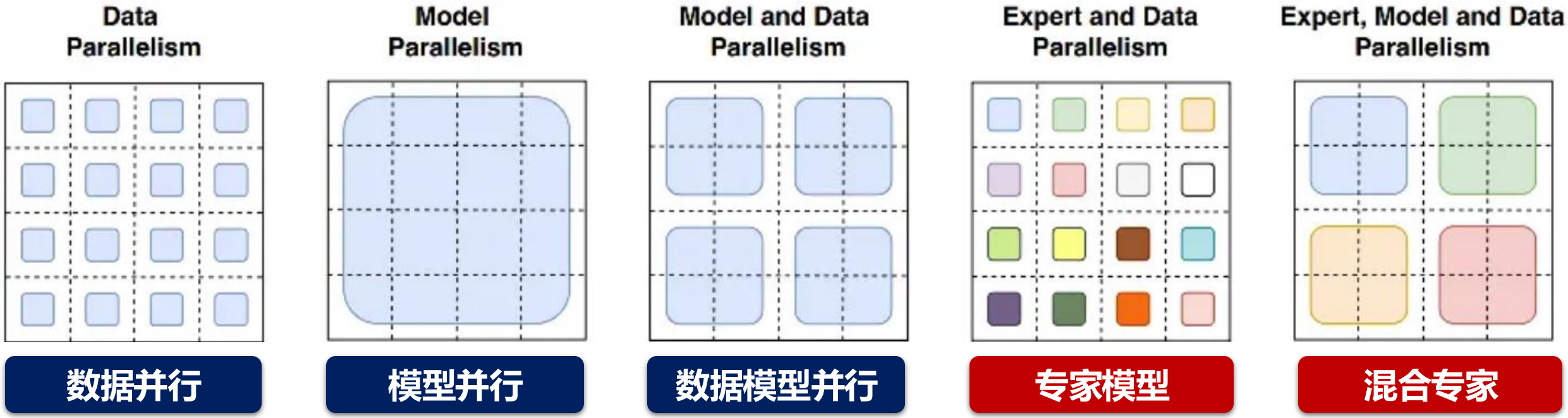
因此需要**可扩展训练技术**



8.3.3 可扩展训练技术

■ 3D 并行

大模型突破上百亿规模参数，传统的单机单卡模式已经无法满足超大模型的训练需求。需要单机多卡、甚至是多机多卡进行分布式大模型训练



利用AI集群，一般需要根据**硬件资源与数据/模型规模**的匹配情况，考虑对**计算任务**、**训练数据**和**模型进行划分**，从而进行分布式存储和分布式训练。

Source：https://juejin.cn/post/7289661052806594618

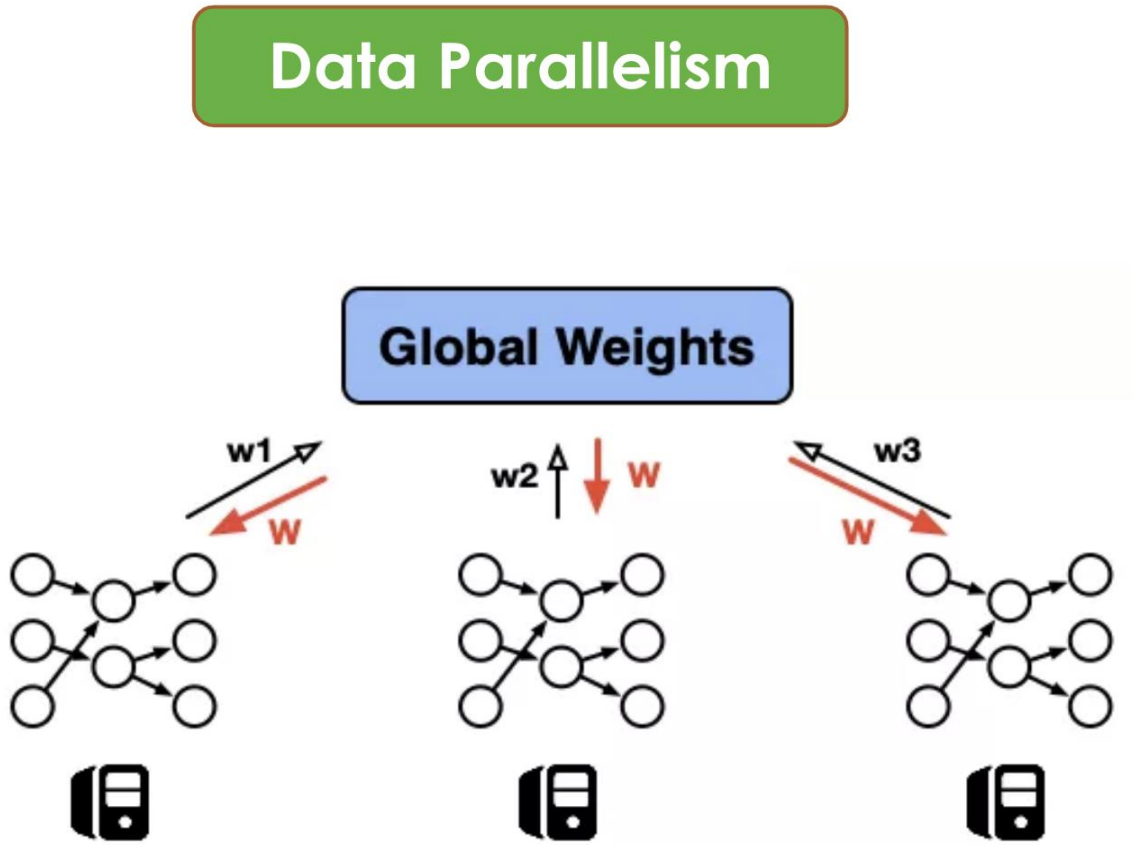
8.3.3 可扩展训练技术

■ 数据并行

■ 大语言模型并行训练方法—数据并行

数据并行的特点：

- 如果**模型太大**无法嵌入到**一台机器**，就将暂时未使用的参数**卸载回CPU**。
- 数据交换传输通常在后端进行（不干扰训练计算），每个Mini-batch计算结束后**worker需要同步梯度或权重**，以保证学习效率。

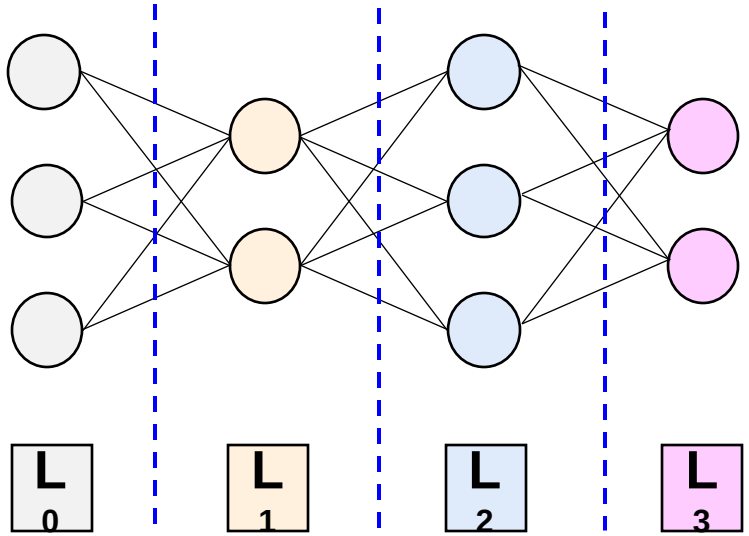


批量同步并行（BSP）：worker在每个Mini-batch结束时同步数据
—优点：保证了模型**权重传递**的**及时性**
—缺点：每台机器都必须排队**等待**其他机器发送**梯度**

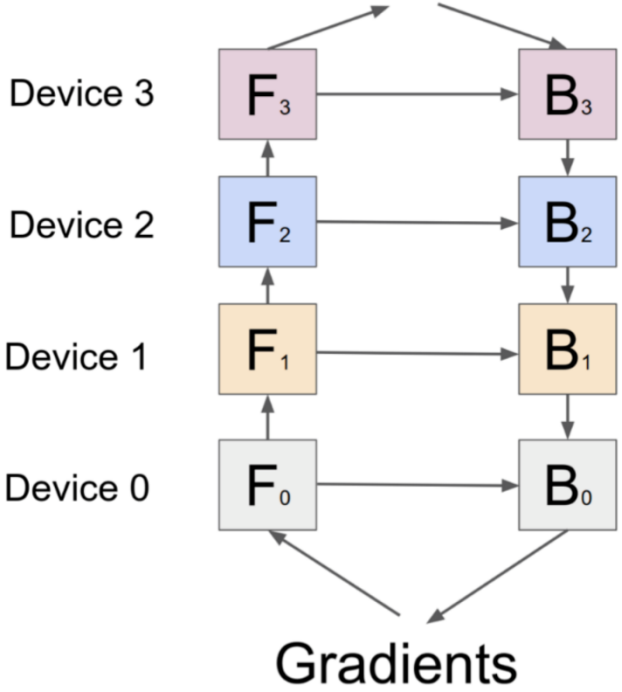
异步并行（ASP）：每个GPU采用异步方式处理数据，异步更新模型
—优点：**避免了**异构机器间的**相互等待**
—缺点：影响了**权重**传递的**时效**，**降低**了统计**学习效率**

8.3.3 可扩展训练技术

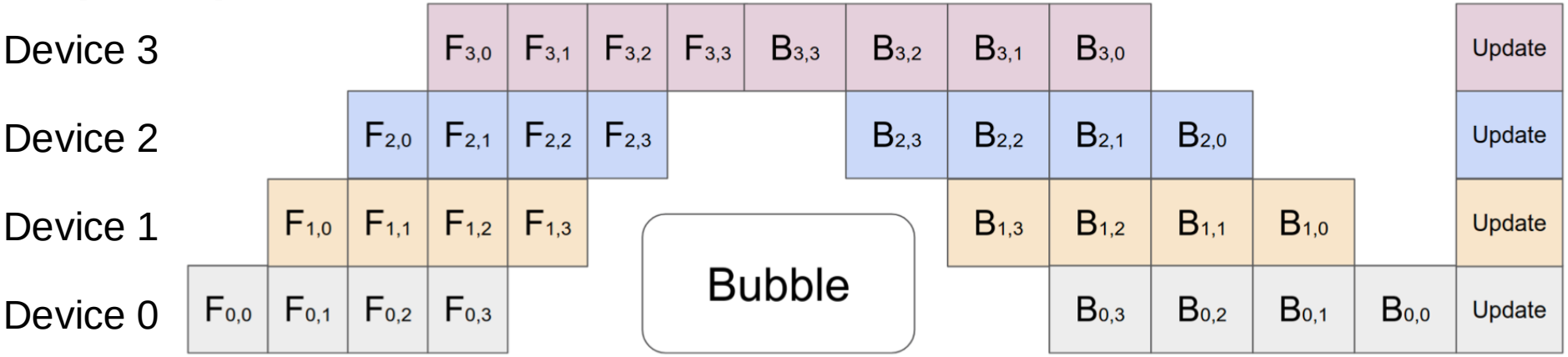
■ 模型并行



■ Workflow Loss



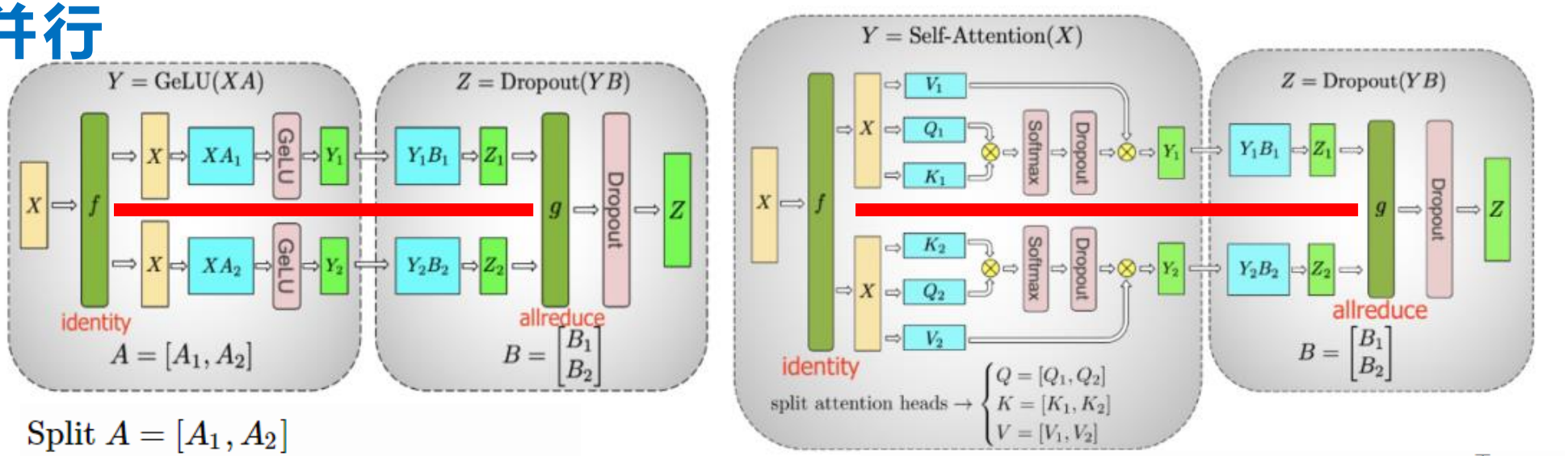
■ Pipeline parallelism



- Lilian Weng OpenAI应用人工智能研究负责人，北大校友，个人主页: https://lilianweng.github.io/lil-log/

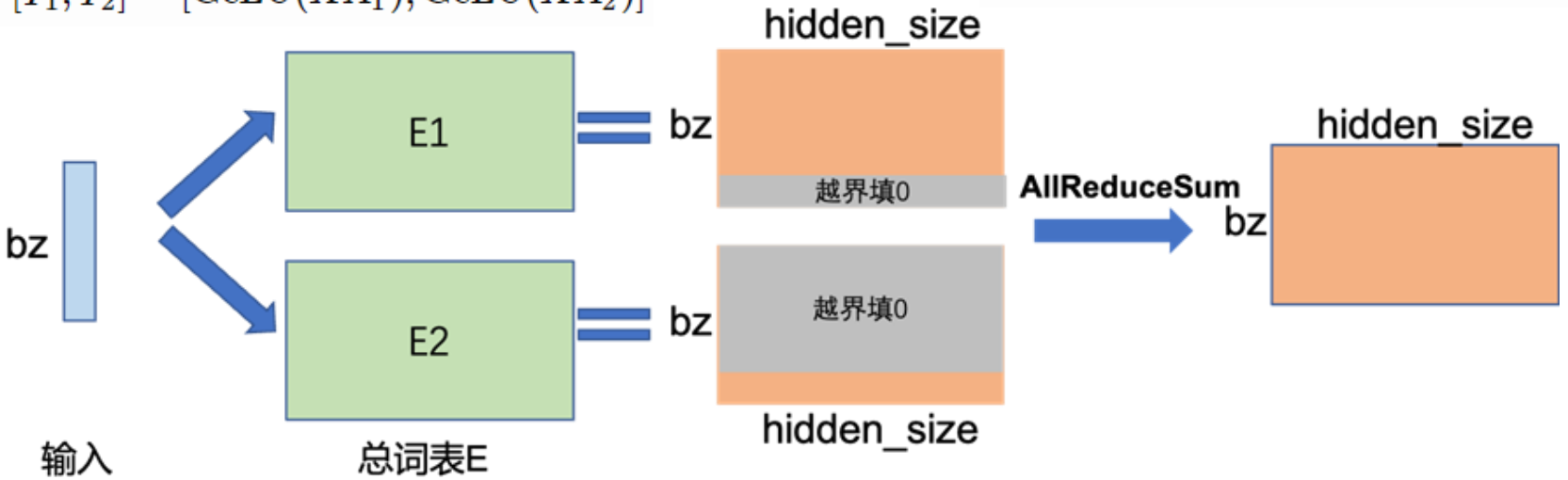
8.3.3 可扩展训练技术

■ 张量并行



$$\text{Split } A = [A_1, A_2]$$
$$Y = \mathrm{GeLU}(XA)$$
$$[Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$

$$\mathrm{Attention}(X, Q, K, V) = \mathrm{softmax}(\frac{(XQ)(XK)^\top}{\sqrt{d_k}})XV$$



- Lilian Weng OpenAI应用人工智能研究负责人，北大校友，个人主页: https://lilianweng.github.io/lil-log/

8.3.3 可扩展训练技术

数据并行+模型并行+张量并行 = NVDIA + Microsoft PTD-P解决方案 (APEX)

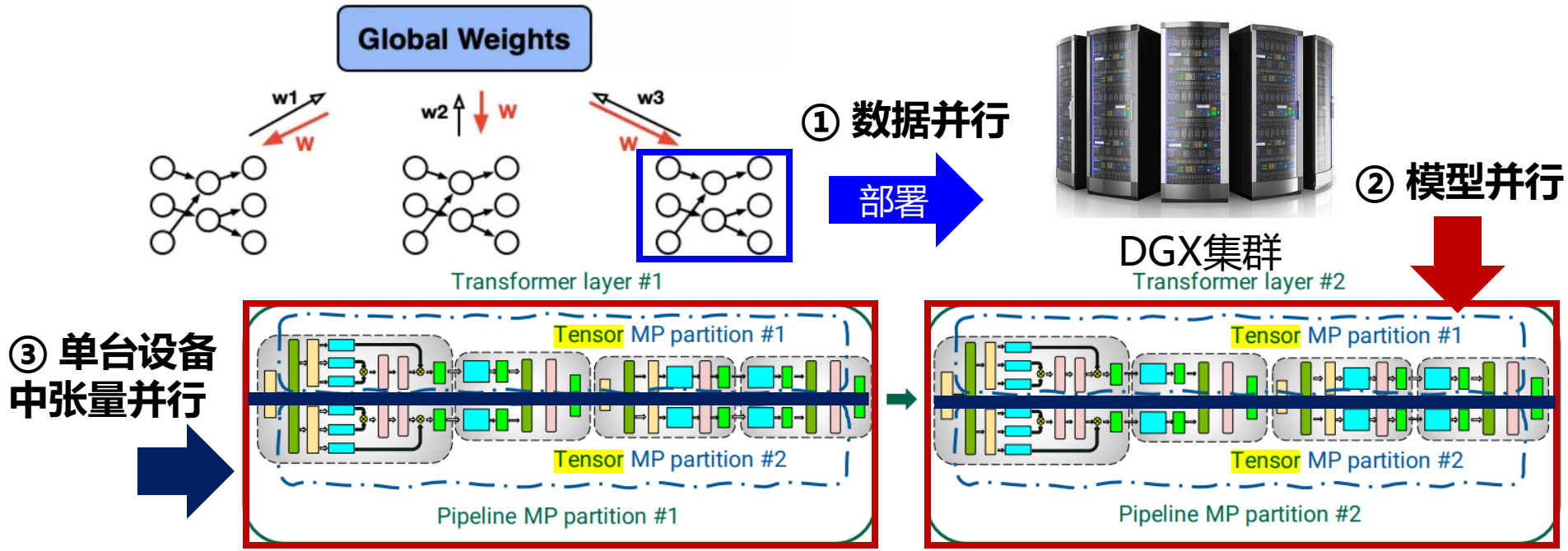
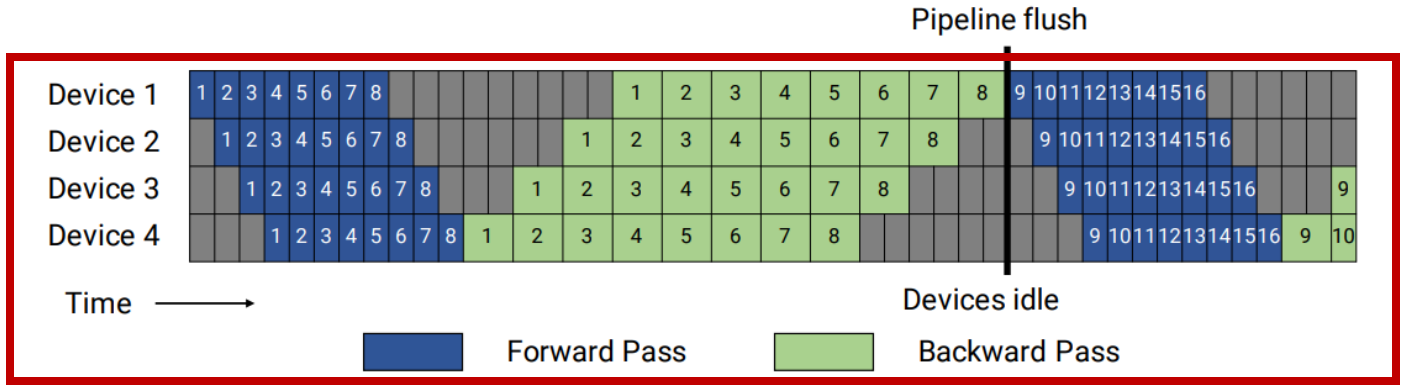


Figure 2: Combination of tensor and pipeline model parallelism (MP) used in this work for transformer-based models.

② 拆分为8个Transformer+MLP 结构，4台设备构建处理流



- Narayanan D, Shoeybi M, Casper J, et al. Efficient large-scale language model training on gpu clusters using megatron-lm[C] //Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021: 1-15.

8.3.3 可扩展训练技术

■ ZeRO-1

训练过程中需要维护的大量中间变量，如动量、平方梯度等。这些状态通常会占用大量内存

方案 **通过将优化器参数分布式存储在多个设备， 使得单设备只保存一部分参数，从而显著减少了内存的使用**

■ ZeRO-2

训练大规模模型时，梯度计算作为反向传播算法的核心，其数据规模非常庞大，会占用大量内存

方案 **梯度也进行拆分，并将其分布到多个 GPU 上。每个 GPU 只需存储和处理一部分梯度**

■ ZeRO-3

大模型的参数规模也非常庞大

方案 **除了拆分优化器状态和梯度外，还通过拆分模型参数来最大化内存效率。**

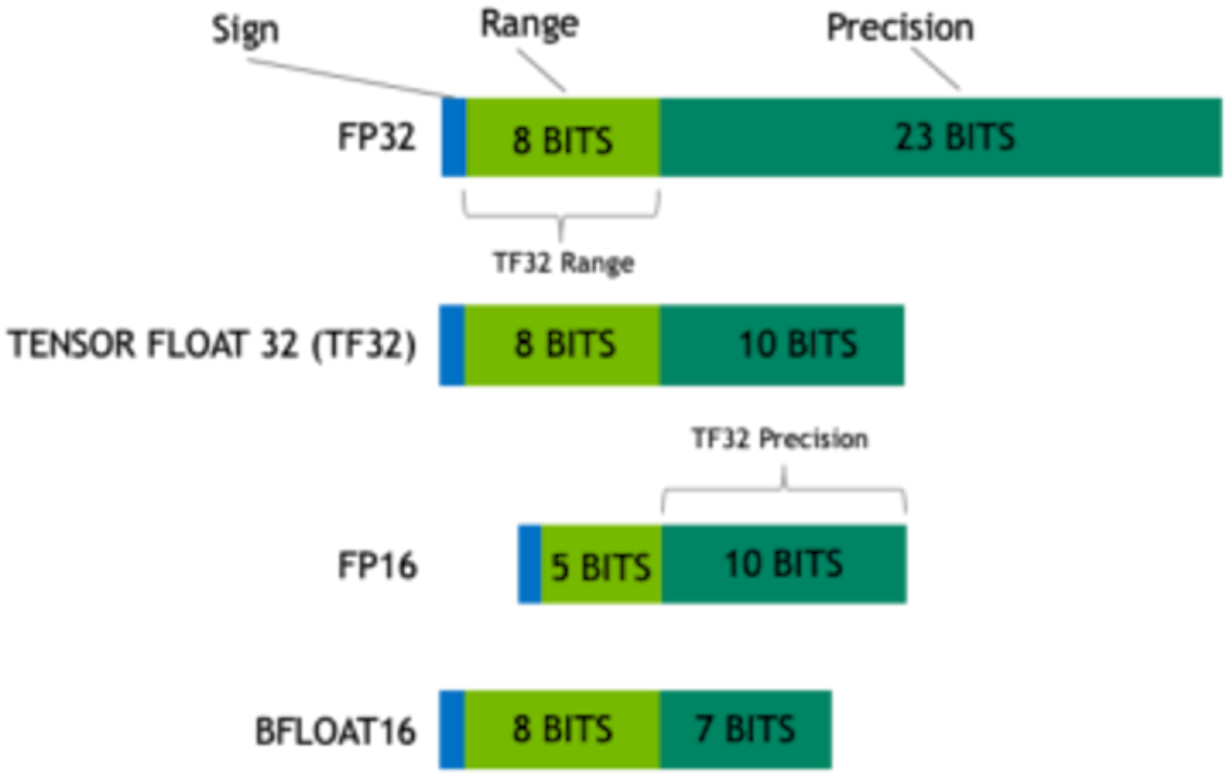
分布式存储和按需加载的机制有效减少了内存占用。ZeRO技术适用于在显存资源极其有限的情况下，仍然需要训练超大规模模型的场景。

8.3.3 可扩展训练技术

■ 混合精度训练

❑ 大模型训练常用数据类型：

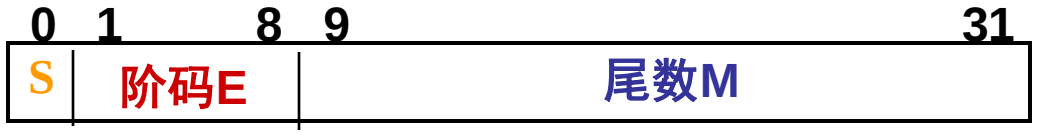
- FP32（Full Precise Float 32） 单精度：单个参数需要 32 bits
- TF32（Tensor Float 32）单精度：由NVIDIA提出的单精度
- FP16 半精度：单个参数需要 16 bits
- INT8 整型： 单个参数需要 8 bits
- BF16 （Brain Floating Point）半精度：单个参数需要16 bits，由google提出



一些研究工作尝试开发更激进的 INT4 量化方法。LLaMA 和 通义千问 等开源模型也均提供了 INT4 量化版本的模型副本

浮点数(Floating Point)的表示范围

例：画出下述32位浮点数格式的规格化数的表示范围。

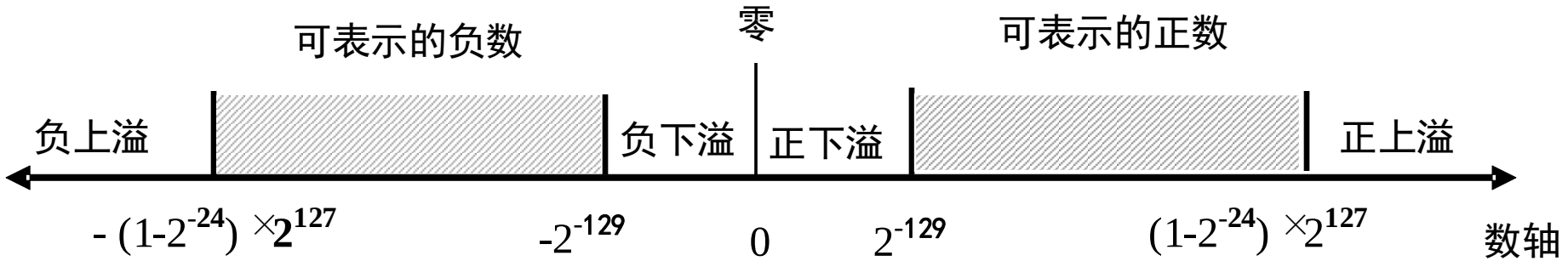


$+/-0.1xxxxx \times 2^{E}$

第0位数符S；第1～8位为8位移码表示阶码E（偏置常数为128）；第9～31位为24位二进制原码小数表示的尾数数值部分M 。规格化尾数的小数点后第一位总是1，故规定第一位默认的“1”不明显表示出来。这样可用23个数位表示24位尾数。

最大正数：$0.11\ldots1 \times 2^{11\ldots1} = (1-2^{-24}) \times 2^{127}$ 最小正数：$0.10\ldots0 \times 2^{00\ldots0} = (1/2) \times 2^{-128}$

因为原码是对称的，所以其表示范围关于原点对称。



机器0：尾数为0 或 落在下溢区中的数

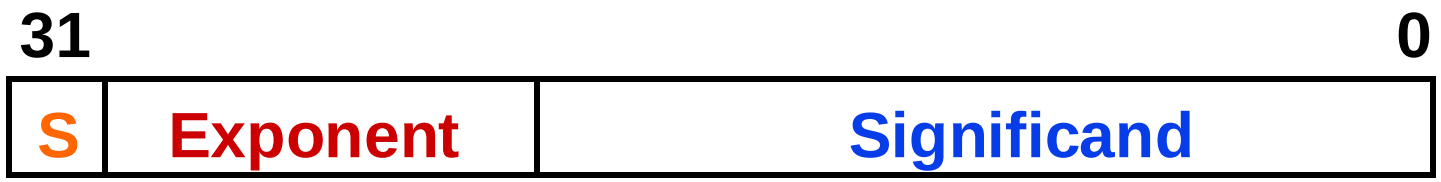
浮点数范围比定点数大，但数的个数没变多，故数之间更稀疏，且不均匀

浮点数的表示

- Normal format（规格化数形式）：

  +/-1.xxxxxxxxxx × $R^{Exponent}$

- 32-bit 规格化数：



S 是符号位（Sign）

Exponent用移码（增码）来表示

Significand 表示 xxxxxxxxxxxxx，尾数部分

(基可以是 2/ 4 / 8 / 16，约定信息，无需显式表示)

- 早期的计算机，各自定义自己的浮点数格式

问题：浮点数表示不统一会带来什么问题？

本章内容

- 8.1 概述
- 8.2 预训练数据工程
  - 8.2.1 预训练数据源
  - 8.2.2 多模态数据集
  - 8.2.3 数据处理
  - 8.2.4 模型性能关系
- 8.3 预训练方法
  - 8.3.1 预训练任务
  - 8.3.2 优化参数设置
  - 8.3.3 可扩展训练技术
- 8.4 讨论

8.4 讨论

思考题1 如何以较小的代价修正大语言模型存储的知识？

思考题2 讨论在大规模语言模型预训练中常用的优化技巧，如学习率调度、混合精度训练、分布式训练等。