中国科学技术大学
University of Science and Technology of China

词法分析

《编译原理(H)》

张昱

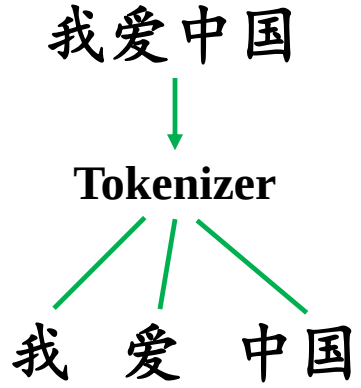
0551-63603804，yuzhang@ustc.edu.cn

中国科学技术大学

计算机科学与技术学院

从词法分析到 LLM Tokenizer

- 在大语言模型（LLM）中，Tokenizer 扮演了类似词法分析器的角色
  - 输入：自然语言（甚至多模态输入，如文本、图片描述、emoji符号）
  - 输出：tokens，即模型的最小处理单元
- LLM的分词特点
  - 统计驱动：分词规则不是人工定义的，而是从大规模语料中训练学习得到
  - 子词分词（subword tokenization）：将频繁出现的字符串作为一个token，将不常见的词拆分成更小的、有意义的子词甚至字符。
  - 文本Tokenizer的主流算法
    - BPE(OpenAI GPT系列使用)
    - WordPiece(Google BERT使用)
    - SentencePiece(Google的T5、DeBERTa使用)



分词算法：BPE (Byte Pair Encoding)

起源

1994年提出的数据压缩算法(减少文件大小)，后被NLP用来做子词分词[ACL2016]

思想

- 从最小单元开始（通常是字符）
- 反复合并高频的相邻单元对；逐步得到越来越大的“子词单元”，作为 Tokenizer 的词汇表

BPE的过程就像在学习一个新词：

- 一开始只认识 “e” 和 “s”，当 “es” 出现次数足够多时，就把它当成整体来记；
- 然后“es”和“t”又经常出现，被合并成“est”，
- 最终可能把 “newest” 当作一个完整的词法单元

分词算法：字节级 BPE (Byte-level BPE)

BPE的不足

- BPE分词以“字符”为基本单位，对于纯英文文本效果很好
- 但对于中文、日文、甚至emoji等非拉丁字符，将Unicode所有字符都纳入词汇表，词汇表会变得非常庞大，且仍然无法覆盖所有可能的字符。

词汇表（Vocabulary）是一个从Token字符串到ID的映射字典。其大小是关键超参数（通常在几万到十几万之间）。

字节级BPE

任何文本都是由字节构成，故可以“字节”为基本单位进行BPE分词。

从而，无论是中文、日文还是emoji，都视为字节序列。

示例：Qwen3系列模型的Tokenizer

- **英文句子：** Hello, I love University of Science and Technology of China
- **分词结果：** ['Hello', ',', 'ĠI', 'Ġlove', 'ĠUniversity', 'Ġof', 'ĠScience', 'Ġand', 'ĠTechnology', 'Ġof', 'ĠChina’]

原句中的单词和标点符号合理地被拆分成了多个token。

其中的字符 `Ġ` 表示空格，是BPE分词实现中的特点。

- **中文句子：** 我爱中国科学技术大学
- **分词结果：** ['æĪij’, 'çĪ±’, 'ä¸ŃåĽ½', 'ç§ijåŃ¦æĬĢæľ¯', 'å¤§åŃ¦’]

Qwen3的BPE分词实际上**字节级BPE**，这些拆分的tokens实际上是字节级的，所以只是看起来像乱码

重新解码： ['我', '爱', '中国', '科学技术', '大学']

分词算法：WordPiece

- 起源：Google BERT
- 原理

  基于概率最大化原则。给定一个词，选择的子词划分应使得训练语料中整体概率最大。

  公式：score = (freq_of_pair) / (freq_of_first_element * freq_of_second_element)

- 优点
  - 对低频词能合理拆分
  - 分词结果稳定，不容易产生奇怪的片段

分词策略：SentencePiece

- 起源：由 Google 提出
- 原理：自上而下
  - 首先用巨大的种子词汇表初始化
  - 逐步移除对整体分词质量贡献最小的词汇，直到词汇表缩小到目标大小

  可以基于 BPE 或Unigram语言模型，直接在原始文本（无需预分词）上训练
- 优点
  - 与语言无关，不依赖空格作为分词边界
  - 常用于多语言模型，支持中文、日文等没有空格的语言

Tokenizer的工作流程

- **规范化Normalization**

  清理文本，如统一转换为小写、去除多余空格、Unicode规范化等

- **预分词Pre-tokenization** “Don't worry.” -> [“Don", "'", "t", " worry", "."]

  根据简单规则(如空格和标点符号)将文本初步分割成“单词”或“粗粒度”的片段

- **分词Tokenization** [“ worry”] -> [“ wor” , “ry”] (BPE示例)

  应用上述BPE/WordPiece算法，将预分词后的片段进一步分成最终的子词Tokens

- **编码Encoding** [“Don” , “ ‘ “ , “t” , “ wor” , “ry” , “.”] -> [1234, 123, 456, 789, 2345, 99]

  将每个Token映射到词汇表中对应的唯一整数ID

- **添加特殊Tokens**

  根据需要添加模型使用的特殊Tokens

解码：是逆过程，将一串Token IDs转换回字符串。

将“ wor” 和“ry” 正确地拼接成“worry” ，而不是“ worry”

分词挑战

中文分词

词间没有空格，预分词步骤需要额外的中文分词工具(如jieba)，
或者直接依赖强大的子词算法(如SentencePiece)将每个汉字视为一个初始符号

一些挑战

- **不一致性：**同一个词在不同上下文中可能有不同的分词方式
- **长度不对等：** Token序列的长度与原始字符串的长度（如字符数）没有固定比例。一个中文字符可能被分成1个或多个Tokens（罕见词）
- **多语言支持：**设计一个能平衡多种语言效率的词汇表非常困难。

通常，像GPT-4这样的模型，其词汇表中会包含大量中、英、法、德等各种语言的常见子词

研究示例：张量程序自动调优的代价模型

Schedule Primitives 3

```
split i=256 to i.0=2, i.1=4, i.2=2, i.3=16
split j=512 to j.0=32, j.1=4, j.2=2, j.3=2
split k=128 to k.0=16, k.1=8
...
reorder i.0, j.0, i.1, j.1, k.0, i.2, ...
compute k.0 at j.1
fuse i.0, j.0, i.1, j.1 to i.0@j.0@i.1@j.1@
parallel i.0@j.0@i.1@j.1@
fuse ax0, ax1 to ax0@ax1@
parallel ax0@ax1@
vectorize j.3
```

预测

代价 0.83

测量

Tensor Program 3

```
parallel i.0@j.0@i.1@j.1@ (0,1024)
  for k.0 (0,16)
    for i.2 (0,2)
      for j.2 (0,2)
        for k.1 (0,8)
          for i.3 (0,16)
            vectorize j.3 (0,2)
              T_dense = ...
parallel ax0@ax1@ (0,131072)
  T_relu = ...
```

split i=256 to i.0=2, i.1=4, i.2=2, i.3=16 保留串和数字

split, i, 256, i.0, 2, i.1, 4, i.2, 2, i.3, 16

编码

1, 0, 0, 0, ..., 0, 0, 1, 256, 2, 2, 3, 4, 4, 2, 5, 16

Type One-hot Table

split	1, 0, 0, 0, ..., 0, 0
reorder	0, 1, 0, 0, ..., 0, 0
compute	0, 0, 1, 0, ..., 0, 0
fuse	0, 0, 0, 1, ..., 0, 0
...	...
parallel	0, 0, 0, 0, ..., 1, 0
vectorize	0, 0, 0, 0, ..., 0, 1

Character Tokens

i	1	j.3	10
i.0	2	k	11
i.1	3	k.0	12
i.2	4	k.1	13
i.3	5	i.0@j.0@i.1@j.1@	14
j	6	ax0	15
j.0	7	ax1	16
j.1	8	ax0@ax1@	17
j.2	9	...	...

(PrimitiveSequence) $S ::= p^*$

(Primitive) $p ::= \tau\ (\text{id} \mid \text{num})^*$

(PrimitiveType) $\mathbf{T} \ni \tau ::= \text{split} \mid \text{reorder} \mid \text{fuse} \mid \ldots$

(NameParam) id

(Number) num

(a) Abstract Schedule Primitive Sequence

(Features) $f = F(p) ::= F_1(\tau)\ (F_2(\text{id}) \mid F_3(\text{num}))^*$

F : Primitive → Features

F_1 : PrimitiveType → OnehotVector

F_2 : NameParam → Token

F_3 : Number → Number

(b) TLP Extractor

TLP-特征提取

调度原语序列➔特征

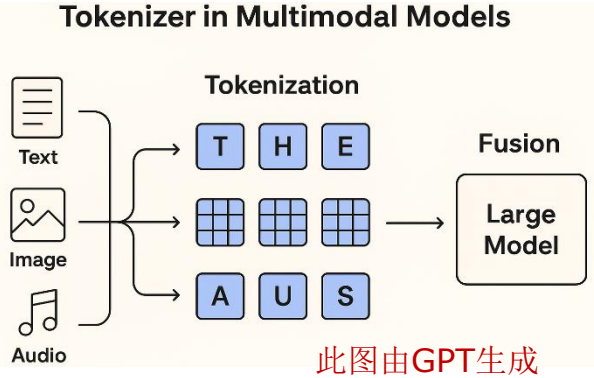
TLP: A Deep Learning-based Cost Model for Tensor Program Tuning. ASPLOS 2023. DOI: 10.1145/3575693.3575737

多模态大模型的分词

把不同模态的原始输入映射为统一的离散表示(token 序列)

图像/语音/视频等是连续模态，需将像素/波形等连续信号编码成离散 token

- **图像 Tokenizer**
  - **使用向量量化，将图像分块编码到离散的codebook(码本，即词典)，如**Imagen、Stable Diffusion
  - 图像切成patch，再编码成token，如ViT Patch Embedding+离散化、Clip(直接embedding)
  - 分层量化：多级量化器编码图像，如RQ-VAE等
- **音频 Tokenizer**
  - 把原始波形压缩成离散码字序列，常用于音乐生成（MusicLM、AudioLM）
  - Residual Vector Quantization，高效表示音频信号
- **视频（时序图像） Tokenizer**
  - **图像 Tokenizer + 时间建模**（逐帧离散化，再建模时序关系）
  - **3D VQ-VAE**：直接在时空域做离散化
- **多模态统一Tokenizer：让不同模态共享词汇表**



此图由GPT生成

多模态大模型的分词

一些挑战

- **模态的差异性：**如何权衡压缩信息量和保持语义细节
- **Token粒度选择：**如何权衡粒度与效率
  - **图像：**patch太小➔token数过多； patch太大➔丢失局部信息
  - **音频：**时间片过短➔序列过长；时间片过长➔难捕捉细节（音色、情感等）

序列长度与计算开销

  - 图像/视频 token 数量远超文本；视频包含时间维度，token 序列可能是文本的 **数百倍**
- **跨模态语义对齐：**如何在统一空间中互相理解
  - 不同模态的 token 在信息含义和统计分布上差异巨大
  - 如何让“文本词元”和“图像 patch token”在共享空间里表示相近语义
- **信息压缩与保真：避免 token 化带来的语义缺失**
  - 离散化不可避免带来信息丢失：丢失纹理、音色等；对下游生成与理解任务都会造成影响