# BPE 算法详解与 LLM Tokenizer 训练流程

Your Name

2025 年 3 月 31 日

## 1 BPE 算法（Byte Pair Encoding）

### 1.1 算法背景

BPE 是一种**数据压缩算法**，由 Philip Gage 于 1994 年提出。在 NLP 领域，Sennrich 等人 (2015) 首次将其应用于子词分词，主要解决：

- 开放词汇表问题 (Open Vocabulary)
- 罕见词表示问题 (Rare Word Representation)
- 多语言统一处理需求

### 1.2 数学形式化定义

给定语料库 $D$，初始化字符集 $V_0$，迭代执行以下操作：

$$(v_i, v_j) = \arg\max_{(x,y)\in P} \text{count}(x, y)$$

其中：

- $P$ 为当前所有可能相邻符号对集合
- $\text{count}(x, y)$ 表示 $(x, y)$ 在 $D$ 中的共现频率

合并操作更新词汇表：

$$V_{k+1} = V_k \cup \{v_i v_j\} \setminus \{v_i, v_j\}$$

# 2 BPE 核心算法流程

**Algorithm 1** BPE 训练过程

**输入:** 原始文本语料 $D$，目标词汇量 $K$

**输出:** BPE 词汇表 $V$

预处理：将 $D$ 中所有单词添加终止符 `</w>`
初始化 $V_0 \leftarrow$ 所有唯一字符 + `</w>`
统计所有单词的词频 $f(w)$
**for** $k = 1$ to $K - |V_0|$ **do**
    统计所有相邻符号对频率 $f(pair)$
    选择最高频 pair $(x, y)$
    将合并操作 $x + y \to z$ 加入合并规则表
    更新 $V_k \leftarrow V_{k-1} \cup \{z\}$
    更新语料中所有 $(x, y)$ 出现位置为 $z$
**end for**

## 2.1 详细步骤解释

**步骤 1 符号化预处理**

- 将文本转换为 Unicode 编码（处理多语言）
- 添加单词边界标记：`low` $\Rightarrow$ `l o w </w>`
- 示例词汇表：$V_0 = \{$`l`, `o`, `w`, `</w>`$\}$

**步骤 2 频率统计阶段**

- 构建共现矩阵：$\text{count}(x, y) = \sum_{w \in D} f(w) \cdot N_w(x, y)$
- 其中 $N_w(x, y)$ 表示单词 $w$ 中 $(x, y)$ 的出现次数

**步骤 3 动态合并过程**

表 1: BPE 合并过程示例

| 迭代次数 | 合并操作 | 新词汇表 |
|---|---|---|
| 1 | (o, w) → ow | `ow, l, o, w, </w>` |
| 2 | (l, ow) → low | `low, ow, l, o, w, </w>` |
| 3 | (e, r) → er | `er, low, ow, ...` |

**步骤 4 终止条件**

- 预设条件：达到目标词汇量 $K$（典型值：32k, 50k）
- 动态条件：$\max f(pair) < \theta$（阈值 $\theta$ 通常设为 2）

# 3 LLM Tokenizer 训练流程

## 3.1 完整训练架构

[width=0.8]bpe$_p$*ipeline.png*

图 1: BPE Tokenizer 训练流程图

## 3.2 关键技术细节

- **字节级处理**（GPT-2 方案）
  - 将文本转换为 UTF-8 字节序列（256 个基础 token）
  - 优点：完全消除未知字符问题
  - 示例：汉字「中」→ `E4 B8 AD`
- **子词正则化**
  - 使用 Unigram 语言模型进行采样
  - 实现概率化的分词结果
- **合并策略优化**

$$\text{Score}(x, y) = \frac{\text{count}(x, y)}{\text{count}(x) \times \text{count}(y)} \tag{1}$$

选择互信息最大的符号对进行合并

# 4 实现对比分析

表 2: BPE 变体对比

| 类型 | BPE | WordPiece | Unigram |
|---|---|---|---|
| 合并策略 | 频率优先 | 最大似然估计 | 概率删除 |
| 训练方式 | 贪婪合并 | 迭代 EM 算法 | 动态规划 |
| 处理未知词 | 子词分解 | 同左 | 同左 |
| 典型应用 | GPT 系列 | BERT | XLNet |

# 附录：代码实现

- Hugging Face 实现核心逻辑：

```
class BPE:
    def train(self, texts, vocab_size):
        merges = []
        vocab = self._build_initial_vocab(texts)
        while len(vocab) < vocab_size:
            pairs = self._get_pairs_with_freq(texts)
            if not pairs:
                break
            best_pair = max(pairs, key=pairs.get)
            texts = self._merge_pair(best_pair, texts)
            merges.append(best_pair)
            vocab.add(''.join(best_pair))
        return merges
```