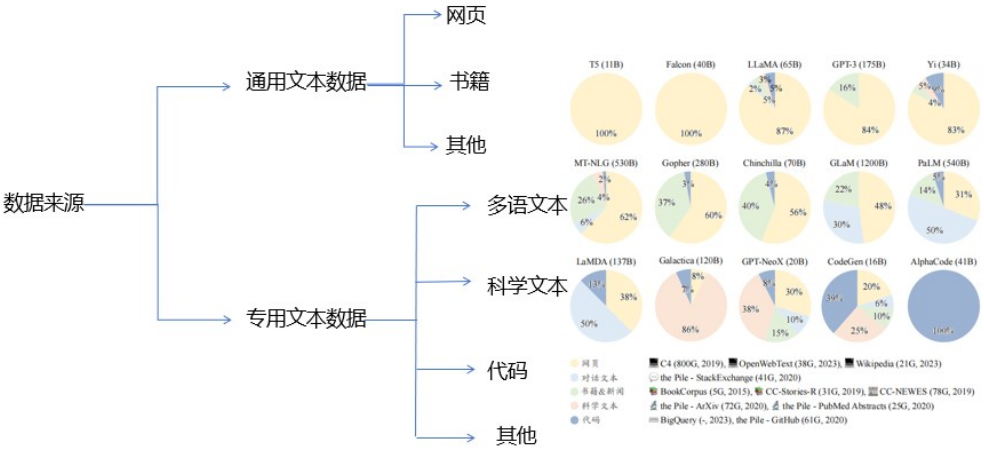
数据准备

刘任强

Wuhan University

March 16, 2025

目录


数据来源
通用文本数据
网页
书籍
其他
专用文本数据
多语文本
科学文本
代码
其他
T5 (11B)
Falcon (40B)
LLaMA (65B)
GPT-3 (175B)
Yi (34B)
MT-NLG (530B)
Gopher (280B)
Chinchilla (70B)
GLaM (1200B)
PaLM (540B)
LaMDA (137B)
Galactica (120B)
GPT-NeoX (20B)
CodeGen (16B)
AlphaCode (41B)
网页
对话文本
书籍&新闻
科学文本
代码
C4 (800G, 2019), OpenWebText (38G, 2023), Wikipedia (21G, 2023)
the Pile - StackExchange (41G, 2020)
BookCorpus (5G, 2015), CC-Stories-R (31G, 2019), CC-NEWES (78G, 2019)
the Pile - ArXiv (72G, 2020), the Pile - PubMed Abstracts (25G, 2020)
BigQuery (-, 2023), the Pile - GitHub (61G, 2020)

目录

两种数据清洗的方法：

- 基于启发式规则的方法
  - 基于语种的过滤
  - 基于简单统计指标的过滤
  - 基于关键词过滤
  - 基于困惑度（Perlexity）过滤
- 基于分类器的方法实现
  - 轻量级模型（如 FastText 等）
  - 可微调的预训练语言模型（如 BERT、BART 或者 LLaMA 等）
  - 闭源大语言模型 API（如 GPT-4、Claude 3）

1. 数据收集与预处理
   数据收集：从各种来源 (如网页，数据库等) 收集大量的原始数据。确保数据的多样性和代表性，以覆盖不同的应用场景和用户需求。
   数据预处理：对数据进行清洗，去除噪声、错误和重复信息。对数据进行格式化和标准化处理，确保数据的一致性和可比性。
2. 质量评估指标确定
   明确质量标准：根据应用场景和模型需求，确定数据质量的评估指标，如准确性、完整性、一致性、相关性和时效性。
   选择评估方法：采用人工评估、自动化评估或两者结合的方式对数据进行质量评估。自动化评估可以基于规则、模型或统计方法来实现。

3. 质量过滤算法与模型应用
   - 基于规则的过滤
   - 基于模型的过滤：训练文本分类器或其他机器学习模型来判数据质量。使用训练好的模型对大量数据进行快速、准确的过滤。常用的模型包括轻量级模型（如 FastText）、可微调的预训练语言模型（如 BERT、LLaMA3）等。
   - 集成过滤方法：结合多种过滤方法（如规则过滤和模型过滤）以达到更好的过滤效果。

   采用多层次、多阶段的过滤策略，逐步剔除低质量数据。
4. 过滤效果评估与优化
   - 评估过滤效果：对过滤后的数据进行质量评估，以验证过滤方法的有效性。
     采用合适的评估指标（如准确率、召回率、F1 分数等）来衡量过滤效果。
   - 优化过滤方法：根据评估结果对过滤方法进行调整和优化。
5. 改进规则设计、模型训练或集成策略，以提高过滤效果。
6. 整合结果

   汇总和分析质量报告：根据分类和处理结果，生成质量报告，包含输出的统计信息、质量指标分布等。
   优化模型的依据：基于质量报告调整模型参数或训练数据，持续改善模型输出质量。

困惑度（Perlexity）：

是衡量语言模型预测能力的一个重要指标，他反映了模型对预测数据的预测好坏程度。困惑度越低，表示模型在预测下一个词时的不确定性越小，模型的性能越好。但在实际应用中，单一使用困惑度效果不佳，所以需要与其他评估指标结合使用，以获得更准确的结果。

其他评估指标：

- BLEU：用于机器翻译任务的评估指标，通过比较机器翻译输出与一组参考翻译之间的 n-gram 重叠程度来评估翻译质量。BLEU 分数越高，表示翻译质量越接近人类翻译。
- ROUGE：这个指标主要用于评估自动摘要的质量，通过计算摘要中与参考摘要共有的 n-gram 数量来评估摘要的准确性和完整性。最常用的是 ROUGE-N 和 ROUGE-L。
- EM：这是一个简单的评估指标，用于检查模型生成的输出是否与参考答案完全匹配。在某些任务中，如问答系统，EM 可以作为一个直接的指标来衡量模型性能

给定一个语言模型和一个测试序列 $w = w_1, w_2, \ldots, w_N$，困惑度 PP 的定义如下：

$$PP(w) = P(w)^{-\frac{1}{N}} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, \ldots, w_{i-1})\right)$$

其中

- $P(w)$ 是生成序列 w 的概率。
- N 是序列中的单词数。
- $P(w_i|w_1, w_2, \ldots, w_{i-1})$ 是给定之前的单词序列 $w_1, w_2, \ldots, w_{i-1}$ 时，模型对当前单词 w_i 的条件概率。

假设我们有一个语言模型，我们想计算句子"I love natural language processing" 的困惑度：

1. 对于句子中的每个单词，使用模型计算条件概率：
   - $P(\text{I})$
   - $P(\text{love}|\text{I})$
   - $P(\text{natural}|\text{I, love})$
   - $P(\text{language}|\text{I, love, natural})$
   - $P(\text{processing}|\text{I, love, natural, language})$
2. 计算对数概率：

$$\text{log\_prob} = \log P(I) + \log P(love|I) + \log P(natural|I, love) + \log P(language|I, love, natural)$$

$$+ \log P(processing|I, love, natural, language)$$

3. 计算总的单词数 $N = 5$。
4. 计算困惑度：

$$PP(w) = \exp\left(-\frac{1}{5} \cdot \text{log\_prob}\right)$$

BLEU 是用于评估机器翻译质量的指标，它基于 **n-gram 精确匹配 ** 计算候选翻译与参考翻译之间的相似度。其计算步骤如下：

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

其中：

- p_n 表示 **n-gram 的精确匹配率**：

$$p_n = \frac{\sum_{\text{ngram} \in C} \min(\text{count}_{\text{cand}}(\text{ngram}), \text{count}_{\text{ref}}(\text{ngram}))}{\sum_{\text{ngram} \in C} \text{count}_{\text{cand}}(\text{ngram})}$$

其中 $\text{count}_{\text{cand}}$ 和 $\text{count}_{\text{ref}}$ 分别表示候选文本和参考文本中的 n-gram 计数。

- w_n 为权重，通常 $w_n = \frac{1}{N}$。
- BP (Brevity Penalty) 为长度惩罚项：

$$\text{BP} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases}$$

其中 c 为候选翻译的长度，r 为参考翻译的最接近长度。

ROUGE 主要用于评估文本摘要任务，常见的变体包括 **ROUGE-N（n-gram 召回率）**、**ROUGE-L（最长公共子序列）**等。

ROUGE-N：

$$\text{ROUGE-N} = \frac{\sum_{\text{ngram} \in R} \min(\text{count}_{\text{cand}}(\text{ngram}), \text{count}_{\text{ref}}(\text{ngram}))}{\sum_{\text{ngram} \in R} \text{count}_{\text{ref}}(\text{ngram})}$$

其中：

- $\text{count}_{\text{cand}}$ 和 $\text{count}_{\text{ref}}$ 分别表示候选摘要和参考摘要中的 n-gram 计数。

ROUGE-L：

$$\text{ROUGE-L} = \frac{\text{LCS}(C, R)}{|R|}$$

其中：

- $\text{LCS}(C, R)$ 表示候选摘要 C 和参考摘要 R 之间的最长公共子序列长度。
- $|R|$ 为参考摘要的长度。

结合 BLEU 分数和困惑度，建立 NLP 文本过滤系统

- **指标设置:**
  - 设定 BLEU 分数阈值: > 0.5
  - 设定困惑度阈值: < 100
- **质量评估:**
  - 计算 BLEU 和困惑度
  - 仅保留 BLEU 分数高且困惑度低的文本

ROUGE 用于摘要质量评估

- **F1 分数:** 结合精确率和召回率，平衡评估文本相似度
- **精确率 vs 召回率:**
  - 高精确率：生成文本包含正确内容但可能有遗漏
  - 高召回率：覆盖更多参考内容但可能冗余
- **基线比较:** 通过与基线模型对比，评估改进效果

目录

敏感内容过滤

- 过滤有毒内容
- 过滤隐私内容

类似于基于启发式规则，如关键字识别来检测和删除私人信息。

数据去重

- 计算粒度
- 用于去重的匹配方法
  - 精确匹配算法（即每个字符完全相同）
  - 近似匹配算法（基于某种相似性度量），可采用局部敏感哈希（LSH），如最小哈希（MinHash）

最小哈希（MinHash）是一种用于估算两个集合相似度的技术，主要用于大规模数据去重、文档相似度计算以及局部敏感哈希（LSH）等领域。其核心思想是通过哈希函数将集合的元素映射为哈希值，并选择最小的哈希值作为集合的签名（Signature）。通过比较两个集合的最小哈希值重合程度，可以高效地估算它们的 Jaccard 相似度。

设有两个集合 A 和 B，目标是估算它们的 Jaccard 相似度：

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

MinHash 通过哈希变换将集合压缩为固定长度的签名，并通过比较签名匹配程度来估算 Jaccard 相似度。它适用于大规模数据集的相似性计算，计算成本较低，能够显著提高相似性搜索的效率。

MinHash 计算过程如下：

1. 选取 k 个不同的哈希函数 $\{h_1, h_2, ..., h_k\}$，这些哈希函数能将集合中的元素映射到一个较大的整数范围。
2. 对于每个集合 S（如 A 或 B），计算每个哈希函数 h_i 对 S 中所有元素的哈希值：

$$h_i(S) = \min\{h_i(x) \mid x \in S\}$$

这意味着，$h_i(S)$ 记录集合 S 中元素在哈希函数 h_i 作用下的最小哈希值。

3. 生成 A 和 B 的 MinHash 签名 $\text{Sig}(A)$ 和 $\text{Sig}(B)$，其维度等于哈希函数的个数 k：

$$\text{Sig}(A) = [h_1(A), h_2(A), ..., h_k(A)]$$

$$\text{Sig}(B) = [h_1(B), h_2(B), ..., h_k(B)]$$

4. 计算 A 和 B 之间 MinHash 签名的匹配比例，作为 Jaccard 相似度的估计值：

$$\hat{J}(A, B) = \frac{\sum_{i=1}^{k} \mathbb{1}[h_i(A) = h_i(B)]}{k}$$

其中，$\mathbb{1}[\cdot]$ 是指示函数，当 $h_i(A) = h_i(B)$ 时取 1，否则取 0。

词元化，或译为分词（Tokenization）旨在将原始文本分割成模型可识别和建模的词元序列，作为大预言模型的输入数据。

在传统的自然语言处理研究主要使用基于词汇的分词方法，但是基于词汇的分词在某些语言 (如中文分词) 中可能对于相同的输入产生不同的分词结果，导致生成包含海量低频词的词表，还可能存在未登陆词。

因此，一些语言模型开始采用字符作为最小单元来分词，目前，子词分词器广泛应用于基于 Transformer 的语言模型中，如 BPE 分词，WordPiece 分词，Unigram 分词。

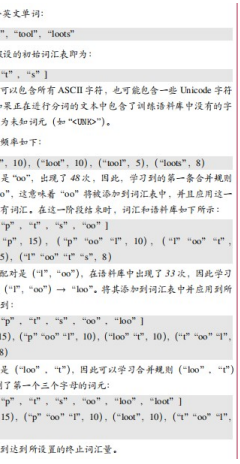
需要关注的因素：

- 分词器须具备无损重构的特性
- 分词器应具有高压缩率

$$\text{压缩率} = \frac{\text{UTF-8 字节数}}{\text{词元数}}$$

BPE 算法

- 统计频率：统计文本中所有相邻字符对的出现频率。
- 合并最频繁的字符对：找到最频繁的字符对并将其合并为新的单个字符。
- 重复上述步骤：反复执行步骤 1 和步骤 2，直到达到预定的词汇表大小或没有更多的字符对可以合并为止。



假设语料中包含了五个英文单词：

"loop"，"pool"，"loot"，"tool"，"loots"

在这种情况下，*BPE* 假设的初始词汇表即为：

["l" , "o" , "p" , "t" , "s"]

在实践中，基础词汇表可以包含所有 ASCII 字符，也可能包含一些 Unicode 字符（比如中文的汉字）。如果正在进行分词的文本中包含了训练语料库中没有的字符，则该字符将被转换为未知词元（如 "<UNK>"）。

假设单词在语料库中的频率如下：

("loop", 15)，("pool", 10)，("loot", 10)，("tool", 5)，("loots", 8)

其中，出现频率最高的是 "oo"，出现了 *48* 次，因此，学习到的第一条合并规则是（"o" , "o"）→ "oo"，这意味着 "oo" 将被添加到词汇表中，并且应用这一合并规则到语料库的所有词汇。在这一阶段结束时，词汇和语料库如下所示：

词汇：["l" , "o" , "p" , "t" , "s" , "oo"]

语料库：("l" "oo" "p" , 15)，("p" "oo" "l" , 10)，("l" "oo" "t" , 10)，("t" "oo" "l", 5)，("l" "oo" "t" "s", 8)

此时，出现频率最高的配对是 {"l", "oo")，在语料库中出现了 *33* 次，因此学习到的第二条合并规则是 {"l", "oo") → "loo"。将其添加到词汇表中并应用到所有现有的单词。可以得到：

词汇：["l" , "o" , "p" , "t" , "s" , "oo" , "loo"]

语料库：("loo" "p", 15)，("p" "oo" "l", 10)，("loo" "t", 10)，("t" "oo" "l", 5)，("loo" "t" "s", 8)

现在，最常出现的词对是（"loo" , "t"），因此可以学习合并规则（"loo" , "t"）→ "loot"，这样就得到了第一个三个字母的词元：

词汇：["l" , "o" , "p" , "t" , "s" , "oo" , "loo" , "loot"]

语料库：("loo" "p", 15)，("p" "oo" "l", 10)，("loot", 10)，("t" "oo" "l", 5)，("loot" "s", 8)

可以重复上述过程，直到达到所设置的终止词汇量。

WordPiece 分词和 BPE 分词想法类似，都是通过迭代合并连续的词元，但是合并的选择标准略有不同。

在合并前，WordPiece 分词算法首先训练一个语言模型，并用这个语言模型对所有可能的词元队进行评分，然后，在每次合并时，它都会选择使得训练数据的似然性增加最多的词元对。计算公式：

$$\text{score} = \frac{\text{frequency of pair}}{\text{frequency of first element} \times \text{frequency of second element}}$$

1. **计算初始词表**：基于训练数据，生成初始的词表，通常由单个字符或已有词汇组成。
2. **拆分训练语料**：将训练数据中的文本拆分成最小单位，例如单个字符或基本子词。
3. **计算合并分数**：根据统计信息计算每对相邻子词的合并分数（如基于出现频率或统计测度）。
4. **选择并合并子词对**：选取得分最高的一对子词进行合并，形成新的子词。
5. **更新词表**：将新的子词加入词表，并更新统计信息。
6. **检查是否达到预定阈值**：
   - 若未达到阈值，则返回步骤 3，继续合并子词对。
   - 若达到阈值，则进入下一步。
7. **进行分词**：使用最终得到的子词表对文本进行分词处理。

Unigram 分词从预料库的一组足够大的字符串或词元初始集合开始，迭代地删除其中的词元，直到达到预期的词表大小。

算法流程：

1. **初始化**：建立初始词表，包含所有可能的子词及其概率。
2. **计算句子概率**：对所有可能的分词方式计算概率：

$$P(W) = \prod_{i=1}^{n} P(w_i)$$

3. **选择最优分词**：找到使 $P(W)$ 最大的子词序列。
4. **迭代优化**：删除低概率子词，重新计算概率并更新词表。

输入句子：”自然语言处理”
候选分词方式：

- (自然, 语言, 处理) $P = P(\text{自然})P(\text{语言})P(\text{处理})$
- (自然, 语言处理) $P = P(\text{自然})P(\text{语言处理})$
- (自然语言, 处理) $P = P(\text{自然语言})P(\text{处理})$

选择概率最大的分词方式作为最终结果。

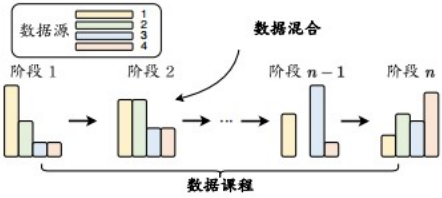
在实际应用中，考虑到对于较长文本或复杂的分词任务，可能需使用动态规划算法 (如维特比算法) 来找到最佳分词路径。

1. 构建模型：首先构造隐马尔可夫模型（HMM），该模型包括初始状态概率，状态转移概率和发射概率。在中文分词任务中，将状态定义为：
   - 词的开始（B, Begin）
   - 词的中间（M, Middle）
   - 词的结束（E, End）
   - 单字词（S, Single）

   观察值为句子中的每个字符。
2. 初始化：对于句子中的第一个字符，计算其处于各个状态（B、M、E、S）的概率，并记录到达该状态的最优路径（即前一个状态）。
3. 递归计算：对于句子中的每个后续字符：
   - 根据前一个字符的状态和当前字符的观察值，计算当前字符处于各个状态的概率。
   - 更新到达当前字符各个状态的最优路径和概率。
4. 终止：在句子的最后一个字符处，找到概率最大的状态（通常是 E 或 S，表示词的结束或单字词）。
5. 回溯：从终止状态开始，根据记录的最优路径回溯，得到整个句子的分词结果。

数据调度：主要关注两个方面

- 各个数据元的混合比例
- 各数据源用于训练的顺序



数据混合：在预训练期间，根据混合比例从不同数据源中采样数据，数据源的权重越大，从中选择的数据越多。常见的几种数据混合策略：

- 增加数据源的多样性
- 优化数据混合
- 优化特定能力