分类号 _____ 密 级 _____

U D C _____ 编 号 10486

武漢大學

硕 士 专 业 学 位 论 文

基于 Neo4j 的研究团体搜索系统设计与实现

研 究 生 姓 名 ：陈小龙

学 号 ：2019282110194

指导教师姓名、职称 ：祝园园 副教授

专业类别（领域） ：计算机技术

二〇二一年五月

Dissertation Submitted to

Wuhan University

Design and Implementation of Research Group Search System Based on Neo4j

By

Xiaolong Chen

Under the Guidance of

Associate Professor Yuanyuan Zhu

May，2021

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

学位论文作者（签名）：

年　　月　　日

摘 要

当下，世界各地高校、研究机构以及不同学者间的交流合作越来越频繁与紧密。他们之间的合作成果往往以学术论文的形式发布，根据合作的密切程度，不同的学者之间会形成一个个的研究团体。

同时，学术文章与文章的创作者之间的关系可以用图（网络）来描述：文章和作者作为图中的顶点，不同作者的合作关系以及文章和作者的从属关系作为图中顶点的边。那么如何从这个网络中方便的检索出作者、文章、研究团体、作者相似度等方面的信息成为了现实的问题。

当前主流的搜索引擎 Google Scholar、Bing、Baidu 等，只能基于关键字匹配查询文章、作者的信息。然而，当用户需要查找的信息无法直接通过关键字匹配得到时，比如找到与某个作者合作发表过文章的人，上述搜索引擎便无法满足查询需求。基于上述文献信息检索问题，本文阐述了如何利用图论相关算法、Web 开发技术以及图数据库 Neo4j[1]，在 DBLP 数据集基础上设计并构建一个信息检索系统，用于解决上述研究团体检索及其关联信息查询的需求。本系统为搜索学术研究团体及其关联信息，进而探寻学术研究的前沿方向提供了一个有效的平台。

关键词：DBLP；Neo4j；社区搜索；图

ABSTRACT

Today's world, Communication and cooperation among universities, research institutions and different scholars around the world are becoming more frequent and close. The results of exchanges and cooperation between them are often published in the form of academic papers, and according to the close degree of exchanges and cooperation, different scholars will form a research group.

At the same time, the relationship between an academic article and the author of an article can be described by a graph. So how to easily retrieve author information, article information, author's research group information, and the similarity between authors from this network has become a real problem. However, the current mainstream search engines cannot solve the above problems well. Based on the above-mentioned literature information retrieval requirements, this article mainly elaborates that using graph theory related algorithms、web development technology and graph database Neo4j[1] to design and construct an information retrieval system based on the DBLP data set to solve the above research group retrieval and its related information query requirements. This system provides an effective platform for searching academic research groups and related information, and then exploring the frontiers of academic research.

Keywords: DBLP; Neo4j;Community Search;Graph

目录

1 绪论

1.1 研究背景和研究意义

随着互联网和学术研究的发展，学术界积累了海量的非结构化文献数据，如何有效管理和利用这些文献数据成为了一个极具价值的热点问题，而高效的学术信息检索正好可以实现对文献数据的充分利用，同时非关系型数据库则是存储非结构化数据的有力工具。

非关系型数据库可以根据其底层使用的数据结构分为不同类别，最常见的有存储键值对的 Redis[2]、列式存储的 HBase[3]、存储文档的 MongoDB[4]；Neo4j[1]则适合用来存储复杂的、相互关联的图数据。学术文献、作者之间具有复杂的关联性，普通的键值对、文档、表格等存储形式尽管在检索速度方面性能尚可，但无法充分的描述数据之间关联性。图作为一种天然具有描述复杂关联关系特性的数据结构，正好适合用于描述学术文献、文献作者之间的关系。同时得益于近年来算力和硬件的发展,计算机对图数据的处理能力有了长足的进步。因此，使用图结构和图数据库对文献数据进行管理和分析是一个相对新颖且可行的方案，在文献数据挖掘方面有着很大潜力。

学术信息检索是一个高频的需求场景，尤其对于高校研究人员和学生而言；但学术文献及作者之间的复杂关联关系衍生出了多样化的信息检索需求，比如文献检索、论文溯源、作者相似度查询、研究团体查询等。其中研究团体查询对促进学术交流、了解学术动态意义重大。传统的搜索引擎 Google Scholar、Bing 以及国内的学者网、知网等都提供了基于关键字匹配的文献检索功能，用户可以使用文章或者作者名称关键字检索对应的文献信息；但是这种方式缺点明显，无法检索与关键词不直接相关的信息，更无法满足数据关联关系的查询，例如无法通过作者的姓名查找与作者有合作关系的人。在关联信息查询方面，清华大学唐杰团队研发的 Aminer 平台利用学术网络图结合人工智能、数据挖掘，实现了更加高级的信息检索功能，如学术排名、人才迁徙、溯源树等。

但无论是基于关键词的 Google Scholar、Bing、学者网还是基于图、人工智能的 Aminer 以及相关的信息检索系统都没有提供研究团体搜索功能。本系统基于 DBLP 文献库使用 Neo4j[1]作为存储工具，通过集成不同的算法（如：中心性算法、社区检测算法），提供了研究团体搜索、作者相似度查询、中心度查

询、作者关联信息查询等复杂学术信息搜索服务，对文献库数据管理与挖掘、弥补目前学术信息检索领域短板具有重要意义。

1.2 国内外研究现状

1.2.1 图数据库的发展概况

根据 DB-Engines 的最新排名显示，在当前在业界使用的主流图数据库中，Neo4j[1]占据榜首位置，并且其市场占有率呈持续上趋势。

include secondary database models　　32 systems in ranking, February 2021

Rank Feb 2021	Rank Jan 2021	Rank Feb 2020	DBMS	Database Model	Score Feb 2021	Score Jan 2021	Score Feb 2020
1.	1.	1.	Neo4j	Graph	52.16	-1.62	+0.96
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	31.66	-1.31	-0.29
3.	3.	3.	OrientDB	Multi-model	5.13	-0.20	+0.19
4.	4.	4.	ArangoDB	Multi-model	5.07	-0.22	+0.22
5.	5.	↑ 7.	JanusGraph	Graph	2.53	-0.05	+0.65
6.	↑ 7.	↓ 5.	Virtuoso	Multi-model	2.37	+0.22	-0.40
7.	↑ 8.	↑ 9.	GraphDB	Multi-model	2.14	+0.03	+1.00
8.	↓ 6.	↓ 6.	Amazon Neptune	Multi-model	2.07	-0.24	+0.11
9.	9.	↑ 10.	FaunaDB	Multi-model	1.91	0.00	+0.92
10.	↑ 11.	↑ 12.	Stardog	Multi-model	1.46	0.00	+0.56
11.	↓ 10.	↓ 8.	Dgraph	Graph	1.40	-0.15	+0.25
12.	12.	↑ 13.	TigerGraph	Graph	1.33	-0.06	+0.46
13.	13.	↑ 14.	AllegroGraph	Multi-model	1.27	+0.08	+0.43
14.	14.	↓ 11.	Giraph	Graph	1.13	-0.01	+0.15
15.	15.	↑ 21.	Nebula Graph	Graph	0.98	+0.06	+0.84
16.	16.	↓ 15.	Blazegraph	Multi-model	0.87	0.00	+0.23
17.	17.	↓ 16.	Graph Engine	Multi-model	0.70	0.00	+0.12
18.	18.	↑ 19.	Grakn	Multi-model	0.67	-0.02	+0.40
19.	19.	↓ 17.	InfiniteGraph	Graph	0.50	0.00	+0.12
20.	↑ 21.	↓ 18.	FlockDB	Graph	0.32	+0.00	+0.03
21.	↓ 20.	↑ 31.	Fluree	Graph	0.30	-0.03	+0.30
22.	22.	↓ 20.	HyperGraphDB	Graph	0.28	+0.01	+0.08
23.	23.	↑ 25.	GraphBase	Graph	0.17	0.00	+0.05
24.	↑ 25.	↓ 23.	AnzoGraph DB	Multi-model	0.17	+0.01	+0.04
25.	↓ 24.	↑ 26.	Sparksee	Graph	0.16	-0.01	+0.04
26.	26.	↓ 22.	TinkerGraph	Graph	0.13	-0.02	-0.01
27.	27.	↑ 31.	TerminusDB	Graph, Multi-model	0.13	+0.01	+0.13
28.	↑ 30.	↑ 29.	HugeGraph	Graph	0.09	+0.02	+0.03
29.	↓ 28.	↓ 28.	VelocityDB	Multi-model	0.08	0.00	+0.01
30.	↓ 29.	30.	Memgraph	Graph	0.07	-0.01	+0.02
31.	31.	↓ 24.	AgensGraph	Multi-model	0.04	-0.02	-0.08
32.	32.	↓ 27.	HGraphDB	Graph	0.01	-0.01	-0.07

图 1.1 2021 年 02 月 DB-Engines 图数据库排名

DB-Engine 会每个月动态更新数据库排名一次。如图 1.1 中所示，截止 2021 年 2 月 Neo4j[1]、Microsoft Azure Cosmos DB 、OrientDB 三种数据库分别占据图数据库排行榜前三名。其中，Neo4j[1]的分数为为 52.16，Microsoft Azure Cosmos DB 的分数为 31.66，OrientDB 的分数为 5.513。显然，Neo4j[1]相对于二三名具有压倒性优势，以极大的优势占有着市场第一的位置，短期内这一趋势仍然会持续，同时，一些小众的图数据库也在不断的发展之中，值得人们持续关注。在可以预见的未来，随着非结构化复杂数据的处理需求上升，图数据库必定会获得更加广泛的应用。

1.2.2 Neo4j 的在国内外的应用案例

2016 年美国大选期间，俄罗斯水军被怀疑渗透了美国的网络空间操纵美国大选。NBC 新闻团队试着弄清水军如何利用推特改变公众的观念进而影响美国的政治。基于推特社交网络本质上是图这一特点，NBC 选择 Neo4j[1]作为社交网络数据存储工具。

社交网络图包含了实体（如：推特、用户、推特标签、链接等）之间的关系。图的算法基于实体之间的连接关系，揭示了实体在网络中的重要性。NBC 团队通过社区检测算法找出频繁与他人沟通的用户；然后，通过 pagerank 算法识别出最有影响力的账号。他们发现水军网络中只有数量很小的核心账号会编辑发送原创推文，这些原创推文大概只占推文总量的 25%。水军利用公共的标签和回复有名气的账号来扩大他们的影响力，增加关注度。最终，研究人员通过 Neo4j[1]发现了这些虚假账户在社交网络中的协作模式，揭示了水军如何影响美国政治。这个项目的意义在于，通过用 Neo4j[1]构建“关系引擎”，政府或者社交平台就能在恶意的水军影响舆情之前，采取行动维护社交秩序。

Neo4j[1]除了应用于社交网络行为分析，也常被应用于社交网络图谱、企业关系图谱，金融机构反欺诈等方面。社交网络是由人和人的关系、人和关联事物的关系组成的复杂网络。在这张图里面，图数据库可以完成一些非常复杂的查询，比如说找到某个人的朋友的朋友，找出有共同爱好的人等。企业关系图谱与社交网络图谱比较类似，在企业关系图谱中，图包含了企业相关的各种信息，如工商信息、组织架构、客户信息等。通过在企业关系图谱上进行复杂的查询可以获取到更加准确有用的信息，帮助经营者做出正确的决策。

在金融反欺诈方面，银行和保险公司每年因欺诈而损失数十亿美元。传统的欺诈检测方法在最小化这些损失方面起着重要作用。但是，仍然有老练的欺诈者利用各种构造虚假身份的手段，开发出更隐蔽的方法来逃避欺诈监测。尽管没有任何欺诈预防措施是完美的，但随着图论的发展以及图数据库的出现，经融机构的目光已经超越了单点数据，它们通过将所有有关系的数据链接起来，从这些链接关系中发现重要线索，以达到更加准确的欺诈检测率。人们发现欺诈行为的内在模式上有着共通之处，图形数据库提供了一种新的发现欺诈事件和其他复杂骗局的方法，并且该方法具有很高的准确性。通过运行适当的实体链接分析查询，它在支票退票、账户创建、余额达到阈值等重要的操作节点检测欺诈环，并以此来识别出可能的欺诈事件。

1.2.3 文献信息抽取的发展状况

经过多年的发展，人们对图的研究取得了许多进展，不断有新的思想和算法被开发出来。这为从海量的数据中获取信息提供了新的方法。典型的例子就是人们利用新的算法和软件工具分析大量文献之间的关系，获取隐藏的深层次知识。在这方面，崔雷、刘伟等人基于生物医学数据库开发了一个“基于文献数据库中书目信息共现关系进行文本挖掘的系统”[5]。此系统使用共现分析的方式，对文献数据库中的高频词、高频引用进行分析；并以此为基础进行关联分析和聚类，进而找出不同词语的联系，实现了基本的文献计量学分析功能。

除了对知识进行共现分析，知识图谱则是一种更加复杂的信息处理方式，其本质是一种知识实体之间关系的语义网络。知识图谱是以图论为基础发展起来的工具，涉及自然语言处理、机器学习等诸多领域，常被用作知识文献的可视化分析工具。文章[6]中，作者通过对大量文献高频词的共现分析以及对文献作者合作关系的分析，实现了对不同学术领域发展趋势、路径以及学术圈人物关系的发现。

图论、数据挖掘、机器学习等技术的不断进步，为文献数据的抽取提供了更加多样的方法，也为不同技术的组合使用提供了可能，清华大学唐杰团队研发的 Aminer 平台就是一个不同技术组合应用实现高级信息抽取的典型例子。在可见的未来，综合使用不同技术对学术信息进行管理和分析是一个必然趋势。

1.3 本文主要工作

本文的工作主要分为四个阶段。

第一个阶段：需求分析阶段。本文在这个阶段对用户需求进行详细分析，并基于用户需求确定系统需要实现的功能；同时确保设计功能的技术可行性以及操作可行性，保证后续的系统开发工作顺利进行。

第二个阶段：系统设计阶段。系统的体系结构和功能设计主要在这个阶段完成，设计要确保该系统健壮可靠并且具有较好的扩展性。系统设计阶段最重要的工作是进行技术选型，在此步骤中，本文对实现系统功能需要用到哪些技术或者工具进行了细致充分的调研，并比较了不同技术、工具的优劣，最终确定了该系统的实现架构以及需要用到的技术、工具。

第三个阶段：系统开发阶段。此阶段依据上一阶段确定的设计方案，进行系统功能和模块的开发。

第四个阶段：测试阶段。此阶段主要对已经完成开发的系统功能进行测试，确保该系统实现需求分析阶段确定的需求；并且修复了测试过程之中发现的若干问题。

1.4 论文组织结构

第一章介绍了基于文献的知识发现的发展状况以及背景。第二章对 Neo4j [1]、Pandas、NLTK 等工具和概念进行了介绍；重点对图数据库、系统架构进行详细阐述。第三章从图信息检索、数据处理、社区搜索等角度进行了需求分析，说明系统需要实现的目标。第四章介绍了系统所采用的图相关的算法，以及算法的扩展。第五章设计了系统的架构和功能模块。第六章则从单元测试、功能测试、性能测试三个方面说明了本文的系统测试工作。第七章概述了论文研究成果，并进一步提出后期可能的发展完善方向。

2 系统相关概念与工具介绍

2.1 图数据库

本系统所涉及的数据，是高度关联的非结构化数据，原始数据以 XML 文档的形式存储。XML 文档虽然具有结构清晰简单的特点，但是其缺点是占用空间大，且很难直接从 XML 标签数据中获取到不同标签数据之间的关联信息，从数据表达能力的角度看 XML 文档相当于是另一种形式的表格数据。

出于上述原因，同时考虑到数据之间的复杂关联和关系数据库的局限；采用 NoSql 数据库处理本系统数据集是合理且必要的方案。另外，图作为一种擅长表现高度关联数据的结构也正好可以满足系统对于数据关系的表达需求。综上所述，系统最终选择 NoSql 数据库家族中的 Neo4j[1]作为数据存储工具，系统测试结果也表明 Neo4j[1]完全可以胜任数据存储工作。

2.1.1 NoSql 数据库

非结构化数据存储工具--NoSql，它的数据模式更加自由，可以很方便的对数据进行水平扩展性，结构简单灵活，适合处理大量的数据。同时，任何事物都有两面性，NoSql 数据库亦有其缺点。首先，NoSql 数据库没公认的统一标准，相互之间不兼容，一般也不支持存储过程；其次，大部分 NosSql 数据库不支持 SQL，更没有标准统一的查询语言。但上述问题并不影响 NoSql 数据库发挥其灵活、适应性强的优势，NoSql 数据库仍有大量的应用场景。图形数据库是非关系型数据库中的一类，专门用来存储和处理图数据，数据表达能力强，方便构建和存储复杂的关系图。

2.1.2 图数据库 Neo4j

图数据库开创了新的数据表示与存储形式，它使我们可以更容易的理解数据之间的关联性。目前市场占有率最高的图数据库是 Neo4j[1]，对事务的支持是其一大特特点。Neo4j[1]按有向图存储数据，同时支持在边和节点上设置多个属性，还有专用的类似 SQL 图数据查询语言 Cypher。

Neo4j[1]很适合存储图数据这种半结构化的数据，它采用原生的图模型。这种存储模式避免了低效的表之间的连接，通过简单的直接遍历算法就可以方便的检索图中节点和边以及节点边的关联信息，查询效率也更高。更重要的是 Neo4j[1]的查询语句对数据之间的关联关系表达能力非常强，对于多重关联关系只

需要简单的类似自然语言的查询语句即可实现。但是，Neo4j[1]并非完全开源，不方便分布式部署。另外，由于 Neo4j[1]采用原生图存储缺乏分片存储机制，在处理极大图的时候存在一定困难，写数据库性能不高。

总的来看，Neo4j[1]性能全面、社区活跃、市场占有率高，并且有丰富的文档和开源的社区版本，很适合用来作为图数据的存储和处理工具。

2.1.3 Cypher 查询语言

受 SQL[7]启发，Neo4j[1]团队开发了专门的查询语言 Cypher。Cypher 设计简单且功能强大、可读性好，能轻松的描述复杂的图数据查询逻辑；它使用模式匹配的方式选择数据，还支持对图进行灵活的修改和更新。Cypher 语句一般由保留关键字对应的若干子句组成，通过不同的子句组合实现对数据的增删改查。

MATCH 关键字用于指定数据库的搜索模式，通常与 WHERE 子句一同使用。WHERE 子句用于给 MATCH 模式中添加限制或谓词，从而对匹配结果进行过滤。RETURN 关键字可以返回模式中的指定数据，包括节点、边、属性以及路径。WITH 关键字则用于对查询数据进行预先处理。ORDER BY 关键字根据节点或关系上的属性对结果集进行排序，使结果集正序或者逆序排列。LIMIT 关键字用于限定返回结果集中的数据项数量，CREATE 关键字用于创建节点和边、索引等。DELETE 关键字用于删除节点和边。

在 Cypher 语法中，除了常用的关键字以外，还有一系列便捷好用的内置函数。这些内置函数进一步的增强了 Cypher 处理数据的能力。典型的内置函数包括：exist 用于判断匹配的模式是否存在、length 可以返回路径的长度、none 用于判断结果集是否为空等等。此外，Cypher 还提供对数据的完整访问控制，通过区分不同的用户和角色实现不同粒度的控制。另外，企业版 Neo4j[1]还支持同时管理多个数据库。

值得注意的是，Cypher 查询语言还在快速的发展之中；有理由相信，随着 Neo4j[1]的广泛应用，Cypher 一定会获得进一步的发展，变得更加成熟和完善。

2.2 WEB 开发相关技术

本文在系统设计开发阶段，经历了详尽的需求分析与技术调研，并在此基础上确定了系统要采用的技术架构与方案。系统采用当下 web 系统开发中主流的 B/S 架构，该架构简单成熟，方便系统的快速构建。

从系统的开发层面考虑，与“客户端/服务端”层次相对应的便是“前端/后端”。前端技术发展很快，先后经历了由静态页面展示到动态页面的过程。当前的主流前端开发模式是前后端分离，其中的代表技术就是基于数据驱动的组件化开发和 MVVM[8]设计模式；而 Vue 框架则是这两种技术的应用典范。由于具有响应式编程和组件化的优点，Vue 框架可以显著提升页面开发速度与用户体验。在后端实现层面，系统采用 REST 风格 API 实现前后端分离。采用 REST[9]风格 API 实现系统交互的最大优势在于，前后端都无需关心彼此的实现逻辑，在接口一致的情况下二者可以独立开发维护。其次，系统后端采用 MVC[10]设计模式，将数据处理放在 Model 层，将前端响应的处理逻辑放在 Controller 层；实现了不同逻辑之间的解耦，大大增强了后端功能的可扩展性。上述技术与设计模式的合理运用是本系统开发实现的基础。

2.2.1 B/S 架构

B/S（浏览器/服务器）是一种两层 web 架构，分别是负责交互的客户端，负责业务逻辑的服务端，如图 2.1 所示。

客户端对数据的访问都通过作为中介的服务器来完成，由服务器解析请求、获取数据库中的数据。对于对外提供数据访问的数据库服务器，可以根据业务需求采用不同类型的数据库，本系统中数据库层就是根据关联数据的处理需求而选择的图数据库。

B/S 架构尽管存在通信开销较大、数据安全等问题，但是由于其扩展性强，开发维护简单方便，仍被广泛应用于 web 系统开发中，且已经被证明是一种成熟可靠的系统架构，本文所述系统正是基于 B/S 架构。

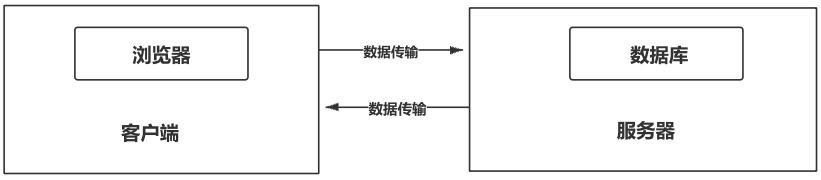

图 2.1 B/S 架构图

2.2.2 MVC 与 MVVM 设计模式

MVC 是一种松耦合的软件设计模式，其典型特征是分为控制、视图、模型三层。三个层次将整个系统的业务逻辑进行细分，其目的在于将业务逻辑中关

联度高的部分放在同一层，同时也将非关联的逻辑放在不同层次，确保层与层之间的业务逻辑不会相互影响。

MVC 设计模式的优势也在于业务分层，每一层负责处理各自的工作，相互之间可以独立开发维护，不会对其他层的业务产生干扰。MVC 模式另一个优点在于可以提高代码的复用性，因为不同的控制器和模型各自处理不同的逻辑，而一个事务可能会有多个逻辑相对独立的事务组成，如此一来不用的业务逻辑代码就可以被组合或者单独复用。因此也可以说，MVC 设计模式的核心在于降低业务逻辑之间的耦合，提高业务逻辑内部的聚合程度。这种低耦合，高内聚的思想也被用于软件设计的方方面面，是系统健壮性的重要保证。

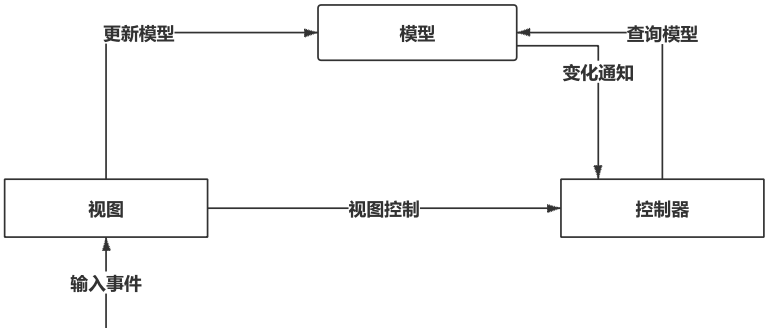

图 2.2 MVC 结构图

尽管 MVC 设计模式已经足够完善与成熟，但是技术的快速迭代发展促进了新需求和工具的产生，传统的 MVC 设计模式已经慢慢无法满足新的软件设计开发需求。举例来说，移动互联网的发展，带来了移动 APP、小程序等应用的爆发式增长，并且这些应用所支持的功能越来越多，交互界面也越来越复杂。多样的功能和复杂的交互界面直接导致要展示的数据变得复杂，进一步产生了数据解析的问题。

在 MVC 设计架构中，并没有单独的层次专职负责数据解析任务，也就是说数据解析任务混杂在了现有的三个层次之中，这也直接导致数据解析任务在各层之间界限不清，变得难以维护。显然，上述问题本质上是系统分层的局限性导致的。这是因为在 MVC 模式提出的初期，以当时的软件开发环境来看，并没有当前的数据解析逻辑归属不清的问题。既然 MVC 模式三层结构都不应该负责数据解析的任务，那么，新增加一个专门用于数据解析的层就是最佳的解决办法，MVVM 设计模式就是这个解决思路的产物。

MVVM 模式在 MVC 模式的基础之上重新划分了一个层次（ViewModel），它将原先 MVC 模式中的数据解析逻辑单独抽取出来放在该层中，这也使得 Contro

ller 层的代码大大简化，确保了各个层次的设计初衷不变。MVVM 设计模式方便测试，便于进行敏捷开发，并且继承了 MVC 模式的可扩展性，已经成为目前最流行的软件系统设计模式之一。本系统在搜索结果可视化的实现上便采用此设计模式，且实现了良好的用户体验。MVVM 结构如图 2.3 所示：

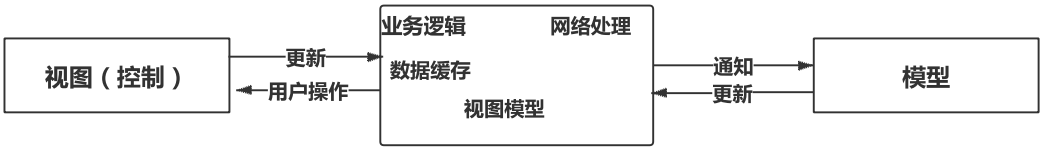

图 2.3 MVVM 结构图

2.2.3 REST 设计风格介绍

在互联网发展初期，页面的请求以及并发量并不高，基本的动态页面（如：jsp、php）就可以满足绝大部分的需求。但是随着移动互联网的蓬勃发展，传统低效的动态页面逐渐被用户体验更佳的 HTML&JavaScript（Ajax）实现的网页所取代。然而安卓、IOS、小程序等形式的移动客户端的大量出现让客户端的形式更加多元化，客户端的需求和功能越发复杂，这就导致 B/S 架构中客户端和服务器的通信接口规范化变成了一个亟待解决的问题。所以，人们普遍认为有必要设计一套标准清晰、简单、易于扩展的接口风格。这个需求直接催生了 RESTful 软件设计风格，正因为具备上述优点，该风格已经成为了当下最流行、使用最广泛的接口设计规范。

REST 是由 Roy Thomas Fielding 提出的一种软件开发思想；它定义了的一系列软件设计的规范和原则。REST 风格主要包含如下内容：客户端和服务端相互分离；服务端不保存客户端的请求状态；服务端响应客户端请求时会告诉客户端是否缓存响应结果；简化系统架构，降低接口的耦合度；系统分层，终端对客户端透明。RESTful API 就是指按照 REST 约束实现的接口。存储在服务器上的资源都可以用资源定位符 URI 进行标识，客户端可以借助于 REST 风格接口完成对服务器资源的状态转化，进而对资源进行操作。客户端使用的 HTTP 协议中有不同功能的操作动词，分别用来获取、更新、修改、删除服务器上的资源。本文基于 REST 风格开发了简洁后端接口，也为后续的功能扩展提供了空间。RESTful 风格应用的典型架构如图 2.4 所示：

图 2.4 RESTful 风格应用架构

2.2.4 Vue 介绍

Vue 是一个 MVVM 结构的前端框架，核心在于数据双向绑定和组件化开发；同时，方便配合其他支持类库和工具，实现复杂的单页面应用。在 MVVM 设计模式中，开发人员无需手动干预数据同步和数据状态管理。这大大加快了应用的开发速度，同时降低了开发的出错概率。另外，Vue 的开发社区非常活跃，并且技术文档丰富，方便开获取各种开发资源以及技术支持。

对于本系统的开发来说，不同功能的可视化页面均可以大致分为获取用户输入的数据采集模块、负责对算法结果进行展示的可视化模块、控制页面展示形式的控制模块。这些模块相互之间相对独立，很适合封装成不同的组件，再基于不同组件的组合进行组件化开发。

因此，结合系统的开发需求，再考虑 Vue 的诸多优点，本系统数据可视化模块使用 Vue 框架实现是一个理想的选择。从数据可视化的性能和效果来看，基于 Vue 框架开发的可视化模块完全符合设计要求。

2.3 数据处理工具

由于 XML 格式文件无法直接用于系统的数据分析，所以需要对原始数据进行预处理，并且根据这些处理过的信息构建图数据模型。另外，上述步骤构建的图数据还需要进一步处理成 Neo4j[1]可以识别的格式，以便可以导入 Neo4j[1]进行存储。

系统使用 xml.sax 工具解析原始的 XML 文档，xml.sax 的主要特点是边读取文件边解析，可以用较少的内存解析大量 XML 数据而不用考虑内存不够的问题。同时，系统使用数据处理工具 Pandas 对从原文档中提取的数据进行表格化处理，并进一步将数据转化成 Neo4j[1]的导入格式，为后续数据的存储扫清障碍。由于系统需要实现属性紧密社区的搜索功能，所以在数据处理阶段从文章标题中提取作者的属性（关键词）是必须的，也就意味着需要对文章标题进行

分词并。本系统采用 NLP 领域常用的分词工具 NLTK 实现对文章标题的分词与关键词提取。

2.3.1 Python 数据处理工具：xml.sax

xml.sax 是一个用于解析 XML 数据的 python 库。因为内存占用的问题，无法采用 DOM 方式解析 DBLP 的 XML 数据，所以系统采用流式解析的方式对数据进行处理。这种方法最大的优点就是内存占用小，不会受文件大小的限制。SAX 就是一种典型的流式解析 XML 数据的工具，任何时刻内存中只会保存当前正在处理的 XML 数据。

考虑到本系统使用的 DBLP 数据量太大，采用 SAX 作为 XML 的解析方式无疑是最佳的方式。测试结果也表明，使用 SAX 方式可以在内存不超过 4G 的机器上高效的解析完 DBLP 数据。

2.3.2 Pandas 数据分析包

Pandas 是一个开源的，基于 Python 的数据分析工具包，内置了很多数据结构和方法，可以高效的处理数据。本系统主要使用 DataFrame 数据结构处理 CSV 文件，它方便对数据列进行插入和删除，也可以实现分组聚合和数据转换。

2.3.3 分词工具 NLTK

NLTK 是一个开源的分词工具，常用于 NLP 领域，并且对中英文都适用。它可以很方便的进行中文分词、词频统计等任务。本系统使用 NLTK 工具对 DBLP 文章标题进行分词，去除无用的介词、连词、语气词，只保留标题的关键字；另外，在分词的过程中统计关键字的词频。经过提取的关键词以及词频最终都以属性的形式保存在图中。

2.4 DBLP 数据集

DBLP 是一个包含国际期刊、会议所发论文、知名高校博士毕业论文等数据的文献库，它收录的文献数据极其丰富并且质量颇高，在学术界被广泛认可，有着很高的声誉；更重要的是 DBLP 数据更新及时，紧跟计算机领域的研究前沿方向。出于 DBLP 数据的权威性以及时效性考虑，本文最终决定以 DBLP 的数据集为基础构建研究团体搜索系统，并提供信息检索功能。

2.5 本章小结

本章主要介绍了系统构建采用的相关工具和技术，首先介绍了图数据库 Neo4j[1]以及 WEB 开发涉及的主要工具和设计模式，然后介绍了系统使用的数据处理工具 Pandas 和 NLTK，最后介绍了系统采用的文献数据库。

3 系统需求分析

3.1 系统需求概述

系统需求分析的主要目的是研究用户的痛点和需求，并以此为功能开发的依据。需求分析的质量也决定着软件系统的开发能否成功[11]。需求分析是整个项目的开端，好的需求分析是设计实现一个优秀软件系统必不可少的一步。另一方面，不完善的需求分析会影响后续的开发以及测试等诸多环节，最终可能会使系统无法达到预期要求，更严重的导致整个项目失败，可以说完善的需求分析是实现一个优秀软件系统的必要条件。

3.1.1 系统目标

对于计算机科学领域的研究人员来说，DBLP 是查阅文献资料的有力工具，它提供基础的文献检索功能，如根据文章标题、作者进行查询。一般情况下，这种基于关键词匹配的搜索模式实现的查询功能可以满足文献检索需求；但是，这种搜索模式比较单一无法实现更加复杂的信息检索需求，也无法从大量的学术文献中提取更高阶的信息（如：查找与某论文作者有合作关系的作者）。

很多情况下学术研究人员希望从海量的科研文献中找出更加复杂的信息，而这些信息无法通过简单的关键词匹配得到。在科学研究领域，往往有很多学者对同一个或者同一类问题进行研究；这些学者之间还会有合作关系，多数时候会以合作论文的形式共同发表研究成果；另外，同一个学者可能会涉及多个学术问题的研究，而对于不同的研究问题同一个学者可能会有和不同的合作者。在上述背景下，随着学术研究的发展，各个学术研究领域的研究人员以及不同研究人员之间的合作关系会变得越来越复杂，最终就形成了一个复杂的庞大网络。这个巨大的网络就隐藏在海量的学术文献之中，就 DBLP 数据集来说，这个由研究人员和他们之间的合作关系形成的巨大网络就隐藏在 XML 元数据之中，而 DBLP 现有的数据检索功能无法有效检索这张网络的各种信息。

本系统在 DBLP 元数据的基础上，抽取学术文献信息构建学术网络，网络以研究人员以及他们所发表的文章为节点，以研究人员的合作关系及论文和作者的从属关系为边；进一步，在这个学术网络的基础之上利用图论的相关算法和工具实现对图信息的挖掘与检索。系统主要实现的功能有：研究团体搜索、查询图中节点的中心性（重要性）、查询节点间的相似性、基于图的遍历的多层

次信息查询等。除此之外，本系统还要实现检索结果可视化模块，提高用户使用体验。

3.1.2 用户特征分析

用户特征分析的目的在于弄清系统用户是何种类型的人，有何种特点。以分析为基础，找到特定用户群体的诉求，而用户的诉求就是系统功能设计的核心。一个满足用户诉求的系统会掌控用户，获得良好的用户反馈。举个例子，抖音、短视频之所以如此流行老少皆宜，其根本在于用户特征分析，并以此为基础使用个性化的推荐方法满足不同用户的需求，最后得以广泛传播。

本系统面向的人群主要是高校学生、教师，他们在日常的学习科研过程中需要查阅大量的学术文件，以补充知识、了解各自领域的研究动态。一个研究领域由一个个研究人员组成，本质上是一个关系网络，教师和学生普遍希望从这个学术关系网络中获取丰富的信息。这些信息不仅包括某一具体文献的信息，也包括研究人员之间的合作关系、某个研究人员研究了哪些内容等等。基于上述用户特征，作为一种描述复杂网络关系的数据结构：图，毫无疑问是最佳的实现用户需求的数据结构，所以本系统以图为基础数据结构构建。

3.2 系统可行性分析

可行性分析包括必要性和可能性两部分，任何一个软件系统的开发都要考虑系统的可行性；因为软件系统的开发会受到资源和技术的限制；合理的可行性分析有助于降低系统开发风险，避免不必要的损失。本章主要从技术和操作两方面出发，分析了系统的可行性。

3.2.1 技术可行性

整个项目的实现需要用到不同的技术，不同的技术是否具有可行性，是否可以相互整合，以及不同技术的应用风险都需要通过技术可行性分析来确定。技术可行性分析如果出错会直接导致项目的失败，所以，对现有技术能力进行细致分析，评估可行性风险是系统开发必不可少的准备工作。

本系统数据处理部分的工作采用开发语言是 Python3，以及 Python3 开源扩展包 Pandas、NLTK；数据处理部分采用 IDE 是社区版 PyCharm。Python 是一种简单方便的弱类型脚本语言，学习方便，上手快速，相关工具 Pandas 等也有着丰富详尽的文档，方便学习，所以数据处理部分不存在技术障碍。

系统的存储数据库采用 Neo4j[1]，系统与图数据库主要的交互方式使用 Cypher 查询语言。另外，图数据计算处理部分用到了 Neo4j[1]自带的算法库 GDS；在 Neo4j[1]算法库扩展部分使需要使用 Java 开发 Neo4j[1]存储过程（Procedure）以及 Unmanaged server 扩展。Neo4j[1]是当下最流行的图数据库，官方文档和实例丰富，并且 Neo4j[1]本身也是基于 Java 开发，所以用 Neo4j[1]作为数据存储工具具备可行性。

系统采用 REST 模式设计 API 接口，并使用主流的 Java 开发框架 Springboot 作为后端快速构建工具，其部署方便且内置 MVC 设计模式。后端使用“Spring Data Neo4j”与数据库进行交互。综合来看，Sprigboot 结合数据交互扩展“Spring Data Neo4j”，完全可以胜任系统后端模块的开发。

系统使用 Vue 开发可视化模块，这个框架方便学习，并且很容易与各种第三方库或者项目进行整合，可以方便快速的构建前端页面。系统的前端端静态页面开发采用目前主流的网页开发技术(H5)，经过多年的发展 H5 已经成为成熟的 web 前端开发方案,高效并且可靠。

在实现社区检测扩展功能时，考虑到算法性能，本系统使用 C++开发的算法，并且将 C++版的算法进行适应性的优化与修改使之可以无缝整合到图数据库 Neo4j[1]中。但是，上述方式会引发 Java 与 C++代码如何交互的问题；经过系统调研，最终系统中的 Java 与 C++的相互调使用 JNI 完成。JNI 是 Andriod 应用开发中广泛使用的技术，专门用于 Java 和本地代码（C、C++）进行交互，技术成熟稳定，方便应用于本系统的开发之中。

3.2.2 操作可行性

本系统操作界面清晰流畅，各个功能分门别类容易使用，不存在复杂的页面操作；用户只需要对 windows 系统运用熟练即可快速掌握其使用方法，不存在操作上的困难。

3.3 功能需求分析

3.3.1 作者文章检索

用户在互联网上查找自己感兴趣的某个研究领域的问题时，常常会检索到和问题相关的学术文章；而对于搜索到的学术文章的作者来说，很可能该文章的作者的主要研究领域就是这篇文章所涉及的领域。因此，这个作者的其他学术文章也可能对用户有很大价值。

基于上述应用场景，该功能用于快速通过作者姓名查询给定作者的发表过的所有文章，并且可以支持加上时间筛选条件，用以筛选某个时期的文章。

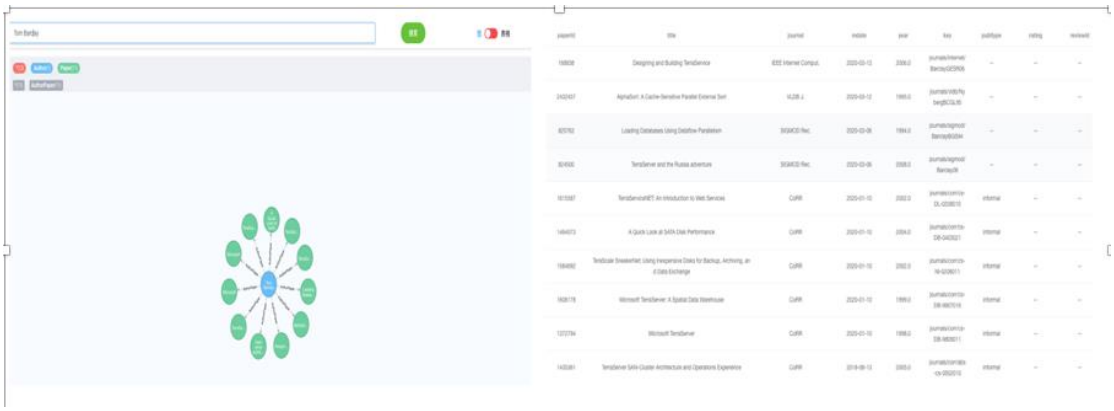

图 3.1 作者文章检索效果示例

3.3.2 作者合作对象检索

一篇学术论文一般有多个作者，这些作者之间是相互合作关系，另外这些作者相互之间的研究领域也一般比较类似；所以，从查询信息的广度上来看，方便快捷的找到一篇论文的所有作者，对整体了解有关学术问题的研究情况有很重要的意义。本功能可以根据文章标题快速查找该文章所有作者并返回，满足上述需求场景的要求。

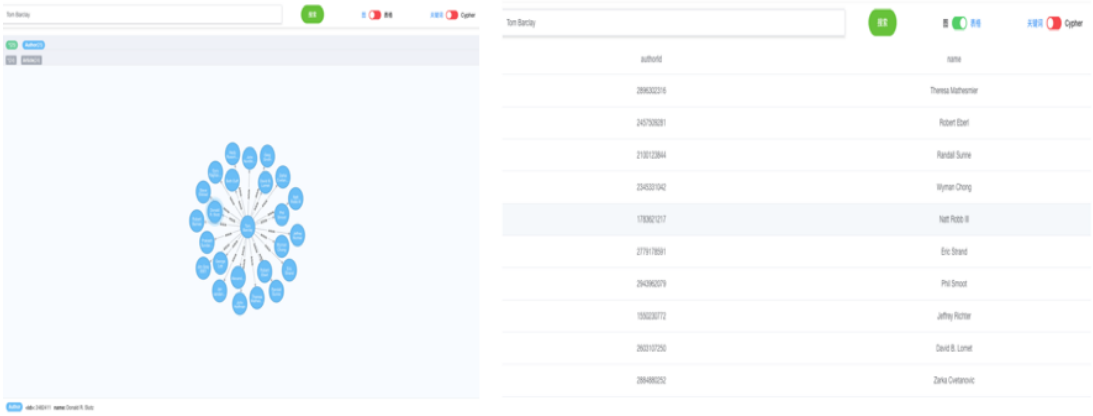

图 3.2 作者合作对象检索效果示例

3.3.3 文章、作者模糊搜索

在一些情形下用户可能无法提供准确的作者姓名或者文章标题，而只能提供不完整的信息，比如部分关键字。在这样的场景下，如果用户仍希望通过部分信息查找文章或者作者，则需要使用模糊匹配来进行查找。

该功能可以根据用户输入的关键字，匹配网络中的一个或者多个节点，返回给用户，可以很好的满足用户对图中节点的模糊搜索需求。

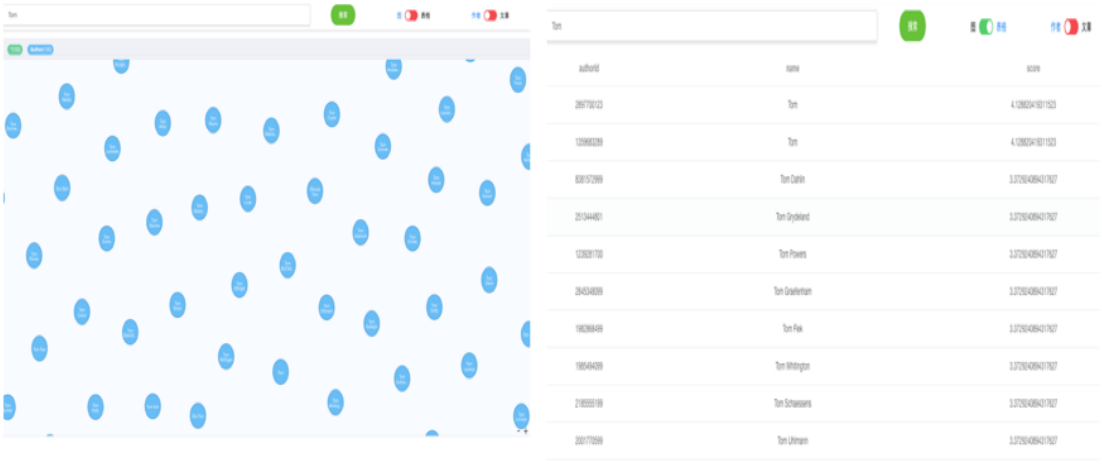

图 3.3 作者、文章模糊搜索效果示例

3.3.4 作者文章关键词搜索

考虑以下场景：用户希望从搜索到的文献中快速识别出论文的大概内容。最简单的方式就是看文章标题，从中提取关键词，这是最简单也是最快的了解文章大致内容的方法。同理，如果将某个作者所发表的所有文章的标题中的关键词提取出来，那么就可以从这些关键词中发现作者大概的研究领域和内容。

该功能通过将前期数据处理时提取的关键词作为图中节点的属性，利用简单的查询就可以快速从节点属性中获取到作者的所有关键词，实现关键词搜索功能。

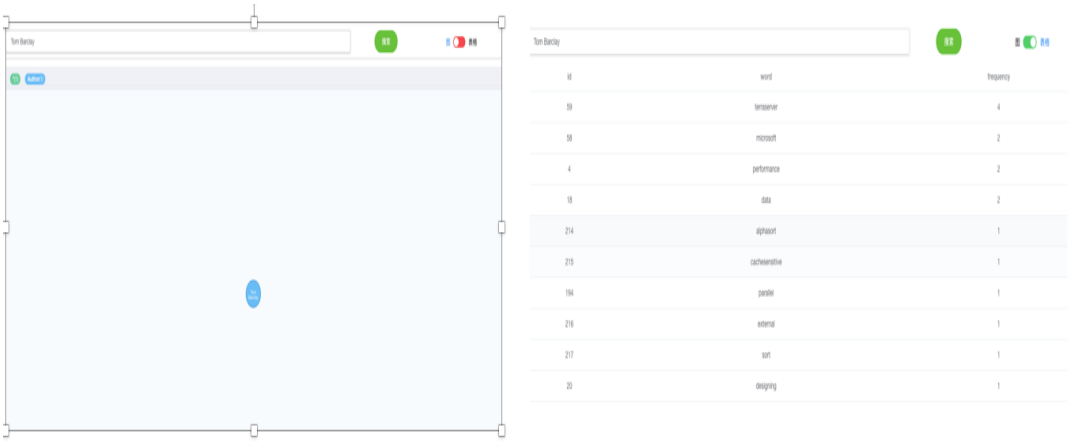

图 3.4 作者关键词搜索效果示例

3.3.5 合作对象文章检索

在科研领域，想要全面了解一个领域的的研究现状，往往需要广泛的调研，查阅大量学术文献，这个过程会花费很多时间，而且还无法保证能够找到足够的有用信息。于是，如何帮助研究学者快速的了解某个学术问题的发展现状，就成为了一个很有必要的工作。一方面可以节省研究人员查找资料的时间，另一方面还可以保证提供的信息的全面性和准确性。

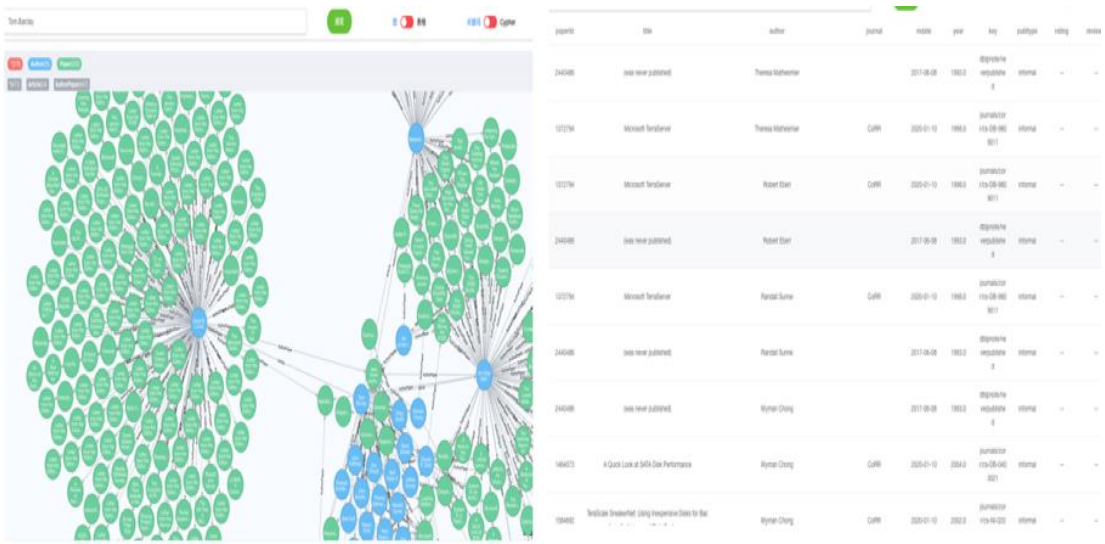

图 3.5 合作对象文章检索效果示例

对于某个特定的研究领域来说，一般会有一个领军学者，这个学者在该领域的有着巨大的影响力。这份影响力还会影响与他有合作的学者，这就可能会出现与领军学者相关联的学者会对同样的学术问题进行研究，并发表学术文章

的情况。因此，找到某个学者所有合作对象所发表的文章，就可以比较方便的对某个研究问题有一个大概的了解。本功能通过给定的作者，快速检索出作者的所有合作对象发表的所有文章，很好的解决了上述问题。

3.3.6 作者相似度查询

在根据 DBLP 学术文献构建的学术网络中，相互关联的作者之间研究的问题总是类似的；在这些有着相似研究领域的学者中，比较这些关联学者在图中结构上的相似度，就可以找出相似度最高的作者，然后通过比较相似度高的学者之间的共同研究内容有助于帮助用户更加准确的追踪研究热点。

本系统通过整合计算节点结构上相似度的算法，来提供图中节点的相似度查询，帮助用户获取更有价值节点关联信息。

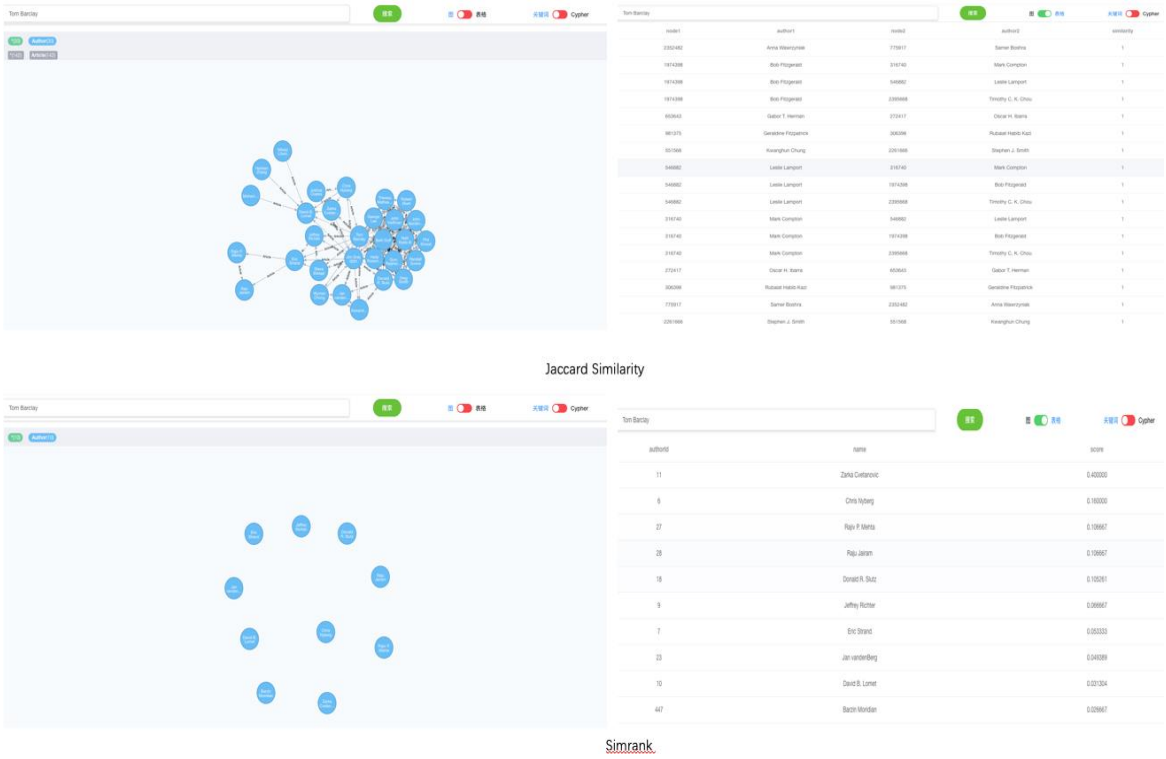

图 3.6 作者相似度查询效果示例

3.3.7 作者中心度查询

用户在查询科研文献时，往往希望查找在某个研究领域影响力最大的学者的论文；一方面可以确保找到的文献资料的权威性，另一方面可以方便的知道

该领域最具有代表性的研究成果。所以，快速找到一组节点之中影响力最大的节点，有助于用户快速获取关键信息。

本系统使用 Neo4j[1]自带的算法库中心性相关的算法计算节点的中心性，并将计算结果反馈给用户，帮助用户找到影响力大的学者。

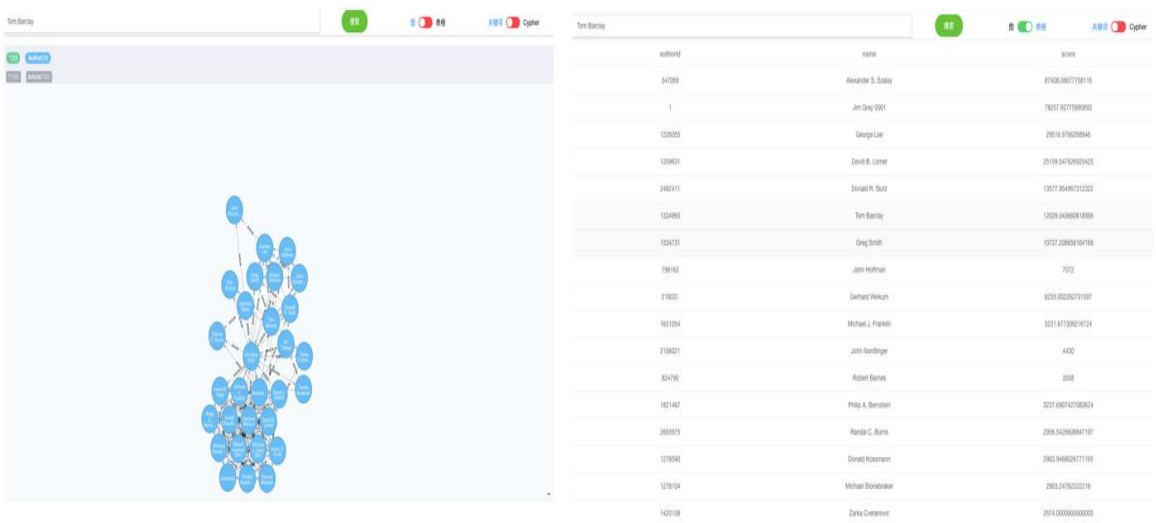

图 3.6 作者中心度查询效果示例

3.3.8 结构上紧密的研究团体查询

DBLP 数据所构建的学术网络中，作者之间合作关系的复杂性导致了图中节点和边的关系也很复杂，在这个基础上必然会形成一个个紧密的研究团体。这样的研究团体出现的基础是学者之间和合作关系，所以研究团体的成员所研究的问题领域也必定是相似的。通过对研究团体的分析，用户很容易找到团体成员，以及成员的学术成果，更能够对研究团体所研究的问题有一个整体的了解。基于上述需求，本功能使用主流的社区检测算法进行图的社区检测，在此基础上开发实现结构上紧密的研究团体搜索功能。

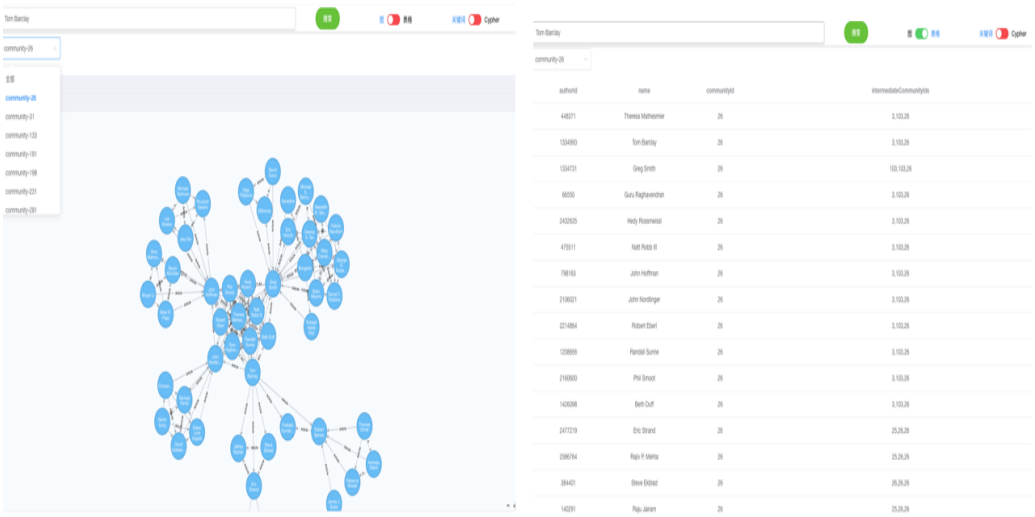

图 3.7 结构紧密的社区查询效果示例

3.3.9 属性紧密的研究团体查询

学术网络中的每个研究人员构成的节点都有若干属性，这些属性来自于其发表过的论文标题中的关键字，代表了该研究人员的主要研究内容。在上一小节中描述了如何从网络中查找结构上紧密的研究团体，但是这样的团体没有考虑团体成员在属性上的紧密性；所谓属性紧密是指团体成员之间有着共同的属性。本系统通过整合相关社区检测算法，来查找不仅在结构上紧密而且在属性上也紧密的研究团体。同时，还可以根据作发表论文的时间，进一步筛选社区中的作者。这个功能在查询出属性密集度高的研究团体的同时，为用户提供多样化的选择。

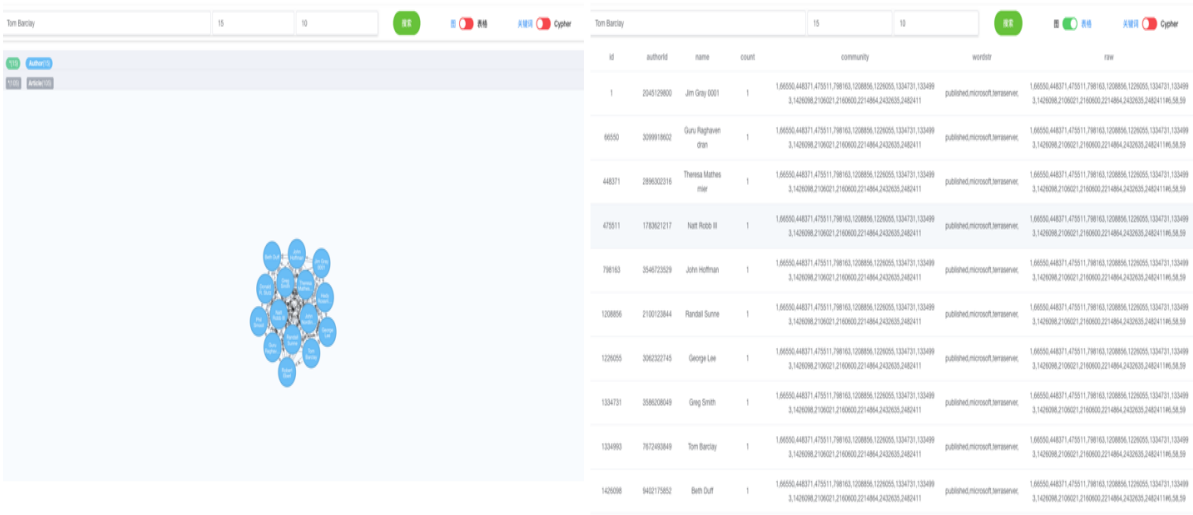

图 3.8 属性紧密的社区查询效果示例

3.4 非功能需求分析

3.4.1 可扩展性

可扩展性是指系统应对未来可能会产生的新需求的能力，如果一个系统在处理新的需求时只需要很少的更改，而无需重构系统，那就说明系统具有较好的扩展性。用户需求的多变性必然导致软件系统的多变性，在一个系统的生命周期中总会有新的需求被不断的提出来，正因为如此，系统的可扩展性就极为重要。但是另一方面也需要对系统的设计进行仔细分析，过分突出设计可能会增加系统的复杂度，导致系统维护成本上升。

考虑到系统的功能扩展，以满足将来的用户需求，本系统的算法扩展采用以 Neo4j[1]存储过程的形式进行整合。当需要新增一种图分析相关的算法时，只需要使用 Java 实现对应的算法，然后将打包后的 Jar 文件注册到 Neo4j[1]，完成这些之后就可以直接通过 Cypher 语法直接调用打包好的算法。另外，系统也支持使用 Java 调用 C 或 C++代码，进一步增加了系统扩展的灵活性。

对于系统后端模块和可视化模块可能的扩展性需求，本文采用前后端分离的方式独立开发和维护这两个模块，二者互不干扰。此外，两个模块之间通过 REST 风格的 API 接口进行交互。前端使用 Vue 框架开发单页面应用，单页面是一个完整的功能对象，方便扩展；后端 API 接口也可以根据需要新增或者修改，同时不影响可视化模块，只需要保证接口数据格式的一致性即可。

综合来说，本系统无论是前后端业务模块迭代，还是算法的新增都具有很高的灵活性，可以满足未来的扩展性需求。

3.4.2 可维护性

软件系统在使用过程中维护的难易程度可以用可维护性来描述，可维护性是软件系统评价体系中的一个重要标准。软件系统的高可维护性可以降低软件使用成本，提高用户体验，对项目的成功有着重要意义。

为了使本系统具有较高的可维护性，在系统设计上，采用前后端分离的方式进行模块化开发。前端可视化模块和后端功能模块相对来说各自独立，可以分别开发；两个模块之间通过事先约定的 API 接口进行交互。由于前后端模块相对独立；所以，在保证接口数据格式不变的情况下，两者之间的开发与维护可以独立进行，不会相互干扰；这也使得系统的维护变得更加简单。

3.4.3 用户界面

用户对系统的体验，最直观的来源于 UI 界面，设计优秀的操作逻辑和观感是实现良好使用体验的基础。本系统采用统一的界面风格样式，页面功能清晰操作简单，没有复杂难懂的功能设计，并且突出核心功能，以此保证良好的用户体验。

3.4.4 软硬件环境

系统使用 Java 开发搜索功能的主要逻辑；数据处理部分使用 Pandas、NLTK 等工具，语言采用 Python；前端模块采用 Vue 实现；Neo4j[1]算法扩展部分使用 Java、C++实现。硬件方面采用 12G 内存以及 1T 的 SSD 硬盘，操作系统为 MacOS。

3.5 本章小结

本章从可行性、功能实现、硬件等方面细致分析了实现系统需要的条件以及要达到的目标，为系统的后续开发打好了基础。

4 系统图算法的应用与扩展

图以及图的算法随着信息技术的发展已经应用到社会生活的方方面面。广义上说，任何使用图这种数据结构进行处理的问题，必然会使用到图的算法，比如查找图中两个节点之间路径、查找一个节点的所有邻节点等；更复杂的图的算法比如节点中心度相关的算法、节点相似度相关的算法、社区检测相关的算法等等。这些算法依据自身特性被用于不同问题的解决。另一方面，近年来图论有关的算法一直在蓬勃的发展之中，不断有新的算法被设计开发出来去解决某些特定的问题；同时，随着硬件和软件技术性能的不断提升，传统的图论算法也迎来了新的发展和应用，各种改进算法和应用领域被不断的提出来。

系统对图的结构、属性进行分析，应用的图算法涉及社区检测、节点相似度度量、节点中心度度量等多个方面，部分算法出自 GDS(Graph Data Science Library)，如中心性算法；另一部分则是通过对 Neo4j[1]进行扩展，将其与不支持的算法进行整合而来，比如：相似度算法 Simrank[12] 、属性图的社区检测算法 ATC[13]。上述算法是整个系统功能实现的核心技术，这些算法在系统中的扩展和应用，使得系统实现对图结构、属性等信息的检索成为可能。

4.1 Neo4j 系统图数据分析算法

4.1.1 中心性（Centrality）算法

中心性是衡量图中顶点重要性的指标，中心性算法可以根据计算中心性方式的区别划分为不同的类别。度中心性[14]算法、接近中心性[15]算法、中介中心性[16]算法是中心性算法的三个主要类别，有各自不同的特点和使用场景。本系统整合了多种中心性算法，用于作者中心性查询功能的实现。

度中心性是指一个点与其他点直接连接的度的总和；它又可以进一步细分为入度中心性和出度中心性。在此方法中，节点的中心性与它的度成正相关。以社交网络为例，某个人在社交圈中的重要性，就可以使用度中心度来衡量。人的中心度与其朋友的数量呈正相关，反应到图中就是节点度的大小。类似的，在社交网络中某个人关注的人的数量和被关注的数量可以用出度中心性和入度中心性来衡量。同理，在本系统基于 DBLP 数据集构建的网络中，也可以使用度中心度计算学术网络中某个学者的影响力。

度中心性只利用了图的局部特征，正因为如此它有一定的局限。在一个网络中，一个节点的连接数多，并不一定代表它处于网络的核心位置。与度中心

性有所区别的是，接近性中心性利用图的整体特征确定节点的重要程度。简单来说，接近中心性与节点到其他节点的距离大小呈负相关，因此接近中心度高的节点距离其他节点的距离也就更短[17]。接近中心性的定义是：某节点到其他节点的最小路径和的倒数值。它在评估信息在网络中传播速度方面有着不错的表现，文章[18]就利用接近中心性实现了对文档关键字的提取。另一方面，在连通图上应用接近性中心性度量的效果要好于非连通图，因为一个不连通的图两个节点之间的距离可能是无限的，这意味着，对于孤立的节点，算法的结果是无限大的分数，这样的结果没有意义。不过也有接近性中心性的变体，如：和谐中心性，可以用于解决此类问题。

中介中心性则常用于在网络中描述连接不同部分的中介型节点，它可以用来计算网络中的节点对网络中信息流动的影响大小。该方式的基本思想类似于找复杂路网中的交汇点，将图看做路网，节点就对应路网中的交叉路口。很显然，通过路口的道路越多，相应的路口在路网中的重要性就越强。对于图而言，节点之间的最短路径就是经过交叉路口的路径，通过衡量经过自身的路径数量就可以间接得到节点的重要性。

中介中心性可以解决很多现实问题，因为其可以评估一个节点在路径上的重要性，所以可以找到中介中心性值高的节点，并且可以认为该节点是网络节点中的瓶颈。在论文[19]中，作者通过找到中介中心性高的岗位来识别犯罪集团中主要的犯罪分子；又比如在论文[20]中，作者利用中介中心性实现推荐引擎，帮助微博用户在 Twitter 上传播自己的影响力。但是，中介中心性也存在一定的局限，它假设网络中的信息的流动路径就是最短路径，且路径选择概率相同；问题在于这个假设在真实的图中并不一定成立。所以中介中心性不能保证找到整个图中最重要的节点，而是提供了一个相对准确的表示方法[21]。

4.1.2 节点相似度（Similarity）度量

图的相似度是图数据挖掘的领域的一个重要度量，图节点的相似度计算常应用于信息检索和文本挖掘中。一个典型的例子就是通过对图相似度进行的度量实现图数据的分类，进而建立数据模型再结合机器学习技术对未知的图数据进行自动识别。图的相似度相关算法依据其采用的数据信息的不同，在计算时间复杂度和准确度方面也有较大差异，不同的算法有着各自不同的适用场景，常见的方法有 Jaccard 相似系数[23]、SimRamk[24]、K-Nearest Neighbors[22]。

网络中两个节点之间的相似度计算主要有两种方式，利用节点属性或者利用节点连接关系。前者将节点属性看做向量空间的一个维度，进而把属性映射到向量空间中，由于一个节点有多个属性，因此节点也可以由属性表示成向量

空间中的一个多维向量。经过上述转换，计算节点的相似度就转化为计算属性向量相似度。然而基于节点属性的相似度度量方法有着一定的局限性，现实的网络中节点的属性可能未知或者缺失，也就无法通过构造多维空间将节点映射为空间中的向量，导致基于属性的度量方法无法使用。

另外一种方案使用节点之间的连接关系来衡量节点相似度，SimRamk[24] 就是典型的基于连接关系的相似度计算方法。节点之间的连接信息实际上是图的结构信息，与属性的局部特性不同，结构信息可以在更大的范围度量节点之间是否相似，不会受单个节点属性缺失的影响。总的来说结构和属性是两个不同的信息维度，对于学术网络而言，图中的连接信息直接反映了学者之间的关联关系，相比属性蕴含了更加丰富的信息。因此，从结构上考虑学术网络节点的相似性有利于获取更多的信息，获取的信息对用户的价值也更大，这正是本系统采用 SimRamk[24]相似度算法的原因。

4.1.3 社区检测（Community Detection）算法

现代海量数据形成的网络结构复杂性不断上升，比如生物蛋白质网络、学术领域的合作以及共同引用等。网络社区是指彼此紧密联系的一组节点，社区内的成员在诸多方面有着相似性。对于学术网络来说，作者形成的社区也就是某个学术问题的研究团体，团体成员有着相似的研究内容或方向。在学术网络上进行社区搜索，对识别、推断作者的研究领域，分析特定研究领域的发展现状有重要作用。

社区检测的算法经过长期的发展，已经出现了不少经典成熟的算法，每种算法都有各自擅长的应用场景，比如：利用标签分组的标签传播算法(Label Propagation)[25]；计算模块度的 Louvain 算法（Louvain Modularity）[26]；以及使用三角形计数(Triangle Count)和聚类系数（Clustering Coefficient）计算关系密度的相关算法等。

直接基于图结构信息的计算方式是最先发展起来的方法。三角形计数(triangle count)算法通过统计节点的三角形数来评估节点之间的紧密程度，基本的思想是：三角形中的任意节点都与其他节点相连，相互之间具有更多三角形的节点之间更有可能构成社区。正因为如此，统计三角形个数是测量网络中节点聚类程度常用方式，此类算法中聚类系数等于统计的三角形个数比上所有可能的关系数。图中节点的聚类系数分为局部和全局的系数，前者衡量节点与邻居连通的概率有多大，后者则是对局部系数归一化求和的结果。聚类系数可以给出随机节点被连接的概率，所以，可以使用聚类系数算法快速得到一组节点或者网络的紧密性；进一步可以通过设置聚类系数的不同阈值来寻找具有不同内

聚性的网络结构或者社区。三角计数和聚类系数在图的分析中应用广泛，如杨长春、俞克非等人在论文[27]中就使用聚类系数对微博社区中博主的影响力进行了研究。此外，基于结构信息的检测方法还有 SCC（Strongly Connected Components）算法，和与之相对的弱连通组件算法(Connected Components)[28]，由 BernardA.Galler 和 MichaelJ.Fischer 在论文[28]中首次提出。后者只要求一个方向上的连通，擅长处理要频繁更新的图，可以快速找到社区的新节点。

标签传播算法（Label Propagation）与三角计数聚类算法和基于 SCC 算法的思想完全不同。它并非直接考虑结构上的紧密性，其基本思想是：在边密集的节点集合中，信息的传播会比边稀疏的区域更快。此算法中在节点之间传播的信息就是所谓的标签，在算法运行结束时，具有相同标签的节点就构成一个社区。算法第一步是对图中所有节点进行唯一标签分配；然后标签在节点之间传播迭代，通过求最大权重更新节点标签；当图中节点都有其相邻节点的大多数标签时，算法就会收敛并结束。在标签传播过程中，紧密节点之间的标签会快速趋同；在算法结束时，按标签的异同划分社区。标签传播算法还可以并行化，所以在图的划分上速度很快。标签传播算法有许多应用场景，比如，利用药物化学特性，找出联合处方药物可能的危险组合[29]；论文[30]使用标签传播算法按兴趣分类检查不同兴趣的社团；微博关系网络可视化分析平台[31]用于关系圈挖掘和可视化分析；类似的还应用于机器学习等诸多领域。

在已经提出的众多社区检测算法中，通过模块度目标函数的优化可以在较快时间内找到最优社区。这类算法既不同于直接测量结构聚类程度，也不同于基于标记的划分；而是定义节点集合模块度，利用模块度来划分节点的归属，通过模块度的大小对节点进行分类，然后通过聚类的强度确定最终的社区。模块化算法首先通过局部聚类形成局部社区，然后再在全局上进行优化，通过在不同粒度的迭代确定不同层次的社区结构。一般来说一个质量高的社区必然是内部节点与外部节点相似度低，内部节点之间相似度高。然而这一类算法存在一定的缺点，一方面算法会将小的社区合并为大的社区，另一方面当迭代过程中有多个类似的候选社区时，可能会形成局部最大值。以整个社区的模块度最大化为优化目标的 Louvain[32]算法是模块化算法的代表。相对其他算法来说 Louvain[32]算法速度更快，适合对点密集但边稀疏的图进行划分。初始时，图中任意节点都被当做独立的社区，然后尝试将节点分别加入到相邻的社区，选取可以使模块度增加最大的社区加入；然后在上述步骤的基础上将小的社区进行折叠构造为超节点重新构建网络，再进行迭代，当所有社区模块度的和不变时算法结束。Louvain[32]算法的特点是可以生成多层次的社区结构，底层的社区划分相对较慢，在进行社区折叠之后节点和边的数目大大减少，算法的速度会提

升。Louvain[32]采用启发式算法，在一般的精确的模块度算法受限的应用场景，该算法有着不错的表现，如分析复杂网络的结构。值得一提的是，Louvain[32]是针对无向图的，尽管存在处理有向图的模块度算法（如：directed Louvain[33]），目前比较成熟并且被广泛应用（比如：大范围社区检测[34]）的算法，包括 Neo4j[1]支持的模块度算法都是面向无向图的。

Louvain[32]算法在图的社区检测方面有大量的应用，如文章[35]中使用利用 Louvain[32]算法检测移动用户网络中的社区，通过分析社区研究用户的行为；又如 D. Meunier 等人在论文[36]中利用 Louvain[32]可以得到层次化社区的特点，研究分层的网络结构在脑功能网络中的特点。

在本系统构建学术网络中，整合了对结构和属性都密集的社区搜索算法，但是社区要同时满足结构和属性都密集的要求，就必然会损失一部分结构上的信息，这不利于用户全面的获取图中研究团体的信息。同时在学术研究领域，同一个领域可以细分为不同的小的领域。这个特征反应到本系统构建的学术网络中就是：网络中的研究团体具有层次性，不同的层次代表着研究领域的细分程度，也就是学术网络中会有不同层次的社区结构。考虑到 Louvain[32]算法对发现层次社区方面的独特优势，本系统也集成了使用该算法实现的社区搜索功能，一方面考虑了充分获取学术网络中研究团体结构信息的需求，另一方面也实现了对不同层次的研究团体的检测，对完善系统的功能具有重要意义。

4.2 Neo4j 算法扩展方案及实现

4.2.1 Java Native Interface(JNI)

JNI 是“Java 本地接口”的简称，其主要的作用的是实现 Java 代码对 C/C++代码的调用，此外，它也使得 Java 可以与其他编程语言互相操作。本系统使用的扩展算法，大都使用 C++编写，但同时系统开发的 Neo4j[1]存储过程使用 Java 编写，所以存在 Java 代码和 C++代码相互调用的过程。本系统使用 JNI 技术解决上述问题。

具体步骤是：1、先在 Java 端注册 native 修饰的方法，确定好方法参数以及返回值；2、使用 javac 编译上述代码获得 native 方法签名（头文件）；3、用 C/C++代码实现步骤一中注册的方法，并用 C/C++实现要扩展的图算法；4、将 C/C++代码编译为动态链接库，并且将动态链接库放到 Java 扩展库目录并注册。经过如上四个步骤后，在 Java 代码中可以通过调用 native 方法，实现对 C/C++代码的调用。具体流程如图 4.1 所示：

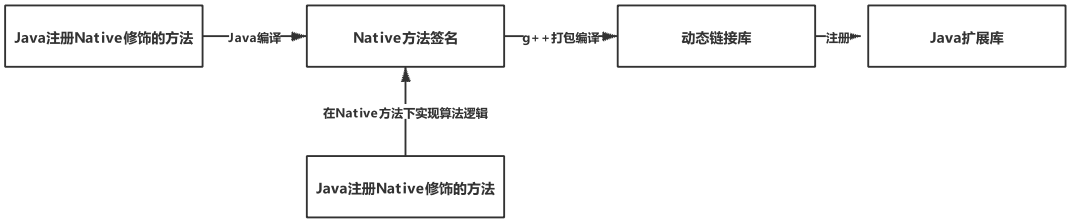

图 4.1 算法扩展执行流

在算法的二次开发中，首要任务是定义 native 方法的输入输出，保证 Java 部分代码和 C/C++代码正常对接。native 方法可以看做是一个通信接口，定义了两部分代码的交流格式，它的参数表中包含算法的输入数据（子图数据、节点数据），还包含了控制算法行为的参数（如：索引类型）；它的返回值就是算法的计算结果。在 Java 代码层面，系统使用用户输入的节点信息，通过遍历算法从图数据库中查询节点数据，再构造与输入节点相关的子图或者其他相关信息。

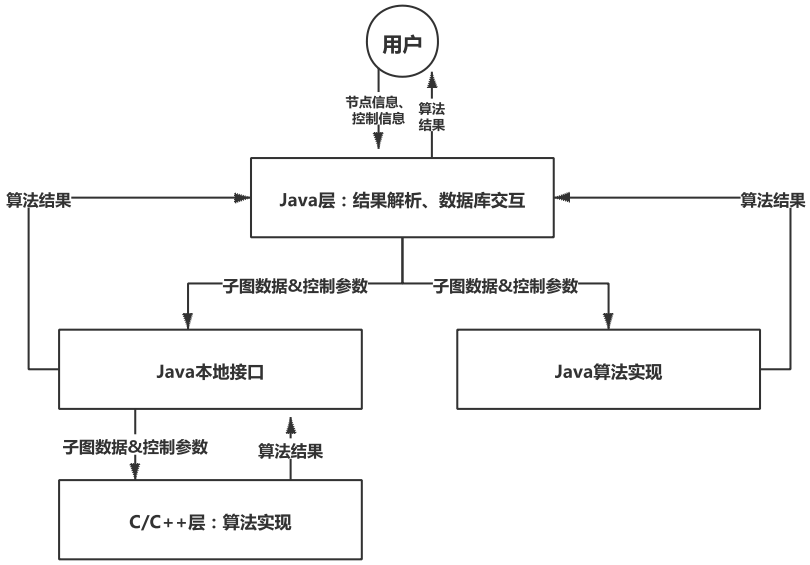

图 4.2 算法扩展执行流

系统将得到的图数据集作为 native 方法的一个入参，同时也从用户输入中个获取用于控制算法行为的参数；然后，两类参数通过 native 方法传递到 C/C++代码层。C/C++算法代码根据输入做适应性开发后可以直接利用 Java 层传递的数据和参数进行计算法，算法完成之后，其计算结果通过 native 方法返回

到 Java 层，最后 Java 层代码对结果进行进一步解析并提供给可视化模块进行展。示。具体执行流程如图 4.2 所示。

4.2.2 SimRank 算法扩展

SimRank[12]是利用图结构信息衡量节点相似度的代表性算法之一，它广泛应用于广告推荐，文本匹配等领域。在 SimRank[12]算法的框架中，节点的相似度取决于其邻居节点的相似度。基于邻居相似的两个节点也相似这一思想，SimRank[12]可以利用图的结构信息衡量节点之间的相似度；进一步可以递归考察多重邻居的相似性使结果更加准确。

因为学术网络结构信息相对重要，所以本文使用 SimRank[12]算法结合图的结构信息计算学者之间的相似度。在由 DBLP 数据集构建的图中，对于任意一个作者节点，所有与它相邻的节点（作者）都与其有着共同的研究内容或研究领域；同理，它的二阶邻居（从自身出发遍历深度为二可以到达的节点）与它的直接邻居也有着相同的研究领域或内容；以此类推，随着距离的增加，节点（作者）之间的研究领域的区别会越来越大。根据上述推断，节点之间结构上的相似度比较，只在距离上相隔较近的节点之间才有参考价值，上述节点距离使用图中节点之间的最短路径长度衡量。

系统根据客户端给定的节点，使用遍历节点的邻居节点，遍历深度为 2；以遍历到的节点和边构建子图。构建子图的步骤由 Java 实现。然后，定义 Java 本地方法，确定输入和返回参数，输入参数包括子图数据。接下来生成 Java 本地方法的签名，并用 C/C++代码实现该方法的真正逻辑。在用 C/C++代码实现 native 方法之前需要用 C/C++实现 SimRank[12]算法，并进行二次开发，确保可以正确接收 Java 层的子图数据并正确返回结果。

上述 C/C++（包含修改过后的 SimRank[12]和 native 方法）码实现后，将此部分代码编译为动态链接库，然后将动态链接库放置到 Java 扩展目录，方便 Java 调用。Java 部分的代码则被打包为 Jar 文件，放置于 Neo4j[1]存储过程指定目录，并在 Neo4j[1]配置文件中进行注册。当上述步骤完成之后，SimRank[12]算法的扩展就正式完成，重启 Neo4j[1]之后就可以在 Cypher 语句中直接使用 SimRank[12]算法技术节点之间的相似度。

4.2.3 ATC 算法扩展

真实网络中的节点有着丰富的属性信息，这样的网络也就是属性图。本系统所构建的学术网络就是一个典型的属性图，图中的 Author 节点包含 word（关键字）、articles（文章）、name（作者姓名）等多个属性。对于研究人

员而言，有着相似研究领域的两个人的关键字属性必定是相似的，也就是两者之间的关键字会有很多重合。因此，找到不仅结构上密集而且关键字属性也密集的社区，就可以更加精确的定位有着相似研究领域的科研人群。

为了解决上述需求，本文中需要实现对属性图的社区检测，保证找的社区在属性上也是紧密的，然而一般的社区检测算法只能满足结构上的密集性，所以需要一种支持属性图社区检测的算法实现上述功能，而 ATC[13]算法正是用于属性图上结构和属性都密集的社区的搜索算法。

考虑到属性密集社区搜索的前提是节点必须有属性，所以在数据处理阶段，系统通过 NLTK、正则匹配等方法对文章标题进行分词，去掉非关键的介词、连词等。由于属性是单词，单词之间直接进行匹配会有性能问题，所以，系统对所有关键字统一编号建立索引，并且将关键词及其索引编号一同写入节点属性，提高后续节点属性匹配的性能。

ATC[13]算法的基本思想是枚举所有的关键字组合，然后检查包含这些公共属性的 k-truss（每个边都至少包含 k-2 个三角形的子图）社区是否存在。由于关键字的组合数太大以及图节点和边的数量众多，会严重影响算法的速度，ATC[13]采取了一系列方法提高算法性能，如：FP-Growth 发现属性频繁项集、构造原图索引提高 k-truss 查找效率等。本系统扩展 ATC[13]算法的步骤与 SimRank[12]类似，关键在于构造包含用户查询顶点的子图，以及修改 ATC[13]算法以适配 Java 存储过程的输入输出。另外，需要注意的是 ATC[13]算法的返回结果中可能会包含多个符合条件的社区，Java 层需要对多个社区的情况进行处理，确保用户可以看到完整的算法结果。

此外，在研究团体搜索的基础上，通过增加以论文时间为筛选条件的方式，进一步筛选研究团体中的节点，通过这种方式搜索到的研究团体中的所有成员都必定是在某一个时间之后发表过论文的学者。这种具有时效性的研究团体搜索更加能够满足系统用户对于信息时效性的需求。

4.3 本章小结

本章主要介绍了系统实现用到的相关算法，包括中心性算法、相似性算法、社区检测算法。同时，详细描述对 Neo4j[1]算法的扩展过程，主要是 Simrank[12]以及 ATC[13]两种算法的扩展整合。

5 系统总体设计

5.1 数据处理

5.1.1 数据采集

数据采集是提取利用数据的开端，也是本文工作的基础。在这一阶段收集到的数据为原始数据，包含全部的信息，一般情况下需要进一步处理以便获得价值密度更高的数据。本文原始数据来源于 DBLP 官方数据源（约 3.05GB），官方提供的数据是 XML 格式，其中包含了各种期刊会议的文章以及作者信息，此外还包含部分硕士博士论文。本系统所用的图主要通过抽取学术期刊文章的信息构建。

5.1.2 数据预处理

在数据采集的过程中得到的原始数据，往往无法满足数据分析的需求。数据预处理是提高数据质量的必要步骤，可以保证数据的正确性、可用性，还能解决脏数据、数值错误等问题。本系统在数据预处理阶段主要对原数据进行转换，使之满足 Neo4j[1]的存储要求并且为后续算法分析提供必要的信息。

本文从 XML 数据中抽取学术期刊、会议论文部分的内容同时过滤不需要的信息；另一方面靠考虑到 Neo4j[1]不支持 XML 格式的数据导入，所以在预处理阶段将清洗出来的数据使用时下流行 CSV 格式进行存储。由于原始 XML 数据太大，对于普通配置的机器而言不适合一次性将全部数据加载进内存进行解析；同时，也考虑到系统的开发成本，本系统采用 SAX 的模式进行逐行数据解析。在 SAX 模式下，无需将全部数据读入内存，只需要逐行读取 XML 数据，因此内存占用很少，处理 XML 文件的大小没有限制。该模式的缺点在于解析时间相对较长，但是结合开发成本等因素该缺点在可以接受的范围内。

XML 解析过程如下：首先逐行读取数据，当匹配到 article 标签，就读取其子标签 author、title、journal、year、url 的内容；然后，去除标签内容中的无效字符，如果内容缺失则使用空白字符串加以代替，当匹配到 article 结束标签时，当前文章的信息解析完成。当文件中的所有标签都被访问之后，整个文件解析结束。XML 解析完成之后生成一个每一行代表一篇文章，包含 article_id、author、title、journal、year、rating 等字段的 CSV 文件。

表 5.1 article 标签保留字段

字段	含义
author	文章作者姓名
title	文章标题
journal	文章发表的期刊
year	发表时间
rating	引用次数

5.1.3 数据处理与分析

本系统的数据分析基于图这种数据结构，图中的节点分两类，一类是 Author 代表文章作者，包含 author_id、articles（所发表过的全部文章）、words（文章标题中出现过的关键词以及词频），一类是 Article 代表文章；图中的边也分为两类，一类是作者之间的边代表两个作者之间有合作关系，边的属性 weight 代表两者合作过的文章数量，另一类边则是 Author 节点与 Article 之间的边，代表文章与作者之间的从属关系。要实现上述图数据模型，需要对数据预处理部分生成的 CSV 数据进行进一步处理。

```
<article mdate="2019-10-25" key="tr/gte/TR-0310-11-95-165" publtype="informal">
  <author>Frank Manola</author>
  <title>Integrating Object-Oriented Applications and Middleware with Relational Databases.</title>
  <journal>GTE Laboratories Incorporated</journal>
  <volume>TR-0310-11-95-165</volume>
  <month>November</month>
  <year>1995</year>
  <url>db/journals/gtelab/index.html#TR-0310-11-95-165</url>
</article>
```

图 5.1 article 标签示例

首先，生成 Author 节点，具体步骤如下：

（1） 找到每一个作者发表过的所有文章，用来生成 Author 节点的 articles 属性。一般来说可以使用 HashMap 作为容器存储作者的文章，但是此种方式需要遍历整个 CSV 文件，由于数据量太大，Java 堆内存不够容纳所有数据，因此这种方式实际上并不可行。为了解决此问题，本文将原 CSV 文件拆分成 100 个小文件，但同时必须保证同一个作者的所有文章必须出现在同一个小文件中。首先将作者的名字转化为一个 1

0 位的整型 hashcode 值，然后用 hashcode 对 100 取模，得到的值就是该条记录应该存放的小文件文件名。

（2） 注意，对一篇文章有多个作者的情况，则会生成多个 Author 节点，同时通过组合的方式得到这多个作者之间的合作关系。

（3） 在上一步生成的小文件的基础上，遍历每一个小文件，统计出每一个作者发表过的全部文章，并且 hashcode 作为作者唯一的 author_id。另外，在遍历过程中使用 NLTK 对文章标题进行分词操作，提取关键字和词频。每一个关键字都有一个全局的 ID，这 ID 通过遍历数据预处理阶段生成的 CSV 文件，并对文章标题进行分词操作得到。通过以上三个步骤就生成了一个包含所有 Author 节点信息的 CVS 文件。

其次，生成 Author 节点之间的边，步骤如下：

（1） 对于一篇文章多个作者的情况，任意两个作者之间都有合作关系，因此通过组合得到所有边的关系，每一条边都包含两个端点，分别都代表一个作者。同样的，使用两个端点的 hashcode 生成每一条边的 10 位 hashcode；边的 hashcode 对 100 取模则得到这条边要存储的文件名。

（2） 遍历所有保存边的关系的小文件，统计每一个边的 hashcode 出现的次数，就得到了任意两个作者总共合作过的次数，也就是边的 weight 属性。

（3） 最后，生成 Article 节点文件以及 Article 的边，具体步骤如下：

（4） 由于数据预处理节点生成的 CSV 文件，每一行就代表一篇文章，所以，只需要遍历该文件，提取每一行需要的字段就可以生成 Article 节点文件。

（5） 然后，在遍历 Author 节点文件，一个作者如果有多篇文章那么就生成多条，Author 节点与 Article 之间的边，遍历完成之后便得到了 Artile 节点的边文件。

在上述数据处理步骤完成之后，得到了以 CSV 文件格式存储的一张学术网络图，因为 Neo4j[1]支持 CSV 数据格式文件的直接导入，所以很容易将上述图数据直接导入 Neo4j[1]数据库进行存储，在导入过程中还需要建立合适的索引以加快图的构建过程。本文后面的工作将基于 Neo4j[1]中存储的图数据展开。上述数据处理步骤如图 5.1 所示：

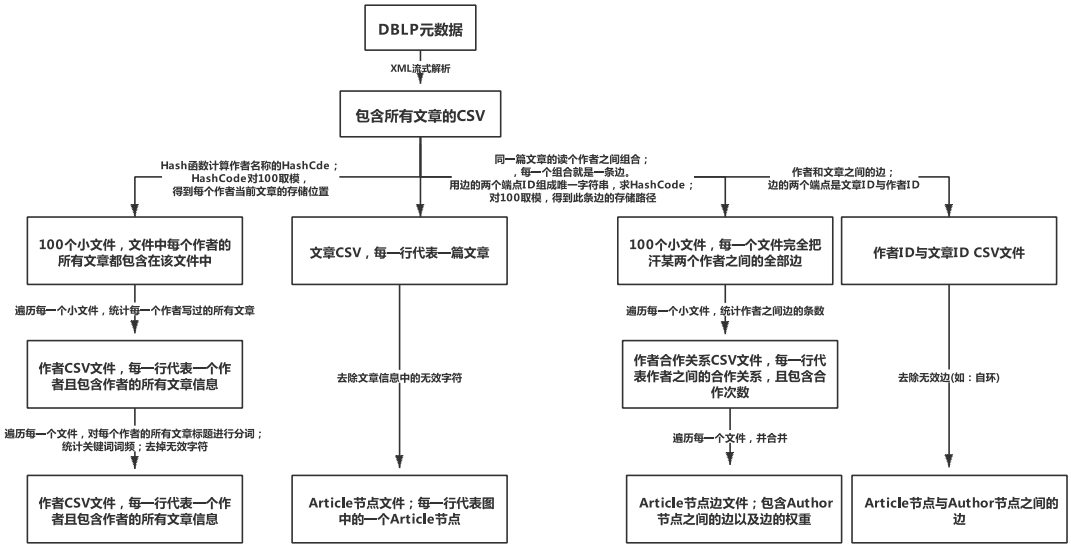

图 5.2 DBLP 数据处理流程

5.2 前端可视化模块

目前主流的前端开发思想是组件化开发，整个项目由各种相对独立的功能组件搭建起来，方便开发维护。本系统也采用组件化开发，不同的功能由各种功能的组件组合而成。前端架构如图 5.1 所示：

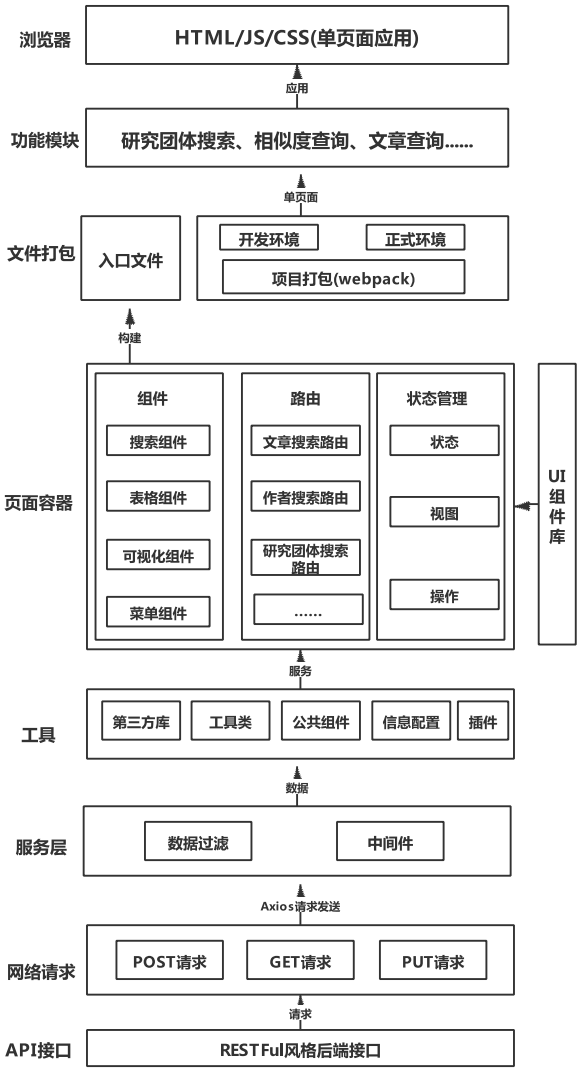

图 5.3 前端总体架构

5.2.1 模块设计

系统可视化模块页面构建划分为两层，首先是页面层，主体页面包含基础查询页面、中心性查询页面、相似度查询页面、研究团体搜索页面。基础查询页面包含基本的文章检索、作者检索等功能；相似度查询页面主要实现作者相似度查询；中心性查询页面实现作者中心度查询；研究团体搜索页面负责实现研究团体搜索功能，包括结构和属性上同时紧密的研究团体。

页面层的下面一层是组件层，所谓组件是指一个独立的页面功能单元，比如搜索按钮、输入框等。组件层主要包含以下组件：1、查询组件，包含搜索按

钮和输入框，负责收集用户输入信息和发起查询请求；2、原生图结构可视化组件，负责将查询结果渲染成图并展示；3、表格组件，负责以表格的形式展示用户查询结果，这类结果无法以图的结构展示，只适合用表格形式展示。组件层相互组合，共同构成页面层不同的功能页面。

5.2.2 接口设计

系统根据页面层不同的功能页面设置不同的接口路由，总体上分为基础查询的接口、相似度查询的接口、中心度查询的接口、研究团体查询的接口。根据路由的不同，页面层的不同功能页面放在各自的文件夹下。这种方式方便路由管理和前端匹配请求页面。

另外，接口设计很重要的一个方面是同步数据的规范，本系统根据不同的功能设计了不同的接口数据字段，比如中心度的查询功能数据接口包含的字段有 author_id、author_name、centrality 等。前后端功能独立实现，不需要考虑交互问题，只要 API 接口定义明确，就可以保证前后端可以正常的工作，节省了大量的联调、测试时间，也降低了前后端代码的耦合度。

5.3 后端业务实现

5.3.1 业务分层

本系统采用分层架构进行后端业务逻辑的实现，主要目的在于各个业务层的职责分离，实现业务层之间的松耦合、高内聚。系统后端业务层主要分为四层：1、domain 层：负责构建 Java 对象和图中实体的映射关系，图中的节点和边这两类实体映射为代码中的 Java 对象，同时，Java 对象中的属性也对应实体的属性。domain 层的主要作用是向上层传递数据源对象；2、resposito ries 层：负责实现具体业务逻辑，用于查询、更新、修改图数据库中的数据，是数据层的抽象。3、controller 层：控制层，负责解析前端的请求并将处理的结果返回。4、support 层：负责封装 controller 层通用的工具和方法，将共用的代码从 controller 分离，比如将数据对象转化为 json 格式。support 层简化了 controler 层的业务逻辑，也提高了代码的复用性。系统最底层是数据库以及算法扩展层，提供最基础的数据存储与处理功能。系统后端分层结构如图 5.3 所示：

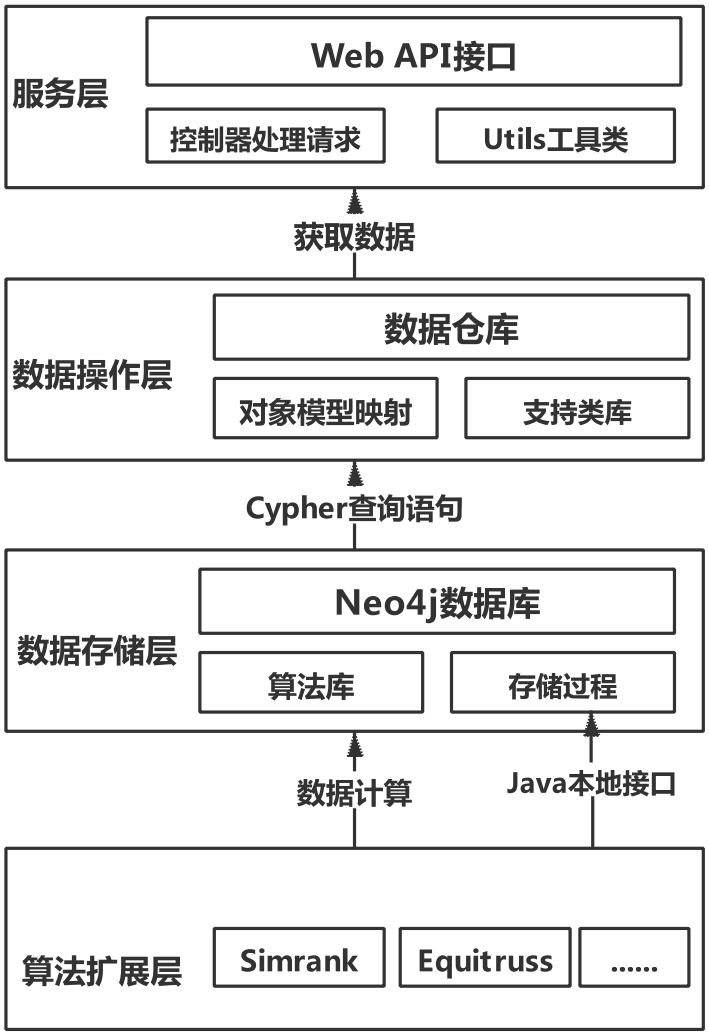

图 5.4 后端总体架构

5.3.2 存储过程实现

存储过程是数据库中的复杂存储程序，它通过封装特定的查询逻辑实现相对复杂的功能，所谓复杂是指通过简单的查询语句无法实现的功能。存储过程也是数据库层面的代码复用。

Neo4j[1]支持使用存储过程对其功能进行扩展。由于本系统的研究团体搜索功能需要使用复杂的社区检测算法，但是 Neo4j[1]本身并不支持，在这种情况下就需要使用存储过程来扩展 Neo4j[1]的功能，以便直接通过 Cypher 语言调用存储过程执行算法。系统存储过程使用 Java 开发，并将存储过程注册到 Neo4j[1]，实现算法的扩展以及 Neo4j[1]查询语句 Cypher 的功能扩充。

5.3.3 系统功能实现

本系统的功能大致分为三类，分别是基础查询功能，包括作者文章检索、作者合作者对象检索、文章&作者模糊搜索；相似度查询，主要是作者相似度查询；研究团体搜索，包括结构密集的研究团体搜索和结构属性都密集的研究团体搜索。

基础查询功能的实现：首先获取用户查询关键字，前端发送请求，请求参数就是用户输入的关键字；后端控制器收到请求，通过解析请求参数获取关键字，再用关键字构造 Cypher 查询语句并执行。Cypher 语句执行的结果由后端返回，最后通过前端可视化模块展示给用户。需要注意的是，用户的输入可能会有非法的或者无效的字符，如果不加处理直接使用原始的输入信息构造 Cypher 查询，会有查询失败和安全性风险。因此，系统在处理用户输入的时候，需要替换输入参数中的无效字符（比如：空格、换行等）和非法字符（主要是 Neo4j[1]中的保留字），确保 Neo4j[1]最终执行的 Cypher 查询是合法有效的。系统主要采用两种方式处理输入，一种是建立 Neo4j[1]保留字的字典，通过字典匹配输入是否包含保留的关键字；另一种是通过正则匹配替换。两种方式配合使用，完成对于输入的过滤操作。

作者相似度查询功能的实现：图节点的相似度查询需要借助于图的相似度算法，图有多种相似度算法，本系统主要采用使用图结构信息的相似度算法 SimRank[12]计算节点的相似度。首先系统从用户输入中获取用户需要查询的节点，后端获取节点后遍历节点所有邻接点以及邻接点之间的所有边、自身与邻接点之间的所有边，通过遍历的结果构造一个子图。得到查询节点相关的子图后，调用事先开发完成的相似度计算存储过程，计算所有节点之间的相似度，最后将结果降序排列返回到客户端。注意，之所以选择查询节点相关的子图，是因为相邻节点之间的相似度比较才有意义，相隔太远的节点（作者）之间的研究领域相关性很低。如果用户输入的节点没有邻居或者输入的节点不存在需要单独进行处理。这种情况下用户输入不会引发算法的调用，而是直接在后端接口层面通过 Cypher 查询结果进行判断，对于无效的节点，返回空的结果。这样一方面可以提高搜索结果的响应速度，另一方面可以降低不必要的计算资源消耗。

研究团体搜索系统的实现：结构紧密的研究团体搜索实现与基础查询功能实现方法类似，通过调用 Neo4j[1]自带的算法库即可，下面主要介绍属性和结构上都紧密的研究团体搜索如何实现。该功能主要使用社区检测算法 ATC[13]实现，算法会返回查询到的社区以及社区中所有节点的公共属性。首先通过用户

输入获取用户想要查询的某一作者，然后遍历该作者的所有邻接点，遍历深度为 2，以遍历到的数据构建子图；在子图的基础上调用 ATC[13]的存储过程，在子图上运行 ATC[13]算法搜索社区。另外，研究团体搜索功能还可以使用时间进一步对筛选社区中的节点，保证查询到的社区中的作者一定是在某个时间之后发表过文章。时间筛选功能使得查询的社区更加具有参考价值。同样，对于无效的输入，系统也进行了单独处理。

5.4 本章小结

本章详细阐述了系统的总体设计与具体实现。首先介绍了系统数据处理部分，如何将原始 DBLP 数据集转化为图数据；接着介绍的前端可视化模块和后端业务的具体实现以及业务架构；最后，介绍了系统对 Neo4j[1]的算法扩展，包括相似度算法 SimRank[12]以及社区检测算法 ATC[13]的扩展实现。

6 环境搭建与系统测试

软件测试是必不可少的开发环节，是保证软件质量的重要方法[37]。软件测试主要作用是验证软件是否符合需求设计和能否正确运行，对发现系统缺陷、评估系统质量具有重要作用。同时，测试结果对系统的可维护性、可扩展性也具有积极的参考意义。

本文对系统的测试主要是对各个功能需求进行单元和功能测试、性能测试，以验证研究团体搜索系统是否实现最终的设计目标。

6.1 实验环境搭建

本系统的实验环境配如下：

表 6.1 实验环境

环境	属性值
Operating System	MacOS 10.15.5
CPU	1.99 GHz 四核 Intel Core i7
Memory	SK Hynix 4 GB 2400 MHz DDR4、Samsung 8 GB 2400 MHz DDR4
Hard Disk	SSD 512GB
Graphics Card	Intel UHD Graphics 620 1536 MB
Software Environment	IntelliJ IDEA 2020.1.2、IntelliJ PyCharm 2020.1.2、IntelliJ CLion 2020.2、Neo4j-community-4.2.1、JDK11、Python 3.7.3、Clang11.0.0

6.2 单元测试

单元测试是软件开发过程中避免 bug 的第一道防线，对确保软件质量至关重要。单元测试一方面可以降低 Bug 数量，另一方面能够有效的发现代码修改造成的问题，确保对问题及时进行修复。本系统采用基于 Junit 发展而来的单元测试框架 Neo4j-harness，Neo4j-harness 是对 Neo4j[1]应用进行开发测试的专用框架。

本文针对每一个功能模块分别编写测试用例，对复杂的功能进一步拆分成若干模块，并且对每个小模块编写测试用例，再对每一个用例进行测试。首先，创建 Neo4j[1]的 Mock（模拟对象）模拟 Neo4j[1]的行为，并在对象中创建一个包含一定数量节点的测试图；系统功能代码中所有对 Neo4j[1]的操作都可以用 Mock 对象模拟。

系统对每一个后端功能模块，如作者文章检索、作者合作对象检索、作者相似度查询以及研究团体搜索分别编写测试用例；对代码执行时间、执行异常等分别进行测试；同时使用 Junit5 的新特性：参数化测试，用不同参数对同一个测试用反复执行。

本文通过对每一个功能单元的测试用例反复执行，发现了系统编码的若干问题，并及时加以处理。最终，系统成功通过所有测试用例，保证系统不会出现明显的功能性 Bug，为系统的稳定运行扫清了障碍。

6.3 功能测试

功能测试也叫黑盒测试，测试过程不关心功能内部的具体实现，只验证系统功能是否达标，只需要设计好输入，再验证系统输出是否符合预期。与单元测试不同，功能测试只关心系统整体运行是否正常。黑盒测试模拟用户对系统的使用，逐一验证系统的所有功能，决定了系统最后能否上线运行。在设计测试用例时，不仅要考虑正常的输入输出情况，也要设计异常的输入输出情况，比如输入图中不存在的节点测试系统反应。通过详尽的测试用例设计，尽可能的暴露系统可能存在的问题。

表 6.2 系统功能测试用例表

功能模块	测试流程	测试结果
作者文章检索	在客户端输入要查询的作者名称，期望返回该作者所有文章的标题；同时测试输入值为随机字符串和空值的情形。	当输入作者名称存在时，正确返回作者文章列表；当输入值不正确时，系统返回值为空。
作者合作对象检索	在合作对象检索页面分别测试输入作者名称和无效字符。	当输入的作者名称存在于图中时，系统正确返回该作者所有邻接点；当输入无效时系统返回值为空。
作者、文章模糊搜索	在模糊搜索页面，以不同作者名称、文章标题的部分内容作为输入值。	系统正确返回匹配结果，返回的结果中包含多个模糊匹配到的作者和文章。
作者文章关键词搜索	在图中随机选取作者节点，以作者名称作为输入；或者以随机字符串作为输入。	对于正确的作者名称输入，系统返回作者关键词列表且包含关键词词频；当输入无效时，系统返回值为空。
合作对象文章检索	在合作对象检索页面，随机选取图中的 Author 节点作为输入；同时测试随机字符串输入。	当输入的作者姓名存在于图中且该作者有合作对象（有邻接点）时，系统返回所有合作的对象的文章.
作者相似度查询	作者相似度查询需要用到相似度算法的存储过程，同时查询结果与节点在图中的结构上下文密切相关。测试以处于不同结构上下文（边密集或者稀疏）中的节点为输入。	根据输入节点的不同，系统成功返回相似度排序前 10 的节点；当输入无效或者输入节点无邻接点时，系统返回为空。
作者中心度查询	随机选取图中结构上下文相差较大的节点最为输入；同时测试无效输入。	对与有效输入系统正确返回节点中心度；输入无效，系统返回值为空。
结构和属性都紧密的研究团体搜索	属性图上结构属性紧密的研究团体搜索需要借助算法 ATC[13]。以不同的节点和不同公共属性个数的不同组合作为输入；同时测试无效输入。	输入有效，当存在符合要求的社区时，系统成功返回社区节点集；对于同一个节点但公共属性个数不同的查询，系统返回不同的社区，且社区中的公共关键字符合输入条件；输入无效，系统返回值为空。

6.4 有效性和性能测试

6.4.1 属性紧密社区搜索算法有效性测试

本小节采用论文[13]实现的 ACCore、KIndexDec、NCTruss 三种算法在 DBLP 数据集上分别测试其搜索到的社区的属性紧密度。这三种算法分别是基于 k-core 和 k-truss 的属性密集社区搜索算法和只考虑结构紧密性的搜索算法。此外，本小节采用方法 CMF[38]和 CPJ[39]作为社区的度量方法，其值的范围是[0,1]，并且和社区属性紧密程度呈正相关。

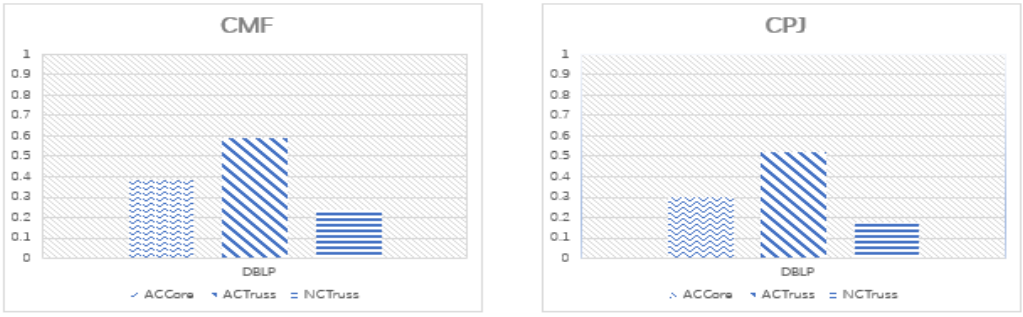

图 6.1 社区属性紧密性

测试结果表明基于 k-core 的算法 ACCore 和基于 k-truss 的算法 ACTruss 所得到的社区相对于只考虑结构紧密性的 NCTruss 算法具有更好的属性紧密性，同时也说明本系统采用的属性社区搜索算法可以有效搜索属性紧密的社区。

此外，通过统计系统搜索到的社区中节点和边的 truss 值均值，本文验证了属性社区搜索算法所得社区在结构上的紧密性。算法测试结果如图 6.2 所示。

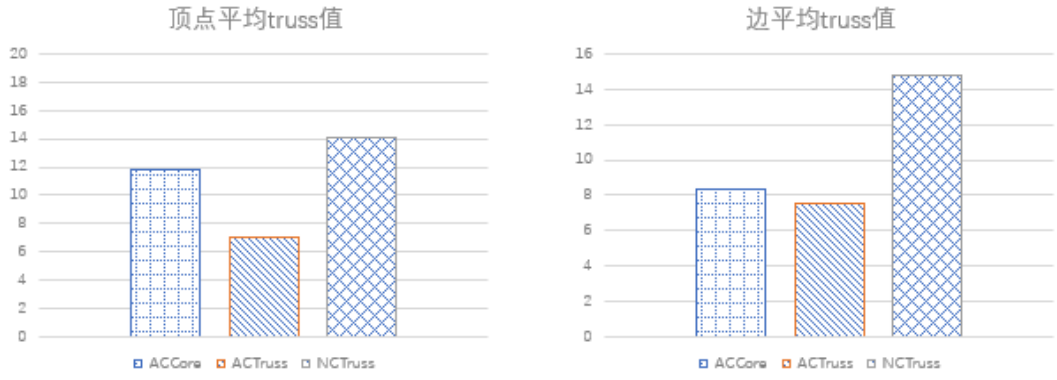

图 6.2 社区结构紧密性

测试结果表明只考虑社区结构的 NCTruss 算法发所得社区相对于 ACCore 和 ACTruss 所得的社区，其边和顶点有更大的平均 truss 值。一个直观的解释是因为后二者通过公共属性过滤了不符合条件的节点。因此得到了规模相对小的社区，但同时社区内的属性是密集的。综上所述，测试结果表明本系统所采用的算法对搜索结构和属性都密集的社区是有效的。

6.4.2 系统性能测试

性能测试的目的在于测试系统的性能，发现系统瓶颈[40]。为保证系统上线后的可靠运行，本小节从多个方面考察系统性能。

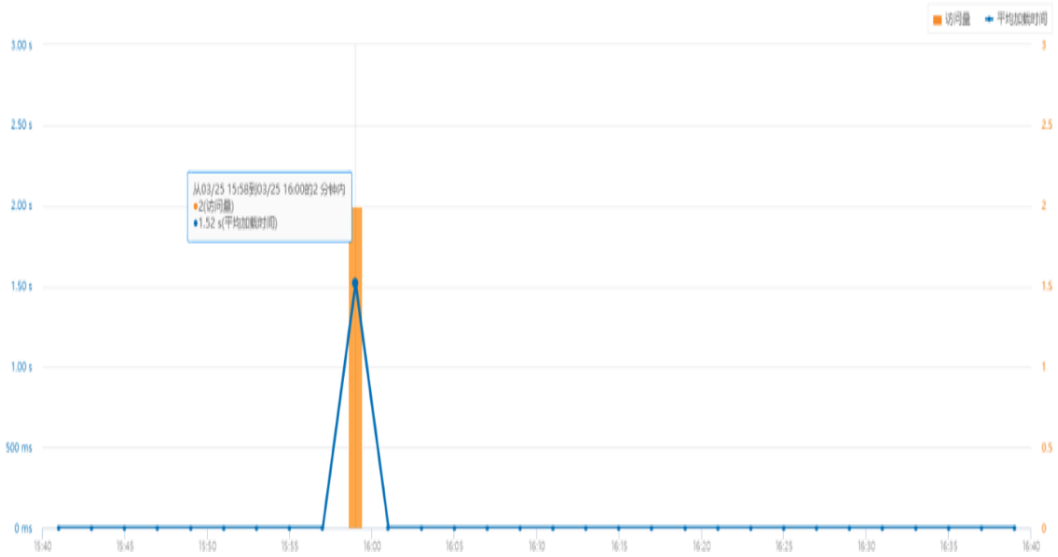

图 6.3 页面访问量&页面平均加载时间趋势图

页面平均加载时间，该指标是用户等待页面视图加载完成的时间，一般来说加载时间超过 5s 用户体验就会变差。本文测试了不同页面的加载时间，得到了系统的平均加载时间在 1.52s 左右。统计信息如图 6.3 所示。

页面加载时间由网络连接和传输时间、加载网页到 DOM 模型建立消耗时间、网页 DOM 模型建立到网页渲染结束的消耗时间几个方面构成。测试发现，页面加载时间中 DOM 构建占加载时间的 96%，网络传输占加载时间的 3%,资源渲染占加载时间的 1%；由此可见平均加载时间的主要瓶颈在于 DOM 渲染，其原因是图的可视化需要复杂的 DOM 操作，会消耗大量时间。测试结果如图 6.4、6.5 所示。

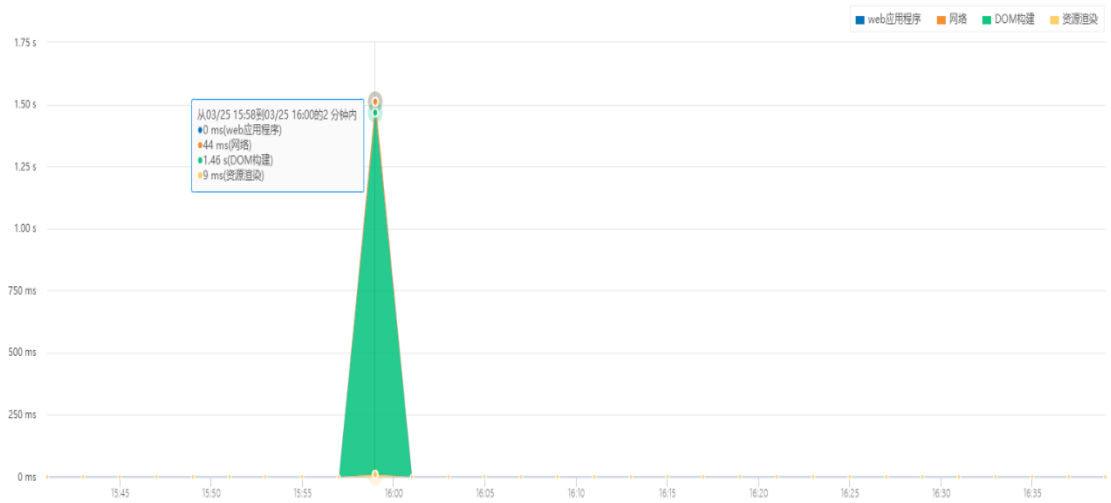

图 6.4 页面平均加载时间构成趋势图

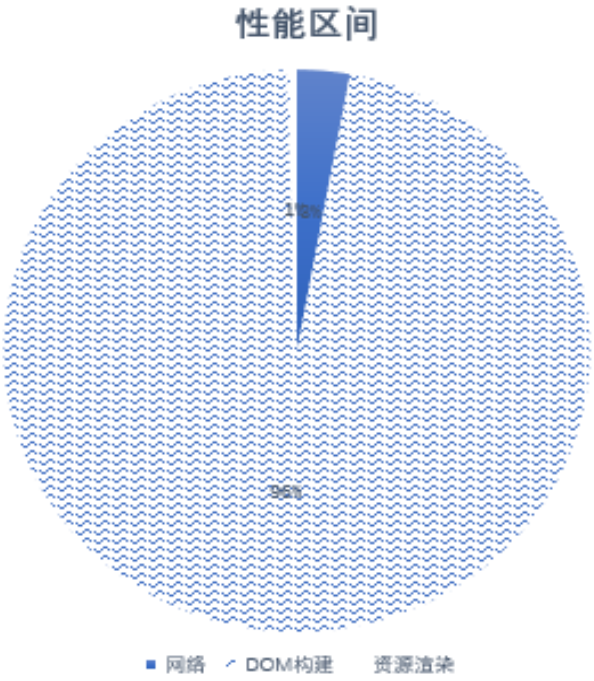

图 6.5 页面平均加载时间构成饼状图

AJAX 调用量以及平均响应时间，这个指标反映了从系统调用算法到算法执行完毕返回结果的平均时间。测试结果显示，本系统算法调用的平均响应时间约为 514ms，表明算法可以在较短时间内返回计算结果。具体测试信息如图 6.6 所示。

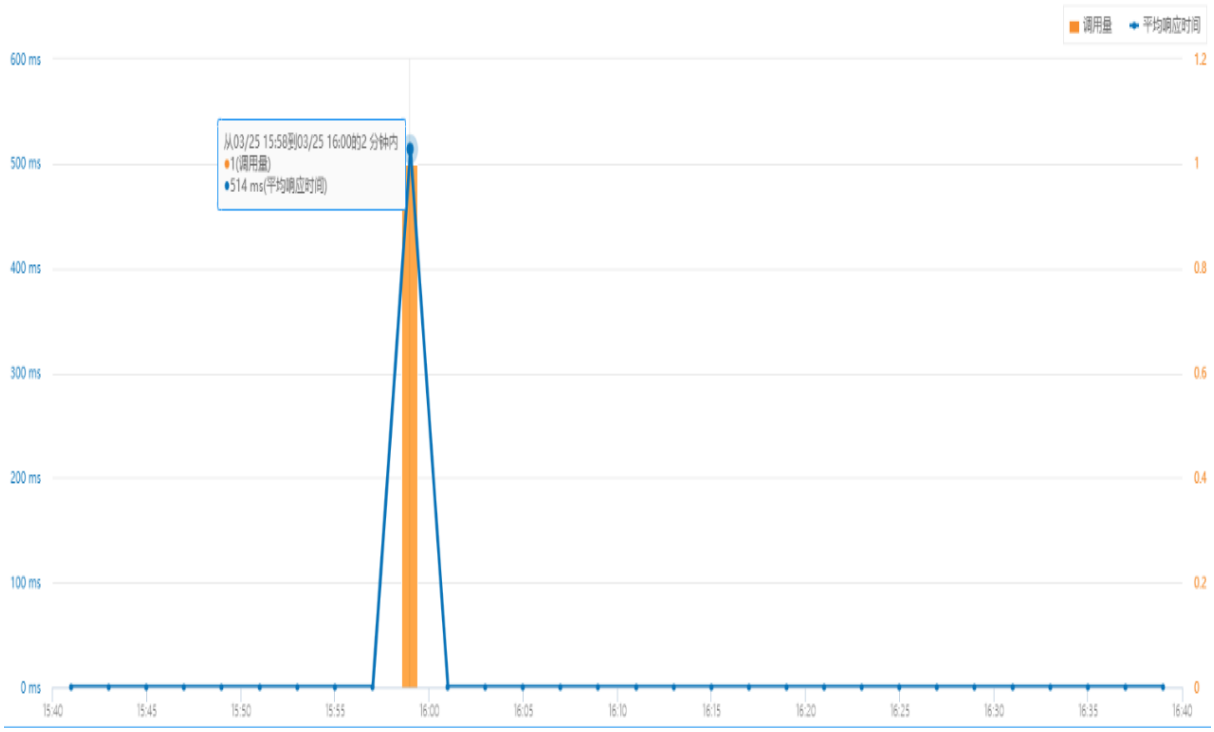

图 6.6 AJAX 调用量&平均响应时间

Apdex（Application Performance Index）指数[41]，是国际通用的度量用户满意度的指标，定义为：Apdex 指数=（1.0*满意样本数+0.5*容忍样本数）/样本总数（值介于 0-1 之间）。该指数根据响应时间长短划分了 3 个满意度区间。响应时间少于 2s 时，用户处于满意区间；响应时间为 2-8s 时，用户处于可以忍受的区间；当响应时间超过 8s，用户会对应用感到失望。本系统 Apdex 指数约为 1.2s 左右，测试结果如图 6.6 所示。

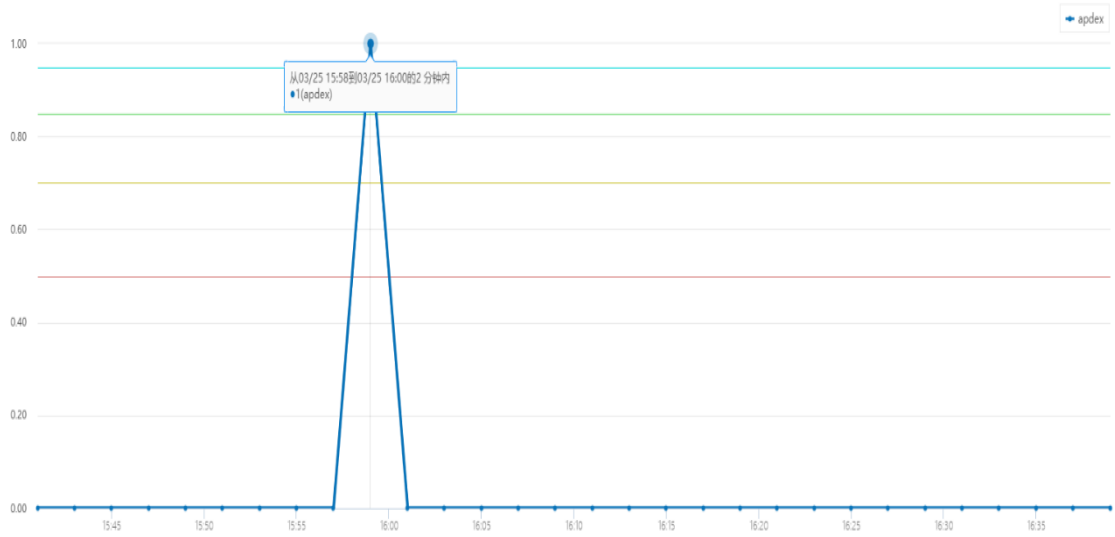

图 6.7 Apdex 指数

综合测试结果可知，系统在页面响应速度、Apdex 指数、算法调用响应速度等方面表现良好，响应时间的主要瓶颈在于 DOM 渲染时间较长，有待进一步优化。

6.5 本章小结

本章从测试环境搭建、系统单元测试、系统功能、有效性以及性能测试等四个主要方面介绍了系统的测试工作。本系统的测试覆盖了需求设计中的所有功能，并且详细验证了系统功能是否满足设计要求，保证了系统可以成功上线运行，也为未来可能的功能扩展打下了基础。

7 总结与展望

7.1 工作总结

大数据时代的来临，预示着新一代的信息检索技术迎来了新的发展。图作为一种复杂的数据结构，由于其方便描述关联性强的非结构化数据，因此在非结构化数据分析和挖掘方面具有很大的优势。随着图论的发展，众多优秀的算法被开发出来，利用这些工具对图数据进行检索和分析也变得愈发高效和重要。

本系统基于 DBLP 数据构造图，并以图和图数据库 Neo4j[1]为基础，进行一系列在常规数据结构上无法实现的数据检索和分析。本系统的主要工作可以总结为如下：

（1） 基础的图数据检索功能，包括：作者文章检索、作者合作对象检索、合作对象文章检索、作者文章关键字查询。其中，作者文章关键字查询要在数据处理阶段对文章标题进行分词，并将分词结果作为图中节点属性。基础检索功能满足了用户对于学术文献的一般检索需求。

（2） 图结构信息查询功能，包括：作者中心度查询、作者相似度查询。作者中心度查询借助 Neo4j[1]算法库自带算法实现。作者相似度查询借助于相似度算法 SimRank[12]实现，需要将 SimRank[12]算法与 Neo4j[1]进行无缝整合。结构化信息查询功能可以让用户直观了解作者在图中结构信息，不同于文本信息，结构信息更有利于对图信息的整体了解。

（3） 研究团体搜索功能，包括：结构上紧密的社区，结构和属性都密集的社区。对于后者，也需要借助在数据处理阶段对作者文章标题进行分词，然后借助于社区搜索算法 ATC[13]进行社区搜索。该功能提供了不同于一般结构紧密的社区搜索的社区检测功能，使用户方便找到结构和属性都密集的社区。

综上所述，本系统通过构造图数据，在利用图数据库以及图论相关算法，设计开发了一套相对完善的图信息检功能；为用户提供了使用图数据结构分析文本信息的能力。

7.2 未来展望

本系统的顺利开发完成代表着这一系统基础功能的完成，也为其之后的发展打下了基础。当前在系统可视化方面还只是实现了基础展示功能，考虑到用户的增多，不同用户需要不同的访问权限，所以 RBAC 权限管理功能也是一个可能的需求。

此外，随着 DBLP 收录的文献不断更新，本系统中储存的信息有过时的风险，因此如何及时同步 DBLP 数据到本系统中也是一个需要考虑和完善的问题。在系统后期的发展中，需要增加数据同步功能模块。

最后，本系统整合的算法只是图论中的一小部分，从功能的多样性上考虑，系统还需要整合更多的经典的或者最新发布的算法，以提供更加丰富的图数据分析功能。

参考文献

[1] Neo4j.Neo4j 官网[EB/OL].https://Neo4j.com/,2020.

[2] Redis Labs.Redis 官网[EB/OL].https://redis.io/,2015-06.

[3] Apache Hbase.The Apache Software Foundation[EB/OL].https://hbase.apache.org/,2007.

[4] MongoDB.MongoDB 简介[EB/OL].https://www.mongodb.com/cn,2021.

[5] 崔雷，刘伟，闫雷，等. 文献数据库中书目信息共现挖掘系统的开发[J]. 现代图书情报技术, 2008, 8: 70-75.

[6] 曹树金，吴育冰，韦景竹，等. 知识图谱研究的脉络，流派与趋势——基于 SSCI 与 CSSCI 期刊论文的计量与可视化[J]. 中国图书馆学报, 2015, 41(5): 16-34.

[7] Egenhofer M J. Spatial SQL: A query and presentation language[J]. IEEE Transactions on knowledge and data engineering, 1994, 6(1): 86-95.

[8] Anderson C. The model-view-viewmodel (mvvm) design pattern[M]//Pro Business Applications with Silverlight 5. Apress, Berkeley, CA, 2012: 461-499.

[9] Fielding R T, Taylor R N. Principled design of the modern web architecture[J]. ACM Transactions on Internet Technology (TOIT), 2002, 2(2): 115-150.

[10] To L R G, Reenskaug F T. THING-MODEL-VIEW-EDITOR an Example from a planningsystem[J]. 1979.

[11] 李超，谢坤武. 软件需求分析方法研究进展[D]. , 2013.

[12] Wang H, Wei Z, Yuan Y, et al. Exact Single-Source SimRank Computation on Large Graphs[C]//Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020: 653-663.

[13] Zhu Y, He J, Ye J, et al. When Structure Meets Keywords: Cohesive Attributed Community Search[C]//Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020: 1913-1922.

[14] Freeman L C. Centrality in social networks conceptual clarification[J]. Social networks, 1978, 1(3): 215-239.

[15] Sabidussi G. The centrality index of a graph[J]. Psychometrika, 1966, 31(4): 581-603.

[16] Freeman L C. A set of measures of centrality based on betweenness[J]. Sociometry, 1977: 35-41.

[17] Needham M, Hodler A E. Graph Algorithms: Practical Examples in Apache Spark and Neo4j[M]. O'Reilly Media, 2019.

[18] Boudin F. A comparison of centrality measures for graph-based keyphrase extraction

[C]//Proceedings of the sixth international joint conference on natural language proces sing. 2013: 834-838.

[19] Morselli C, Roy J. Brokerage qualifications in ringing operations[J]. Criminology, 20 08, 46(1): 71-98.

[20] Wu S, Gong L, Rand W, et al. Making recommendations in a microblog to improve the impact of a focal user[C]//Proceedings of the sixth ACM conference on Recomm ender systems. 2012: 265-268.

[21] Newman M. Networks[M]. Oxford university press, 2018.

[22] Altman N S. An introduction to kernel and nearest-neighbor nonparametric regression [J]. The American Statistician, 1992, 46(3): 175-185.

[23] Jaccard P. The distribution of the flora in the alpine zone. 1[J]. New phytologist, 19 12, 11(2): 37-50.

[24] Jeh G, Widom J. Simrank: a measure of structural-context similarity[C]//Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and d ata mining. 2002: 538-543.

[25] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect communit y structures in large-scale networks[J]. Physical review E, 2007, 76(3): 036106.

[26] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008 (10): P10008.

[27] 杨长春，俞克非，叶施仁，等. 一种新的中文微博社区博主影响力的评估方法[J]. 计算机工程与应用, 2012 (2012 年 25): 229-233+ 248.

[28] Galler B A, Fisher M J. An improved equivalence algorithm[J]. Communications of the ACM, 1964, 7(5): 301-303.

[29] Zhang P, Wang F, Hu J, et al. Label propagation prediction of drug-drug interactions based on clinical side effects[J]. Scientific reports, 2015, 5(1): 1-10.

[30] 冯晓楠. 社区问答系统中的社团发现技术研究及其应用[D]. 中国科学技术大学, 201 4.

[31] 黄焕坤. 基于微博互动的关系圈发现及其可视化研究[D]. 广东工业大学, 2015.

[32] Dugué N, Perez A. Directed Louvain: maximizing modularity in directed networks[D]. Université d'Orléans, 2015.

[33] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical review E, 2007, 76(3): 036106.

[34] Que X, Checconi F, Petrini F, et al. Scalable community detection with the louvain

algorithm[C]//2015 IEEE International Parallel and Distributed Processing Symposium. IEEE, 2015: 28-37.

[35] 谷红勋，杨珂．基于大数据的移动用户行为分析系统与应用案例[J]．电信科学, 2016, 32(3): 139-146.

[36] Meunier D, Lambiotte R, Fornito A, et al. Hierarchical modularity in human brain functional networks[J]. Frontiers in neuroinformatics, 2009, 3: 37.

[37] 马瑞芳，王会燃．计算机软件测试方法的研究[D]. , 2003.

[38] Fang Y, Cheng R, Luo S, et al. Effective community search for large attributed graphs[J]. Proceedings of the VLDB Endowment, 2016, 9(12): 1233-1244.

[39] Huang X, Lakshmanan L V S. Attribute-driven community search[J]. Proceedings of the VLDB Endowment, 2017, 10(9): 949-960.

[40] 何正玲. Web 系统性能测试研究及应用[J]．科技信息, 2013, 15: 95-96.

[41] Sevcik P. Defining the application performance index[J]. Business Communications Review, 2005, 20.

致 谢

在武大的求学时光即将结束，研究生开学典礼仿佛就在昨日，想到即将和老师同学分别有些不舍，但除了不舍更多的是感动与温暖，感动的是老师同学的无私帮助，温暖的是老师同学在学习生活上的教导与关心。

在这里要特别感谢祝老师，在论文的选题立意、研究资料的提供、系统实现思路等许多方面，她都给予了我悉心的帮助与指导。祝老师严谨的治学态度以及过硬的科研能力始终是我求学道路上努力学习，迎难而上的榜样。

另外，在论文写作过程中我的同窗和朋友们也给我提供了力所能及的帮助；并且在日常的学习生活中教会了我诸多宝贵的经验。我由衷的感谢他们，愿天涯海角，友谊长青。

最后想说，读研不仅是知识的提高，也是自我的重新认知，更是迈向新征程的起点，我会坚持自己的道路，努力前进，成为更好的自己！

武汉大学学位论文使用授权协议书

（一式两份，一份论文作者保存，一份留学校存档）

本学位论文作者愿意遵守武汉大学关于保存、使用学位论文的管理办法及规定，即：学校有权保存学位论文的印刷本和电子版，并提供文献检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在以教学与科研服务为目的前提下，学校可以在校园网内公布部分及全部内容。

1、 在本论文提交当年，同意在校园网内以及中国高等教育文献保障系统（CALIS）高校学位论文系统提供查询及前十六页浏览服务。

2、 在本论文提交□当年/□一年/□两年/☑三年以后，同意在校园网内允许读者在线浏览并下载全文，学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。（保密论文解密后遵守此规定）

论文作者（签名）：陈小龙

学　　号：2019282110194

学　　院：计算机学院

日期：2021 年 5 月 27 日