GPT 报告

目　录

插图清单

附表清单

1 研究背景及国内外发展现状

1.1 研究的关键问题

搜集整理来源丰富、大规模、高质量的中文语料。

1.2 主要工作

为了得到在中文数据上具有较高性能的预训练语言模型，本文主要工作如下：

1) 数据收集：利用互联网、书籍等多种渠道收集海量语料

2) 数据预处理：中文数据提取，去除重复、低质量语料

2 数据及技术调研

2.1 数据处理流程

盘古α搭建了面向大型语料库预处理的分布式集群，通过数据清洗过滤、去重、质量评估等处理流程，利用近 80TB 原始数据构建了一个约 1.1TB 的高质量中文语料数据集。整体流程图如图 1：

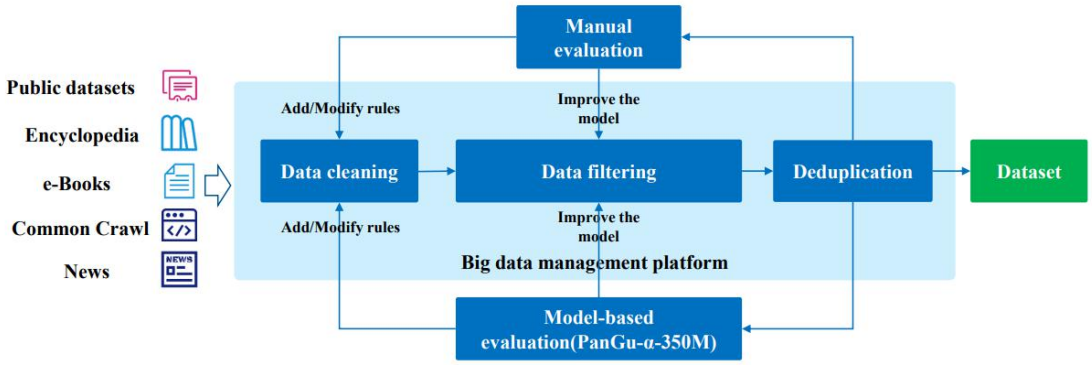

图 1 盘古α数据来源及预处理流程

此外，facebook 也构建了一个完整的数据处理流程，能够完成从各种语言的 Common Crawl 中提取大量高质量的单语数据集。首先是对数据进行下载得到 wet 文件并计算 hash 值分组保存为二进制文件，对二进制文件中的重复数据根据段落进行删除并识别其语言，然后利用高质量语料筛选步骤来选择接近高质量语料库（如维基百科）的数据，最后根据文档语言和高质量分数重新对文件进行整合。其整体流程如下图 2 所示：

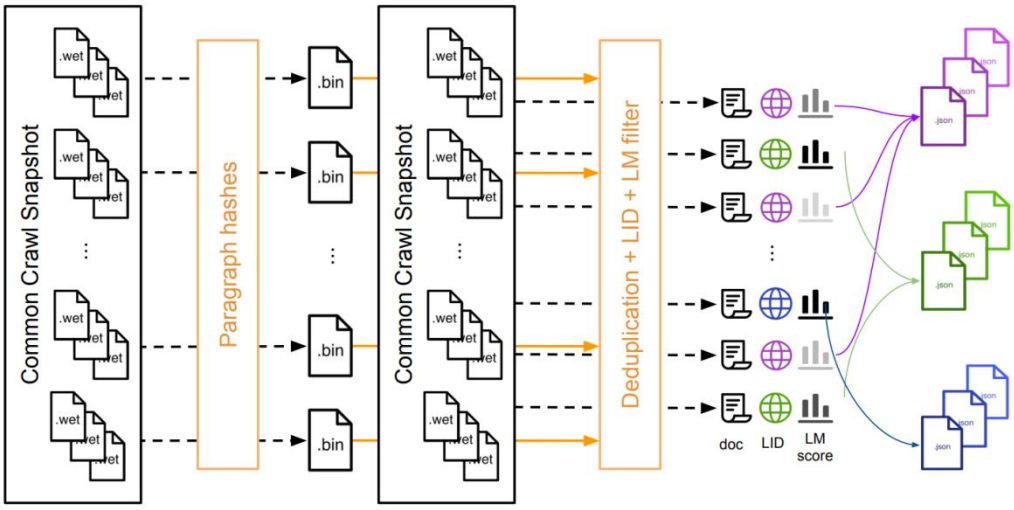

图 2 CC-Net 数据处理流程

3 设计方案及优化思路

3.1 数据来源

目前开源的中文数据集主要为 nlp_chinese_corpus[1] 项目中收集的 Wikipedia 中结构良好的中文词条以及 CLUECorpus2020[2] 项目中对 Common Crawl 2019 年 7 月到 12 月爬取的数据进行处理后得到的约 100GB 的中文语料。

根据 Common Crawl 上的数据统计，40%多的语言为英语，而仅有约 6%的语言为汉语，且根据对数据的观察，整体数据质量不高，因此，可用中文数据较少。

表格 1 2021 年 1 月-4 月语言占比统计（%）

月份	英语	俄语	汉语	德语
1 月	44.2	7.5	5.5	5.5
2 月/3 月	44.2	7.5	5.8	5.5
4 月	43.9	7.7	5.3	5.7

为满足大型中文预训练语言模型训练时的海量数据需求，仍需自行搜集整理大量的中文语料，目前选定的数据来源主要有四个：

a) Common Crawl 中文数据，并进行后续清洗工作，作为具有较高质量、数据量庞大的数据集

[1] https://github.com/brightmart/nlp_chinese_corpus

[2] https://github.com/CLUEbenchmark/CLUECorpus2020

b) Wikipedia、百度百科中的结构良好的中文词条

c) 中文书籍、报纸等高质量数据

d) 开源中文数据

3.2 数据下载、清洗

参考 CLUECorpus2020 项目，对 Common Crawl 上的更长时间爬取的数据进行下载和提取处理得到高质量的中文语料。本方案构建了一个完整的数据下载和清洗流程，并且，考虑到部分数据无需进行下载或者中文提取等处理步骤，整体流程可从任意步骤开始执行或结束。

3.2.1 整体流程

整体数据处理流程包括以下五个步骤，在多台设备上工作：

- 下载/爬取：从互联网上通过直接下载或者使用网络爬虫爬取的方式获取原始的文本数据。
- 中文提取：由于本方案的最终目的为训练一个适用于在多项任务上具有较好表现的中文大型语言模型，因此提取出所有语言类型主要为中文的文本数据。
- 清洗：根据一定的规则去掉不良文本。不良内容既包括色情、反动等违反公序良俗的内容，也包括过短的句子、网页的导航栏文字等无意义内容。

- 去重：首先使用哈希算法计算文本的特征值，对特征值相同的文本进行相似度计算，超过一定阈值则认为是的重复、冗余语料，需要进行删除。
- 筛选：使用语言模型通过计算文本困惑度的方式去掉不成文、不通顺或者质量低的语料。

图 3 整体流程

图 3 简要展示了该自然语言处理数据预处理系统的总体流程。首先位于数据下载服务器从互联网上下载语料数据并进行中文提取工作。接着，正则清洗服务器上进行不良语料清洗工作，这一步骤也是并行进行的。数据清洗完毕以后按照对应的月份被备份到不同的离线备份磁盘上。第三个步骤是去重，去重时，系统需要能够实时访问所有的语料文件便于计算文本相似度，因此使用连接一个在线硬盘阵列的去重服务器上运行多个进程的方式实现去重并行，同样，去重完毕的数据也会按月份被备份到对应的离线磁盘中去。后续高质量语料筛选、去重下游任务相关标签也可使用对应的服务器进行并行处理。并且上述提到的数据下载服务器、正则清洗服务器、去重服务器仅仅是为了区分功能进行的命名，不对应物理机器，实际使用时可根据实际机器进行合理安排。

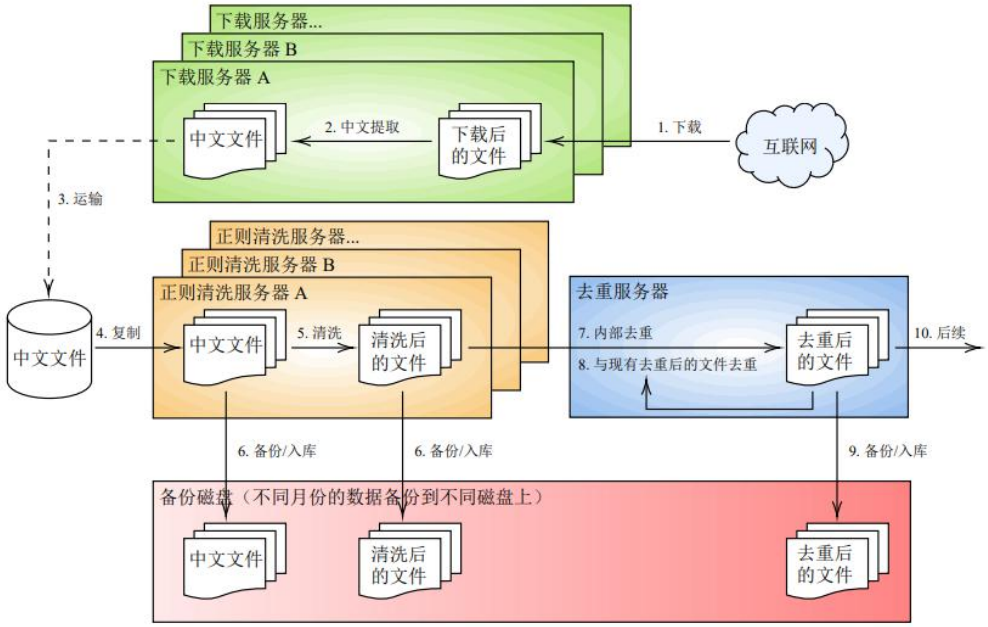

图 4 系统处理流程

为了确保数据处理的效率，尽可能利用起更多的计算资源，每个处理步骤都需要支持多个处理节点上的多个处理进程并行进行处理。而这就会引入高并发时的访问冲突问题，可能会带来数据一致性方面的隐患。因此，需要保证在高并发（数百个进程同时运行）时，整个系统的稳定性与数据一致性。对于单个处理模块（例如下载、提取、清洗等），具体流程如所示。该流程充分考虑到了多个处理进程之间的协调问题，还保证了出错时能够尽可能地减小损失。

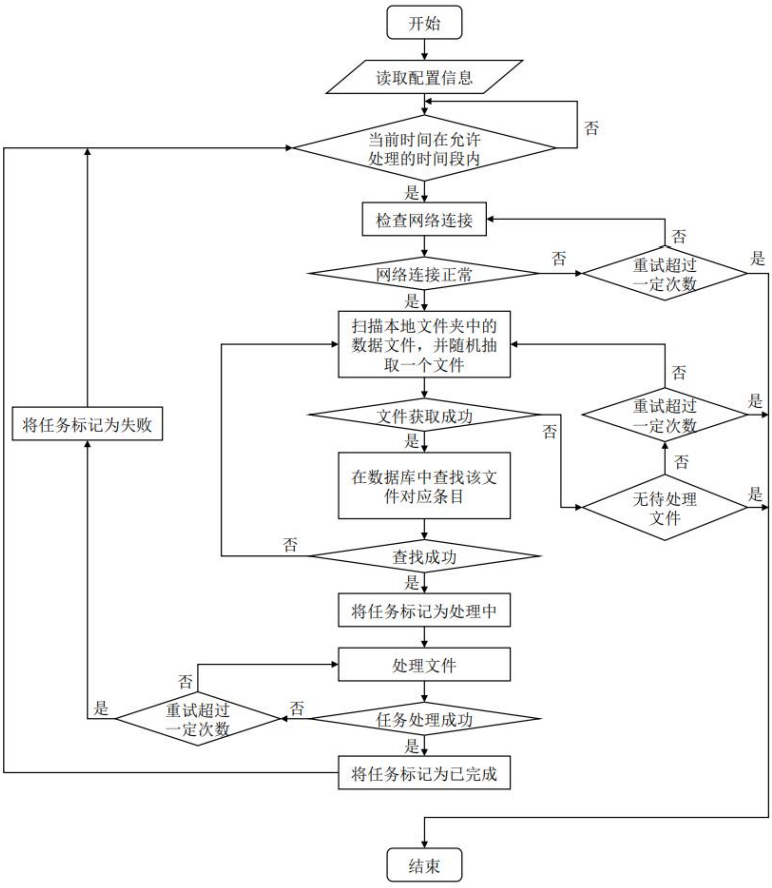

图 5 单个处理模块流程图

3.2.2 数据下载

互联网文本数据的获取有两种途径。一种方案是自行编写爬虫并爬取数据。然而自行爬取互联网数据存在以下困难：

- 难以获取较全的待爬取网址列表
- 难以获取网页历史数据
- 难以反制反爬措施
- 网页版权问题

另一种方案是直接下载或购买公开数据。例如 Common Crawl 数据集。Common Crawl 是一个非营利性组织，每月定期爬取网络数据并向公众免费提供数据集。Common Crawl 的网络档案包含自 2011 年以来收集的 PB 级的数据。计划从 Common Crawl 下载 400TB 大小的原始数据。

表格 2 2017 年-2021 年数据量统计

年份	压缩后大小（TB）
2021	18.10
2020	79.19
2019	96.61
2018	101.02
2017	109.66
总计	404.88

3.2.3 中文提取

比较三种不同的语言检测方法，最终选定 faseText 作为语言提取算法。

表格 3 语言检测算法对比

	运行效率	准确率	短文本文准确率
langid[30]	低	高	较高
langdetect	高	低	低
fastText[31][32]	较高	高	较高

Buck et al.（2014）指出，网页文本中通常会包含多种不同类型的语言。对于整个网页文本进行判断可能会丢失许多中文信息。因此，本方案选择记录行作为中文提取的单位。对于每行，定义中文占比 prop_{ch} 为

$$prop_{ch} = \frac{num_{ch}}{len_{data}} \text{ s.t.len}_{\text{data}} > \text{len}_{i} \#(4)$$

其中，num_{ch} 是一行文本中中文字符数量（简体中文、繁体中文、中文标点 符号），len_{dato} 是一行文本中除特殊字符（空格符、占位符、缩进等）外的字符数量，len_{i} 表示数据长度的不同范围。对于不同长度范围的数据采用不同的中文占比是有意义的。数据长度较长时，采取较低的比例限定可以保留很多含有其他语言的有意义的中文句子；数据长度较短时，采取高比例限定可以避免含有中文的短标题逃脱筛选，具体参数如表格 4 所示：

表格 4 参数选择

prop_{ch}	len_{i}
80%	任意
70%	70
60%	230

通过表格 5 可以看到，相比于对整条记录筛选，以行为单位可以得到更多的中文语料。

表格 5 以记录为单位和以行为单位中文提取比例

	提取比例
针对记录	约 3.8%
针对行	约 4.6%

3.2.4 正则清洗

正则清洗主要分为不良信息去除、无关文本去除两部分。

中文词汇中包含有色情、暴力、反动和其他不良词汇，不良文本对于训练一个正面的语言模型有非常负面的影响，因此需要去除不良信息以提高数据集的质量。首先从公开来源搜集了一份包含色情、贪腐、民生、暴恐和其他词汇的中文不良词汇表，并根据语料进行了适当的词汇删除和补充，各类不良词汇数量统计如表格 6。然后通过检测文本内容中的不良词汇出现的频率，根据不良分类和比例对内容进行删除，得到不良信息过滤后的文本。不良频率计算公式如下：

$$\text{prop}_{\text{dirty\_type}} = \text{len}_{\text{dirty\_type}} / \text{len}_{\text{data}} \#(5)$$

$\text{prop}_{\text{dirty\_type}}$ 表示某种类型敏感词出现的比例，$\text{len}_{\text{dirty\_type}}$ 表示某种类型敏感词的累计长度，len_{data} 表示文本长度。在不良信息去除步骤，由于对于行进行不良认定会造成许多的误判，为了保留尽量多的中文信息，因此选择记录作为检测单位。即使是人工判断时也是更倾向于去定义一段文本

是否是不良文本，而不是去定义某一行是关于不良信息。同时，以记录为单位也避免了处理后数据内容上的不连续。

无关内容去除主要是将所有不成句子和与自然语言处理无关的内容去除。主要包括以下几个部分：

- 导航栏、短链接等不成句子的内容。
- 网页提示信息、声明信息等网页模板类内容。
- 时间、阅读量等对自然语言处理无意义的内容。
- 访问网页错误、无返回网页等访问错误信息
- 重复的新闻标题等多次出现的信息。
- 不成连续内容的的网页。

为了剔除这些无用信息，设置了四个过滤器来对内容进行过滤：

- 网页结尾截取
- 完整句子检测
- 乱码去除
- 长度限定

网页结尾通常为版权信息、备案信息、导航信息、网站主体信息等无意义的内容，通常不成句，因此可以利用中文符号对结尾进行判定。找到全文最后一个结束符，并将结束符之后的内容进行删除，本方案选定的结束符如表格 7：

表格 6 各类不良词汇数量统计

种类	词数
色情	1594
反动	792
暴恐	254
民生	736
贪腐	97
其他	2559

表格 7 中文结束符

句号	。
问号	？
双引号	”

在网页首部和网页中间包含着大量的标签和断句，这些通常为一些导航、标题、链接、日期等对自然语言处理无意义的信息。因此，需要对句子的完整性进行检测，以提取出连续有意义的内容。本方案根据一行文本是否包含中文标点符号来判断该行文本是否是有效句。

以及，由于网页恶意写入或 unicode 识别错误等原因，大量网页文本中存在如“\x01\x02\x03”等不应出现在正常文本中的乱码符号，本方案采用正则匹配、替换的方式将乱码去除。

通过观察数据发现，每行的内容质量是与长度呈正相关的，因此，对内容进行一定的长度限定是有意义的。通过观察不同长度范围内的数据，最终选择将长度为 150 以下的记录删除。取样发现，其中包括大量的访问错误返回信息、不成连续内容、网页声明、网页引导类内容。

3.2.5 去重

本方案针对文本进行篇章级去重。对于完全相同的数据，使用 MD5 之类的哈希函数也可以非常容易地判断是否重复并且时间复杂度很低，然而对于相似而非完全相同的数据则无法很好地进行判断。另一个简单方法是对所有数据两两计算 Jaccard 相似度来判断是否重复。Jaccard 相似度指集合的交与集合的并之比；对于给定的两条数据，使用滑动窗口得到数据切片，对比两条数据切片集合的 Jaccard 相似度，相似度高则认为两文档是重复文档。但这种方法时间复杂度时间复杂度为 $O(N^2)$, 时间复杂度过高，对于大数据量数据集难以接受。

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \#(6)$$

为了保证去重效果并且考虑算法时间复杂度可接受，本方案使用 MinHash 算法结合 Jaccard 的方案来进行大规模数据的去重处理。由于使用 MinHash 表示数据时，发生哈希冲突的概率等于 Jaccard 相似度。因此首先利用 MinHash 为每个数据生成一组 MinHash 值，MinHash 值完全相同的数据被存入相同的“桶”中，然后对每个“桶”

内的数据计算 Jaccard 相似度。Jaccard 相似度高于阈值的数据即被丢弃。

3.2.6 高质量语料筛选

参考 CCNet 的处理筛选方式，本方案首先使用维基百科等少量高质量数据训练了一个较小规模（约 9000 万参数）的中文 GPT 模型。并利用小规模文本对待筛选数据进行困惑度计算，利用困惑度分数对数据分级。首先筛出困惑度 230 以上的数据保留其余所有数据并记录困惑度分数，使用根据需要使用的数据数量利用困惑度分数进行筛选。例如当整体数据量很大而需求的数据很少时，可以将困惑度阈值设置的比较低一些，仅保留质量非常高的数据；而当整体数据量较少，而需求数据较多时，可以将困惑度阈值设置的比较高，接受质量较为一般的数据。

4 技术实现细节

4.1 数据收集、预处理

在处理大量的语料数据时，难免会出现错误，而数据无价。在错误发生时，必须得保证数据不发生错误与丢失，而且所有任务的状态不能混乱，保证任务 的可恢复性。具体而言，该自然语言处理数据预处理系统的出错处理有如下几点：

- 由于需要通过网络连接中央数据库，所以需要保证网络的可用性。处理进程开始时需要检测网络状态是否中断，如果失败则进行重试。重试超过一定次数表示网络不通，此时报错并退出程序。
- 在整个程序的运行过程中，如果出现异常，捕获该异常并进行重试，重试超过一定次数将该任务标记为处理失败，并领取新任务。
- 捕获 Ctrl + C 等中断指令，正常退出程序。
- 在任何时候出现异常或中断时，回滚数据库，保证数据一致性与正确性。

数据预处理部分选用了 MySQL 8 数据库和 MongoDB 数据库。MySQL 8 及以上版本支持 FOR UPDATE SKIP LOCKED 语句，可以自动跳过被上锁的数据条目，方便进行进程间同步，避免访问冲突。然而，由于在数据去重阶段需要计算大量的哈希值并进行存储，然而 MySQL 这种关系型数据库无法支撑如此大量（TB 级） 的数据存

储，因此最终选用了 MongoDB 进行哈希值的存储。同时，两种数据库都建立了科学高效的索引，使得数据的查询速度基本上不会受到数据量增大带来的影响。

代码使用 Python 进行开发。Python 拥有许多实用且高效的自然语言处理相关的工具包，大大简化了编码流程。并且为了方便数据库存取代码的编写，也为了保证数据库访问的安全性，使用了 ORM 风格的数据库连接方式。即不必书写裸 SQL，而是将数据库表映射到 Python 对象上，极大简化了操作，也通过隐藏 SQL 语句加强安全性。

该系统在设计时对系统的模块进行了拆分以保证复用性。如图 6 所示，每 个预处理模块都共用了许多的基础模块。其中，配置读取模块、数据库连接模 块、进度监控模块被所有数据预处理模块使用，而正则模块被正则清洗与数据 去重模块使用。网络传输模块只被文件下载模块使用，中文检测模块只被中文 提取模块使用，哈希模块只被数据去重模块使用。文件下载模块、中文提取模 块、正则清洗模块、数据去重模块共同组成了数据预处理系统。

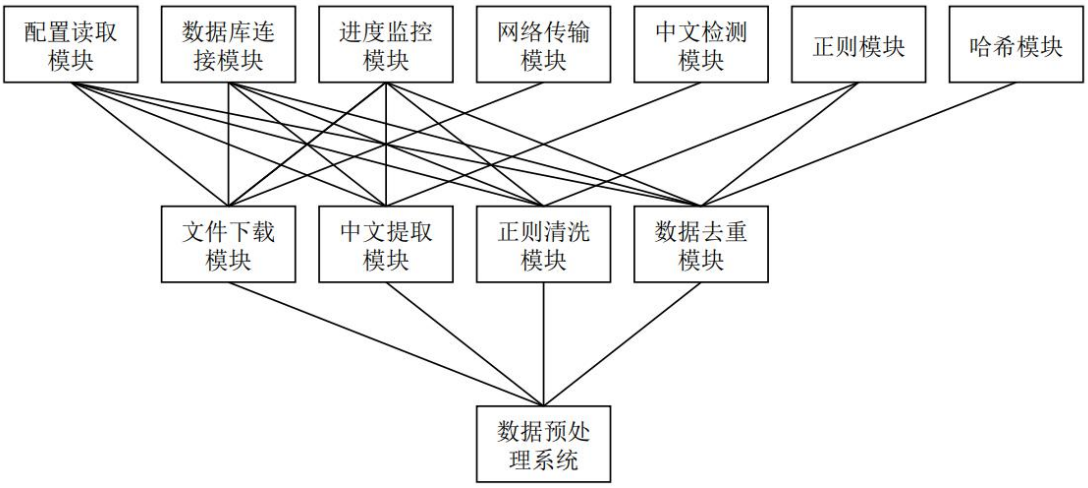

图 6 数据预处理系统模块图