AI 大模型开发工程师
之大模型微调核心之数据

讲师：李希沅

掌握大模型核心三要素

目录

1 预训练数据准备之词表准备

大模型预训练之数据

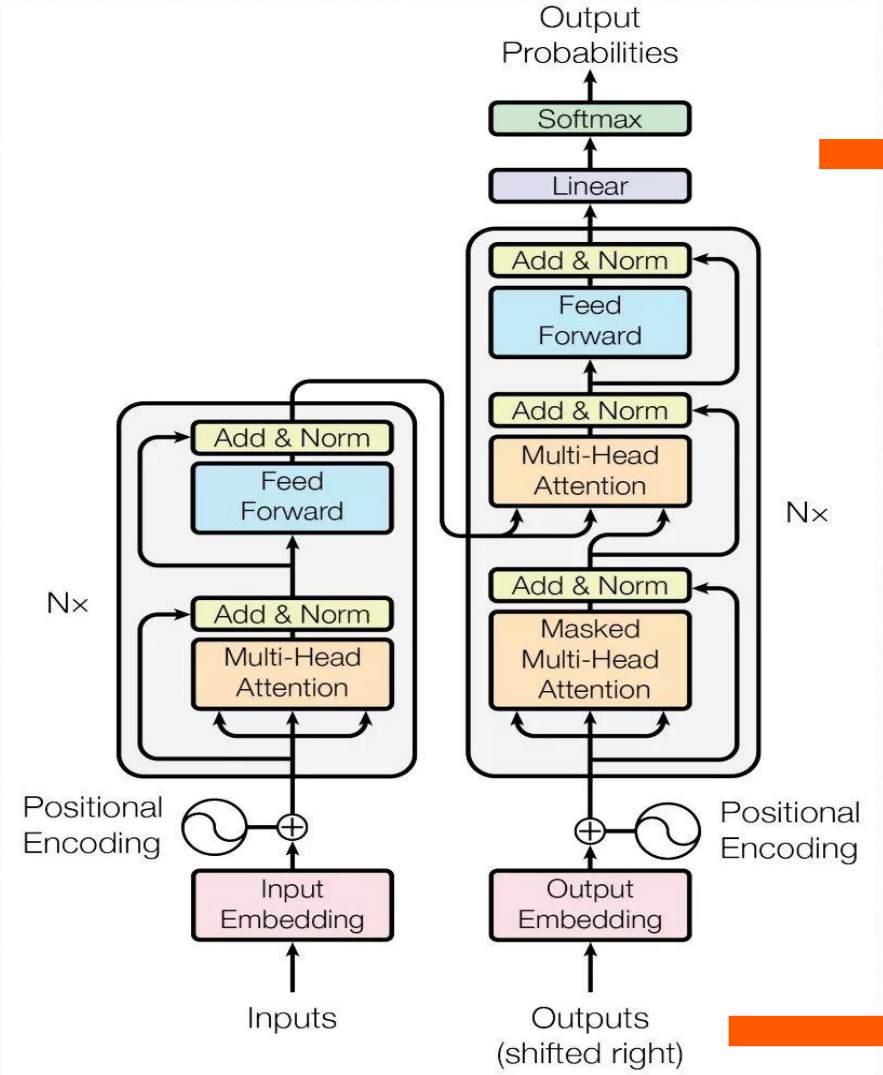


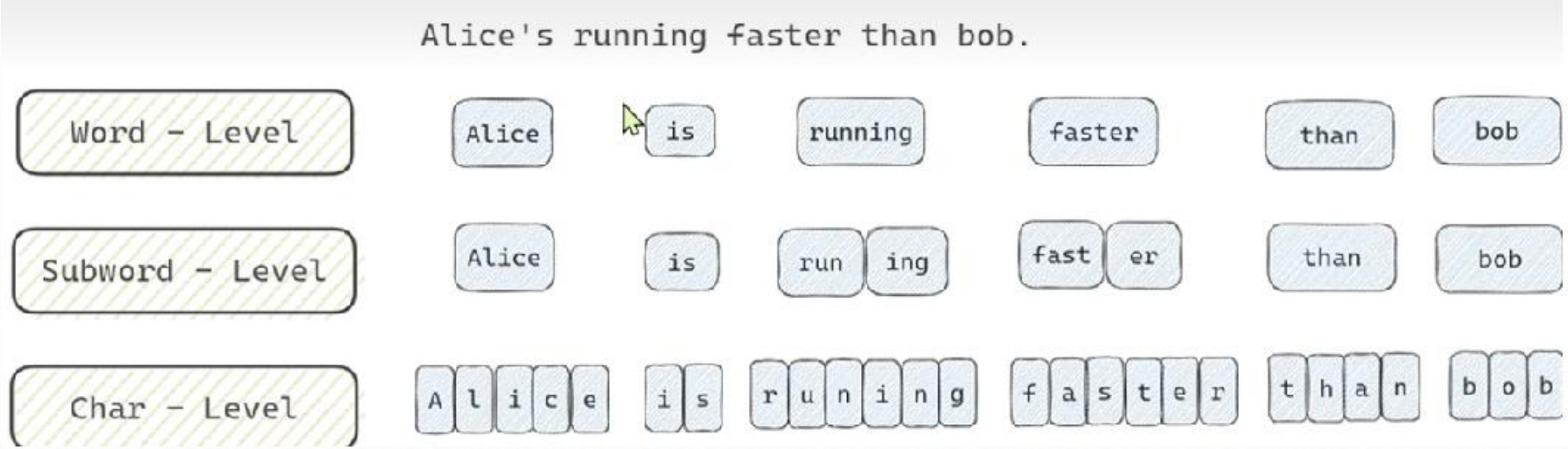
Figure 1: The Transformer - model architecture.

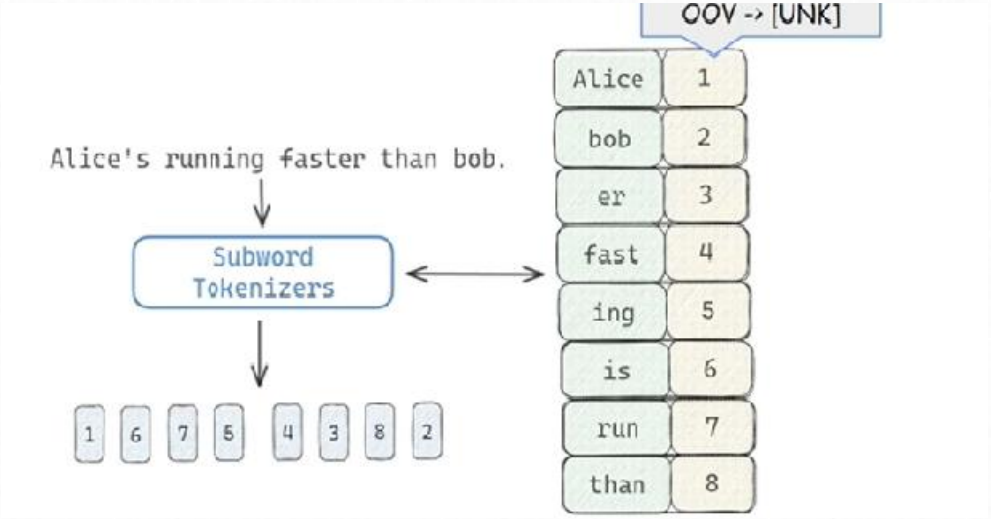
词表

我爱中?

训练数据

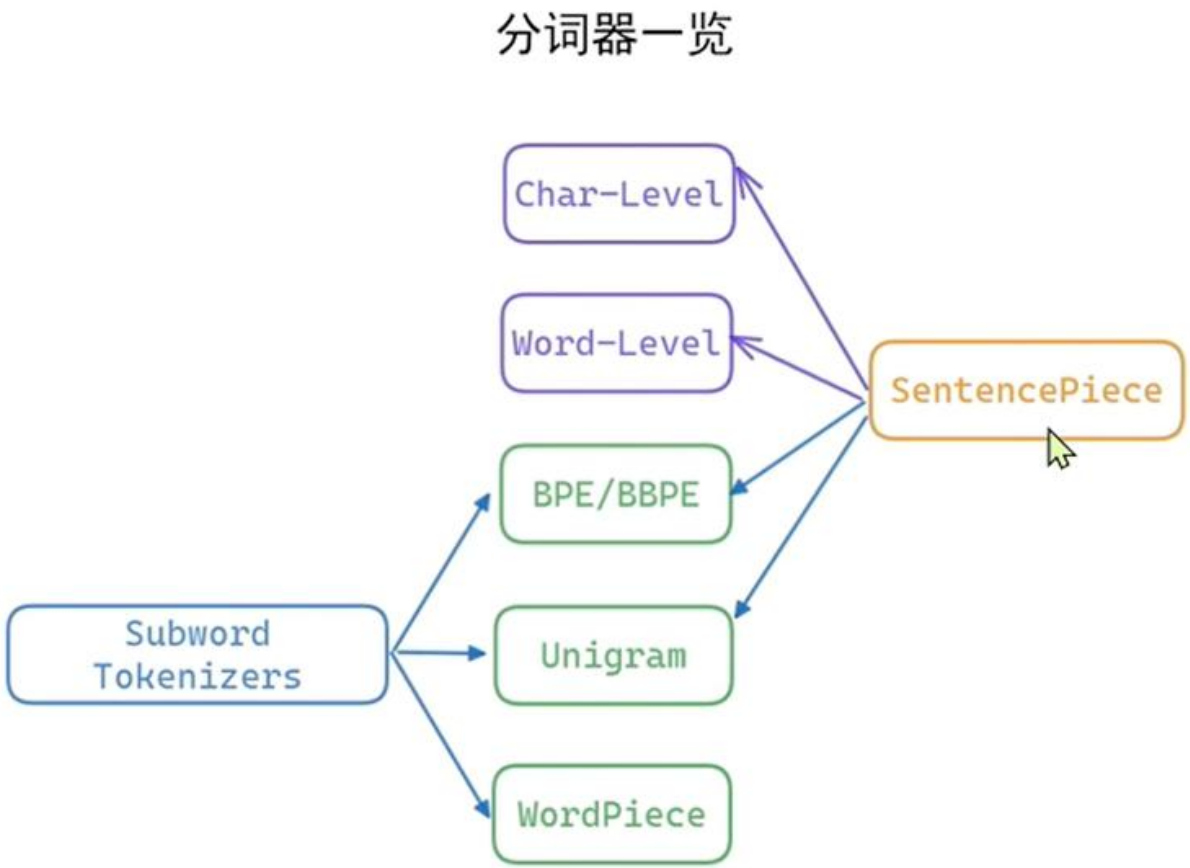
分词的粒度





OOV是“Out-Of-Vocabulary”的缩写，直译为“词汇表外的”，在自然语言处理中，表示的是那些在词汇表中没有的单词

分词器的类型

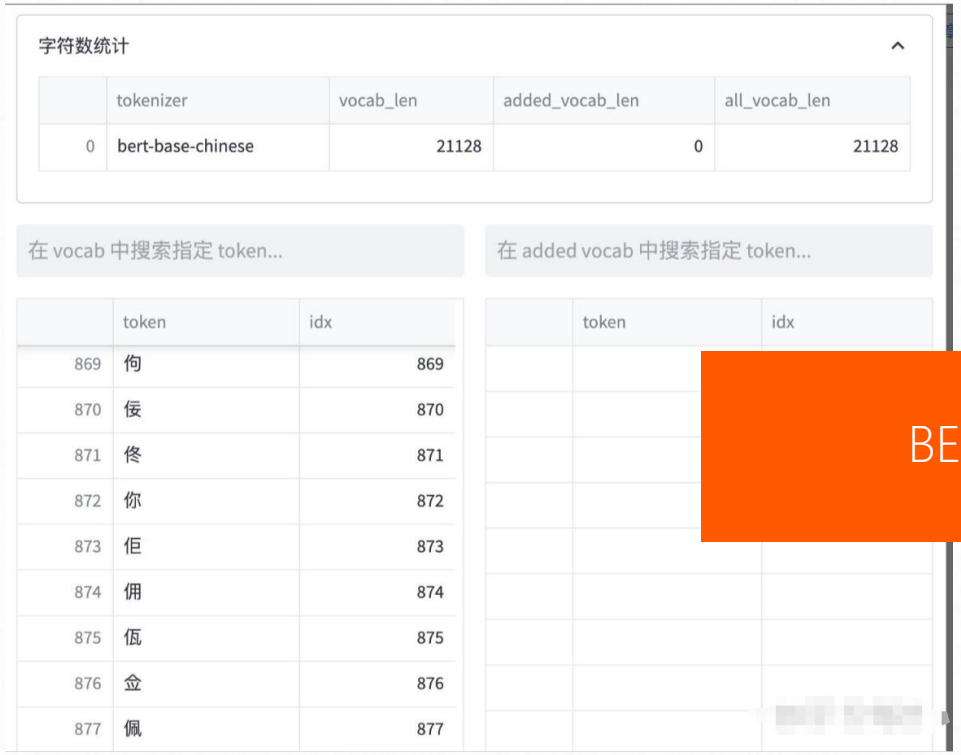


Model	Type of Tokenizer
fast MPNet	WordPiece
PhoBERT	Byte-Pair-Encoding
T5	SentencePiece
fast T5	Unigram
fast MBART	BPE
fast PEGASUS	Unigram
PEGASUS	SentencePiece
XLM	Byte-Pair-Encoding
TAPAS	WordPiece
BertGeneration	SentencePiece
BERT	WordPiece
fast BERT	WordPiece
XLNet	SentencePiece
GPT-2	byte-level Byte-Pair-Encoding
fast XLNet	Unigram
fast GPT-2	byte-level Byte-Pair-Encoding
fast ALBERT	Unigram
ALBERT	SentencePiece
CTRL	Byte-Pair-Encoding
fast GPT	Byte-Pair-Encoding
Flaubert	Byte-Pair Encoding
FAIRSEQ	Byte-Pair Encoding
Reformer	SentencePiece
fast Reformer	Unigram
Marian	SentencePiece

WordPiece分词法

tokenizer 有 3 种常用的分词形式：WordPiece，BPE和BBPE（SentencePiece）。

WordPiece ： 就是将所有的「常用字」和「常用词」都存到词表中，当需要切词的时候就从词表里面查找即可。



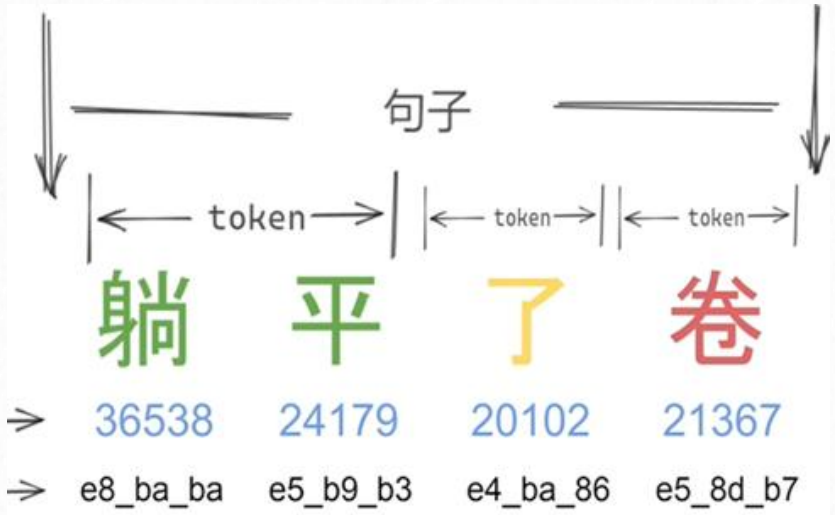
BERT用的就是这种方式

WordPiece 的方式很有效，但当字词数目过于庞大时这个方式就有点难以实现了。

对于一些多语言模型来讲，要想穷举所有语言中的常用词（穷举不全会造成 OOV）

BPE和BBPE分词法

BPE和WordPiece很像，但是在合词的时候有细小的区别：比如unable，BPE 只关心 token pair 的出现频率，即 freq_of_pair；WordPiece 还考虑了每个 token 的出现频率。即使 unable 出现频率很高，但如果 un 和 able 单个 token 的出现频率都很高，也不会合并它们。BPE在支持多语言的时候，经常也会有OOV的情况。



BBPE（Byte-Level BPE）分词：对于英文、拉美体系的语言来说使用BPE分词足以在可接受的词表大小下解决OOV的问题，但面对中文、日文等语言时，其稀有的字符可能会不必要的占用词汇表，因此考虑使用字节级别byte-level解决不同语言进行分词时OOV的问题。具体的，BBPE将一段文本的UTF-8编码(UTF-8保证任何语言都可以通用)中的一个字节256位不同的编码作为词表的初始化基础Subword。

BBPE的分词

[>>>] Encoding Test:

待编码

```
encode token: ['<unk>', '▁', '<0xE5>', '<0xBE>', '<0x85>', '编', '码']
encode token ids: [0, 29871, 232, 193, 136, 31795, 31183]
```

编

Token 「编」 存在于当前词表中，token_idx 为 31795。

码

Token 「码」 存在于当前词表中，token_idx 为 31183。

待

Token 「待」 不存在于当前词表中。

BBPE的优点：不会出现 OOV 的情况。不管是怎样的汉字，只要可以用字节表示，就都会存在于初始词表中。

BBPE的缺点：一个汉字由3个字节组成，一个汉字就会被切成多个token，但实际上这多个token没必要进行训练。

游泳池是杭州西湖的一个游泳池，

BPE和BBPE效果是相当的

SentencePiece里BPE/BBPE

基于BPE构建词表流程

1. 准备语料库，确定期望的 subword 词表大小等参数
2. 在每个单词末尾添加**前缀**或者**后缀**，标记单词的边界，并统计每个单词出现的频率。

   解决 ab-> [a1, a2, b1, b2] 后，复原文本时 [a1, a2] [b1, b2] 还是 [a1] [a2,b1] [b2]？的问题

   Decode(Encode(Normalize(text))) = Normalize(text).
3. 将语料库中所有单词拆分为单个字符，用所有单个字符建立最初的词典
4. 统计每个字符的频率挑出频次最高的符号对，比如说 t 和 h 组成的 th，将新字符加入词表
5. 然后将语料中所有该字符对融合(merge)，即所有 t 和 h 都变为 th。
6. 重复遍历 4 和 5 操作，直到词表中单词数达到设定量 / 最高符号对的频数为 1，退出合并流程

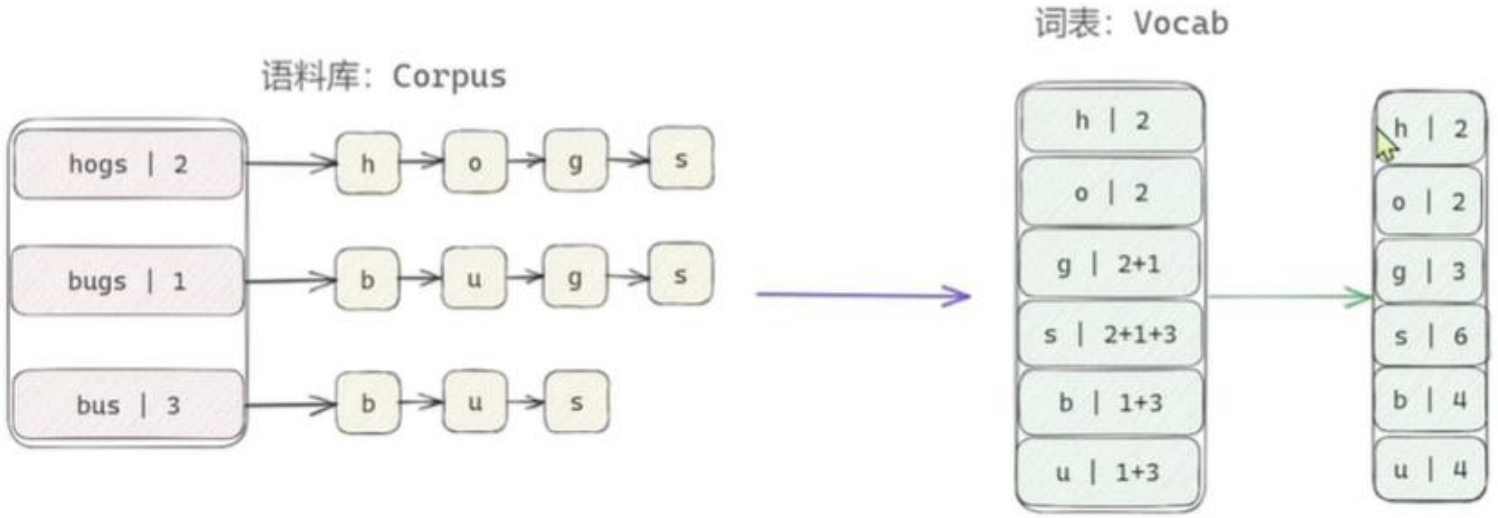
基于BPE的词表构建流程

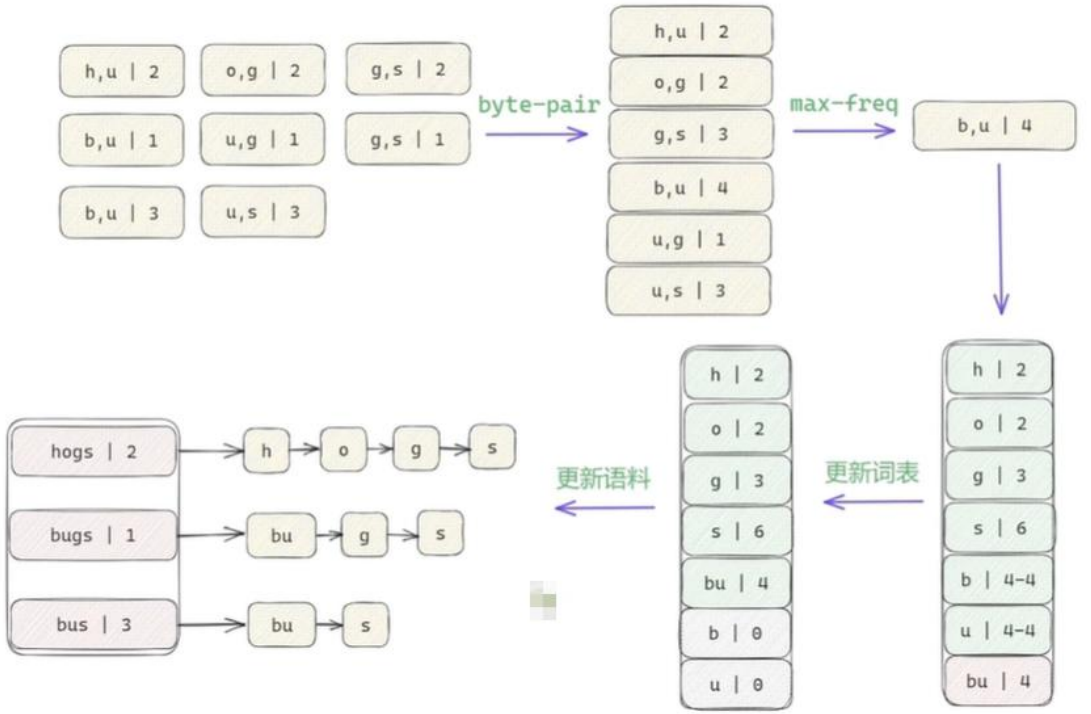
一、加入"元字符" (meta_chars)
1）控制字符(<unk>/<s>/</s>)
2）用户自定义字符

```
//////////////////////////////////////////////////////////////
// Reserved special meta tokens.
// * -1 is not used.
// * unk_id must not be -1.
// Id must starts with 0 and be contigous.
optional int32 unk_id = 40 [default = 0];   // <unk>
optional int32 bos_id = 41 [default = 1];   // <s>
optional int32 eos_id = 42 [default = 2];   // </s>
optional int32 pad_id = 43 [default = -1];  // <pad> (padding)
optional string unk_piece = 45 [default = "<unk>"];
optional string bos_piece = 46 [default = "<s>"];
optional string eos_piece = 47 [default = "</s>"];
optional string pad_piece = 48 [default = "<pad>"];

 const char32 TrainerInterface::kWSChar = L'\u2581';
 const char TrainerInterface::kWSStr[] = "\xe2\x96\x81";
```

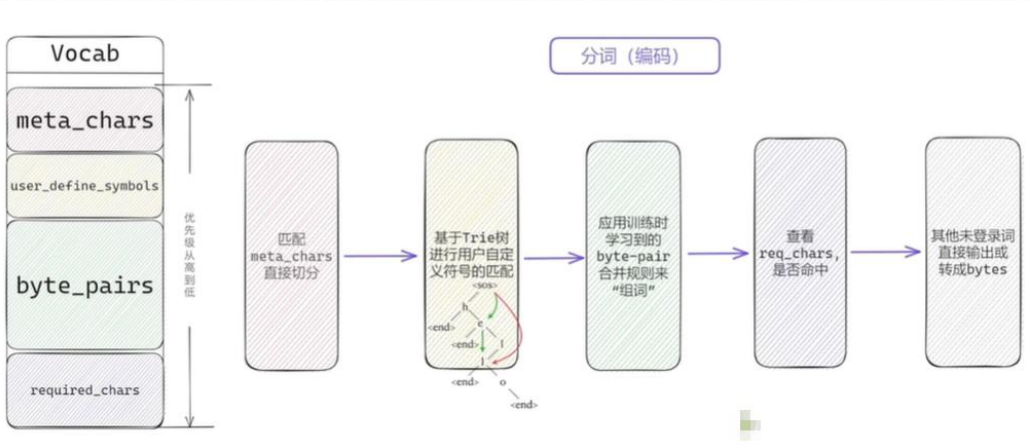
二、对输入文本进行字符级别切割，
根据覆盖度计算出要保留的topk个字符
(required_chars)，得到初始词表。

```
// Determines required_chars which must be included in the vocabulary.
int64 accumulated_chars_count = 0;
// Sorted() sorts the chars_count values in the descending order of pair<>.
// I.e. characters are sorted in the order of required characters and then
// frequent characters.
for (const auto &w : Sorted(chars_count)) {
  const float coverage = 1.0 * accumulated_chars_count / all_chars_count;
  if (!trainer_spec_.use_all_vocab() &&
      coverage >= trainer_spec_.character_coverage()) {
    LOG(INFO) << "Done: " << 100.0 * coverage << "% characters are covered.";
    break;
  }
  accumulated_chars_count += w.second.second;
  CHECK_NE_OR_RETURN(w.first, 0x0020)
      << "space must not be included in normalized string.";
  if (w.first == kUPPBoundaryChar) continue;  // Tab is not included.
  required_chars_.emplace(w.first, w.second.second);
}
```





BPE的分词流程



可以自己写分词器

也可以用现成的

https://github.com/google/sentencepiece

```
/d/code/sentencepiece/build/doc  master  ?                                                    10:49:13
../src/spm_train --input=corpus.txt --model_prefix=tokenizer --vocab_size=23 --character_coverage=0.99 --model_type=bpe

''src/spm_trainer_main.cc(279) [_status.ok()] Internal: ''src/trainer_interface.cc(581) [(static_cast<int>(required_chars_.size() + meta_pieces
_.size())) ≤ (trainer_spec_.vocab_size())] Vocabulary size is smaller than required_chars. 23 vs 1197. Increase vocab_size or decrease chara
cter_coverage with --character_coverage option.
Program terminated with an unrecoverable error.

/d/code/sentencepiece/build/doc  master  ?                                                    INT X  10:50:38
../src/spm_train --input=corpus.txt --model_prefix=tokenizer --vocab_size=1197 --character_coverage=0.99 --model_type=bpe
```

File: tokenizer.vocab

```
1   <unk>   0
2   <s> 0
3   </s>    0
4   :"      -0
5   七公    -1
6   . "     -2
7   黄蓉    -3
8   洪七公  -4
9   郭靖    -5
10  ..      -6
11  ?"      -7
12  子翁    -8
13  梁子翁  -9
14  一招    -10
15  ....    -11
16  爹爹    -12
17  笑道    -13
18  说道    -14
19  甚么    -15
20  洪七公道    -16
21  了一    -17
22  黄蓉道  -18
23  功夫    -19
24  怎么    -20
25  叫化    -21
26  武功    -22
27  不是    -23
28  娃娃    -24
29  叫道    -25
30  !"      -26
31  也不    -27
32  掌法    -28
33  人家    -29
34  这么    -30
35  ......  -31
36  一个    -32
```

编码（分词）

```
/d/code/sentencepiece/build/doc  master  ?
echo "郭靖用从洪七公那里学会的降龙十八掌拍晕了特朗普" | ../src/spm_encode.exe --model=tokenizer.model
_ 郭靖 用 从 洪七公 那 里 学 会 的 降龙十八掌 拍 晕 了 特 朗普

/d/code/sentencepiece/build/doc  master  ?
echo "郭靖用从洪七公那里学会的降龙十八掌拍晕了特朗普" | ../src/spm_encode.exe --model=tokenizer.model --output_format=id
219 8 281 361 7 181 238 196 232 135 89 388 794 139 812 0
```

解码（复原）

```
/d/code/sentencepiece/build/doc  master  ?
echo "219 8 281 361 7 181 238 196 232 135 89 388 794 139 812 0" | ../src/spm_decode.exe --model=tokenizer.model
郭靖用从洪七公那里学会的降龙十八掌拍晕了特 ⁇
```

```
/d/code/sentencepiece/build/doc  master  ?
../src/spm_train --input=corpus.txt --model_prefix=tokenizer --vocab_size=1323 --character_coverage=0.99 \
  --model_type=bpe  --user_defined_symbols="特朗普"
```

File: tokenizer.vocab

```
1   <unk>   0
2   <s> 0
3   </s>    0
4   特朗普  0
5   :"      -0
6   七公    -1
7   . "     -2
8   黄蓉    -3
9   洪七公  -4
10  郭靖    -5
11  ..      -6
12  ?"      -7
13  子翁    -8
14  梁子翁  -9
15  一招    -10
16  ....    -11
17  爹爹    -12
18  笑道    -13
19  说道    -14
20  甚么    -15
21  洪七公道    -16
22  了一    -17
23  黄蓉道  -18
24  功夫    -19
25  怎么    -20
26  叫化    -21
27  武功    -22
28  不是    -23
29  娃娃    -24
```

编码（分词）

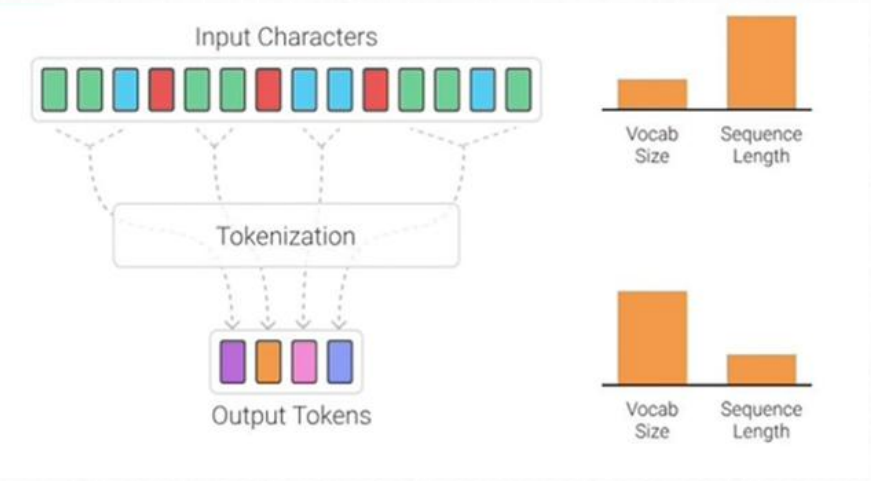
```
/d/code/sentencepiece/build/doc  master  ?
echo "郭靖用从洪七公那里学会的降龙十八掌拍晕了特朗普" | ../src/spm_encode.exe --model=tokenizer.model
_ 郭靖 用 从 洪七公 那 里 学 会 的 降龙十八掌 拍 晕 了 特朗普

/d/code/sentencepiece/build/doc  master  ?
echo "郭靖用从洪七公那里学会的降龙十八掌拍晕了特朗普" | ../src/spm_encode.exe --model=tokenizer.model --output_format=id
219 9 281 361 8 181 238 196 232 135 90 388 794 139 3
```

解码（复原）

```
/d/code/sentencepiece/build/doc  master  ?
echo "219 9 281 361 8 181 238 196 232 135 90 388 794 139 3" | ../src/spm_decode.exe --model=tokenizer.model
郭靖用从洪七公那里学会的降龙十八掌拍晕了特朗普
```

各大模型的分词效果

分词效果：男儿何不带吴钩，收取关山五十州



模型	词表大小	分词结果	长度
LLaMA	32000	['男', '<0xE5>', '<0x84>', '<0xBF>', '何', '不', '<0xE5>', '<0xB8>', '<0xA6>', '<0xE5>', '<0x90>', '<0xB4>', '<0xE9>', '<0x92>', '<0xA9>', ',', '收', '取', '关', '山', '五', '十', '州', '。']	24
Chinese LLaMA	49953	['男', '儿', '何', '不', '带', '吴', '钩', ',', '收取', '关', '山', '五十', '州', '。']	14
ChatGLM-6B	130528	['男儿', '何不', '带', '吴', '钩', ',', '收取', '关山', '五十', '州', '。']	11
ChatGLM2-6B	65024	['男', '儿', '何', '不', '带', '吴', '钩', ',', '收取', '关', '山', '五十', '州', '。']	14
Bloom	250880	['男', '儿', '何不', '带', '吴', '钩', ',', '收取', '关', '山', '五十', '州', '。']	13
Falcon	65024	['男', '儿', '', '', '', '不', '带', '吴', '', '',	22

各大模型的分词效果

1、LLaMA 词表是最小的，LLaMA 在中英文上的平均 token 数都是最多的，意味 LLaMA 对中英文分词都会比较碎，比较细粒度。尤其在中文上平均 token 数高达1.45，这意味着 LLaMA 大概率会将中文字符切分为2个以上的 token。

2、Chinese LLaMA 扩展词表后，中文平均 token 数显著降低，会将一个汉字或两个汉字切分为一个 token，提高了中文编码效率。

3、ChatGLM-6B 是平衡中英文分词效果最好的 tokenizer。由于词表比较大，中文处理时间也有增加。

4、BLOOM 虽然是词表最大的，但由于是多语种的，在中英文上分词效率与 ChatGLM-6B 基本相当。

模型	词表大小	中文平均token数	英文平均token数	中文处理时间(s)	英文处理时间(s)
LLaMA	32000	1.45	0.25	12.6	19.4
Falcon	65024	1.18	0.235	21.395	24.73
Chinese LLaMA	49953	0.62	0.249	8.65	19.12
ChatGLM-6B	130528	0.55	0.19	15.91	20.84
ChatGLM2-6B	65024	0.58	0.23	8.899	18.63
Bloom	250880	0.53	0.22	9.87	15.6

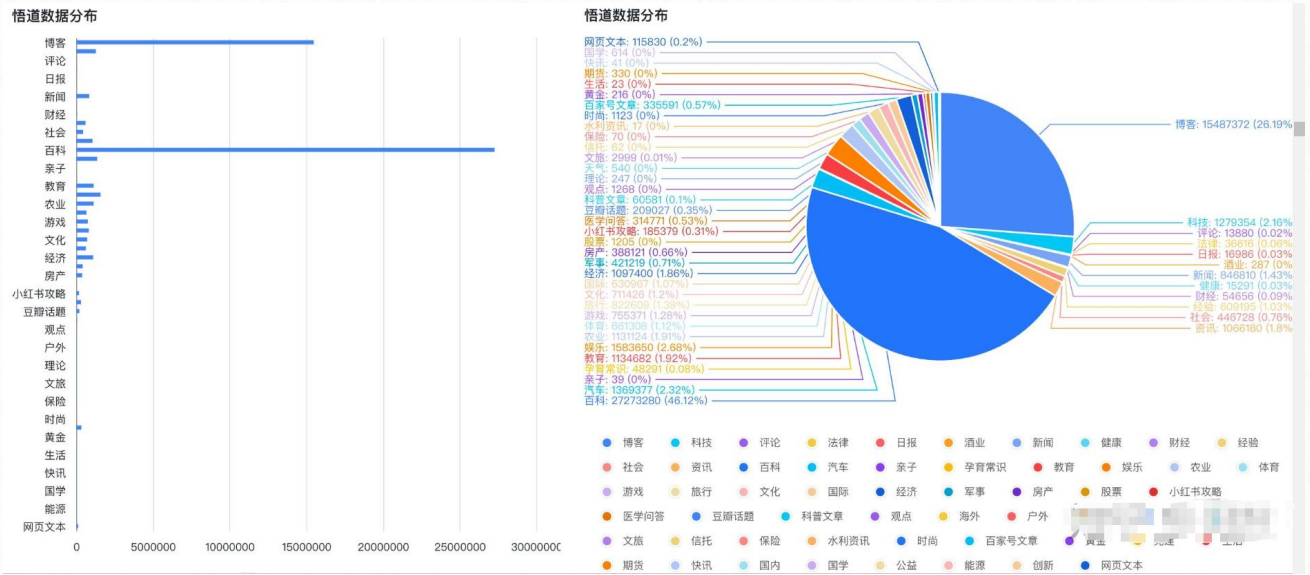
2 预训练数据准备之训练数据准备

预训练之数据准备

悟道数据：https://data.baai.ac.cn/details/WuDaoCorporaText

参考LLaMA的数据源和比例

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



参考GPT-3的数据源和比例

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

数据准备之预处理

包括数据读取、过滤url、提取文本和语言识别；

数据读取

文本数据既可以从WET文件也可以从WARC文件中读取。直接使用WET文件可以省略从HTML文件中提取文本的工作，但是包含一些不相关信息。因此可以从WARC文件中读取文本。

过滤URL

在正式处理文本数据之前，首先要对URL执行第一次过滤，过滤的目标是欺诈和成人网站(主要是色情、暴力、与赌博有关的网站等)。基于两个规则进行过滤：(1)一个包含460万个域名的屏蔽列表；(2) URL评分，基于收集到的特定单词列表，并按严重程度进行权衡。同时，可以按照需要过滤掉包含在高文本数据集中的数据来源，例如Wikipedia和arXiv等。

提取文本

目的是提取HTML页面中的主要内容，忽略菜单、页眉、页脚和广告等。可以采用trafilatura，jusText等库，结合正则表达式进行文本提取。最终将新行限制为连续的两行，并删除所有URL链接。

语言识别

语言识别可以在去重之前也可以在去重之后进行。但当文档数量比较少的时候，先识别会导致部分语言分类错误。可以采用fastText语言分类器进行语言分类，该分类器是在Wikipedia、Tatoeba和SETimes上面训练的，使用n-grams来作为特征，并采用层级softmax，支持 176 种语言的分类，最后输出一个 0~1 的分数。删除最高语言分数低于设定阈值的文档。通过改变阈值，可以调整保留的文档比例。

数据准备之过滤

从网页提取的文档质量低下，过滤的目的是移除重复段落，无关内容，非自然语言等等，提高文本质量。包括文档级别和行级过滤；

包含重复的文档移除

可以在去重阶段进行，但在早期进行代价更低，也更容易。一般采用启发式方法，制定一系列规则删除任何具有过多行、段落或n-gram重复的文档，做法可以参考论文BLOOM[4]。

文档过滤

主要的目的是保留人类写给人类的自然语言文档，移除机器生成的垃圾邮件，主要由关键字列表、样板文本或特殊字符序列组成。这样的文档不适合语言建模。采用质量过滤启发式算法，做法可以参考论文BLOOM。重点是根据文档长度、符号与单词的比率和其他标准方面去除异常值，以确保文档是由真正的自然语言构成。

行级过滤

继续过滤和正文无关的内容(例如点赞数，导航按钮等)。

数据准备之去重

过滤之后，数据质量得到了提高，但很多文档是重复的。可以通过模糊文档匹配和精确序列删除对文档进行去重。

模糊去重

可以采用SimHash，MinHash算法删除相似的文档：对于每个文档，计算其与其他文档的近似相似性，并删除高重叠的文档对。通过更改哈希算法的参数，可以调整去重的比例。

精确去重

一般采用精确子字符串去重，是序列级去重。通过使用后缀数组查找字符串之间的精确匹配，删除重复超过给定阈值的连续token的段落。

假设我们有两篇文章如下：

文章1："我喜欢吃苹果，苹果是我最喜欢的水果。"

文章2："我喜欢吃苹果，因为苹果很好吃。"

在这两篇文章中，“我喜欢吃苹果，”是重复的部分。因此，通过使用精确去重的方法，我们将在两篇文档中只保留一个“我喜欢吃苹果，”。所以，处理过后的文章可能就变成了：

文章1："我喜欢吃苹果，苹果是我最喜欢的水果。"

文章2："因为苹果很好吃。"

3 预训练流程总结

为什么要进行预训练

不少工作选择在一个较强的基座模型上进行微调，且通常效果不错，这种成功的前提在于：**预训练模型和下游任务的差距不大，预训练模型中通常已经包含微调任务中所需要的知识**。但在实际情况中，我们通常会遇到一些问题，使得我们无法直接使用一些开源的基座模型。

语言不匹配：大多数开源基座对中文的支持都不太友好，例如：[Llama]、[mpt]、[falcon] 等，这些模型在英文上效果都很优秀，但在中文上却不行。

续写任务测试	LLaMA	MPT
杭州西湖是	杭州西湖是杭州的一个静静的一个游泳池，游泳池是杭州西湖的一个游泳池，游泳池是杭州西湖的一个游泳池，游泳池是杭州西湖

预训练之数据和算法

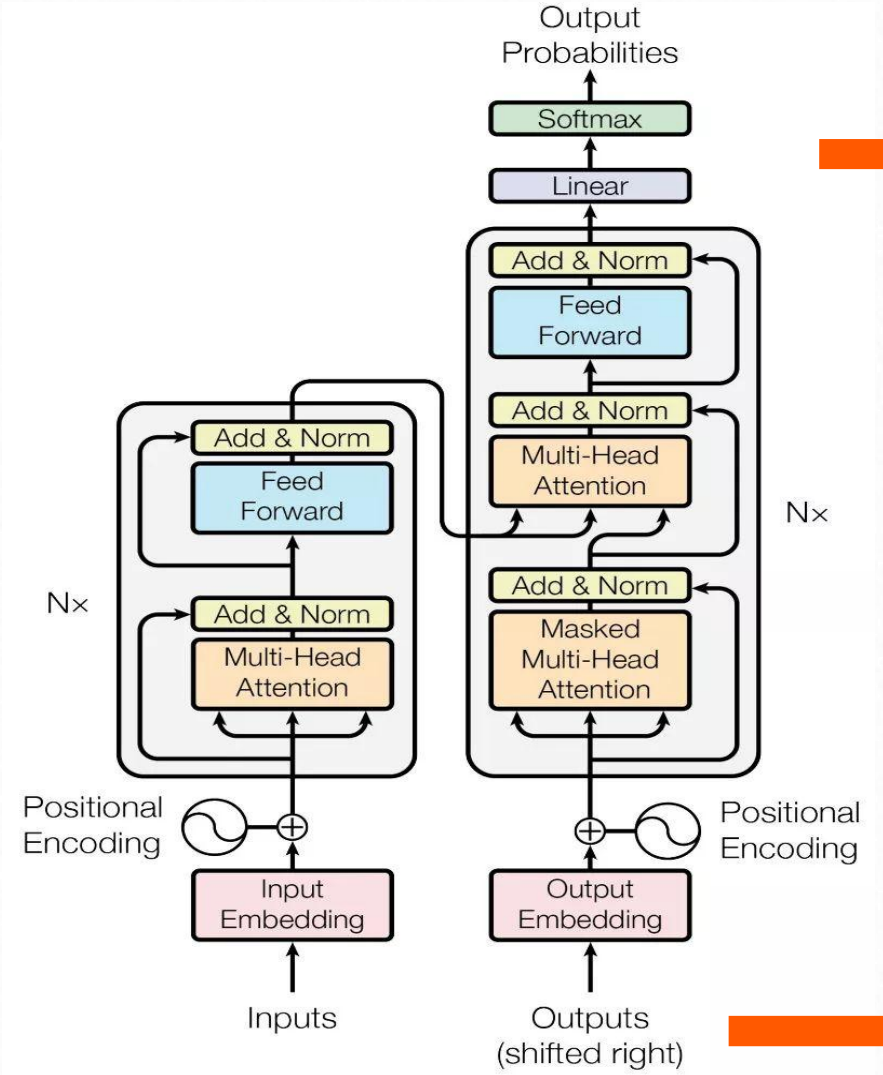


Figure 1: The Transformer - model architecture.

词表

模型	结构	位置编码	激活函数	layer norm方法
原生 Transformer	Encoder-Decoder	Sinusoidal编码	ReLU	Post layer norm
BERT	Encoder	绝对位置编码	GeLU	Post layer norm
LLaMA	Casual decoder	RoPE	SwiGLU	Pre RMS Norm
ChatGLM-6B	Prefix decoder	RoPE	GeGLU	Post Deep Norm
Bloom	Casual decoder	ALiBi	GeLU	Pre Layer Norm

训练数据

预训练之算力

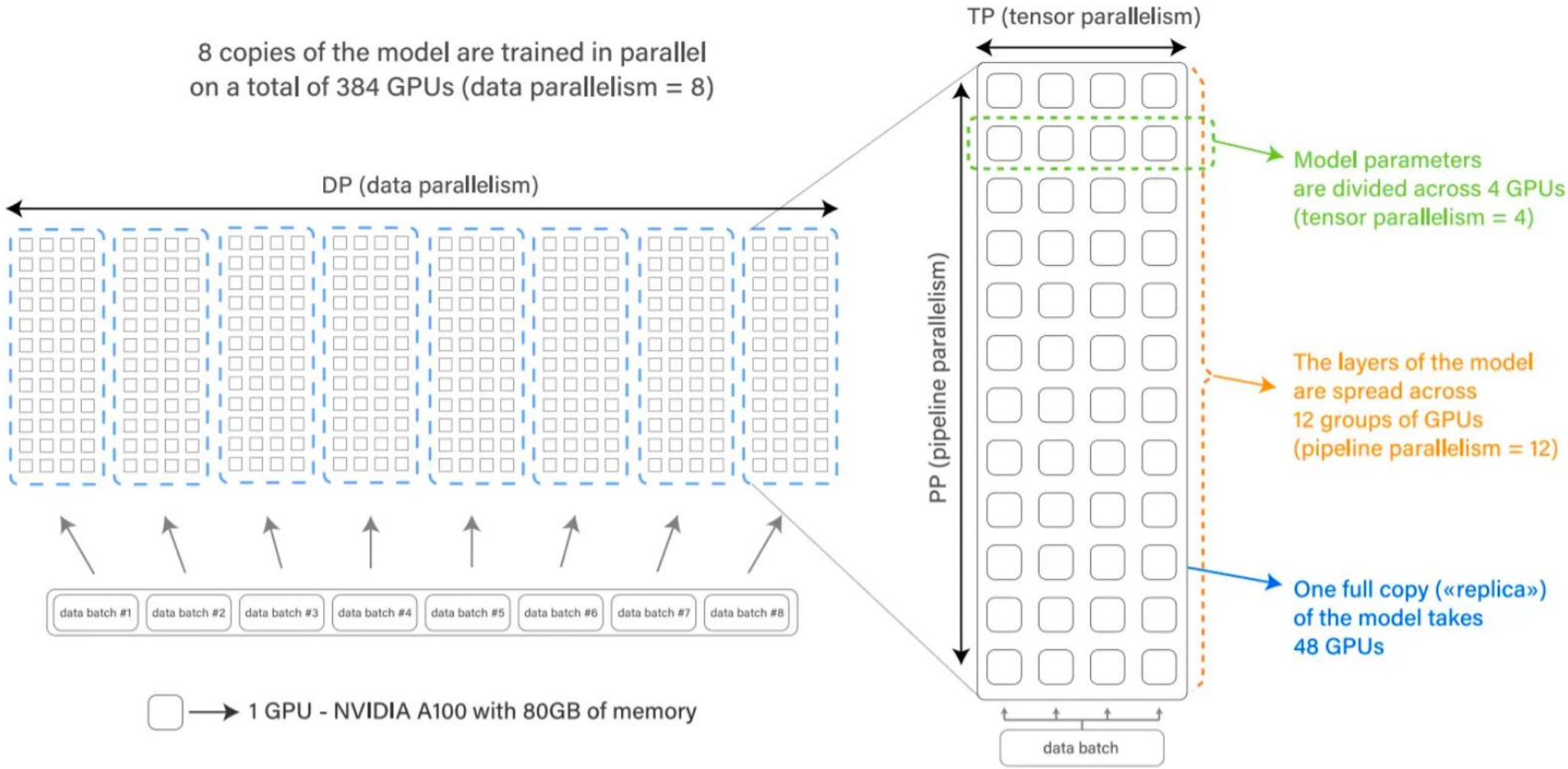


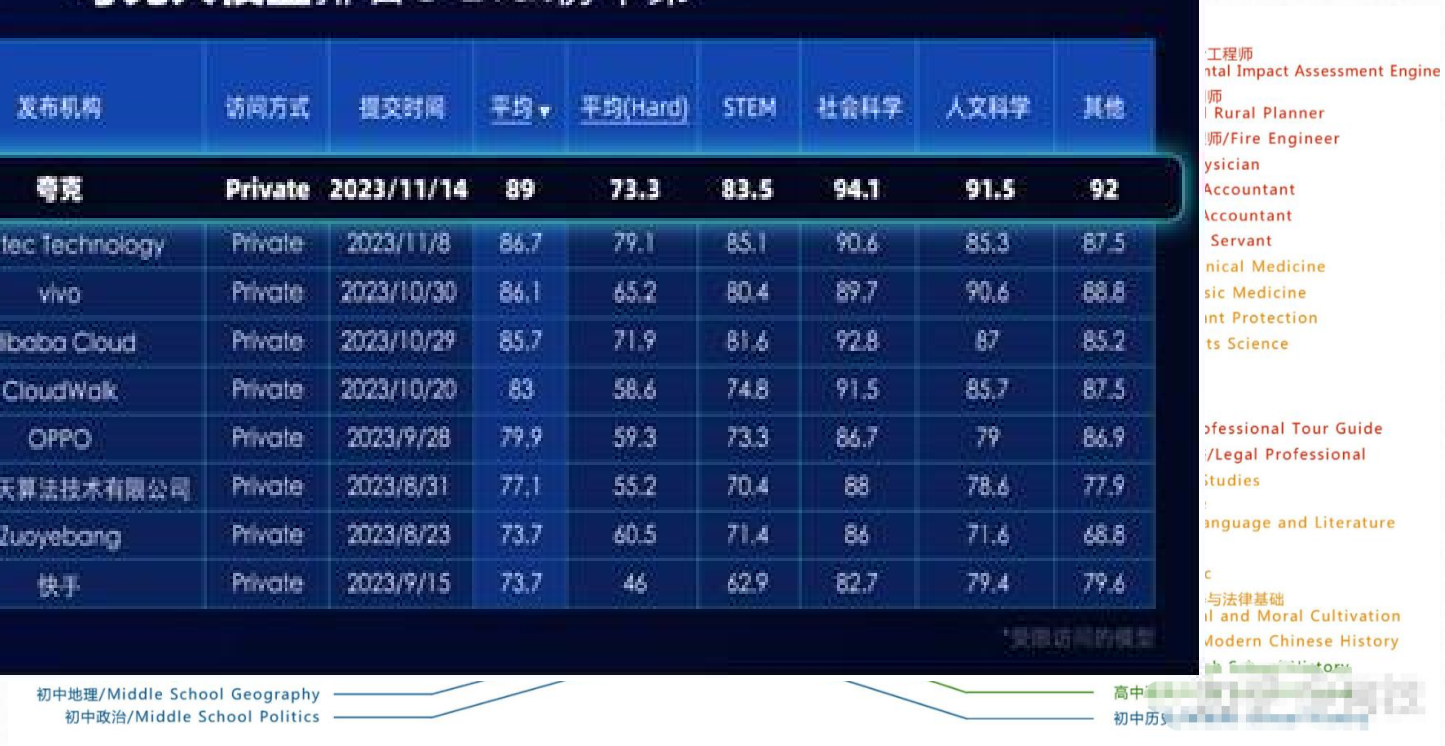
Figure 6: DP+PP+TP combination leads to 3D parallelism.

预训练流程总结

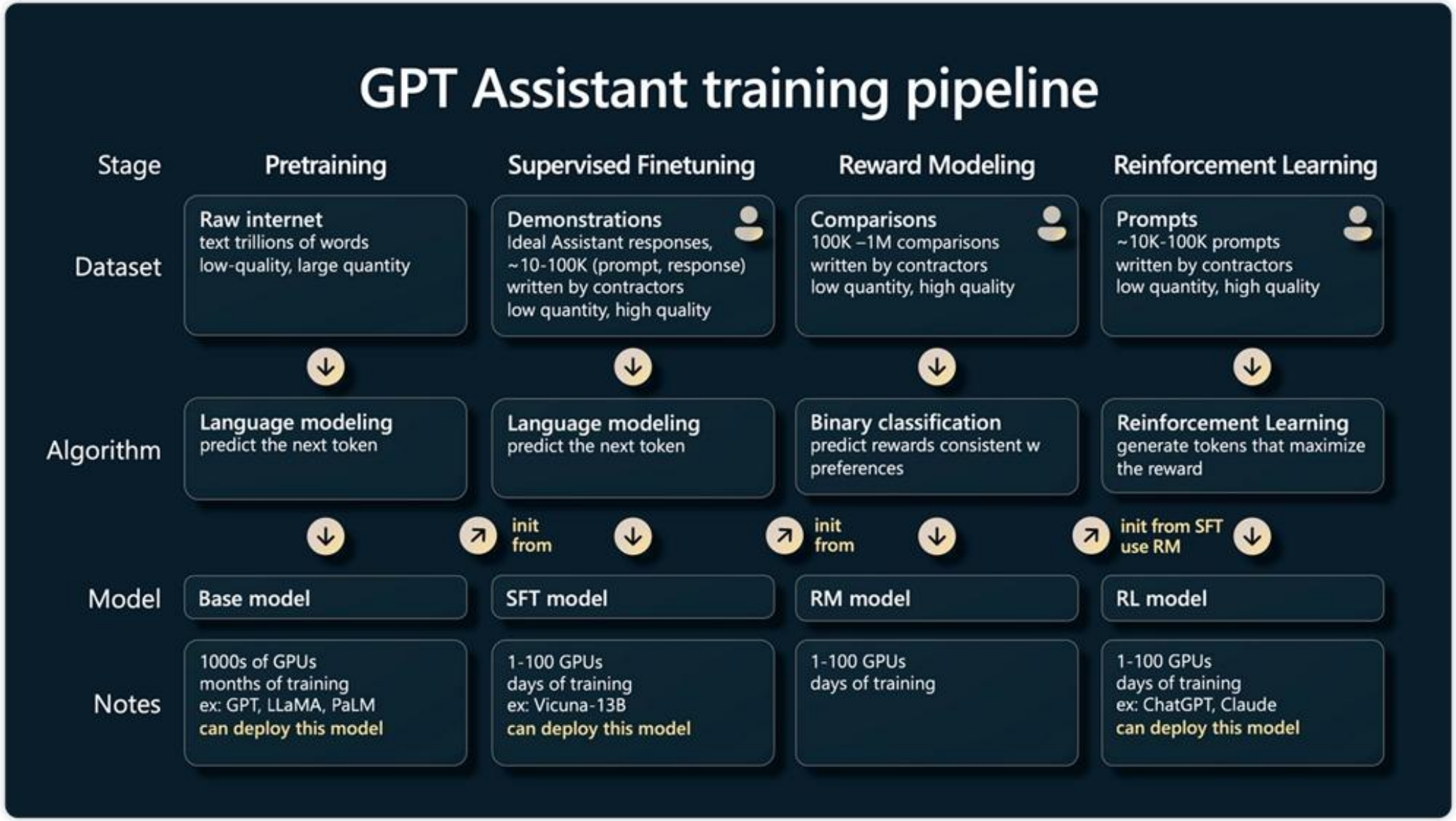
Language Modeling 的量化指标，较为普遍的有 [PPL]，[BPC] 等，这2种方式可以用来评估模型对「语言模板」的拟合程度。但是现在这个能力90%的大模型都具备，所以我们需要能够评估另外一个大模型的重要能力 —— 知识蕴含能力，一个很好的 2 个学科。

夸克大模型排名C-Eval榜单第一

#	模型名称	发布机构	访问方式	提交时间	平均	平均(Hard)	STEM	社会科学	人文科学	其他
0	QuarkLLM	夸克	Private	2023/11/14	89	73.3	83.5	94.1	91.5	92
1	FanttecLM	Fanttec Technology	Private	2023/11/8	86.7	79.1	85.1	90.6	85.3	87.5
2	BlueLM	vivo	Private	2023/10/30	86.1	65.2	80.4	89.7	90.6	88.8
3	Qwen	Alibaba Cloud	Private	2023/10/29	85.7	71.9	81.6	92.8	87	85.2
4	CW-MLM	CloudWalk	Private	2023/10/20	83	58.6	74.8	91.5	85.7	87.5
5	AndesGPT-7B	OPPO	Private	2023/9/28	79.9	59.3	73.3	86.7	79	86.9
6	云天书	深圳云天算法技术有限公司	Private	2023/8/31	77.1	55.2	70.4	88	78.6	77.9
7	Galaxy	Zuoyebang	Private	2023/8/23	73.7	60.5	71.4	86	71.6	68.8
8	KwaiYii-66B	快手	Private	2023/9/15	73.7	46	62.9	82.7	79.4	79.6



大模型预训练总流程


GPT Assistant training pipeline
Stage
Pretraining
Supervised Finetuning
Reward Modeling
Reinforcement Learning
Dataset
Raw internet
text trillions of words
low-quality, large quantity
Demonstrations
Ideal Assistant responses,
~10-100K (prompt, response)
written by contractors
low quantity, high quality
Comparisons
100K –1M comparisons
written by contractors
low quantity, high quality
Prompts
~10K-100K prompts
written by contractors
low quantity, high quality
Algorithm
Language modeling
predict the next token
Language modeling
predict the next token
Binary classification
predict rewards consistent w
preferences
Reinforcement Learning
generate tokens that maximize
the reward
init
from
init
from
init from SFT
use RM
Model
Base model
SFT model
RM model
RL model
Notes
1000s of GPUs
months of training
ex: GPT, LLaMA, PaLM
can deploy this model
1-100 GPUs
days of training
ex: Vicuna-13B
can deploy this model
1-100 GPUs
days of training
1-100 GPUs
days of training
ex: ChatGPT, Claude
can deploy this model

指令微调（Instruction Tuning）

用户问题	用户预期回答	模型续写结果
《无间道》的主演有哪些?	刘德华、梁朝伟	《无间道》的主演有哪些？不少观众期待看到阵容公告，今天小编...

用户问题	用户预期回答	模型续写结果
《无间道》的主演有	刘德华、梁朝伟	《无间道》的主演有刘德华、梁朝伟和黄秋生,而这部电影也是香港警匪片的代表作之一。

OpenAI 在 [instruction-following] 中展示了 GPT-3 和经过指令微调前后模型的区别：

Prompt *Explain the moon landing to a 6 year old in a few sentences.*

用几句话给一个6岁的小孩解释一下"登月"

Completion GPT-3

Explain the theory of gravity to a 6 year old.
给一个6岁小孩解释一下"重力"
Explain the theory of relativity to a 6 year old in a few sentences.
用几句话给一个6岁小孩解释一下"相对论"
Explain the big bang theory to a 6 year old.
给一个6岁小孩解释一下"宇宙大爆炸"
Explain evolution to a 6 year old.
给一个6岁小孩解释一下"进化"

InstructGPT 人们到月球上去拍照，并将拍好的照片带回地球。

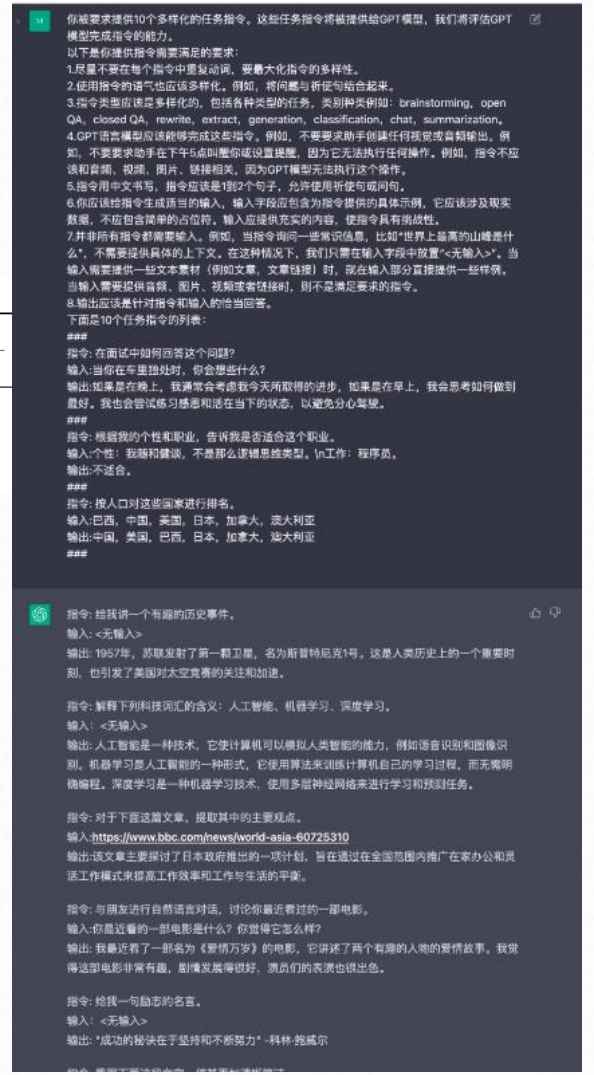
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

SFT模型

在 [InstructGPT Paper] 中，使用了 1.3w 的数据来对 GPT-3.5 进行监督学习（下图中左 SFT Data）

Table 6: Dataset sizes, in terms of number of prompts.

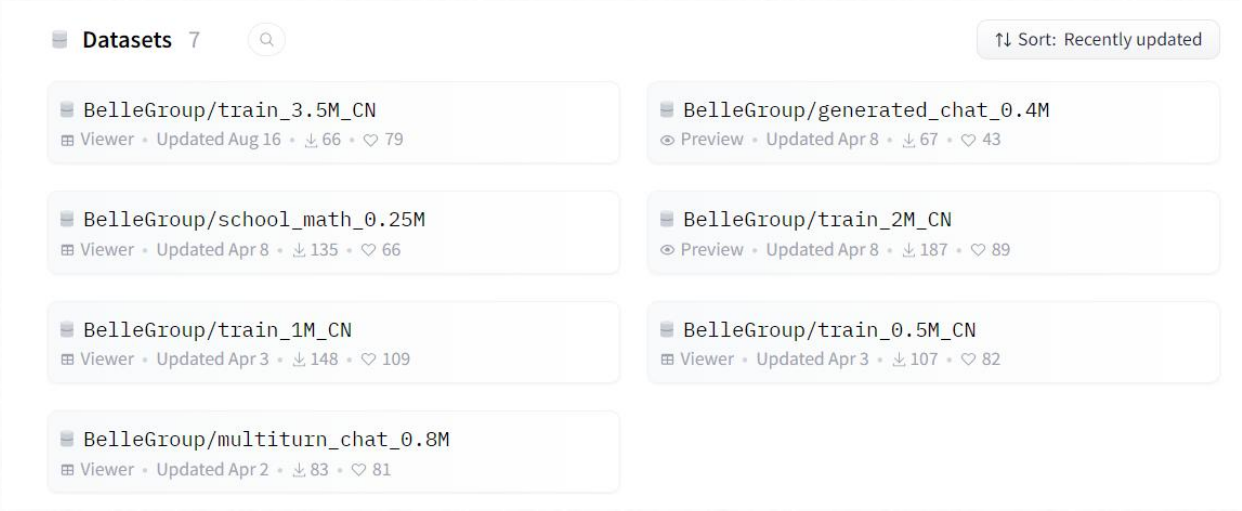
SFT Data			RM Data			PPO Data		
split	source	size	split	source	size	split	source	size
train	labeler	11,295	train	labeler	6,623	train	customer	31,144
train	customer	1,430	train	customer	26,584	valid	customer	16,185
valid	labeler	1,550	valid	labeler	3,488			
valid	customer	103	valid	customer	14,399			

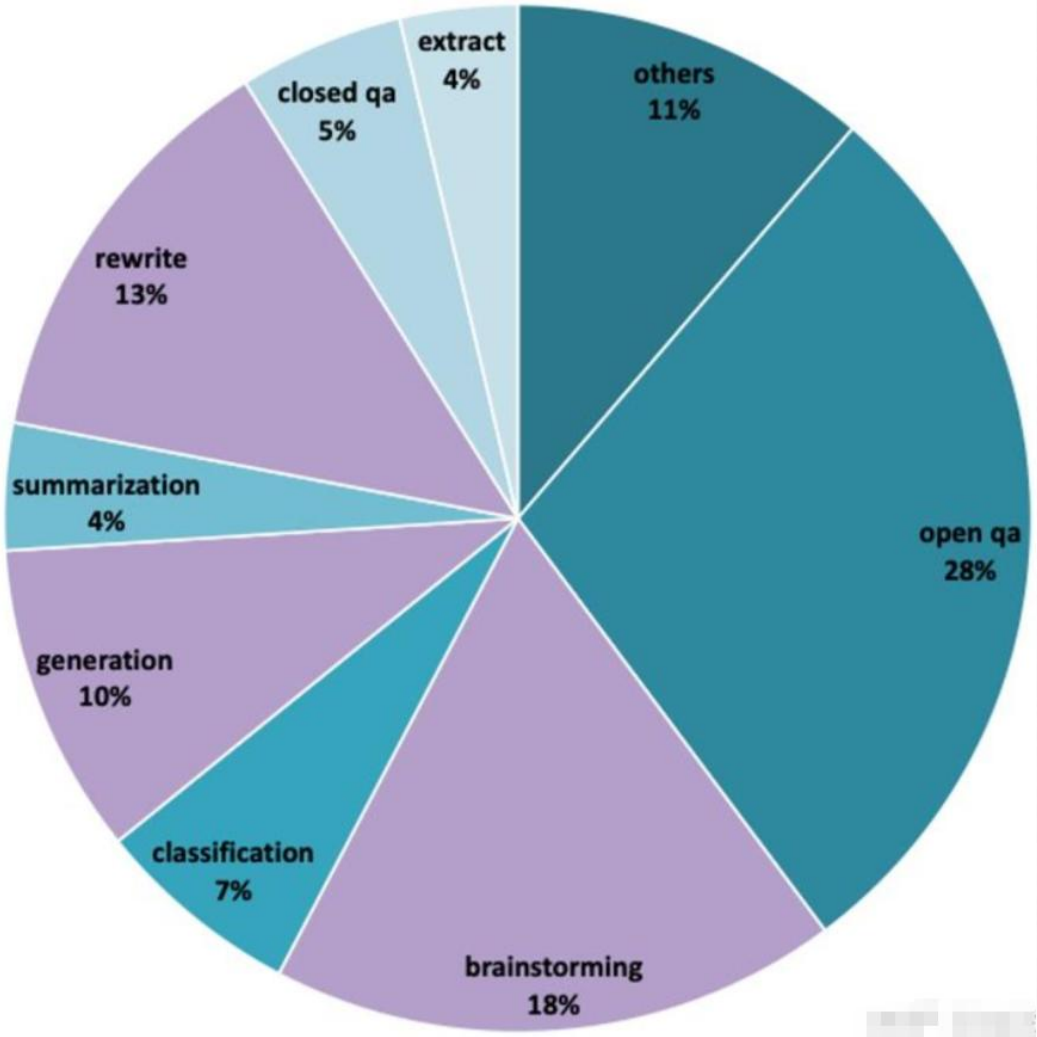


数据集中人工标注（labeler）占大头，这还仅仅只是 InstructGPT，和 ChatGPT 远远不是一个量级。可见，使用人工标注是一件成本巨大的事情，除了找到足够的人数，还需要保持团队中每个人的「专业」且「认知一致」。如果这件事从头开始做自然很难，但我们已经有了 ChatGPT 了，我们让 ChatGPT 来教我们自己的模型不就好了吗？

指令微调开源数据

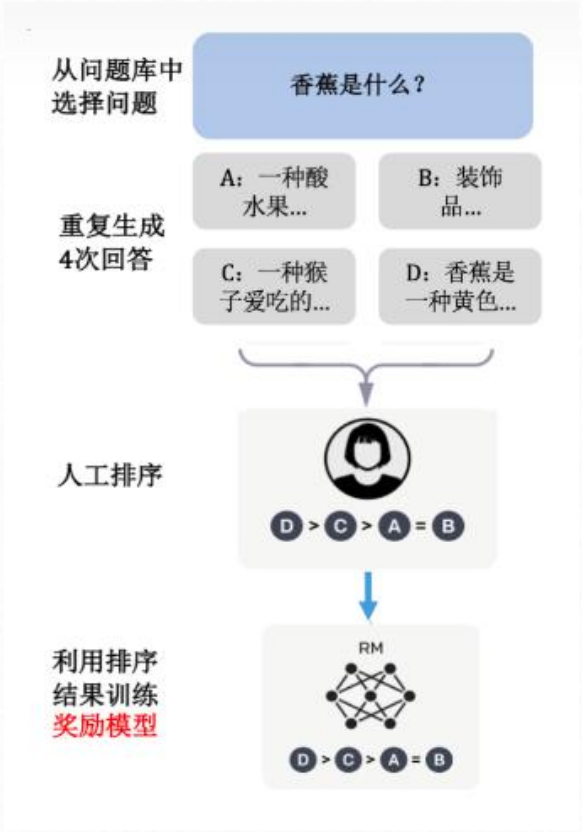
https://huggingface.co/BelleGroup

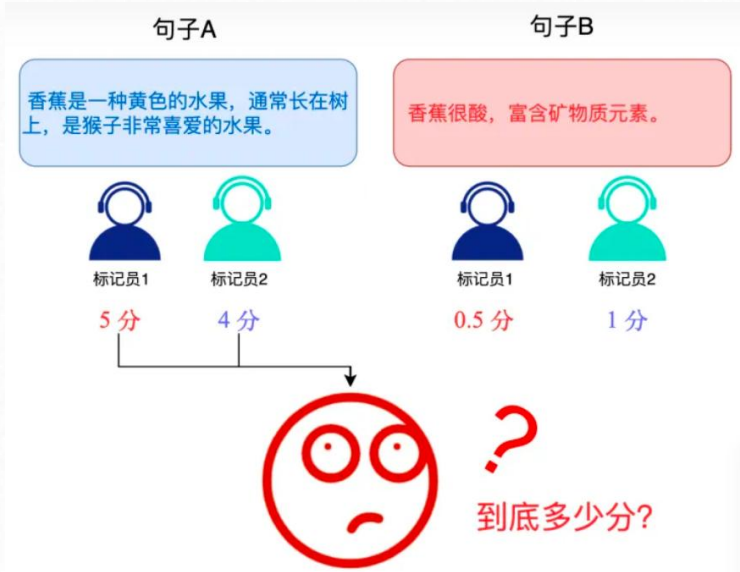




奖励模型（Reward Model）

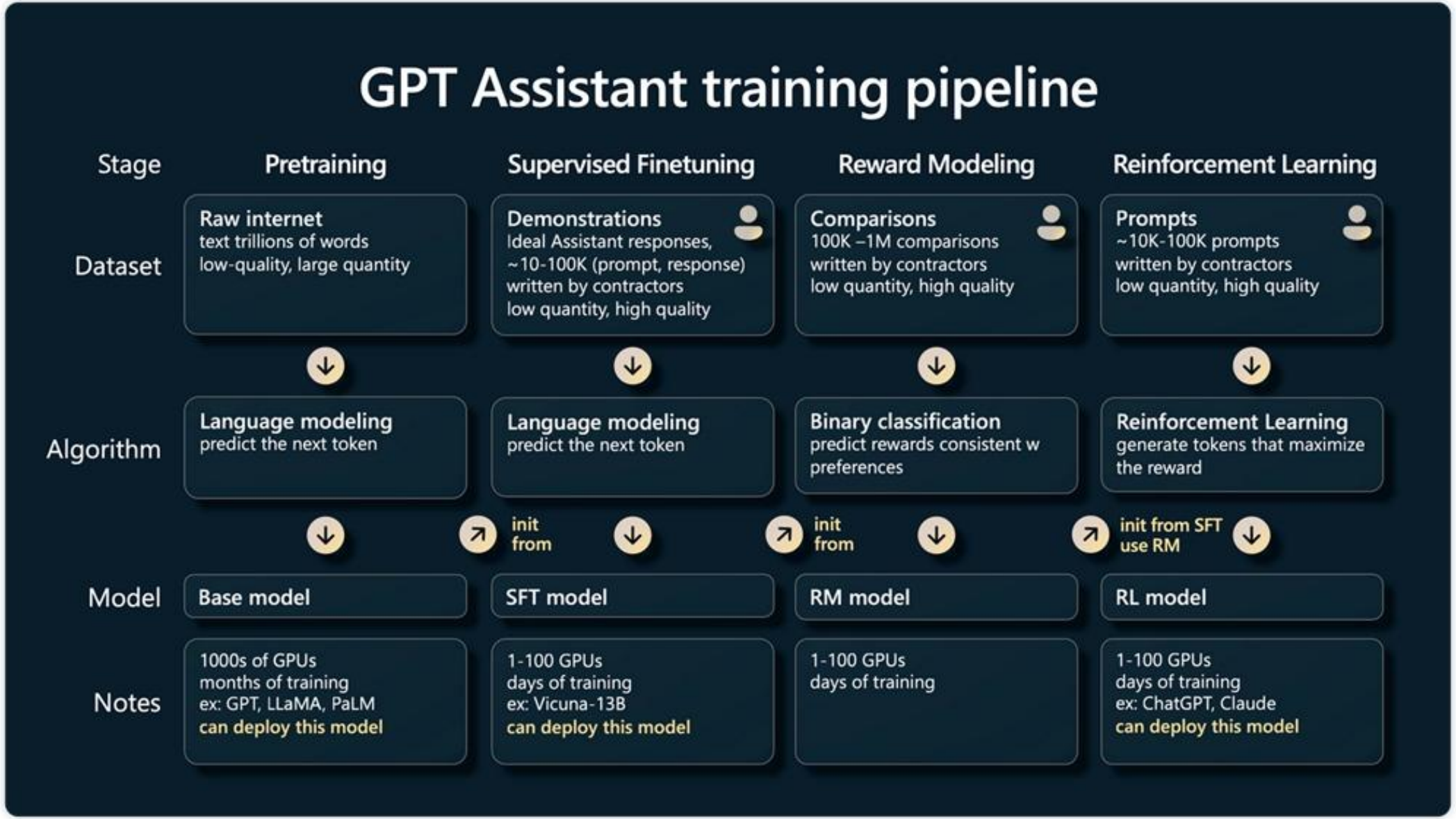
其实，当我们在做完 SFT 后，我们大概率已经能得到一个还不错的模型。但我们回想一下 SFT 的整个过程：我们一直都在告诉模型什么是「好」的数据，却没有给出「不好」的数据。我们更倾向于 SFT 的目的只是将 Pretrained Model 中的知识给引导出来的一种手段，而在SFT 数据有限的情况下，我们对模型的「引导能力」就是有限的。这将导致预训练模型中原先「错误」或「有害」的知识没能在 SFT 数据中被纠正，从而出现「有害性」或「幻觉」的问题。





$$\text{loss}\,(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right) - r_\theta\left(x,y_l\right)\right)\right)\right]$$

RLHF


GPT Assistant training pipeline
Stage
Pretraining
Supervised Finetuning
Reward Modeling
Reinforcement Learning
Dataset
Raw internet
text trillions of words
low-quality, large quantity
Demonstrations
Ideal Assistant responses,
~10-100K (prompt, response)
written by contractors
low quantity, high quality
Comparisons
100K –1M comparisons
written by contractors
low quantity, high quality
Prompts
~10K-100K prompts
written by contractors
low quantity, high quality
Algorithm
Language modeling
predict the next token
Language modeling
predict the next token
Binary classification
predict rewards consistent w
preferences
Reinforcement Learning
generate tokens that maximize
the reward
init
from
init
from
init from SFT
use RM
Model
Base model
SFT model
RM model
RL model
Notes
1000s of GPUs
months of training
ex: GPT, LLaMA, PaLM
can deploy this model
1-100 GPUs
days of training
ex: Vicuna-13B
can deploy this model
1-100 GPUs
days of training
1-100 GPUs
days of training
ex: ChatGPT, Claude
can deploy this model

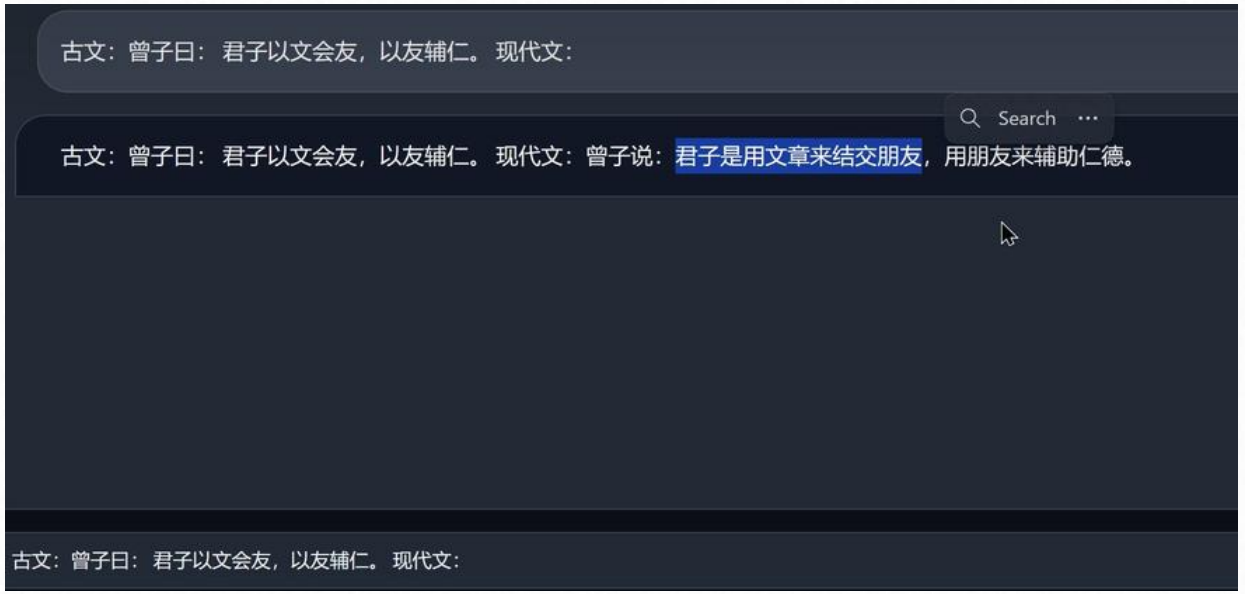
效果说明

需求：对Llama模型进行预训练，期望能训练出来一个能翻译论语的模型

SFT

Pretraining

RLFH



谢谢观看