大模型和大数据

数据要求数量大，多多益善

数据的分类：

- 一类是网页数据（web data），这类数据的获取最为方便，各个数据相关的公司比如百度、谷歌等每天都会爬取大量的网页存储起来。其特点是量级非常大，质量参差不齐。

- 第二类称之为专有数据（curated high-quality corpora），为某一个领域、语言、行业的特有数据。比如对话、书籍、代码、技术报告、论文考试等数据。这类数据比较难获取，如果在中国那么最优代表性的就应该是在我们的图书馆、国家数字档案馆、国家数字统计局等机构和地方。

专有数据优点：一种普遍观点认为“GPT、PaLM等模型的成功很大程度源自于其他模型难以企及的大量的、高质量的专有数据”。

网页数据优点：

- 网页数据的量级比公开数据大的多，仅用专有数据模型模型训练不到最佳效果：
- 网页数据有固定的格式，我们可以根据html上面的标签进行处理，而专有数据因为来源很杂，格式不统一等原因，甚至需要一份数据，一种处理方式很费时间。
- 大部分专有数据其实在网页数据中也能找到：比如书籍数据，也可能在某些盗版书网站上就有网页版本的。

大模型有两种：
基座模型：GLM，gpt 具备语言理解能力，但是不具备对话能力

通过QA问答训练

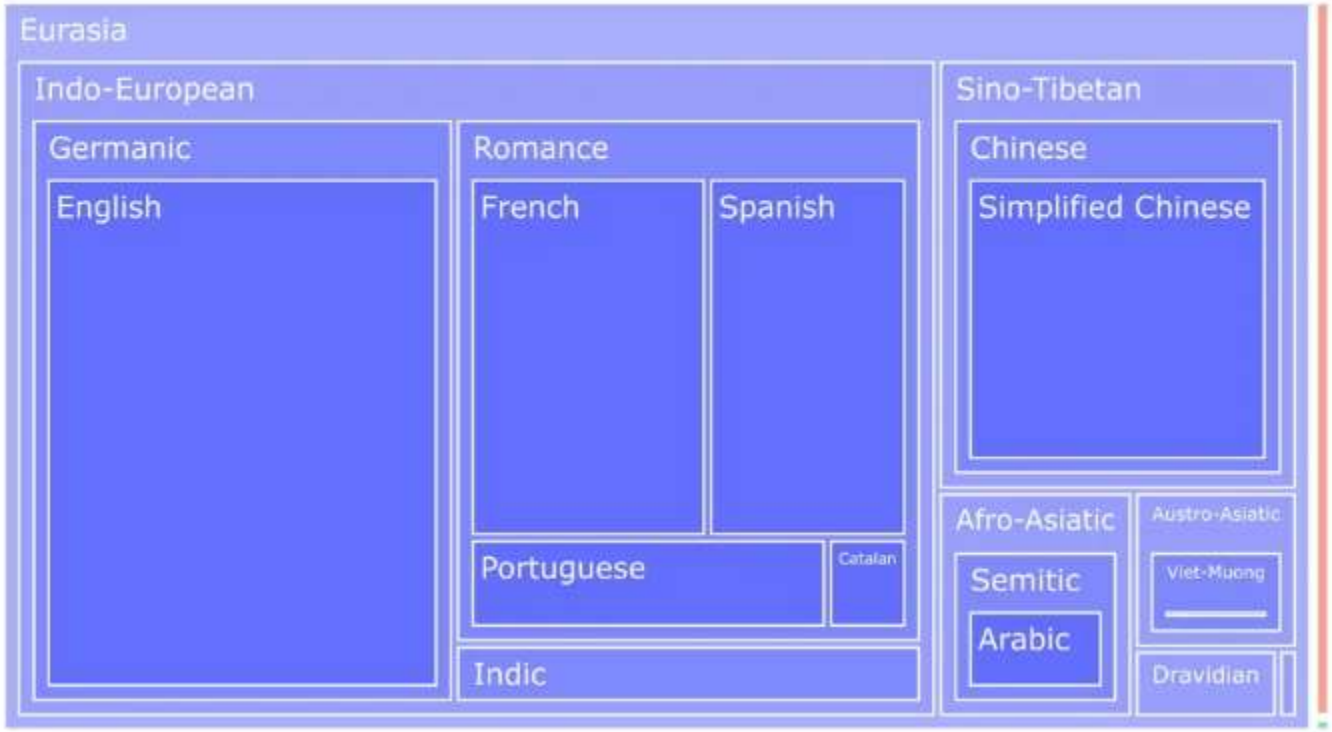
对话模型：chatGLM，chatGPT在基座模型的基础上，进行对话的专项训练

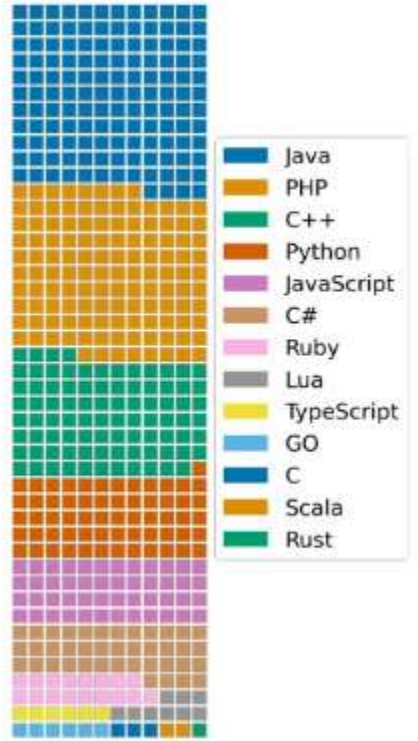
glm130B
baichuan

对BLOOM大模型训练的时候使用了1321.89 GB数据，一共超过40+不同国家的语预语言，对于代码Code有10+不同的编程语言。

Sql不是客观语言，和数据库高度绑定，几乎没有通用性

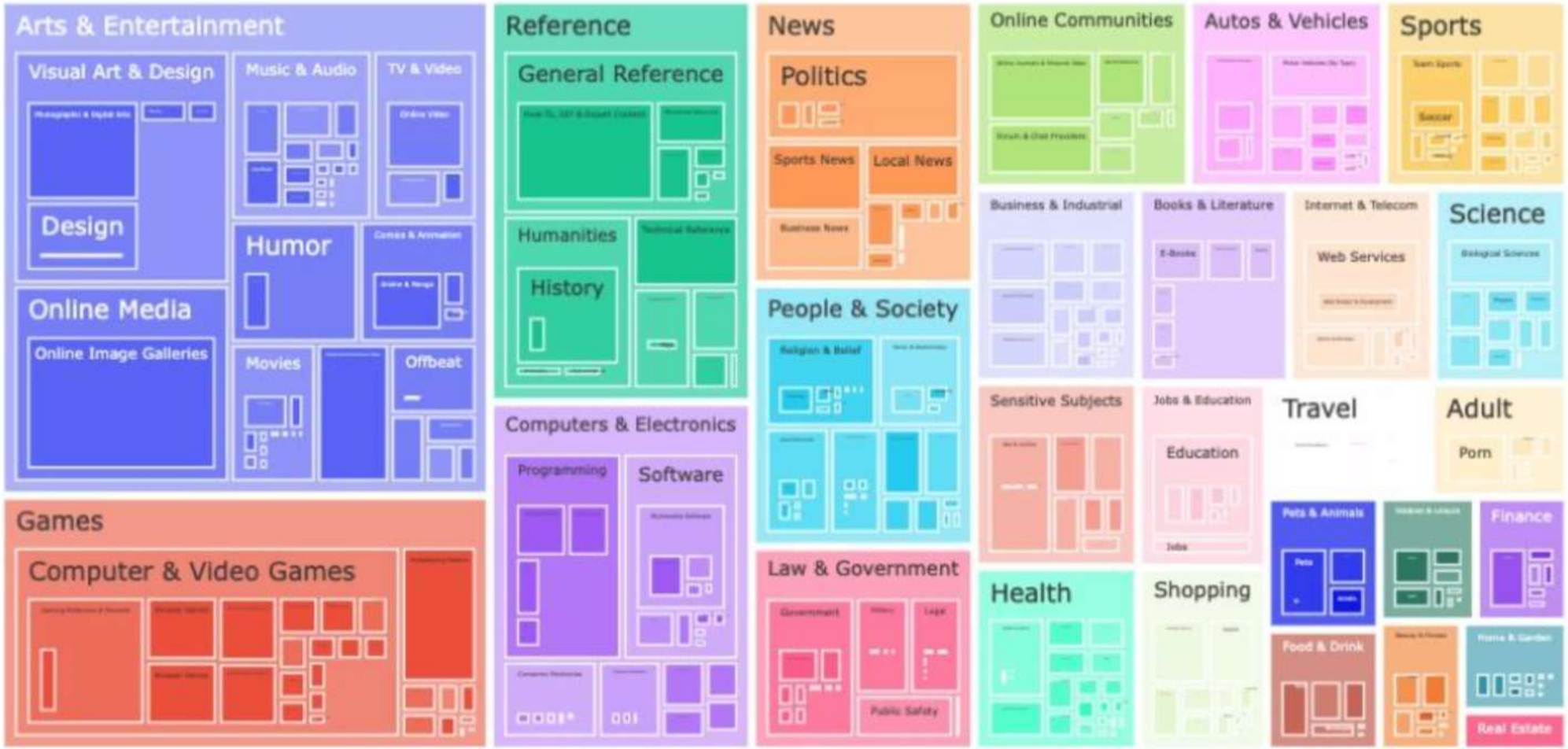
为什么没有sql





扫码了解更多

PALM大模型数据来源



常见数据集

1 English CommonCrawl

使用模型：LLaMA（67%）、LaMDA、PaLM

下载地址 https://github.com/karust/gogetcrawl

2 Wikipedia

使用模型：LLaMA（4.5%）、GPT-NEOX（1.53%）、LaMDA、PaLM

下载地址 https://huggingface.co/datasets/wikipedia

3 C4

使用模型：LLaMA（15%）、LaMDA、PaLM

下载地址 https://huggingface.co/datasets/c4

4 Github

使用模型：LLaMA（4.5%）、GPT-NEOX（7.59%）、PaLM、OPT、GLM130B

下载地址 https://github.com/EleutherAI/github-downloader

美洲驼LLAMA数据
模型与数据参数量：30B/65B 1.4T 300B tokens
数据下载地址
https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T

BLOOM
模型与数据参数量：1.6TB 350B tokens
数据下载地址
https://docs.google.com/forms/d/e/1FAIpQLSdq50O1x4dkdGI4dwsmchFuNI0KCWEDiKUYxvd0r0_sl6FfAQ/viewform?pli=1

中文数据相对比较少

中文数据集

1 中文文本分类数据集THUCNews
http://thuctc.thunlp.org

2 清华大学NLP实验室开放数据集
http://thuocl.thunlp.org/

3 wiki百科中文
https://zh.wikipedia.org

4 WuDaoCorpora
https://openi.pcl.ac.cn/BAAI/WuDao-Data/

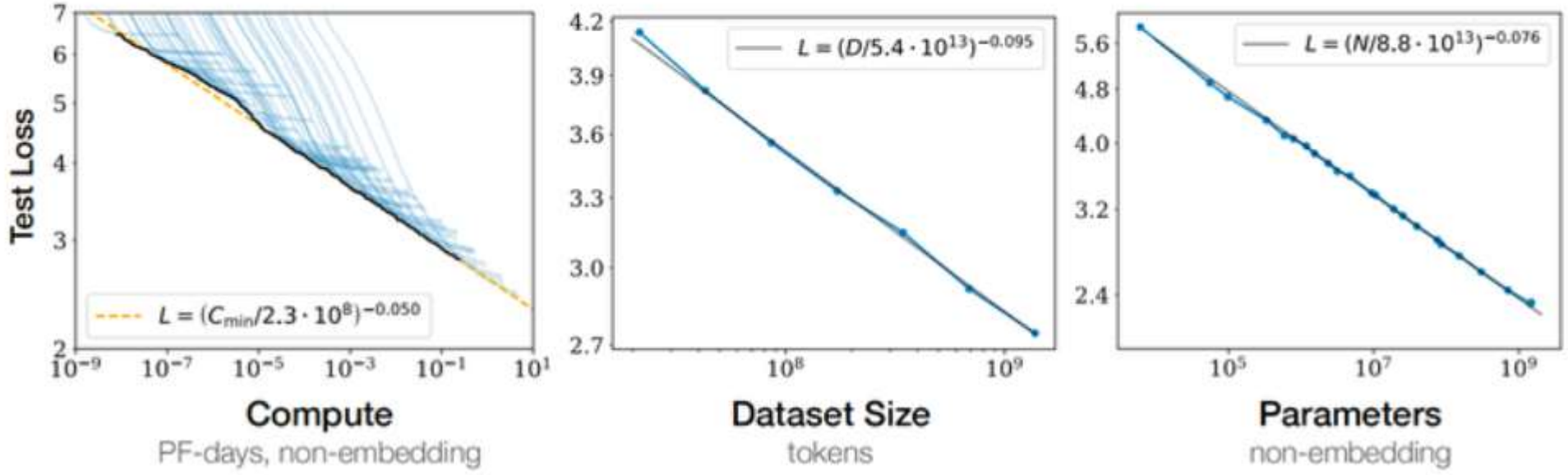
5 Chinese book
https://link.zhihu.com/?target=https%3A//github.com/JiangYanting/Chinese_book_dataset

6 千言
https://www.luge.ai/

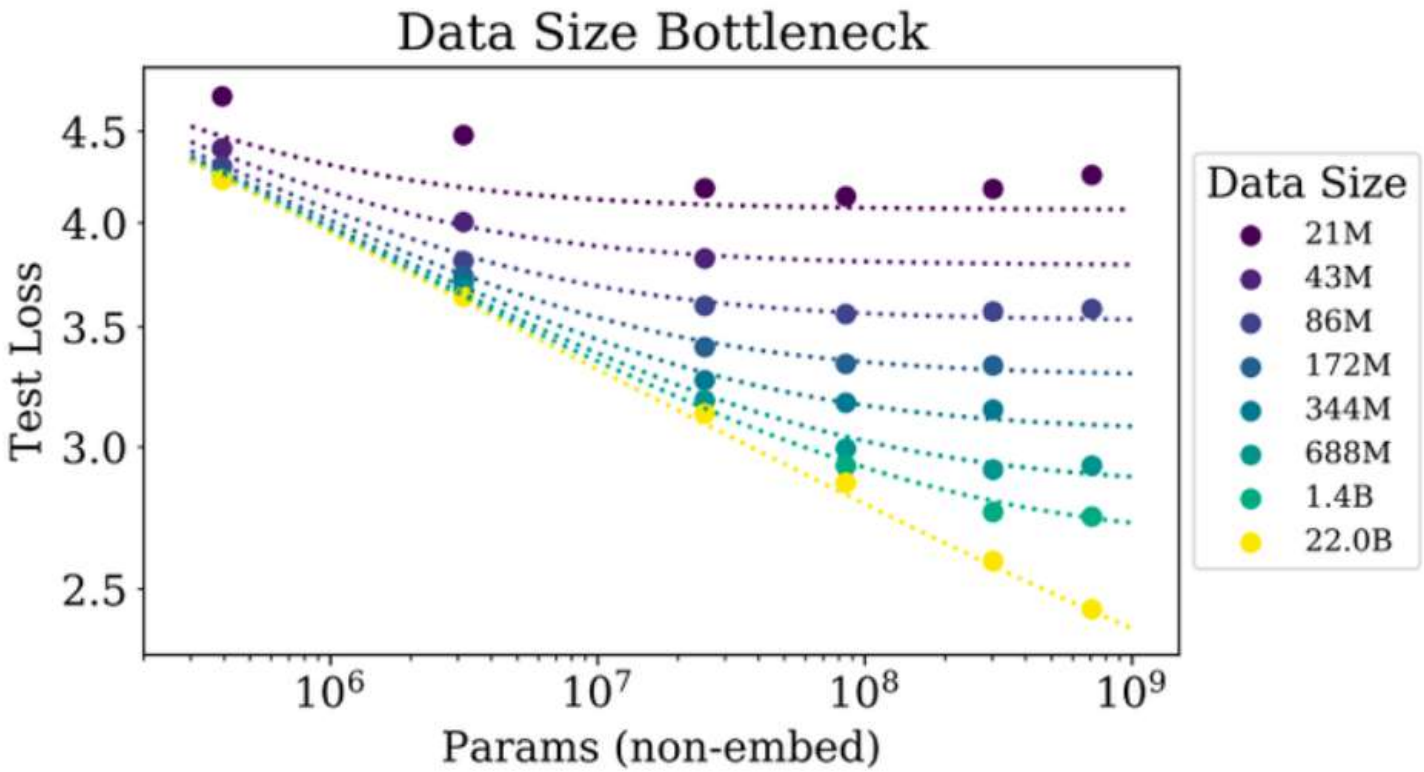
Scaling Laws简单介绍就是：随着模型大小、数据集大小和用于训练的计算浮点数的增加，模型的性能会提高。并且为了获得最佳性能，所有三个因素**必须同时放大**。当不受其他两个因素的制约时，模型性能与每个单独的因素都有**幂律关系**

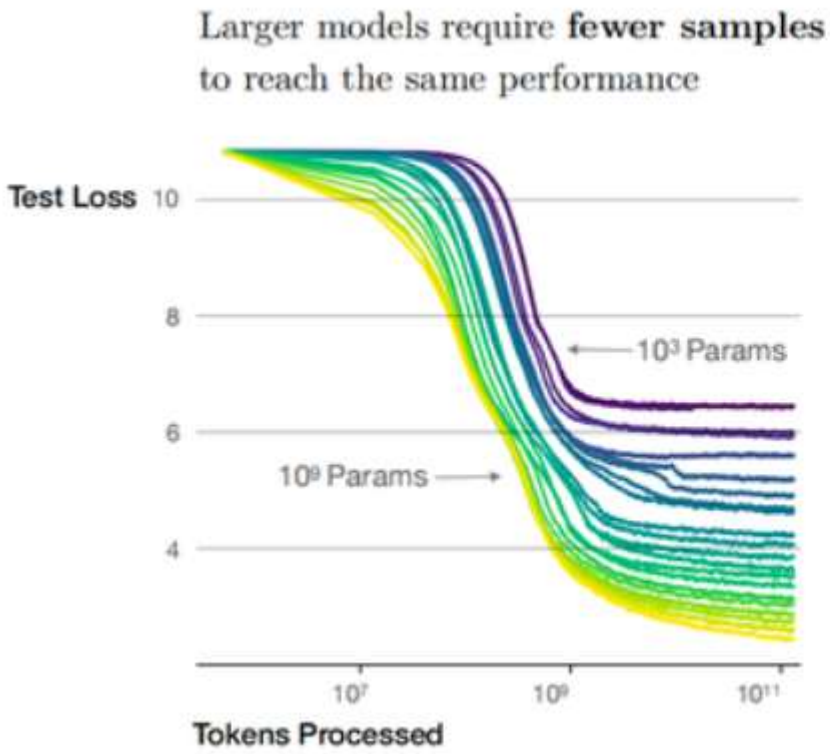
决定效果：模型参数量，数据集大小，计算量

- 模型表现和规模强相关，和模型的shape弱相关：规模包括模型参数量N（不包括embedding）、数据集大小D和计算量C，模型shape指模型depth、width、number of self-attention heads
- 幂方法则：对于模型参数量N、数据集大小D和计算量C三个因素，如果其他两个充足的前提下，模型表现和第三个因素成幂方关系。实验曲线如下：

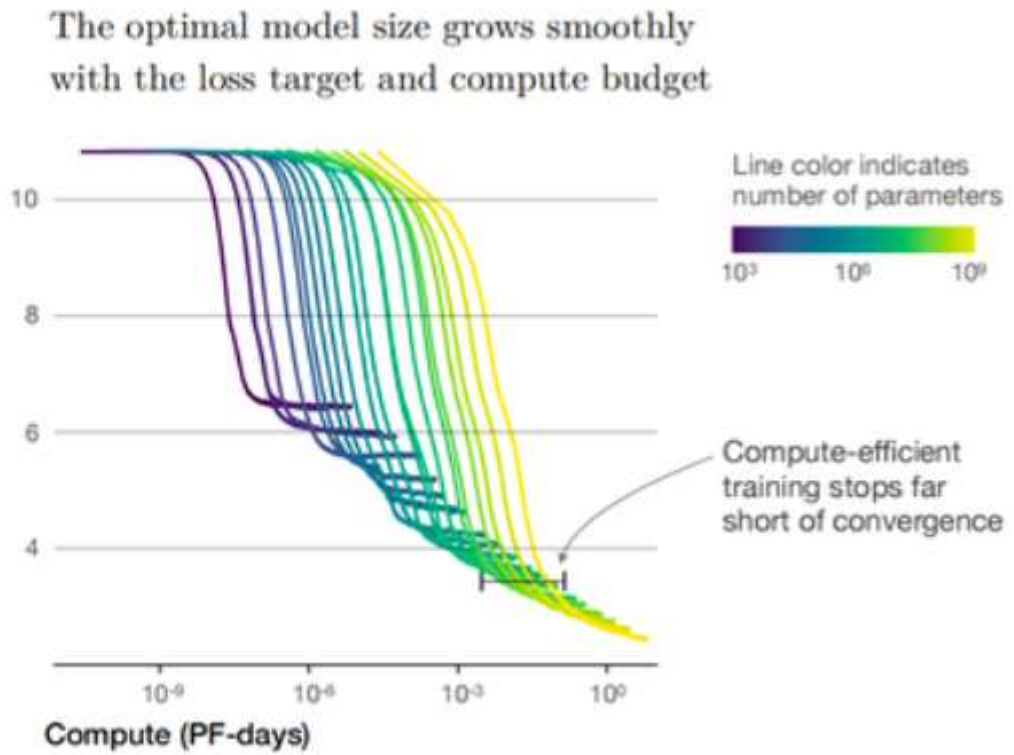


当同时增加数据量和模型参数量时，模型表现会一直变好。当其中一个因素受限时，模型表现随另外一个因素增加变好，但是会逐渐衰减。





数据



计算量

计算量和模型参数之间的关系

在算力不足的情况下
小模型收敛的更快
但是大模型
效果上限更高（损失函数更小）

扫码了解更多

数据处理

分词粒度：
1.单词分词法：英文（空格分词），中文（jieba分词 or 分字）
2.单字分词法：英文（字母），中文（分字）
3.子词分词法：BPE，WordPiece，Unigram

字词分词法：有个词表，根据词表来分词

词表生成：根据训练数据生成

英文词表：est er ing
中文词表：中国 人民币

从GPT-2开始一直到GPT-4，OpenAI一直采用BPE（Byte Pair Encoding）分词法。

假设我们有一个语料库，其中包含单词（pre-tokenization之后）—— old, older, highest, 和 lowest，我们计算这些词在语料库中的出现频率。假设这些词出现的频率如下：

{ “old” : 7, “older” : 3, “finest” : 9, “lowest” : 4}

让我们在每个单词的末尾添加一个特殊的结束标记 “</w>” 。

{ “old</w>” : 7, “older</w>” : 3, “finest</w>” : 9, “lowest</w>” : 4}

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13
11	t	13
12	w	4

迭代 1：我们将从第二常见的标记“e”开始。 在我们的语料库中，最常见的带有“e”的字节对是“e”和“s”（在finest和lowest两个词中），它们出现了 9 + 4 = 13 次。 我们将它们合并以形成一个新的token“es”并将其频率记为 13。我们还将从单个token（“e”和“s”）中减少计数 13，从而我们知道剩余的“e”或“s”token数。 我们可以看到“s”不会单独出现，“e”出现了 3 次。 这是更新后的表格：

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13
12	w	4
13	es	9 + 4 = 13

迭代 2：我们现在将合并token“es”和“t”，因为它们在我们的语料库中出现了 13 次。 因此，我们有一个频率为 13 的新token“est”，我们会将“es”和“t”的频率减少 13。

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13

选取出现频率最高的合并
ol 10次
es 13次

迭代 3：让我们现在考虑“</w>”token，我们看到字节对“est”和“</w>”在我们的语料库中出现了 13 次。

Number	Token	Frequency
1	</w>	23 - 13 = 10
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13 - 13 = 0
15	est</w>	13

迭代 4：查看其他token，我们看到字节对“o”和“l”在我们的语料库中出现了 7 + 3 = 10 次。

Number	Token	Frequency
1	</w>	23
2	o	14 - 10 = 4
3	l	14 - 10 = 4
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13
15	ol	7 + 3 = 10

Number	Token	Frequency
1	</w>	10
2	o	4
3	l	4
4	e	3
5	r	3
6	f	9
7	i	9
8	n	9
9	w	4
10	est</w>	13
11	old	10

假设单词的序列是[“the</w>”,“highest</w>”,“range</w>”,“in</w>”,“Seattle</w>”]。我们将遍历我们在语料库中找到的所有token——从最长到最短，并尝试使用这些token替换给定单词序列中的子字符串。最终，我们将遍历所有token，并且我们的子字符串将被替换为我们token列表中已经存在的token组合。如果会留下几个子串（我们的模型在训练中没有看到的词），我们将用unknown token替换它们。

会准备一个大数据集，生成字词

通过训练数据 得到了子词

字词：挨着一起出现，且频率较高

bpe
我喝
我吃
我玩
我讲
。。。

Word-Piece

Word-Piece和BPE非常相似，BPE使用**出现最频繁的组合**构造子词词表，而Wordpiece使用**出现概率最大的组合**构造子词词表。换句话说，WordPiece每次选择合并的两个子词，通常在语料中以**相邻方式**同时出现。比如说 P(ed) 的概率比P(e) + P(d)单独出现的概率更大（可能比他们具有最大的互信息值），也就是两个子词在语言模型上具有较强的关联性。这个时候，Word-Piece会将它们组合成一个子词。

$$logP(t_z) - (logP(t_x) + logP(t_y)) = log(\frac{P(t_z)}{P(t_x)P(t_y)})$$

Unigram

与BPE或者WordPiece不同，Unigram的算法思想是**从一个巨大的词汇表出发，再逐渐删除trim down其中的词汇**，直到size满足预定义。

开始很多字词
慢慢剔除

初始的词汇表可以**采用所有预分词器分出来的词，再加上所有高频的子串**。
每次从词汇表中删除词汇的**原则是使预定义的损失最小**。训练时，计算loss的公式为：

$$Loss = -\sum_{i=1}^{N} log\left(\sum_{x \in S(x_i)} p(x)\right)$$

假设训练文档中的所有词分别为 $x_1; x_2, \ldots, x_N$ ，而**每个词tokenize的方法**是一个集合 $S(x_i)$ 。
当一个词汇表确定时，每个词tokenize的方法集合 $S(x_i)$ 就是确定的，而每种方法对应着一个概率 $p(x)$ 。
如果从词汇表中删除部分词，则某些词的tokenize的种类集合就会变少，log(*)中的求和项就会减少，从而增加整体loss。

Unigram算法每次**会从词汇表中挑出使得loss增长最小的10%~20%的词汇**来删除。
一般Unigram算法会与SentencePiece算法连用。

信息熵：混乱程度，越低越好

$$Loss = -\sum_{i=1}^{N} log\left(\sum_{x \in S(x_i)} p(x)\right)$$

遍历所有句子，所有词
N词

子词x的概率p(x)

去除一个子词
Loss就会增加

我晚上吃面条

字词：我 晚上 面条 吃面条 吃面

不同的分词结果：
我 晚上 吃面条
我 晚上 吃 面条
我 晚上 吃面 条

遍历所有子词，找到让Loss增加最少的字词
然后去掉

各大LM用的tokenizer和对应的词汇表大小：

LM	Tokenizer	Vocabulary Size
BERT	Word-Piece	30k
ALBERT	Sentence-Piece	30k
RoBERTa	BPE	50k
XLM-RoBERTa	Sentence-Piece	30k
GPT	SpaCy	40k
GPT-2	BPE	50k
GPT-3	BPE	50k
GPT-3.5 (ChatGPT)	BPE	-
GPT-4	BPE	-
T5	Sentence-Piece	30k
Flan T5	Sentence-Piece	30k
BART	Word-Piece	50k

生成完子词之后，大模型一般还会添加一些特殊字符：
开始符
结束符
未登录词
模型特殊要求的词 对于bert [cls] [sep]

不同的模型，使用不同的子词词表 每个词对应一个ID

同一个子词 在不同的词表里 对应的id不一样

另外，有大佬做了各大LLM的词汇表大小和性能：

名称	词表长度↑	中文平均长度↓	英文平均长度↓	中文处理时间↓	英文处理时间↓
LLaMA	32000	62.8	32.8	02:09	01:37
BELLE	79458	24.3	32.1	00:52	01:27
MOSS	106072	24.8	28.3	07:08	00:49
GPT4	50281	49.9	27.1	00:07	00:08
BLOOM/Z	250680	23.4	27.6	00:46	01:00
ChatGLM	130344	23.6	28.7	00:26	00:39

分词方法
子词词表：按长度从高到底排
看句子里面是否有子词，如果有，则分开，没有遍历下一个

1 2 3

字词：我:1 晚上:2 吃面条:3

不同的分词结果：
我 晚上 吃面条
我 晚上 吃 面条
我 晚上 吃面 条

比如我训练的场景极度垂直，

我是否需要自己训练子词词表，分词器

1.需要训练：
会生成新的子词
a.完全替换以前的词表（从基础模型开始，整个模型全部重新训练）
b.补充以前的词表（只需要训练新的子词对应的embeding）

2.不训练词表：
专业词汇 通过模型本身学习（建议：成本低，微调）