第 3 章　训练数据的采集与清洗

数据是大模型训练的基石，其质量直接决定了模型的性能上限。在当前技术条件下，数据工程的重要性甚至超过了模型架构的创新。本章将深入探讨训练数据的获取、评估与处理流程，为构建高质量训练数据集提供系统性的工程指导。

3.1　数据源评估与采集策略

【学习目标】

（1）掌握开源数据集的多维度质量评估框架；

（2）理解网络爬取的法律合规要求与技术实现；

（3）了解专有数据的规范化采集流程；

（4）建立数据采集的系统化思维。

数据源的选择与评估是构建高质量训练语料的第一步，需要综合考虑数据规模、质量、多样性、合法性等多个维度，建立科学的评估体系与规范的采集流程。

3.1.1　开源数据集的质量评估框架

开源数据集为大模型训练提供了重要的基础资源，但其质量参差不齐。近期研究表明，高质量数据集应具备 13 个关键特征，包括准确性、多样性、时效性、完整性、一致性、可扩展性等。建立系统的评估框架能够帮助筛选出真正有价值的训练数据。

数据质量评估需要从多个维度进行综合考量，涵盖内容质量、技术指标、领域覆盖等方面，如图 3-1 所示。

数据质量评估体系包含六个核心维度：①内容质量关注数据的准确性、原创性和专业性；②技术指标衡量去重率、噪声比和格式规范性；③领域覆盖考查学科广度、语言多样性和场景丰富度；④合规性确保版权合法性、隐私保护和正确的价值导向；⑤时效性追踪数据的更新频率；⑥数据规模评估样本数量的充足程度。这些维度相互关联，共同构成了完整的评估框架。

在实际评估过程中，需要为每个维度设置具体的量化指标。例如，去重率可通过 MinHash 算法计算文档间的相似度，设定阈值进行判断；噪声比可通过统计特殊字符、乱码、重复模式等异常内容的比例来衡量；领域覆盖度可通过主题建模技术分析数据的主题分布。

以 Qwen 系列模型的数据工程实践为例，在构建万亿 token 级别的训练语料时，建立了严格的多层次质量评估标准。在内容质量维度，通过语言模型困惑度过滤，仅保留困惑度低

图 3-1　数据质量评估维度

于阈值的高质量文本；在技术指标维度，使用 MinHash 算法进行全局去重，确保去重率达到 95%以上；在领域覆盖维度，构建了涵盖科学、技术、人文、社会等 12 个一级学科的知识图谱，通过主题建模确保各领域数据的均衡分布。这种系统化的评估框架使得 Qwen 能够从海量原始数据中精选出真正有价值的训练语料。

根据最新的实践经验，主流开源数据集在这些维度上表现各异，需要综合评估后选择合适的数据源，如表 3-1 所示。

表 3-1　主流开源数据集质量评分表

数据集名称	规模(TB)	内容质量	技术指标	领域覆盖	合规性	综合评分
Common Crawl	100+	7.2	6.8	9.5	8.0	7.8
The Pile	0.8	8.5	8.8	8.7	7.5	8.4
RedPajama	1.2	8.2	7.9	8.3	8.5	8.2
SlimPajama	0.6	8.6	9.2	8.0	8.5	8.6
CCI 2.0(中文)	0.5	8.8	8.5	7.8	9.2	8.6
WuDaoCorpora	3.0	7.5	7.2	8.9	7.8	7.8
The Stack v2	67.5	8.3	8.0	7.5	8.8	8.2

注：评分采用 10 分制，综合评分为各维度加权平均，其中内容质量的权重为 30%，技术指标的权重为 25%，领域覆盖的权重为 20%，合规性的权重为 25%。

表 3-1 显示了主流开源数据集的质量评估结果。SlimPajama 通过深度去重处理获得了最高的技术指标得分(9.2 分)，其采用精确去重(13-Grams)和模糊去重(MinHashLSH)相结合的方法，成功去除了 RedPajama 数据集中 49.6%的重复内容。The Pile 在内容质量方面表现优异(8.5 分)，得益于其严格的数据筛选标准和多源整合策略。CCI 2.0 作为专门针对中文环境构建的数据集，在合规性方面得分最高(9.2 分)，充分考虑了国内法律法规要求和价值导向。

从评估结果可以看出，不同数据集各有优势：规模庞大的 Common Crawl 提供了极其

丰富的领域覆盖，但内容质量相对较低；经过精心筛选的高质量数据集，如 The Pile 和 SlimPajama，虽然规模较小，但在训练效果上往往更胜一筹。实践中应根据具体需求选择合适的数据集，或将多个数据集组合使用。

3.1.2 网络爬取的法律边界与技术实现

网络爬取是获取训练数据的重要途径，但必须在法律框架内规范进行。随着数据保护法规的不断完善，合规性已成为数据采集不可逾越的红线。

Robots 协议作为互联网行业的技术规范，虽然不具有法律强制力，但已被司法实践广泛认可为行业内应当遵守的商业道德。违反 Robots 协议可能构成不正当竞争，需要承担相应的民事责任。除了遵守 Robots 协议外，爬虫还需要注意不能突破网站设置的反爬虫技术措施，避免对目标网站造成过重负担。

在技术实现上，分布式爬虫架构能够有效提升数据采集效率，同时通过合理的并发控制避免对目标网站造成压力。

如下为分布式爬虫配置示例。

```
yaml
crawler_config:
  # 基础设置
  user_agent: "ResearchBot/1.0 ( + http://example.com/bot)"
  timeout: 30
  retry_times: 3

  # 并发控制
  concurrency:
    max_workers: 10
    rate_limit: 2                # 每秒请求数
    delay_range: [1, 3]          # 请求间隔(秒)

  # Robots 协议遵守
  robots_policy:
    obey: true
    cache_size: 1000
    cache_expiry: 86400          # 24 小时

  # 分布式设置
  distributed:
    mode: "redis"
    redis_url: "redis://localhost:6379"
    queue_key: "crawler:queue"

  # 数据过滤
  filters:
    min_text_length: 100
    max_text_length: 50000
    languages: ["zh", "en"]
```

此配置文件展示了规范化爬虫的关键参数设置。其中，并发控制通过限制最大工作线程数和请求速率来避免对目标服务器造成压力；Robots 协议遵守设置确保爬虫行为符合网站规范；分布式设置通过 Redis 队列实现任务调度，支持多节点协同工作；数据过滤规则

在采集阶段就进行初步质量控制，提高后续处理效率。

需要特别注意的是，爬虫的使用目的不能是实质性替代被爬取方提供的产品或服务。在采集数据时，应当充分评估数据的使用方式是否会侵犯原网站的合法权益。对于包含个人信息的数据，还需要遵守《中华人民共和国个人信息保护法》等相关法规，确保数据处理的合法性。

3.1.3　专有数据的获取与处理流程

专有数据通常具有更高的质量和针对性，是提升模型特定领域能力的重要资源。企业内部文档、行业数据库、合作伙伴提供的数据等都属于专有数据范畴。这类数据的获取需要建立规范的流程，确保数据来源的合法性和处理的规范性。

专有数据采集 Pipeline 需要覆盖从数据源接入到质量验证的完整流程，如图 3-2 所示。

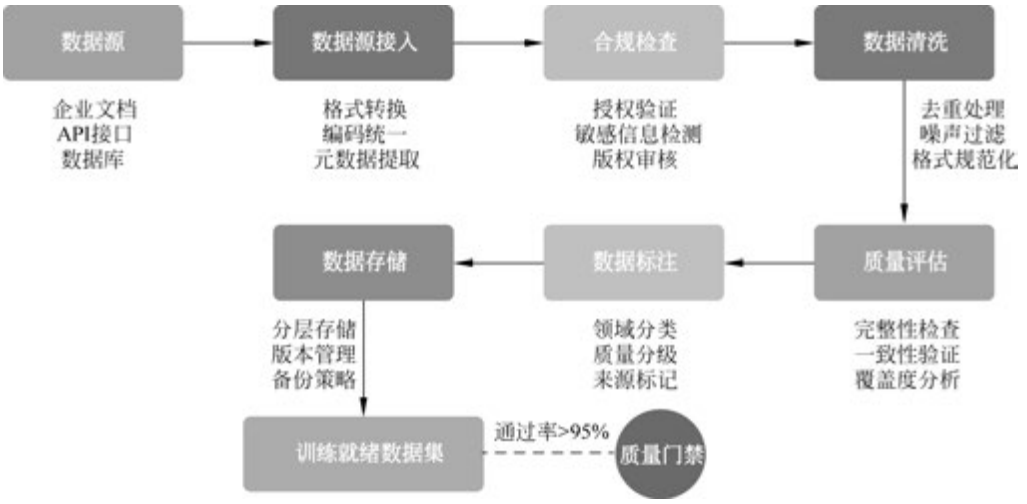

图 3-2　数据采集 Pipeline 流程

数据采集 Pipeline 遵循严格的流程控制，确保每个环节的质量和合规性。数据源接入阶段需要进行格式转换和编码统一，将异构数据转换为统一的处理格式。合规检查是关键控制点，通过授权验证、敏感信息检测和版权审核确保数据使用的合法性。数据清洗环节采用多种技术手段去除重复、过滤噪声并规范化格式。质量评估阶段进行完整性检查、一致性验证和覆盖度分析。只有通过质量门禁（通过率大于 95%）的数据才能进入最终的训练数据集。

在实际操作中，专有数据的处理还需要建立完善的数据治理体系。这包括明确的数据分类分级制度，将数据按照敏感程度和重要性进行分类管理；建立数据使用审批流程，确保数据访问和使用符合内部管理规定；实施数据脱敏处理，对包含个人信息或商业机密的内容进行技术处理；建立数据生命周期管理机制，明确数据的采集、存储、使用、归档和销毁等各个环节的管理要求。

专有数据的质量控制尤为重要。由于这类数据通常规模较小但价值密度高，需要投入更多资源进行精细化处理。可以采用人工审核与自动化检测相结合的方式，确保数据的准确性和相关性。同时，需要建立数据更新机制，定期对数据进行更新和补充，保持数据的时

效性和完整性。

通过建立规范的数据采集流程，能够确保专有数据的质量和合规性，为模型训练提供高价值的数据资源。这种系统化的数据工程方法不仅提升了数据质量，也为后续的模型训练奠定了坚实基础。

3.2 数据清洗 Pipeline 设计

【学习目标】

(1) 理解文本规范化的技术原理与实现方法；

(2) 掌握基于统计的噪声识别算法；

(3) 学习敏感信息检测与脱敏技术；

(4) 建立完整的数据清洗流程体系。

数据清洗是将原始文本转换为高质量训练语料的关键环节，通过系统的处理流程去除噪声、统一格式、保护隐私，为模型训练奠定坚实的数据基础。

3.2.1 文本规范化与编码统一

文本规范化是数据预处理的基础步骤，其目标是将异构的文本数据转换为统一、标准的格式。在大模型训练场景下，规范化处理直接影响词表构建、分词效果和模型的泛化能力。近期研究表明，基于大语言模型的规范化方法相比于传统的基于规则的方法，错误率降低了约 40%，显示出显著的性能提升。

文本规范化涉及多个层面的处理，包括字符级别的编码统一、词汇级别的标准化以及格式级别的规范化。在处理大规模多源数据时，需要设计高效且鲁棒(Robust)的处理算法。

以下是文本规范化处理算法的伪代码示例。

```
算法: TextNormalization(text)
输入: 原始文本 text
输出: 规范化文本 normalized_text

// Unicode 规范化
text ← ConvertToUTF8(text)
text ← ApplyNFKC(text)                    // 规范化组合字符

// 字符级处理
for each char in text:
    if IsFullWidth(char):
        char ← ToHalfWidth(char)
    if IsControlChar(char):
       char ← RemoveOrReplace(char)

// 大小写规范化(保留首字母大写)
words ← Tokenize(text)
for each word in words:
    if not IsProperNoun(word) and not IsAcronym(word):
        word ← word.lower()

// 数字和日期规范化
```

```
    text ← NormalizeNumbers(text)              // 统一数字格式
    text ← NormalizeDates(text)                // 统一日期格式
    text ← NormalizeURLs(text)                 // 规范化 URL

    // 空白字符处理
    text ← CollapseWhitespace(text)
    text ← TrimSpaces(text)

    return text
```

该算法首先进行 Unicode 规范化，采用 NFKC(Normalization Form KC)形式统一字符表示，解决同一字符的多种编码问题。字符级处理阶段将全角字符转换为半角，移除或替换控制字符。大小写规范化保留专有名词和缩略词的原始形式，其他词汇统一转换为小写。数字、日期和 URL 的规范化确保格式一致性，便于后续的模式识别和语义理解。最后进行空白字符的规范化处理，去除多余空格并统一换行符。

在实际工程实践中，文本规范化需要根据语言特点进行调整。对于中文数据，需要处理繁简转换、异体字统一等问题；对于多语言混合文本，需要识别语言边界并采用相应的规范化策略。同时，规范化过程应保持可逆性或半可逆性，避免信息损失。

3.2.2　噪声识别与过滤算法

噪声数据的存在会严重影响模型训练效果，导致模型学习到错误的模式。基于统计方法的噪声检测能够自动识别异常文本，其中困惑度(Perplexity)是最常用的指标之一。困惑度衡量了语言模型对文本的“困惑”程度，异常高的困惑度通常意味着文本质量存在问题。

噪声检测的核心在于建立正常文本的统计模型，通过异常检测算法识别偏离正常分布的样本。图 3-3 展示了基于困惑度分布的噪声检测阈值选择方法。

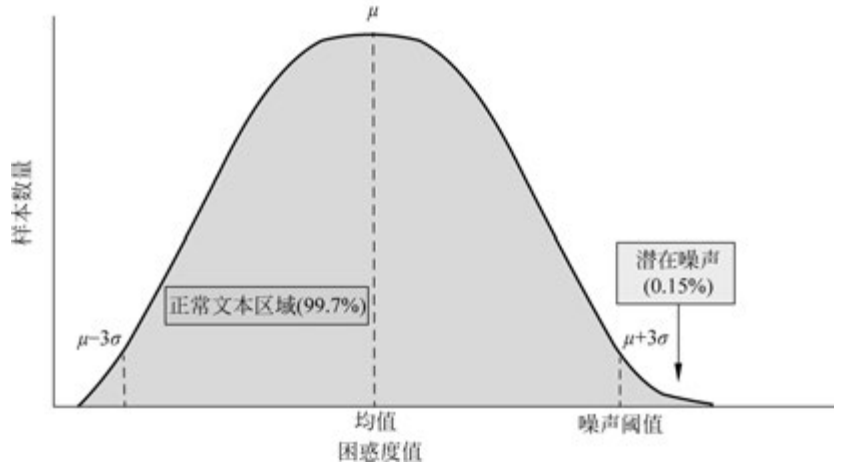

图 3-3　噪声检测阈值选择

如图 3-3 所示，正常文本的困惑度呈现近似正态分布，大部分样本集中在均值附近。通过统计分析确定均值 μ 和标准差 σ 后，可以设定异常检测阈值。实践中常用 3σ 准则作为判定标准，即将困惑度超过 $\mu+3\sigma$ 的文本标记为潜在噪声。

困惑度异常检测：

$$\mathrm{PPL} > \mu + 3\sigma$$

异常检测的判定条件为：当文本的困惑度 PPL 大于均值 μ 加上 3 倍标准差 σ 时，将该文本标记为异常。这种方法在统计学上对应于 99.7%的置信区间，能够有效识别显著偏离正常分布的异常样本。在实际应用中，可以根据数据质量要求调整系数，例如使用 2.5σ 或 3.5σ 作为阈值。

困惑度异常检测的数学基础是统计假设检验。对于给定文本序列，其困惑度计算公式为：

$$\mathrm{PPL}(W) = P(w_1, w_2, \cdots, w_n)^{(-1/n)} = \exp\left(-1/n \sum_{i=1}^{n} \log P(w_i \mid w_1, \cdots, w_{i-1})\right)$$

其中，W 表示文本序列，n 为序列长度，$P(w_i \mid w_1, \cdots, w_{i-1})$ 表示在给定前文条件下第 i 个词的条件概率。

在工程实践中，通常使用预训练的语言模型（如 GPT-2）计算困惑度。对于检测出的异常文本，需要进一步分析其异常原因。高困惑度可能源于多种因素：格式错误（如 HTML 标签未清理）、语言混杂（多语言混合）、内容质量低（机器翻译错误）或领域特殊性（专业术语密集）。因此，异常检测阈值的设置需要根据具体任务和数据特点进行调整，一般建议在标记数据上进行验证，选择召回率和准确率的最佳平衡点。

3.2.3 敏感信息检测与脱敏处理

保护用户隐私和数据安全是大模型训练的基本要求。敏感信息检测需要识别并处理个人身份信息（PII）、商业机密、医疗记录等各类敏感数据。不同类型的敏感信息需要采用不同的检测策略和脱敏方法，如表 3-2 所示。

表 3-2 敏感信息类别与处理策略表

信息类别	识别方法	脱敏策略	示例
个人身份信息 （姓名、身份证号）	正则表达式匹配＋NER 模型	替换为标记符 （[NAME]，[ID]）	张三→[NAME] 110102 *** 543→[ID]
联系方式 （电话、邮箱）	模式匹配，正则表达式	部分掩码，保留格式	138 *** 5678 a ** @example. com
地址信息 （详细地址）	地址解析器＋地名词典	泛化处理，保留省市	北京市朝阳区 xxx 路 →北京市[ADDRESS]
金融信息 （银行卡、账号）	Luhn 算法验证＋格式检测	完全替换或加密存储	6222 **** 1234 →[CARD]
医疗信息 （病历、诊断）	医疗实体识别 （BioBERT 模型）	类别替换，保留语义	糖尿病Ⅱ型 →[DIAGNOSIS]
商业机密 （内部代码）	关键词匹配＋上下文分析	删除或替换通用描述	项目代号 Alpha →[PROJECT]
生物特征 （指纹、面部）	特征检测算法 （哈希匹配）	不可逆变换 （哈希处理）	生物数据 →[BIOMETRIC]
认证凭据 （密码、密钥）	熵值检测＋格式分析	完全移除，零知识替换	API_KEY_xxx →[CREDENTIAL]

敏感信息的识别通常结合规则匹配、正则表达式、命名实体识别(NER)等多种技术。近年来,基于深度学习的方法在敏感信息检测方面取得了显著进展,能够识别上下文相关的隐式敏感信息。

表 3-2 系统化地展示了八类主要敏感信息的识别方法与脱敏策略。个人身份信息采用正则表达式结合 NER 模型进行识别,通过标记符替换实现脱敏;联系方式使用模式匹配技术,采用部分掩码保留格式信息;地址信息通过地址解析器识别,泛化处理后仅保留省市级信息;金融信息利用 Luhn 算法验证,采用完全替换或加密存储保护;医疗信息使用专业的 BioBERT 模型识别,通过类别替换保留语义信息;商业机密通过关键词匹配和上下文分析识别,替换为通用描述;生物特征使用特征检测算法,进行不可逆哈希变换;认证凭据通过熵值检测识别,采用完全移除策略确保安全。

敏感信息脱敏的实施需要在数据可用性和隐私保护之间寻求平衡。过度脱敏会损失重要信息,影响模型训练效果;脱敏不足则可能泄露隐私信息,带来合规风险。在实际应用中,建议采用分级脱敏策略:对于高敏感度信息(如认证凭据)采用完全移除或强加密;对于中等敏感度信息(如联系方式)采用部分掩码保留结构信息;对于低敏感度信息(如公开的机构名称)可以保留或进行轻度处理。

脱敏处理还需要考虑上下文关联性。单个信息可能不敏感,但多个信息组合可能构成敏感信息。例如,“在某医院工作”和“肿瘤科医生”单独看都不敏感,但结合特定时间地点信息可能识别出具体个人。因此,需要进行关联分析和组合脱敏,确保无法通过推理恢复原始信息。

数据脱敏后的质量验证同样重要。需要建立脱敏效果评估机制,包括脱敏完整性检查(是否遗漏敏感信息)、可逆性测试(是否能恢复原始信息)、语义保持度评估(脱敏后文本是否保持原有语义)等。通过自动化检测结合人工抽检,确保脱敏处理的有效性和一致性。

3.3　去重技术与实现策略

【学习目标】

(1) 理解 MinHash 算法的数学原理与工程优化;

(2) 掌握基于向量的语义去重方法;

(3) 学习大规模数据去重的分布式架构;

(4) 建立高效去重的系统化实现策略。

数据去重是提升训练质量的关键步骤,通过识别和移除重复内容,不仅能减少计算资源消耗,更能显著改善模型的泛化能力和训练稳定性。

3.3.1　MinHash 算法原理与优化

MinHash 算法通过将文档映射为固定长度的签名向量,实现了 Jaccard 相似度的高效近似计算。该算法的核心思想是利用哈希函数的最小值保持集合相似性的概率特性。最新研究表明,在处理万亿级别 token 的训练数据时,MinHash 结合 LSH(局部敏感哈希)能够

将去重时间复杂度从 $O(N^2)$ 降低到接近线性的 $O(N)$。

MinHash 算法的基本原理是通过多个独立的哈希函数对文档的 shingle 集合进行映射，每个哈希函数产生一个最小哈希值，这些最小值组成的向量即为文档的 MinHash 签名。两个文档的 Jaccard 相似度可以通过比较它们的 MinHash 签名的相同位置数量来估计。

Jaccard 相似度估计：

$$J(A,B) \approx |\boldsymbol{h}(A) \cap \boldsymbol{h}(B)|/k$$

其中，$J(A,B)$ 表示集合 A 和 B 的 Jaccard 相似度，$\boldsymbol{h}(A)$ 和 $\boldsymbol{h}(B)$ 分别表示集合 A 和 B 的 MinHash 签名向量，k 是签名向量的维度（哈希函数的数量）。该公式表明，两个集合的 Jaccard 相似度可以通过计算它们的 MinHash 签名中相同元素的比例来近似。

在工程实践中，k 值的选择需要平衡准确性和计算效率。研究表明，对于大规模文本去重，$k=128$ 能够在保持较高准确率的同时控制内存开销；当需要更高精度时，可以将 k 值增加到 256 或 512。对于特定领域的高质量数据集，如学术论文或技术文档，建议使用更大的 k 值（如 450），以降低误判率。签名向量通常使用 uint32 类型存储，避免使用 float32 可能导致的精度问题。

LSH 通过将签名向量划分为多个 band，将相似文档快速聚集到同一 bucket 中，大幅减少需要比较的文档对数量。

以下为 LSH 近似去重算法的伪代码示例。

```
算法：LSH_Deduplication(documents, num_perm, bands, threshold)
输入：文档集合 documents，排列数 num_perm，带数 bands，相似度阈值 threshold
输出：去重后的文档集合

// 生成 MinHash 签名
signatures ← []
for doc in documents:
    shingles ← GenerateShingles(doc, n = 5)        // 5 - gram
    minhash ← MinHash(num_perm)
    for shingle in shingles:
        minhash.update(shingle)
    signatures.append(minhash.signature)

// LSH 分带处理
rows_per_band ← num_perm / bands
buckets ← defaultdict(list)

for idx, sig in enumerate(signatures):
    for band_idx in range(bands):
        start ← band_idx * rows_per_band
        end ← start + rows_per_band
        band_sig ← hash(tuple(sig[start:end]))
        buckets[band_sig].append(idx)

// 识别重复文档
duplicates ← set()
for bucket_docs in buckets.values():
    if len(bucket_docs) > 1:
        for i, j in combinations(bucket_docs, 2):
            if JaccardSimilarity(signatures[i], signatures[j]) > threshold:
                duplicates.add(max(i, j))        // 保留较小索引
```

```
28.
29. // 返回去重结果
30. return [doc for idx, doc in enumerate(documents) if idx not in duplicates]
```

该算法首先为每个文档生成 MinHash 签名，使用 5-gram 作为基本单元能够在捕获语义信息和计算效率之间取得平衡。LSH 阶段将签名划分为多个 band，每个 band 独立进行哈希映射。当两个文档在任意一个 band 中产生相同的哈希值时，它们被视为候选重复对。最后通过精确计算 Jaccard 相似度进行验证，超过阈值的文档对被标记为重复。

最新的优化技术包括使用 Bloom Filter 减少内存占用，在 600GB 数据集上可以减少 87.2%的内存使用；采用 GPU 加速的 MinHash 实现，相比 CPU 版本可以获得 3～5 倍的速度提升；使用增量式处理策略，支持流式数据的实时去重。在 Milvus 2.6 中集成的原生 MinHash LSH 索引，通过优化的云原生架构，能够处理数百亿文档的去重任务。

3.3.2　语义去重的向量化方法

语义去重通过深度学习模型生成的向量表示来识别语义相似的文本，即使它们在字面上完全不同。这种方法对于处理改写、翻译、同义替换等情况特别有效。基于 BERT 等预训练模型的句子嵌入能够捕获文本的深层语义信息，通过聚类算法识别语义重复的内容。

语义去重的核心流程包括嵌入生成、向量聚类和相似度计算三个主要步骤。图 3-4 展示了语义向量聚类的可视化结果。

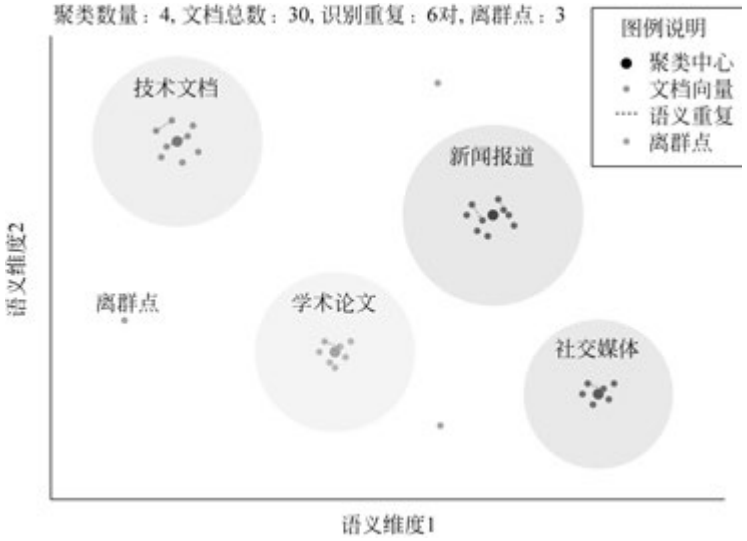

图 3-4　语义向量聚类可视化

从图 3-4 中可以看出，不同主题的文档在语义空间中形成了明显的聚类结构，技术文档、新闻报道、学术论文和社交媒体内容各自聚集成簇。每个聚类内部的点表示语义相似的文档，虚线连接表示识别出的语义重复对。聚类中心（大的圆点）代表该类别的典型语义表示。离群点则表示与任何主要类别都不相似的独特内容。这种可视化有助于理解语义去重的工作原理：通过将文档映射到高维语义空间并进行聚类分析，能够有效识别语义相似但表述不同的重复内容。

语义去重的实现通常采用 Sentence-BERT 等专门优化的模型生成句子嵌入。与原始 BERT 不同，Sentence-BERT 通过孪生网络架构和对比学习训练，能够生成更适合相似度计算的固定维度向量（通常为 768 维）。在聚类阶段，k-means 算法是最常用的选择，聚类数量 k 的设置需要根据数据集规模调整，一般建议每 10 万个文档设置 100～500 个聚类。

在每个聚类内部，通过计算余弦相似度识别语义重复。阈值的选择至关重要：对于一般文本，余弦相似度阈值为 0.8～0.85 能够平衡召回率和准确率；对于技术文档等专业内容，可以降低到 0.75，以捕获更多变体表述；对于新闻等时效性内容，可以提高到 0.9，以避免误删相关但不同的报道。

最新的 SemDeDup 算法引入了多项优化：使用批量余弦相似度计算减少内存占用；采用“hard”和“soft”两种保留策略，前者保留与聚类中心最近的文档，后者基于概率采样保留多样性；支持增量处理，能够处理流式数据。在 LAION 数据集上的实验表明，语义去重能够减少 20%～50%的数据量，同时保持甚至提升模型性能。

3.3.3 大规模去重的分布式实现

在处理 TB 乃至 PB 级别的训练语料时，单机去重方案难以满足性能和扩展性需求。分布式去重架构能够充分利用集群的计算和存储资源，实现高效的大规模数据处理。Qwen 系列模型在数据工程中积累了丰富的去重实践经验，其方法论对构建高质量训练数据集具有重要参考价值。

1. 分布式 MinHash 去重架构

大规模 MinHash 去重的核心挑战在于如何高效地进行签名计算和相似文档检索。典型的分布式实现采用两阶段架构：第一阶段并行计算所有文档的 MinHash 签名，第二阶段通过 LSH（局部敏感哈希）索引进行近似最近邻检索。

以下展示了融入 Qwen 实践经验的大规模去重任务配置。

```yaml
# 大规模去重任务配置(融入 Qwen 经验)
spark:
  app_name: "LargeScaleDedup"
  executor_memory: "32g"
  executor_cores: 8
  num_executors: 256

minhash:
  num_perm: 256                          # 置换函数数量
  num_bands: 32                          # LSH 分段数
  rows_per_band: 8                       # 每段行数
  threshold: 0.8                         # 相似度阈值

processing:
  partition_strategy: "domain_aware"     # 按域名分区
  chunk_size: 100000                     # 每批处理文档数
  enable_exact_dedup: true               # 先精确去重
  hash_algorithm: "xxhash64"             # 精确去重哈希算法
```

该配置体现了 Qwen 团队的核心经验：首先，采用 256 个置换函数生成 MinHash 签名，在计算效率和去重精度之间取得平衡；其次，将签名划分为 32 个 band 进行 LSH 索引，

每个 band 包含 8 行，这一配置使得相似度超过 0.8 的文档对有很高的概率被召回；第三，采用“domain_aware”分区策略，将同一域名的文档分配到相同分区，利用网站内容的局部相似性提高去重效率。

2. 去重策略效果对比

不同的去重策略对最终模型性能有显著影响。表 3-3 总结了多种去重策略在实践中的效果对比。

表 3-3 去重策略效果对比表

去重策略	去重率	下游任务	训练稳定性	计算成本	推荐度
无去重	0%	基准线	易过拟合	最低	不推荐
精确去重	15%～25%	+1.2%	明显改善	低	一般
MinHash(0.9)	25%～35%	+2.1%	良好	中等	推荐
MinHash(0.8)	**35%～50%**	**+3.5%**	**最佳**	中等	最佳选择
MinHash(0.7)	50%～65%	+2.8%	可能欠拟合	较高	一般
语义去重	40%～55%	+4.2%	良好	极高	一般

如表 3-3 所示，去重策略的选择对模型性能有显著影响。Qwen 团队的实践表明，MinHash 相似度阈值设置为 0.8 是一个较优的平衡点：去重率达到 50%，下游任务性能提升约 3.5%，同时保持良好的训练稳定性。

过于严格的去重（阈值 0.7 及以下）虽然能够移除更多近似重复内容，但也可能错误删除包含重要变体信息的文档，导致数据多样性下降和潜在的欠拟合风险。而过于宽松的去重（阈值 0.9 及以上）则可能保留过多冗余内容，影响训练效率和模型泛化能力。

语义去重虽然在效果上表现最优（下游任务提升 4.2%），但其计算成本极高，需要为每个文档生成稠密向量并进行向量相似度计算。在 PB 级数据规模下，语义去重的成本可能是 MinHash 方案的 10～100 倍。因此，实践中通常采用“精确去重→MinHash 近似去重”的两阶段方案，在有限预算内实现最佳的去重效果。

3. 增量去重与数据更新

对于持续更新的训练数据集，增量去重是一个重要的工程需求。Qwen 的实践采用了基于布隆过滤器的增量方案：将已处理数据的 MinHash 签名存储在分布式布隆过滤器中，新数据只需与布隆过滤器进行匹配即可快速判断是否为重复内容。这种方案将增量去重的时间复杂度降至近似 $O(1)$，支持每日 TB 级新数据的高效处理。

3.4 数据质量评估体系

【学习目标】

（1）掌握语言质量的自动化评分方法；

（2）理解知识覆盖度的量化评估框架；

（3）学习毒性与偏见的系统化检测技术；

（4）建立多维度的数据质量评估体系。

数据质量评估贯穿整个数据工程流程，通过建立科学的评估指标和自动化的检测方法，

能够及时发现问题并指导优化方向，确保训练数据的高质量标准。

3.4.1 语言质量的自动化评分

语言质量是训练数据最基础的要求，涵盖语法正确性、流畅度、连贯性等多个维度。最新研究表明，基于大语言模型的评估方法相比于传统的统计指标，在准确性上提升了 40% 以上，能够更好地捕捉语言的细微特征。自动化评分系统通过结合多种评估技术，实现对海量文本的高效质量控制。

语言质量评分采用分层评估策略，从词汇、句子到篇章三个层次进行综合分析。在词汇层面，检测拼写错误、词汇多样性和专业术语使用的准确性；在句子层面，评估语法正确性、句式复杂度和语义清晰度；在篇章层面，分析逻辑连贯性、主题一致性和结构完整性。

图 3-5 所示的质量分布直方图清晰地展示了数据集的整体质量状况。

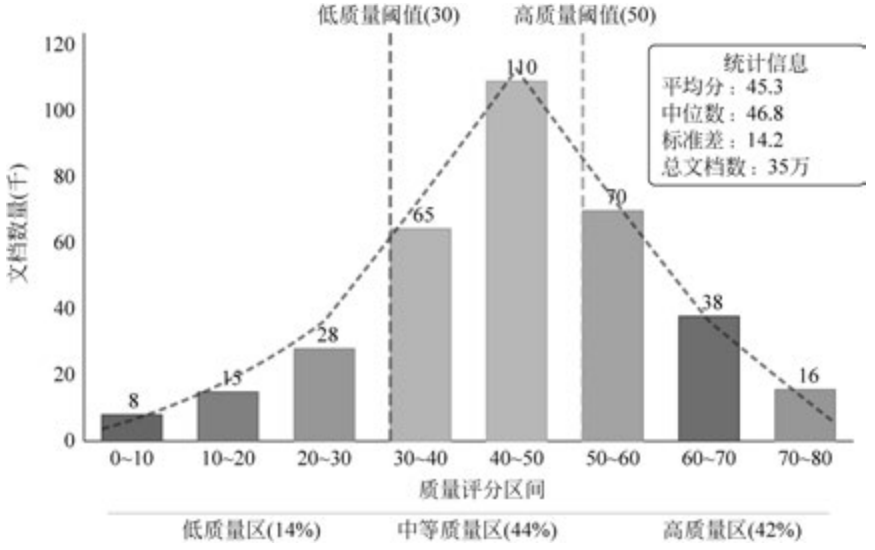

图 3-5 质量分布直方图

从图 3-5 中可以看出，分布呈现近似正态分布特征，主要集中在 40～50 分区间，表明大部分文档质量处于中等偏上水平。低质量文档(得分低于 30)仅占总量的 14%，需要重点清洗或移除；高质量文档(得分超过 50)占 42%，可以直接用于训练；中等质量文档占 44%，需要进一步优化提升。虚线标注的质量阈值帮助快速识别不同质量级别的文档，指导后续的数据处理策略。

自动化评分系统采用多模型融合策略提高准确性。基础层使用 ROUGE、METEOR 等统计指标快速筛选明显的低质量文本；中间层采用 BERT 等预训练模型计算语义相似度和流畅度得分；顶层使用 GPT-4 等大语言模型进行综合评估，通过 G-Eval 框架生成细粒度的质量评分。实验表明，这种分层评估方法相比于单一指标，F1 得分提升了 15～20 个百分点。

在工程实践中，语言质量评分需要根据领域特点进行调整。技术文档更注重术语准确性和逻辑严谨性，权重可以设置为 40%；新闻文本强调时效性和可读性，流畅度权重提高到 35%；学术论文侧重引用规范和论证完整性，结构分析权重达到 30%。通过动态调整评分策略，能够更准确地反映不同类型文本的质量特征。

在大规模数据处理实践中，多层次质量过滤的效果显著。以 Qwen 的数据工程为例，通过构建包括基础规则过滤、统计指标筛选、语言模型评分在内的三层过滤 Pipeline，将原始网络抓取数据的可用率从不足 10%提升至超过 40%。第一层使用正则表达式快速过滤明显的噪声和格式错误，处理速度可达每秒数万条；第二层采用困惑度、重复率等统计指标进行质量评分，筛除低质量样本；第三层使用 fine-tuned 的评分模型对剩余数据进行精细化评估，确保进入训练集的数据符合高质量标准。这种渐进式过滤策略在保证数据质量的同时，显著提升了处理效率。

3.4.2　知识覆盖度的量化指标

知识覆盖度反映了训练数据在不同领域和主题上的分布均衡性，直接影响模型的泛化能力和知识完整性。通过构建多维度的知识图谱和主题模型，能够精确量化数据集的知识分布特征，识别知识盲区并指导数据补充策略。

知识覆盖度评估采用层次化的分类体系，从学科大类、专业领域到具体主题逐层细化。利用 BERTopic 等主题建模技术自动识别文本主题，结合领域专家构建的知识本体进行映射和标注。

如图 3-6 所示的知识领域分布热力图直观展示了训练数据在不同学科和子领域的覆盖情况。

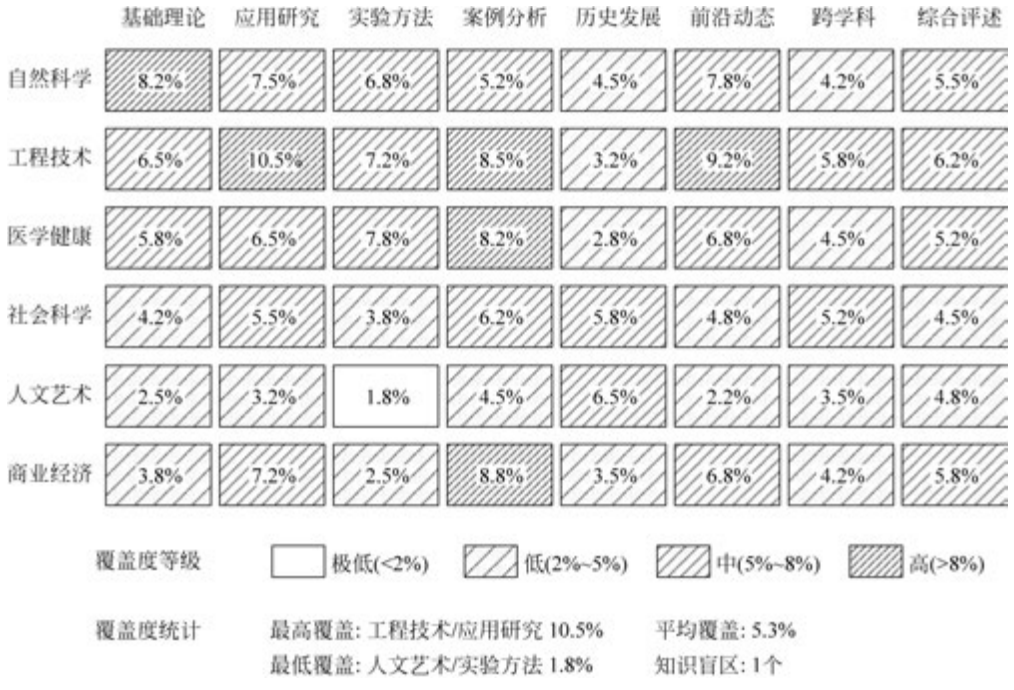

图 3-6　知识领域分布热力图

知识领域分布热力图直观展示了训练数据在不同学科和子领域的覆盖情况。图中采用四级斜线密度表示覆盖程度：空白表示极低覆盖（＜2%），稀疏斜线表示低覆盖（2%～5%），中等斜线表示中覆盖（5%～8%），密集斜线表示高覆盖（＞8%）。工程技术领域的应用研究覆盖度最高（10.5%），反映了技术类内容的丰富性；人文艺术的实验方法覆盖度最

低(1.8%)，是唯一低于2%阈值的知识盲区，提示需要重点补充相关数据。从整体分布看，自然科学和工程技术领域覆盖相对充分，而人文艺术和社会科学的某些细分领域存在明显不足。通过热力图的斜线密度分布，能够快速识别知识盲区(图3-6中的空白区域)，为后续的定向数据采集提供明确指导。

知识覆盖度的量化评估采用多项指标综合衡量。基尼系数用于衡量知识分布的均衡性，理想值为0.2～0.3；香农熵反映知识的多样性，值越高表示覆盖越全面；稀有度指数识别小众但重要的知识领域，确保长尾知识的保留。通过构建知识图谱，还能计算知识点之间的连通性和依赖关系，评估知识体系的完整性。

实践中发现，不同训练阶段对知识覆盖度的要求存在差异。预训练阶段追求广泛覆盖，各领域分布相对均衡；领域微调阶段则需要有针对性地增强特定领域的覆盖深度。建议采用分层采样策略，确保每个知识领域都有最低比例的数据(如不低于2%)，同时允许重点领域有更高的覆盖度。对于识别出的知识盲区，可以通过定向爬取、专家生成或跨领域迁移等方法进行补充。

3.4.3 毒性与偏见的检测方法

毒性和偏见检测是确保训练数据安全性和公平性的关键环节。最新研究表明，未经过滤的训练数据可能导致模型输出有害内容的概率提高3～5倍。通过建立多层次的检测体系，能够有效识别和过滤包含仇恨言论、歧视性内容、暴力倾向等有害信息的文本。

偏见检测需要从多个维度进行系统评估，包括性别、种族、宗教、年龄、职业等敏感属性，如表3-4所示。

表3-4 偏见检测指标体系表

偏见类型	检测方法	评估指标	阈值标准	处理策略
性别偏见 (Gender)	代词替换测试 BERT分类器 Word2Vec相似度	性别对称性 职业关联度 属性差异度	对称性＞0.8 关联差＜0.2	平衡重写 中性化处理
种族偏见 (Racial)	名称替换分析 情感分析模型 ToxiGen检测	情感偏差值 刻板印象分数 毒性得分	偏差＜0.15 毒性＜0.3	敏感词过滤 上下文重构
年龄偏见 (Age)	年龄词关联 共现矩阵分析 语义距离计算	能力关联偏差 描述词极性	关联＜0.25 极性均衡	年龄中立化 多样性增强
宗教偏见 (Religious)	信仰词汇检测 BOLD数据集 情感倾向分析	宗教公平度 负面关联率	公平度＞0.85 负面率＜0.1	平衡采样 敏感内容标记
职业偏见 (Professional)	职业-属性映射 WinoBias测试 嵌入空间分析	职业中立性 性别关联强度	中立性＞0.7 关联＜0.3	职业去关联 描述规范化
毒性内容 (Toxicity)	Perspective API RealToxicity LITI模型	毒性概率 仇恨言论分数 威胁等级	毒性＜0.2 仇恨＝0	直接删除 人工审核

续表

偏见类型	检测方法	评估指标	阈值标准	处理策略
隐性偏见 (Implicit)	上下文推理 GPT-4 评估 交叉验证	隐含偏见度 微妙歧视指数	偏见度<0.3 歧视指数<0.2	上下文重写 专家评审
政治倾向 (Political)	立场检测器 主题建模 情感极性分析	政治中立度 观点平衡性	中立度>0.75 平衡性>0.8	观点平衡 多源验证

注 1：阈值标准应根据具体应用场景和法规要求进行调整。

注 2：建议采用多种检测方法交叉验证，提高检测准确率。

注 3：处理策略优先级：删除>重写>标记>保留。

如表 3-4 所示的偏见检测指标体系表全面展示了八种主要偏见类型的检测方法、评估指标和处理策略。每种偏见类型都配置了多层次的检测手段：性别偏见通过代词替换测试和职业关联度分析识别，要求对称性大于 0.8；种族偏见使用 ToxiGen 等专门工具检测隐含的歧视性内容，毒性得分需低于 0.3；年龄偏见关注能力描述的公平性；宗教偏见评估不同信仰的情感倾向差异；职业偏见使用 WinoBias 等标准测试集验证；毒性内容采用 Perspective API 等多种工具交叉验证，确保零容忍；隐性偏见通过 GPT-4 等大模型进行深层语义分析；政治倾向评估立场的中立性和观点的平衡性。

在实际应用中，偏见检测需要采用分级响应策略。对于明确的毒性内容和仇恨言论，采取零容忍政策直接删除；对于存在偏见但有价值的内容，通过重写或中性化处理保留其信息价值；对于边界模糊的案例，添加标记并提交人工审核。建立反馈循环机制，将人工审核结果用于改进自动检测模型，不断提升检测准确率。

最新的 MBIAS 等技术通过指令微调和强化学习，能够在保持上下文连贯性的同时有效降低偏见。实验表明，经过系统化的偏见检测和处理，训练数据的公平性指标可以提升 30%～40%，模型输出有害内容的概率降低到 0.1%以下。需要注意的是，偏见检测标准应根据不同文化背景和应用场景进行调整，避免过度纠正导致的信息损失。

3.5 本章小结

本章系统介绍了大模型训练数据的采集与清洗技术体系。在数据源评估方面，建立了涵盖内容质量、技术指标、领域覆盖、合规性等多维度的评估框架，通过对主流开源数据集的系统评估，为数据源选择提供了科学依据。网络爬取需要在 Robots 协议等法律框架内规范进行，通过合理的并发控制和分布式架构实现高效采集。专有数据的获取建立了从数据源接入到质量验证的完整 Pipeline，确保数据来源的合法性和处理的规范性。

数据清洗 Pipeline 的设计涵盖了文本规范化、噪声过滤和敏感信息脱敏三个关键环节。文本规范化通过 Unicode 统一、字符处理和格式标准化确保数据的一致性；基于困惑度的噪声检测能够有效识别异常文本，通过 3σ 准则设定合理阈值；敏感信息检测覆盖八大类别，采用差异化的脱敏策略平衡隐私保护和数据可用性。

去重技术方面，MinHash 结合 LSH 算法实现了近线性时间复杂度的大规模去重，通过优化 band 数量和阈值设置达到召回率和准确率的平衡。语义去重利用 BERT 等模型生成

的向量表示，通过聚类分析识别语义相似的内容，能够处理改写和同义替换等复杂情况。分布式实现基于 Spark 等框架，通过合理的资源配置和优化策略，支持 TB 级数据的高效处理。

数据质量评估体系从语言质量、知识覆盖度、毒性与偏见三个核心维度建立了系统的评估方法。语言质量采用多层次评分策略，结合统计指标和深度学习模型实现准确评估；知识覆盖度通过热力图可视化展示领域分布，识别知识盲区，指导数据补充；毒性与偏见检测建立了八类偏见的检测指标体系，通过多工具交叉验证和分级处理策略，确保训练数据的安全性和公平性。

通过本章介绍的技术方法和工程实践，能够构建高质量、大规模的训练数据集，为大模型训练奠定坚实的数据基础。数据工程不仅是技术挑战，更需要在合规性、伦理性和实用性之间寻求平衡，这需要持续的技术创新和规范完善。