文本预处理

内容提要

- 什么是语言模型？

数据与预处理

- 语料获取与清洗
- 分词
- 词表构建
- 数据格式化

语料获取与清洗

- 过去: 主要使用标准、干净的新闻语料（如华尔街日报）
- 现在: 从互联网（如Common Crawl）获取海量、多样但充满噪声的数据。

- 清洗：去除HTML标签、模板化文本、低质量内容、有害信息等。这是决定现代LLM质量的关键一步

分词

- 过去 (N-gram/RNN): 基于空格或词典的简单分词。遇到未登录词（OOV, Out-of-Vocabulary）是个大问题。
- 现在 (Transformer): 子词分词 (Subword Tokenization) 是主流，如 BPE (Byte-Pair Encoding), WordPiece (BERT), SentencePiece (T5)。
- 优点: 没有真正的OOV问题（任何词都可以被拆分成已知的子词或字符），有效平衡了词表大小和序列长度。例如，“tokenization” -> ["token", "ization"]。
- 重要性: 分词方式直接影响模型的性能和效率，是现代LM预处理的核心。

词表构建

- 词表构建 (Vocabulary Construction)
  - 根据子词分词的结果，构建一个包含几万到十几万个token的词表。特殊token（如 [CLS], [SEP], [PAD], [UNK]）的添加也是在这一步。

数据格式化

- 将文本token序列转换为模型可以接受的张量（Tensors），包括input_ids, attention_mask, token_type_ids等。对于指令微调（Instruction Tuning），还需要将数据构造成特定的prompt格式

文本预处理

Tokenization

- 将原始文本字符串转化为模型可以处理的整数ID序列
- 策略对比
  - 词级 (Word-level): 简单，但词汇表爆炸，且无法处理未登录词 (OOV)
  - 字符级 (Character-level): 无OOV问题，但序列过长，模型学习效率低
  - 子词级 (Subword-level): 黄金标准
    - 算法： BPE, WordPiece, SentencePiece
    - 原理： 通过数据驱动的方式，将文本切分为最优的子词单元组合。高频词是单个token，低频词和未见词由多个子词token组合表示
    - 优势： 在词汇表大小、序列长度和OOV处理之间取得了最佳平衡

读取长序列数据

- 整个语料库（如一本书）太大，无法一次性载入内存
- 解决方案： 将长序列切分为固定长度的小批次 (mini-batches)
  - 随机采样 (Random Sampling)
    - 每次从语料库中随机抽取一段序列
    - 适用场景： 无状态模型（如前馈神经LM），批次间无需传递信息
  - 顺序分区 (Sequential Partitioning)
    - 将整个语料库按顺序切分。后一个批次的数据在原文中紧接着前一个批次
    - 适用场景： 有状态模型（如RNN），需要在批次间传递“记忆”（隐状态）
    - 这是训练高性能RNN语言模型的标准做法