
第 5 章

数据采集与预处理

大数据本身是一座金矿、一种资源，但“沉睡”的资源是很难创造价值的，它必须经过采集、清洗、处理、分析、可视化等加工处理之后，才能真正产生价值。而数据采集和预处理是具有关键意义的第一道环节。通过数据采集，我们可以获取传感器数据、互联网数据、日志文件、企业业务系统数据等，用于后续的数据分析。采集得到的数据需要进行预处理，数据预处理包括数据清洗、数据转换和数据脱敏。数据清洗是发现并纠正数据文件中可识别错误的一道程序，该步骤针对数据审查过程中发现的明显错误值、缺失值、异常值、可疑数据，选用适当方法进行“清理”，使“脏”数据变为“干净”数据，有利于通过后续的统计分析得出可靠的结论。数据转换是把原始数据转换成符合目标算法要求的数据。数据脱敏的目的是实现可靠保护敏感隐私数据。

本章首先介绍数据采集，然后介绍数据清洗和数据转换，最后介绍数据脱敏。

5.1 数据采集

本节介绍数据采集的概念、数据采集的三大要点、数据采集的数据源、数据采集方法和网络爬虫。

5.1.1 数据采集的概念

数据采集是大数据产业的基石。大数据具有很高的商业价值，但是，如果没有数据，价值就无从谈起，就好比不开采石油，就不会有汽油。数据采集，又称“数据获取”，是数据分析的入口，也是数据分析过程中相当重要的一个环节。它通过各种技术手段把外部各种数据源产生的数据实时或非实时地采集并加以利用。在数据大爆炸的互联网时代，被采集的数据的类型也是复杂多样的，包括结构化数据、半结构化数据、非结构化数据。结构化数据最常见，就是保存在关系数据库中的数据。非结构化数据是数据结构不规则或不完整，没有预定义的数据模型，包括所有格式的传感器数据、办公文档、文本、图片、XML、HTML、各类报表、图像和音频/视频信息等。

大数据采集与传统的数据采集既有联系又有区别，大数据采集是在传统的数据采集基础之上

发展起来的，一些经过多年发展的数据采集架构、技术和工具被继承下来，同时，由于大数据本身具有数据量大、数据类型丰富、处理速度快等特性，使得大数据采集又表现出不同于传统数据采集的一些特点（见表 5-1）。

表 5-1 传统数据采集与大数据采集的区别

	传统数据采集	大数据采集
数据源	来源单一，数据量相对较少	来源广泛，数据量巨大
数据类型	结构单一	数据类型丰富，包括结构化、半结构化和非结构化数据
数据存储	关系数据库和并行数据仓库	分布式数据库，分布式文件系统

5.1.2 数据采集的三大要点

数据采集的三大要点如下。

(1) 全面性。全面性是指数据量足够具有分析价值、数据面足够支撑分析需求。比如对于“查看商品详情”这一行为，需要采集用户触发时的环境信息、会话以及用户 ID，最后需要统计这一行为在某一时段触发的人数、次数、人均次数、活跃比等。

(2) 多维性。数据更重要的是能满足分析需求。数据采集必须能够灵活、快速自定义数据的多种属性和不同类型，从而满足不同的分析目标要求。比如“查看商品详情”这一行为，通过“埋点”，我们才能知道用户查看的商品是什么、商品价格、商品类型、商品 ID 等多个属性，从而知道用户看过哪些商品、什么类型的商品被查看得多、某一个商品被查看了多少次，而不仅是知道用户进入了商品详情页。

(3) 高效性。高效性包含技术执行的高效性、团队内部成员协同的高效性，以及数据分析需求和目标实现的高效性。也就是说，采集数据一定要明确采集目的，带着问题搜集信息，使信息采集更高效、更有针对性。此外，采集数据还要考虑数据的及时性。

5.1.3 数据采集的数据源

数据采集的主要数据源包括传感器数据、互联网数据、日志文件、企业业务系统数据等。

1. 传感器数据

传感器是一种检测装置，能感受到被测量环境的信号，并能将感受到的信号按一定规律转换成电信号或其他所需形式的信号输出，以满足信息的传输、处理、存储、显示、记录和控制等要求。在工作现场，我们会安装很多的各种类型的传感器，如压力传感器、温度传感器、流量传感器、声音传感器、电参数传感器等。传感器对环境的适应能力很强，可以应对各种恶劣的工作环境。在日常生活中，如 DV 录像、手机拍照等都属于传感器数据采集的一部分，支持音频、视频、图片等文件或附件的采集工作。

2. 互联网数据

互联网数据的采集通常借助于网络爬虫来完成。所谓“网络爬虫”（简称爬虫），就是一个在

网上不定向或定向抓取网页数据的程序。抓取网页数据的一般方法是，定义一个入口页面，一般一个页面中会包含指向其他页面的 URL，于是从当前页面获取这些网址并将其加入爬虫的抓取队列中，然后进入新页面后递归地进行上述的操作。爬虫数据采集方法可以将非结构化数据从网页中抽取出来，将其存储为统一的本地数据文件，并以结构化的方式存储。它支持图片、音频、视频等文件或附件的采集，附件与正文可以自动关联。

3. 日志文件

许多公司的业务平台每天都会产生大量的日志文件。日志文件一般由数据源系统产生，用于记录数据源执行的各种操作活动，比如网络监控的流量管理、金融应用的股票记账和 Web 服务器记录的用户访问行为。从这些日志文件中，我们可以得到很多有价值的信息。通过对这些日志文件进行采集，然后进行数据分析，我们就可以从公司业务平台日志文件中挖掘到具有潜在价值的信息，为公司决策和公司后台服务器平台性能评估提供可靠的数据保证。系统日志采集系统做的事情就是收集日志文件，提供离线和在线的实时分析使用。

4. 企业业务系统数据

一些企业会使用传统的关系数据库 MySQL 或 Oracle 等来存储业务系统数据，除此之外，Redis 和 MongoDB 这样的 NoSQL 数据库也常用于数据的存储。企业每时每刻产生的业务数据，以数据库行记录的形式被直接写入数据库中。企业可以借助于 ETL 工具，把分散在企业不同位置的业务系统的数据，抽取、转换、加载到企业数据仓库，以供后续的商务智能分析使用。通过采集不同业务系统的数据并将这些数据统一保存到一个数据仓库，就可以为分散在企业不同地方的商务数据提供一个统一的视图，满足企业的各种商务决策分析需求。

在采集企业业务系统数据时，由于采集的数据类型复杂，对不同类型的数据进行数据分析之前，我们必须通过数据抽取技术，将复杂格式的数据进行数据抽取，从而得到需要的数据，这里可以丢弃一些不重要的数据。经过数据抽取得到的数据，由于数据采集可能存在不准确的情况，所以，必须进行数据清洗（预处理）；对那些不正确的数据进行过滤、剔除。针对不同的应用场景，对数据进行分析的工具或者系统不同，我们还需要对数据进行转换操作，将其转换成不同的数据格式，最终按照预先定义好的数据仓库模型，将数据加载到数据仓库中。

5.1.4 数据采集方法

数据采集是数据系统必不可少的关键操作，也是数据平台的根基。根据不同的应用环境及采集对象，有多种不同的数据采集方法，包括系统日志采集、分布式消息订阅分发、ETL、网络数据采集等。

1. 系统日志采集

Flume 是 Cloudera 公司提供的具有高可用、高可靠、分布式的海量日志采集、聚合和传输的工具，Flume 支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume 提供对数据进行简单处理，并写到各种数据接收方（可定制）的能力。

Flume 运行的核心是 Agent。Flume 以 Agent 为最小的独立运行单位，一个 Agent 就是一个 Java

虚拟机 (Java Virtual Machine, JVM)。Agent 是一个完整的数据采集工具, 包含三个核心组件, 分别是数据源 (Source)、数据通道 (Channel) 和数据槽 (Sink) (见图 5-1)。通过这些组件, “事件” (Event) 可以从一个地方流向另一个地方。每个组件的具体功能如下。

(1) 数据源是数据的收集端, 负责将数据捕获后进行特殊的格式化处理, 将数据封装到事件里, 然后将事件推入数据通道。

(2) 数据通道是连接数据源和数据槽的组件, 可以将它看作一个数据的缓冲区 (数据队列)。它可以事件暂存到内存, 也可以持久化保存到本地磁盘上, 直到数据槽处理完该事件。

(3) 数据槽取出数据通道中的数据, 存储到文件系统和数据库, 或者提交到远程服务器。

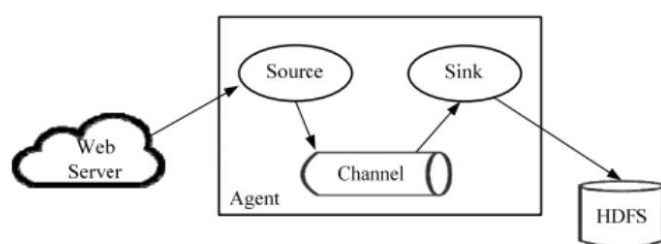


图 5-1 Flume 的核心组件

2. 分布式消息订阅分发

分布式消息订阅分发也是一种常见的数据采集方式, 其中, Kafka 就是一种具有代表性的产品。Kafka 是由 LinkedIn 公司开发的一种高吞吐量的分布式发布/订阅消息系统。用户通过 Kafka 系统可以发布大量的消息, 同时也能实时订阅消费消息。Kafka 设计的初衷是构建一个可以处理海量日志、用户行为和网站运营统计等的数据处理框架。为了满足上述应用需求, 数据处理框架就需要同时提供实时在线处理的低延迟和批量离线处理的高吞吐量等功能。现有的一些数据处理框架, 通常设计了完备的机制来保证消息传输的可靠性, 但是由此会带来较大的系统负担, 在批量处理海量数据时无法满足高吞吐量的要求。另外有一些数据处理框架则被设计成实时消息处理系统, 虽然可以带来很高的实时处理性能, 但是在批量离线场合时无法提供足够的持久性, 即可能发生消息丢失。同时, 在大数据时代涌现的新的日志收集处理系统 (如 Flume、Scribe 等) 往往更擅长批量离线处理, 而不能较好地支持实时在线处理。相对而言, Kafka 可以同时满足在线实时处理和批量离线处理的要求。

Kafka 的架构包括以下组件 (见图 5-2)。

- (1) 话题 (Topic): 特定类型的消息流。
- (2) 生产者 (Producer): 能够发布消息到话题的任何对象。
- (3) 服务代理 (Broker): 保存已发布的消息的服务器, 被称为代理或 Kafka 集群。
- (4) 消费者 (Consumer): 可以订阅一个或多个话题, 并从服务代理拉数据, 从而 “消费” 这些已发布的消息。

从图 5-2 中可以看出, 生产者将数据发送到服务代理, 服务代理有多个话题, 消费者从服务代理获取数据。

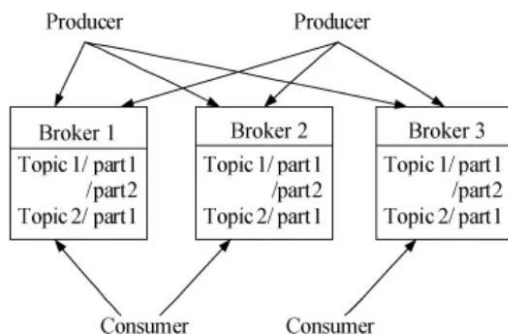


图 5-2 Kafka 的组件

3. ETL

ETL 常用于数据仓库中的数据收集和预处理环节。顾名思义，ETL 从原系统中抽取数据，并根据实际商务需求对数据进行转换，把转换结果加载到目标数据存储中。可以看出，ETL 既可用于数据采集环节，也可用于数据预处理环节。ETL 的源和目标通常都是数据库和文件，但也可以是其他类型的数据，比如消息队列。ETL 是实现大规模数据初步加载的理想解决方案，它提供了高级的转换能力。ETL 任务通常在“维护时间窗口”进行，在 ETL 任务执行期间，数据源默认不会发生变化，这就使得用户不必担忧 ETL 任务开销对数据源的影响，但同时意味着，对于商务用户而言，数据和应用并非任何时候都是可用的。目前，市场上主流的 ETL 工具包括 DataPipeline、Kettle、Talend、Informatica、Datax、Oracle Goldengate 等。其中，Kettle 是一款开源的 ETL 工具，使用 Java 编写，可以在 Windows、Linux、UNIX 上运行，数据抽取高效、稳定。Kettle 是“Kettle E.T.T.L. Environment”首字母的缩写，它可以实现抽取、转换和加载数据。Kettle 的中文含义是“水壶”，顾名思义，开发者希望把各种数据放到一个“壶”里，然后以一种指定的格式流出。Kettle 包含 Spoon、Pan、Chef、Encr 和 Kitchen 等组件。

4. 网络数据采集

网络数据采集是指通过网络爬虫或网站公开应用程序编程接口等方式从网站上获取数据信息。该方法可以将非结构化数据从网页中抽取出来，将其存储为统一的本地数据文件，并以结构化的方式存储。它支持图片、音频、视频等文件的采集，文件与正文可以自动关联。网络数据采集的应用领域十分广泛，包括搜索引擎与垂直搜索平台的搭建与运营，综合门户与行业门户、地方门户、专业门户网站数据支撑与流量运营，电子政务与电子商务平台的运营，知识管理与知识共享领域，企业竞争情报系统的运营，商业智能系统，信息咨询与信息增值，信息安全和信息监控等。

5.1.5 网络爬虫

网络爬虫是用于网络数据采集的关键技术，本节介绍什么是网络爬虫、网络爬虫的组成、网络爬虫的类型、Scrapy 爬虫以及反爬机制。

1. 什么是网络爬虫

网络爬虫是自动抓取网页的程序，它为搜索引擎从万维网上下载网页，是搜索引擎的重要组

成部分。网络爬虫的工作原理如图 5-3 所示，爬虫从一个或若干个初始网页的 URL（也叫种子 URL）开始，获得初始网页上的 URL，在抓取网页的过程中，不断从当前页面上提取新的 URL 加入任务队列，直到满足系统的特定停止条件。实际上，网络爬虫的行为和人们访问网站的行为是类似的。举个例子，用户平时到天猫商城购物（PC 端），他的整个活动过程就是打开浏览器→搜索天猫商城→单击链接进入天猫商城→选择所需商品（站内搜索）→浏览商品（价格、详情、评论等）→单击链接→进入下一个商品页面……周而复始。现在，这个过程不再由用户手动去完成，而是由网络爬虫自动去完成。

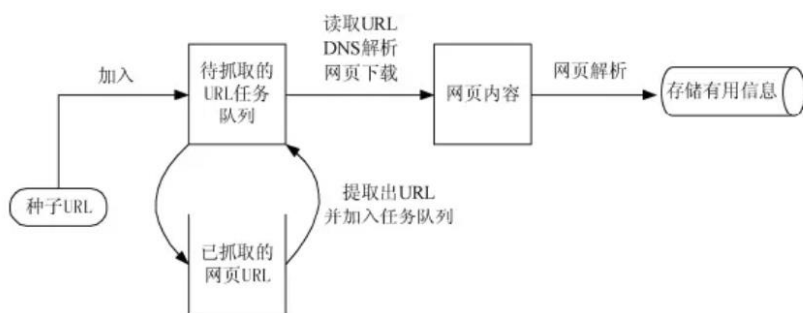


图 5-3 网络爬虫的工作原理

2. 网络爬虫的组成

网络爬虫由控制节点、爬虫节点和资源库构成。网络爬虫的控制节点和爬虫节点的结构关系如图 5-4 所示。从图 5-4 可以看出，网络爬虫中可以有多个控制节点，每个控制节点下可以有多个爬虫节点，控制节点可以互相通信。同时，控制节点和其下的各爬虫节点也可以互相通信，属于同一个控制节点的各爬虫节点亦可以互相通信。

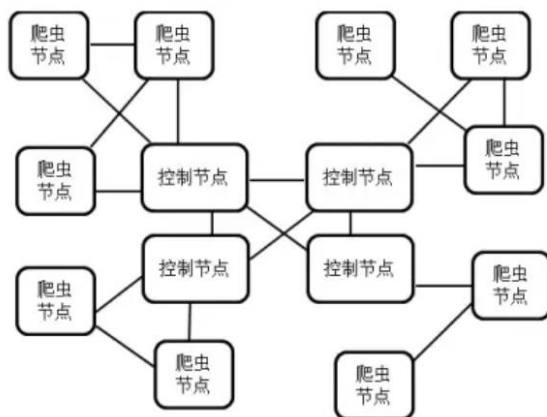


图 5-4 网络爬虫的控制节点和爬虫节点的结构关系

控制节点也叫作“爬虫的中央控制器”，主要负责为 URL 地址分配线程，并调用爬虫节点进行具体的抓取。爬虫节点会按照相关的算法，对网页进行具体的抓取，主要包括下载网页和对网页的文本进行处理，并在抓取后，将对应的抓取结果存储到对应的资源库中。

3. 网络爬虫的类型

网络爬虫可以分为通用网络爬虫、聚焦网络爬虫、增量式网络爬虫、深层网络爬虫 4 种类型。

(1) 通用网络爬虫。通用网络爬虫又称“全网爬虫”(Scalable Web Crawler), 抓取对象从一些种子 URL 扩充到整个 Web, 该类爬虫主要为门户网站搜索引擎和大型 Web 服务提供商采集数据。通用网络爬虫的结构大致可以包括页面抓取模块、页面分析模块、链接过滤模块、页面数据库、URL 队列和初始 URL 集合。为提高工作效率, 通用网络爬虫会采取一定的抓取策略。常用的抓取策略有: 深度优先策略和广度优先策略等。

(2) 聚焦网络爬虫。聚焦网络爬虫(Focused Crawler), 又称“主题网络爬虫”(Topical Crawler), 是指选择性地抓取那些与预先定义好的主题相关的页面的网络爬虫。和通用网络爬虫相比, 聚焦网络爬虫只需要抓取与主题相关的页面, 极大地节省了硬件和网络资源, 保存的页面也由于数量少而更新快, 还可以很好地满足一些特定人群对特定领域信息的需求。聚焦网络爬虫的工作流程较为复杂, 需要根据一定的网页分析算法过滤与主题无关的 URL, 保留有用的 URL 并将其放入等待抓取的 URL 队列。然后, 它将根据一定的搜索策略从队列中选择下一步要抓取的网页, 并重复上述过程, 直到达到系统的特定条件时停止。另外, 所有被爬虫抓取的网页将会被系统存储, 进行一定的分析、过滤, 并建立索引, 以便用于之后的查询和检索。对于聚焦网络爬虫来说, 这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。聚焦网络爬虫常用的策略包括: 基于内容评价的抓取策略、基于链接结构评价的抓取策略、基于增强学习的抓取策略和基于语境图的抓取策略等。

(3) 增量式网络爬虫。增量式网络爬虫(Incremental Web Crawler)是指对已下载页面采取增量式更新和只抓取新产生的或者已经发生变化页面的爬虫, 它能够在一定程度上保证所抓取的页面是尽可能新的页面。与周期性抓取和刷新页面的网络爬虫相比, 增量式网络爬虫只会在需要的时候抓取新产生或发生更新的页面, 并不重新下载没有发生变化的页面, 可有效减少数据下载量, 及时更新已抓取的页面, 减小时间和空间上的耗费, 但是增加了抓取算法的复杂度和实现难度。增量式网络爬虫有两个目标: 保持本地页面集中存储的页面为最新页面和提高本地页面集中页面的质量。为实现第一个目标, 增量式爬虫需要通过重新访问网页来更新本地页面集中页面内容。为了实现第二个目标, 增量式爬虫需要对网页的重要性排序, 常用的策略包括广度优先策略和 PageRank 优先策略等。

(4) 深层网络爬虫。深层网络爬虫将 Web 页面按存在方式分为表层网页(Surface Web)和深层网页(Deep Web, 也称 Invisible Web Page 或 Hidden Web)。表层网页是指传统搜索引擎可以索引的、超链接可以到达的静态网页。深层网页是那些大部分内容不能通过静态链接获取的、隐藏在搜索表单后的、只有用户提交一些关键词才能获得的 Web 页面。深层网络爬虫体系结构包含 6 个基本功能模块(抓取控制器、解析器、表单分析器、表单处理器、响应分析器、LVS 控制器)和两个爬虫内部数据结构(URL 列表、LVS 表)。

4. Scrapy 爬虫

Scrapy 是一套基于 Twisted 的异步处理框架, 是纯 Python 实现的爬虫框架, 用户只需要定制

开发几个模块就可以轻松地实现一个爬虫，可用来抓取网页内容或者各种图片。Scrapy 可运行于 Linux/Windows/macOS 等多种环境，具有速度快、扩展性强、使用简便等特点。即便是新手，也能迅速学会使用 Scrapy 编写所需要的爬虫程序。Scrapy 可以在本地运行，也可以部署到云端实现真正的生产级数据采集系统。Scrapy 用途广泛，可以用于数据挖掘、监测和自动化测试。Scrapy 吸引人的地方在于它是一个框架，任何人都可以根据需求对它进行修改。当然，Scrapy 只是基于 Python 的一个主流爬虫框架，除了 Scrapy 外，还有其他基于 Python 的爬虫框架，包括 Crawley、Portia、Newspaper、Python-goose、Beautiful Soup、Mechanize、Selenium 和 Cola 等。

(1) Scrapy 体系架构

图 5-5 所示的 Scrapy 体系架构由以下几个部分组成。

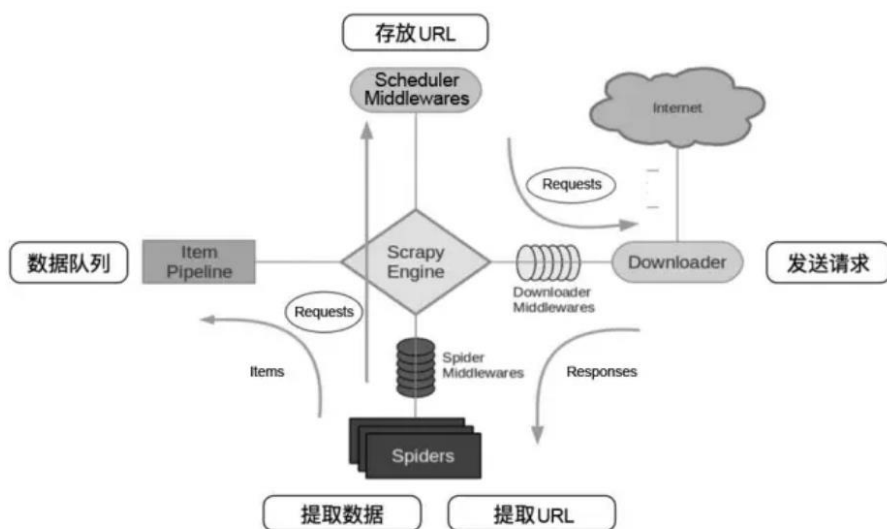


图 5-5 Scrapy 体系架构

① Scrapy 引擎 (Engine)。Scrapy 引擎相当于一个中车站，负责调度器、项目管道、下载器和爬虫 4 个组件之间的通信。例如，将接收到的爬虫的请求发送给调度器，将爬虫的存储请求发送给项目管道。调度器发送的请求，会被 Scrapy 引擎提交到下载器进行处理，而下载器处理完成后会发送响应给 Scrapy 引擎，Scrapy 引擎将其发送至爬虫进行处理。

② 爬虫 (Spiders)。爬虫相当于一个解析器，负责接收 Scrapy 引擎发送过来的响应，对其进行解析，开发者可以在其内部编写解析规则。解析好的内容可以发送存储请求给 Scrapy 引擎。在爬虫中解析出的新的 URL，可以向 Scrapy 引擎发送请求。注意，入口 URL 也存储在爬虫中。

③ 下载器 (Downloader)。下载器用于下载搜索引擎发送的所有请求，并将网页内容返回给爬虫。下载器建立在 Twisted 这个高效的异步模型之上。

④ 调度器 (Scheduler)。调度器可以被理解成一个队列，存储 Scrapy 引擎发送过来的 URL，并按顺序取出 URL 发送给 Scrapy 引擎进行请求操作。

⑤ 项目管道 (Item Pipeline)。项目管道是保存数据用的，它负责处理爬虫中获取的项目，比如去重、持久化存储 (如存数据库或写入文件)。

⑥ 下载器中间件(Downloader Middlewares)。下载器中间件是位于引擎和下载器之间的组件,主要用于处理引擎与下载器之间的请求和响应。类似于自定义扩展下载功能的组件。

⑦ 爬虫中间件(Spider Middlewares)。爬虫中间件是介于引擎和爬虫之间的组件,主要工作是处理爬虫的响应输入和请求输出。

⑧ 调度器中间件(Scheduler Middlewares)。调度器中间件是介于引擎和调度器之间的中间件,用于处理从引擎发送到调度器的请求和响应,可以自定义扩展和操作搜索引擎与爬虫中间“通信”的功能组件(如进入爬虫的请求和从爬虫出去请求)。

(2) Scrapy 工作流

Scrapy 工作流也叫作“运行流程”或“数据处理流程”,整个数据处理流程由 Scrapy 引擎进行控制,其主要的运行步骤如下。

- ① Scrapy 引擎从调度器中取出一个 URL 用于接下来的抓取。
- ② Scrapy 引擎把 URL 封装成一个请求并传给下载器。
- ③ 下载器把资源下载下来,并封装成应答包。
- ④ 爬虫解析应答包。
- ⑤ 如果解析出的是项目,则交给项目管道进行进一步的处理。
- ⑥ 如果解析出的是 URL,则把 URL 交给调度器等待抓取。

5. 反爬机制

为什么会有反爬机制?原因主要有两点:第一,在大数据时代,数据是十分宝贵的财富,很多企业不愿意让自己的数据被别人免费获取,因此,它们为自己的网站设计反爬机制,防止网页上的数据被抓取;第二,简单低级的网络爬虫,数据采集速度快,伪装度低,如果没有反爬机制,它们可以很快地抓取大量数据,甚至因为请求过多,造成网站服务器不能正常工作,影响企业的业务开展。

反爬机制是一把“双刃剑”,一方面可以保护企业网站的数据,但是,另一方面,如果反爬机制过于严格,可能会误伤到真正的用户请求,也就是真正的用户请求被错误当成网络爬虫而被拒绝。如果既要和“网络爬虫”死磕,又要保证很低的误伤率,那么会增加网站搭建的成本。

通常而言,伪装度高的网络爬虫速度慢,对服务器造成的负担也相对较小。所以,网站反爬的重点是针对那种简单粗暴的数据采集行为。有时反爬机制会允许伪装度高的网络爬虫获得数据,毕竟伪装度很高的数据采集行为与真实用户请求没有太大差别。

5.2 数据清洗

数据清洗对于获得高质量分析结果而言,其重要性不言而喻,正所谓“垃圾数据进,垃圾数据出”,没有高质量的输入数据,那么输出的分析结果价值会大打折扣,甚至没有任何价值。数据清洗是指将大量原始数据中的“脏”数据“洗掉”,它是发现并纠正数据文件中可识别的错误的最

后一道程序,包括检查数据一致性,处理无效值和缺失值等。比如,在构建数据仓库时,由于数据仓库中的数据是面向某一主题的数据的集合,这些数据从多个业务系统中抽取而来,而且包含历史数据,就避免不了有的数据是错误数据,有的数据相互冲突,这些错误的或有冲突的数据(称为“脏数据”)显然是我们不想要的。我们要按照一定的规则把“脏数据”给“洗掉”,这就是“数据清洗”。

5.2.1 数据清洗的内容

数据清洗主要是对缺失值、异常值、数据类型有误的数据和重复值进行处理。数据清洗的主要内容如下。

1. 缺失值处理

由于调查、编码和录入误差,数据中可能存在一些缺失值,需要给予适当的处理。常用的处理方法有:估算、整列删除、变量删除和成对删除。

① 估算:最简单的办法就是用某个变量的样本均值、中位数或众数代替缺失值。这种办法简单,但没有充分考虑数据中已有的信息,误差可能较大。另一种办法就是根据调查对象对其他问题的回答,对变量之间的相关分析或逻辑推论进行估计。例如,某一产品的拥有情况可能与家庭收入有关,可以根据调查对象的家庭收入推算拥有这一产品的可能性。

② 整列删除:剔除含有缺失值的样本。由于很多问卷都可能存在缺失值,这种做法可能导致有效样本量大大减少,无法充分利用已经收集到的数据。因此,整列删除只适合关键变量缺失,或者含有异常值或缺失值的样本比重很小的情况。

③ 变量删除:如果某一变量的缺失值很多,而且该变量对于所研究的问题不是特别重要,则可以考虑将该变量删除。这种做法减少了供分析用的变量数目,但没有改变样本量。

④ 成对删除:用一个特殊码(通常是9、99、999等)代表缺失值,同时保留数据集中的全部变量和样本。但是,在具体计算时只采用有完整答案的样本,因而不同的分析因涉及的变量不同,其有效样本量也会有所不同。这是一种保守的处理方法,最大限度地保留了数据集中的可用信息。

2. 异常值处理

异常值处理是指根据每个变量的合理取值范围和相互关系,检查数据是否合乎要求,发现超出正常范围、逻辑上不合理或者相互矛盾的数据。例如,用1~7级量表测量的变量出现了0值,体重出现了负数,都应视为超出正常值域。SPSS、SAS和Excel等计算机软件都能够根据定义的取值范围,自动识别每个超出范围的变量值。逻辑上相互矛盾的答案可能以多种形式出现:例如,许多调查对象说自己开车上班,又报告自己没有汽车;或者调查对象报告自己是某品牌的重度购买者和使用者,但同时在熟悉程度量表上给了很低的分值。发现矛盾时,要列出问卷序号、记录序号、变量名称、错误类别等,便于进一步核对和纠正。

3. 数据类型转换

数据类型往往会影响到后续的数据处理分析环节,因此,需要明确每个字段的数据类型,比如,来自A表的“学号”是字符型,而来自B表的“学号”是字符串型,在数据清洗的时候就需

要对二者的数据类型进行统一处理。

4. 重复值处理

重复值的存在会影响数据分析和挖掘结果的准确性，所以，在数据分析和建模之前需要进行数据重复性检验，如果存在重复值，还需要删除重复值。

5.2.2 数据清洗的注意事项

在进行数据清洗时，需要注意如下事项。

(1) 数据清洗时可优先进行缺失值、异常值和数据类型转换的操作，最后进行重复值处理。

(2) 在对缺失值、异常值进行处理时，要根据业务的需求进行处理，这些处理并不是一成不变的。常见的填充包括：统计值填充（常用的统计值有均值、中位数、众数）、前/后值填充（一般在前后数据存在关联时使用，比如数据是按照时间进行记录的）、零值填充。

(3) 在数据清洗之前，最重要的是对数据表进行查看，要了解表的结构和发现需要处理的值，才能将数据清洗彻底。

(4) 数据量的大小也关系着数据的处理方式。如果总数据量较大，而异常的数据（包括缺失值和异常值）的量较少时，可以选择直接删除，因为这通常并不太会影响到最终的分析结果；但是，如果总数据量较小，则每个数据都可能影响分析结果，这个时候就需要认真去对数据进行处理（可能需要通过其他的关联表去找到相关数据进行填充）。

(5) 在导入数据表后，一般需要对所有列依次地进行清洗，来保证数据处理的彻底性。有些数据可能看起来是正常可以使用的，实际上在进行处理时可能会出现（比如某列数据在查看时看起来是数值类型，但是其实这列数据是字符串类型，这就会导致在进行数值操作时无法使用）。

5.3 数据转换

数据转换就是将数据进行转换或归并，从而构成一个适合数据处理的形式。本节首先介绍常见的数据转换策略，然后重点介绍数据转换策略中的平滑处理和规范化处理。

5.3.1 数据转换策略

常见的数据转换策略如下。

(1) 平滑处理：帮助除去数据中的噪声。常用的方法包括分箱、回归和聚类等。

(2) 聚集处理：对数据进行汇总操作。例如，每天的数据经过汇总操作可以获得每月或每年的总额。这一操作常用于构造数据立方体或对数据进行多粒度的分析。

(3) 数据泛化处理：用更抽象（更高层次）的概念来取代低层次的数据对象。例如，街道属性可以泛化到更高层次的概念，如城市、国家，再比如年龄属性可以映射到更高层次的概念，如青年、中年和老年。

(4) 规范化处理: 将属性值按比例缩放, 使之落入一个特定的区间, 比如 0.0~1.0。常用的数据规范化方法包括 Min-Max 规范化、Z-Score 规范化和小数定标规范化等。

(5) 属性构造处理: 根据已有属性集构造新的属性, 后续数据处理直接使用新增的属性。例如, 根据已知的质量和体积属性, 计算出新的属性——密度。

5.3.2 平滑处理

噪声是指被测变量的一个随机错误和变化。平滑处理旨在帮助去掉数据中的噪声。常用的方法包括分箱、回归和聚类等。

1. 分箱

分箱 (Bin) 利用被平滑数据点的周围点 (近邻点), 对一组排序数据进行平滑处理, 排序后的数据被分配到若干箱子中。

典型的分箱方法一般有两种: 一种是等高方法, 即每个箱子中元素的个数相等; 另一种是等宽方法, 即每个箱子的取值间距 (左右边界之差) 相同, 如图 5-6 所示。

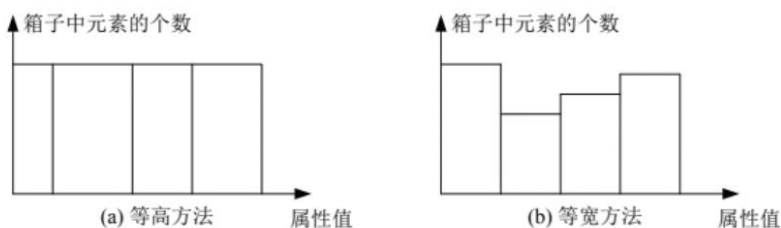


图 5-6 两种典型分箱方法

这里给出一个实例介绍分箱。假设有一个数据集 $X=\{4,8,15,21,21,24,25,28,34\}$, 这里采用基于平均值的等高方法对其进行平滑处理, 则分箱处理的步骤如下。

(1) 把原始数据集 X 放入以下 3 个箱子。

箱子 1: 4,8,15。

箱子 2: 21,21,24。

箱子 3: 25,28,34。

(2) 分别计算每个箱子的平均值。

箱子 1 的平均值: 9。

箱子 2 的平均值: 22。

箱子 3 的平均值: 29。

(3) 用每个箱子的平均值替换该箱子内的所有元素。

箱子 1: 9,9,9。

箱子 2: 22,22,22。

箱子 3: 29,29,29。

(4) 合并各个箱子中的元素得到新的数据集 $\{9,9,9,22,22,22,29,29,29\}$ 。

此外,还可以采用基于箱子边界的等高方法对数据进行平滑处理。利用边界进行平滑处理时,对于给定的箱子,其最大值与最小值就构成了该箱子的边界,利用每个箱子的边界值(最大值或最小值)替换该箱子中的所有值。这时的分箱结果如下。

箱子 1: 4,4,15。

箱子 2: 21,21,24。

箱子 3: 25,25,34。

合并各个箱子中的元素得到新的数据集 {4,4,15,21,21,24,25,25,34}。

2. 回归

回归是利用拟合函数对数据进行平滑处理。例如,借助线性回归方法(包括多变量回归方法),可以获得多个变量之间的拟合关系,从而达到利用一个(或一组)变量值来预测另一个变量值的目的。利用线性回归方法所获得的拟合函数,能够平滑数据并除去其中的噪声。对数据进行线性回归拟合如图 5-7 所示。

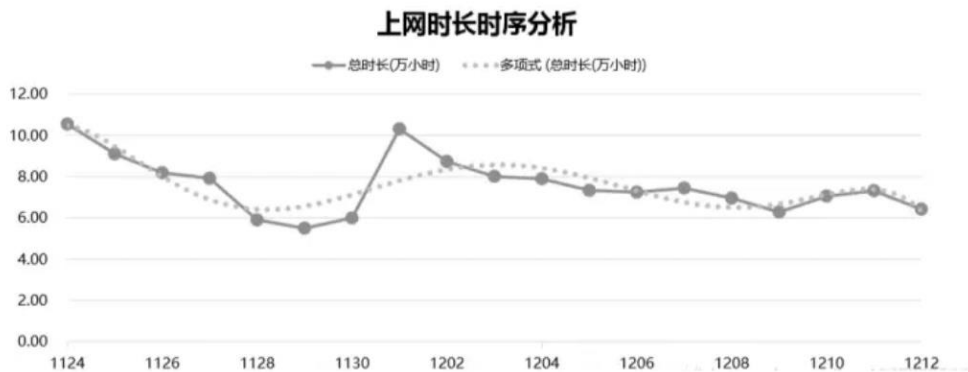


图 5-7 对数据进行线性回归拟合

3. 聚类

聚类可帮助发现异常数据。如图 5-8 所示,相似或相邻的数据聚合在一起形成了各个聚类集合,而那些位于这些聚类集合之外的数据对象,被认为是异常数据。

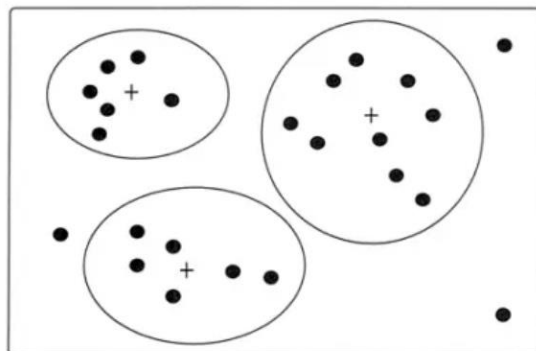


图 5-8 基于聚类的异常数据监测

5.3.3 规范化处理

规范化处理是一种重要的数据转换策略。它将一个属性取值范围投射到一个特定范围，以消除数值型属性因大小不一而造成挖掘结果的偏差，常用于神经网络、基于距离计算的最近邻分类和聚类挖掘的数据预处理等。对于神经网络，采用规范化后的数据，不仅有助于确保学习结果的正确性，而且会帮助提高学习的效率。对于基于距离计算的挖掘，规范化方法可以帮助消除因属性取值范围不同而影响挖掘结果的公正性的情况。

常用的规范化处理方法包括 Min-Max 规范化、Z-Score 规范化和小数定标规范化。

1. Min-Max 规范化

Min-Max 规范化方法对被转换数据进行一种线性转换，其转换公式如下：

$$x = (\text{待转换属性值} - \text{属性最小值}) / (\text{属性最大值} - \text{属性最小值})$$

例如，假设属性的最大值和最小值分别是 87 000 元和 11 000 元，现在需要利用 Min-Max 规范化方法，将“顾客收入”属性的值映射到 0~1 的范围内，则“顾客收入”属性的值为 72 400 元时，对应的转换结果如下：

$$(72400 - 11000) / (87000 - 11000) \approx 0.808$$

Min-Max 规范化比较简单，但是也存在一些缺陷，当有新的数据加入时，可能导致最大值和最小值的变化，需要重新定义属性最大值和最小值。

2. Z-Score 规范化

Z-Score 规范化的主要目的是将不同量级的数据统一转化为同一个量级的数据，统一用计算出的 Z-Score 值衡量，以保证数据之间的可比性。其转换公式如下：

$$z = (\text{待转换属性值} - \text{属性平均值}) / \text{属性标准差}$$

假设我们要比较学生 A 与学生 B 的考试成绩，A 的考卷满分是 100 分（及格 60 分），B 的考卷满分是 700 分（及格 420 分）。很显然，A 考出的 70 分与 B 考出的 70 分代表着完全不同的意义。但是从数值来讲，A 与 B 在数据表中都是用数字 70 代表各自的成绩。那么如何能够用一个同等的标准来比较 A 与 B 的成绩呢？Z-Score 就可以解决这一问题。

假设 A 班级的平均分是 80，标准差是 10，A 考了 90 分；B 班的平均分是 400，标准差是 100，B 考了 600 分。通过上面的公式，我们可以计算得出，A 的 Z-Score 是 1（即 $(90-80)/10$ ），B 的 Z-Score 是 2（即 $(600-400)/100$ ），因此，B 的成绩更为优异。若 A 考了 60 分，B 考了 300 分，则 A 的 Z-Score 是 -2，B 的 Z-Score 是 -1，A 的成绩比较差。

Z-Score 的优点是不需要知道数据集的最大值和最小值，对离群点规范化效果好。此外，Z-Score 能够应用于数值型的数据，并且不受数据量级的影响，因为它本身的作用就是消除量级给分析带来的不便。

但是 Z-Score 也有一些缺陷。首先，Z-Score 对于数据的分布有一定的要求，正态分布是最有利于 Z-Score 计算的。其次，Z-Score 消除了数据具有的实际意义，A 的 Z-Score 与 B 的 Z-Score 与他们各自的分数不再有关系，因此，Z-Score 的结果只能用于比较数据间的结果，探究数据的真

实意义时还需要还原数据。

3. 小数定标规范化

小数定标规范化通过移动属性值的小数位置来达到规范化的目的。所移动的小数位数取决于属性绝对值的最大值。其转换公式为：

$$x = \text{待转换属性值} / 10^k$$

其中， k 为能够使该属性绝对值的最大值的转换结果小于 1 的最小值。

比如，假设属性的取值范围是 $-957 \sim 924$ ，则该属性绝对值的最大值为 957，很显然，这时 $k=3$ 。当属性的值为 426 时，对应的转换结果如下：

$$426 / 10^3 = 0.426$$

小数定标规范化的优点是直观简单，缺点是并没有消除属性间的权重差异。

5.4 数据脱敏

数据脱敏是在给定的规则、策略下对敏感数据进行变换、修改的技术，能够在很大程度上解决敏感数据在非可信环境中使用的问题。它会根据数据保护规范和脱敏策略，通过对业务数据中的敏感信息实施自动变形，实现对敏感信息的隐藏和保护。在涉及客户安全数据或者一些商业性敏感数据的情况下，在不违反系统规则的前提下，需对身份证号、手机号、银行卡号、客户号等个人信息进行数据脱敏。数据脱敏不是必需的数据预处理环节，可以根据业务需求对数据进行脱敏处理，也可以不进行脱敏处理。

5.4.1 数据脱敏原则

数据脱敏不仅需要执行“数据漂白”，抹去数据中的敏感内容，同时需要保持原有的数据特征、业务规则和数据关联性，保证开发、测试以及大数据类业务不会受到脱敏的影响，达成脱敏前后的数据一致性和有效性，具体如下。

(1) 保持原有数据特征。数据脱敏前后必须保持原有数据特征，例如：身份证号码由 17 位数字本体码和 1 位校验码组成，分别为区域地址码（6 位）、出生日期码（8 位）、顺序码（3 位）和校验码（1 位）。那么身份证号码的脱敏规则就需要保证脱敏后依旧保持这些特征信息。

(2) 保持数据之间的一致性。在不同业务中，数据和数据之间具有一定的关联性。例如：出生年月或出生日期和年龄之间的关系。同样，身份证信息脱敏后仍需要保证出生日期字段和身份证中包含的出生日期之间的一致性。

(3) 保持业务规则的关联性。保持数据业务规则的关联性是指数据脱敏时数据关联性和业务语义等保持不变，其中数据关联性包括：主外键关联性、关联字段的业务语义关联性等。特别是高度敏感的账户类主体数据，往往会贯穿主体的所有关系和行为信息，因此需要特别注意保证所有相关主体信息的一致性。

(4) 多次脱敏数据之间的数据一致性。对相同的数据进行多次脱敏,或者在不同的测试系统进行脱敏,需要确保每次脱敏的数据始终保持一致性,只有这样才能保障业务系统数据变更的持续一致性和广义业务的持续一致性。

5.4.2 数据脱敏方法

数据脱敏主要包括以下方法。

(1) 数据替换。用设置的固定虚构值替换真值。例如将手机号码统一替换为 139***10002。

(2) 无效化。通过对数据值的截断、加密、隐藏等方式使敏感数据脱敏,使其不再具有使用价值,例如将地址的值替换为“*****”。数据无效化与数据替换所达成的效果基本类似。

(3) 随机化。采用随机数据代替真值,保持替换值的随机性以模拟样本的真实性。例如用随机生成的姓和名代替真值。

(4) 偏移和取整。通过随机移位改变数字数据,例如把日期“2018-01-02 8:12:25”变为“2018-01-02 8:00:00”。偏移取整在保持了数据的安全性的同时,保证了范围的大致真实性,此项功能在大数据利用环境中具有重大价值。

(5) 掩码屏蔽。掩码屏蔽是针对账户类数据的部分信息进行脱敏时的有力工具,比如对银行卡号或是身份证号的脱敏。例如,把身份证号码“220524199209010254”替换为“220524*****0254”。

(6) 灵活编码。在需要特殊脱敏规则时,可执行灵活编码以满足各种可能的脱敏规则。例如用固定字母和固定位数的数字替代合同编号真值。

5.5 本章小结

数据采集与预处理是大数据分析全流程的关键一环,直接决定了后续环节分析结果的质量高低。近年来,以大数据、物联网、人工智能、5G 为核心特征的数字化浪潮正席卷全球。随着网络和信息技术的不断普及,人类产生的数据量正在呈指数级增长,大约每两年翻一番,这意味着人类在最近两年产生的数据量相当于之前产生的全部数据量。世界上每时每刻都在产生大量的数据,包括物联网传感器数据、社交网络数据、企业业务系统数据等。面对如此海量的数据,如何有效收集这些数据并且进行清洗、转换,已经成为巨大的挑战。因此需要用相关的技术来收集数据,并且对数据进行清洗、转换和脱敏。

本章介绍了数据采集、数据清洗、数据转换和数据脱敏的方法。

5.6 习题

1. 请阐述传统数据采集与大数据采集的区别。

2. 请阐述数据采集的三大要点。
3. 请阐述数据采集的数据源有哪些。
4. 请阐述典型的数据采集方法有哪些。
5. 请阐述什么是网络爬虫。
6. 请阐述网络爬虫的组成。
7. 请阐述网络爬虫的类型。
8. 请阐述 Scrapy 爬虫的体系架构。
9. 请阐述数据清洗的主要内容。
10. 请阐述数据清洗的注意事项。
11. 请阐述数据转换包括哪些策略。
12. 请阐述数据规范化包含哪些方法。
13. 请阐述数据脱敏的原则。
14. 请阐述数据脱敏的方法。