Tokenizer篇：

LLM的基本运作原理是：将输入内容进行分词映射，变为数字之后输入到模型当中，最后将计算结果编码生成文本。

Tokenizer则是用于将输入内容分词，并且映射到词表中。

目前主流的分词算法就那么几个，我们这里使用BPE进行分词。

借助huggingface的tokenizer库，调用现有分词器进行微调或是从零开始训练都是可以的。

参阅代码以及注释，修改源数据格式，然后即可使用Qwen分词器进行重新生成。

由于BPE算法的特点，在相同的数据集上进行相同的分词，最终得到的结果也一定是相同的。

所以我们需要的就是足够多的数据。

LLM各个层（Layer）作用：

第1层：Embedding层：

这一层主要有两个工作，一个是将我们的文字分词映射成数字形式，另一个是对内容进行一个“定位”。

具体来说，在Attention的机制下，“我喜欢你”和“你喜欢我”，这两个词组对于LLM来说意义是一样的，但是在自然语言的语境中，这很明显有较大的区别，那我们就需要对词进行一个“定位”，将每个词的位置信息连同它的词表信息一并输入模型，这样我们的模型就能读懂真正含义了。

第2~N-1层：Transformer层：

核心内容，具体原理参考论文：Attention is all you need

对输入内容进行运算并且储存参数的层，是模型的主要部分。

第N层：归一化，线性层：

最后将输出的参数转化为概率，然后输出一个[a,b,c,d…]这样的列表，通过词表中的映射关系反向编码成文字形式：

这是一个例子：

[‘I’,’Like’,’The’,’World’]→[1,2,3,4]

假设最后编码结果是：

[3,1,2,4]，那么输出的内容就是The I Like World这样。