# LLM文档

目录

[LLM文档 1](#_Toc199190521)

[一、Tokenizer 1](#_Toc199190522)

[1.1 Word-level分词器 1](#_Toc199190523)

[1.2 Character level 1](#_Toc199190524)

[1.3 Subword-level 1](#_Toc199190525)

[1.4 Byte level 4](#_Toc199190526)

## 一、Tokenizer

### 1.1 Word-level分词器

1992年，分词Tokenization概念首次出现。Jonathan J. Webster等人在论文[Tokenization as the initial phase in NLP](https://link.zhihu.com/?target=https%3A//aclanthology.org/C92-4173.pdf" \t "_blank)中指出，是文本的基本单元。

Word-level每个英文单词（word）作为一个 token，使用空格进行分割。中文常配合字典进行匹配（如结巴分词）。

缺点：此表过大，容易OOV

### 1.2 Character level

将每个字符作为一个 token，从而解决 OOV 问题。

优点：词表小，没有OOV

缺点：学习难度大

### 1.3 Subword-level

(1) WordPiece：代表模型Bert.

WordPiece 的训练过程步骤：

1. 初始化词表：从所有单字符开始构建。



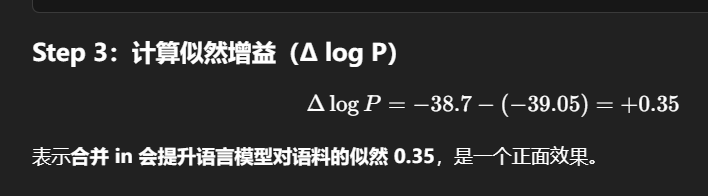




1. 计算子词对合并带来的语言建模概率提升（似然增益）。



1. 选择带来最大似然增益的子词对进行合并，词表中子词后缀用 ## 表示非词首。



4.重复该过程直到词表大小达到目标值。

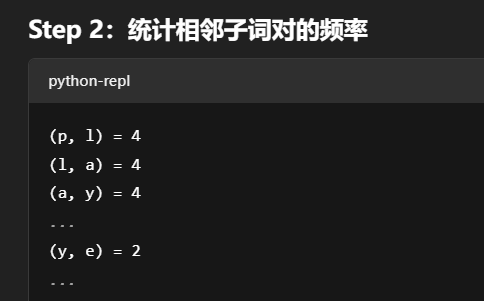
训练后使用时采用贪心策略。

(2)BPE:代表模型Llama, GPT

1.**初始化词表**：将语料中的所有词拆分为单字符加终止符（如 l o w </w>），词表初始化为所有可能的字符。



2.**统计子词对频率**：遍历所有词，统计相邻子词对的出现频率。



**3.合并最频繁子词对**：将出现频率最高的子词对合并为一个新子词（如 e 和 r 合并为 er），并更新所有词中对应位置。



4.**重复合并**：继续步骤 2–3，直到词表达到预设大小（例如 30,000 个 token）。



### 1.4 Byte level

