学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

密级\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**东北大学本科毕业论文**

基于知识图谱的学术资源

推荐系统设计与实现

学 院 名 称：计算机科学与工程

专 业 名 称：计算机科学与技术

学 生 姓 名：孙季斌

指 导 教 师：王大玲 教授

二○一九年六月

# **郑 重 声 明**

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名： 日期：

# 摘 要

随着大数据时代到来，人们逐渐从信息缺乏的时代走向了信息过载的时代。如今，学术资源每年以亿数量级增长，这给用户（科研人员）的学术研究提供了宝贵的资源财富。但与此同时，大量的资源也使得如何使用户获取自己感兴趣的资源以及如何高效地获取这些资源成为目前亟待解决的一个问题。学术资源推荐系统是解决这一问题的最有效的方法之一，基于此，本文开发一个基于知识图谱的学术资源推荐系统。

传统的推荐系统方法分为协同过滤推荐、基于内容推荐，而本文所采用的基于知识图谱的学术资源推荐系统则是在传统推荐方法的基础上，将知识图谱作为辅助工具，以知识图谱的三元组信息使推荐系统更精确、更具解释性和多样性。

本文基于DBLP学术资源网站所提供的原始数据，完成了以下主要工作：

（1）对原始数据进行了解析并进行数据预处理，包括数据清洗等工作；

（2）导入neo4j图数据库，并通过neo4j实现了知识图谱的展示；

（3）使用推荐算法itemCF，UserCF以及Cypher语言实现了论文的推荐；

（4）使用python库py2neo实现了前后端交互；

（5）使用python库tkinter实现了用户界面可视化。

测试表明，本文最终实现的原型系统很好地完成了上述功能

关键词：推荐系统，知识图谱，neo4j，itemCF算法，UserCF算法

# **ABSTRACT**

With the advent of the era of big data, people have gradually moved from an era of lack of information to an era of information overload. Today, academic resources grow at an annual order of billions, which provides valuable resources for academic research for users (i.e. researchers). However, a large number of resources also make it an urgent problem to be solved how to make the users access the resources they are interested in and how to obtain them efficiently. The recommender system for academic resource is one of the most effective ways to solve this problem. In this case, a recommender system for academic resources based on knowledge graph is designed and implemented in this thesis.

The traditional method of recommender system is divided into collaborative filtering recommendation and content-based recommendation. The recommender system for academic resource with knowledge graph adopted in this thesis is based on the traditional recommendation method which is using knowledge graph as an auxiliary tool, and the recommendation system is more accurate, more explanatory and more diversity based on the triple of the knowledge graph.

Based on the original data provided by the DBLP Academic Resources website, this project completed the following main work.

(1) Parses the original data and performing data preprocessing, including data cleaning.

(2) Imports the neo4j graph database and display the knowledge graph through neo4j.

(3) Applies the recommended algorithm itemCF ,userCF and Cypher language to implement the paper's recommendation.

(4) Uses the python library py2neo for front-end interaction.

(5) Uses the python library Tkinter to visualize the user interface.

The test results to the prototype implemented by this thesis show that above functions have been completed well.

**Key Words**: Recommender System, Knowledge Graph, Neo4j, ItemCF, UserCF

# 

# 目 录

摘要 III

**ABSTRACT** V

1 绪论

1.1 课题的研究背景及意义 1

1.2 国内外研究现状 2

1.3 本文研究内容 4

1.4 本文组织结构 4

2 相关理论与技术

2.1 数据获取 7

2.2 Neo4j数据库介绍 7

2.3 前后端交互python库py2neo介绍 8

2.4 前端界面python库Tkinter介绍 8

2.5 知识图谱 8

2.6 推荐算法 8

2.7 本章小结 10

3 系统分析

3.1 系统任务概述 11

3.2 系统运行环境 11

3.3 系统开发环境 11

3.3.1 程序语言 11

3.3.2 数据库 11

3.4 数据来源 12

3.5 功能性需求描述 13

3.6 非功能性需求描述 18

3.7 本章小结 18

4 系统设计

4.1 系统架构设计 19

4.2 数据库设计 20

4.2.1 概念设计 20

4.2.2 逻辑设计 21

4.2.3 物理设计 21

4.3 系统功能结构 23

4.3.1 数据获取模块 24

4.3.2 数据预处理模块 24

4.3.3 用户登录模块 24

4.3.4 用户功能模块 24

4.3.5 知识图谱构建模块 24

4.3.6 论文查询模块 24

4.3.7 论文推荐模块 25

4.4 本章小结 25

5 系统实现

5.1 DBLP数据获取 27

5.2 数据预处理 28

5.3 知识图谱构建与展示 30

5.4 用户界面编写 32

5.5 用户功能实现 34

5.6 论文查询功能实现 36

5.7 论文推荐功能实现 39

5.7.1 用户端推荐功能实现 39

5.7.2 作者端推荐功能实现 42

5.8 本章小结 43

6 结论

6.1 本文工作总结 45

6.2 进一步展望 45

参考文献 47

致谢 49

# 1 绪论

## 1.1 课题的研究背景及意义

由于互联网技术的高速发展，人们由信息匮乏的时代走进信息过载的时代，这为人们带来了海量的信息资源。但与此同时，也使得如何从海量资源中高效获取想要的信息成为一大难点。学术资源亦是如此。如今每年学术资源以数以亿计增长，这为学者、研究人员如何从中获取自己需要的论文成为难题。

为解决信息过载的问题，诞生了许多方法，如分类目录、搜索引擎。而推荐系统是继分类目录和搜索引擎后，又一为解决这一问题而产生的有效途径。与搜索引擎不同的是，推荐系统更注重当用户无法准确描述自己的需求时，通过用户的历史行为，注册信息为其高效发现有用的资源[1]。

推荐系统自1994年由明尼苏达大学研究组GroupLens第一次提出后，经过二十多年的发展，已经逐渐趋于成熟[2]。传统的推荐系统领域将推荐方法共分为三种，一是协同过滤推荐，二是基于内容推荐，三是混合推荐[3]。

然而目前，推荐系统仍然存在着例如冷启动、稀疏性等问题，本课题采用基于知识图谱进行推荐的意义即在于此。知识图谱是由实体以及它们相互之间的关系构成的语义网，节点表示实体，边则表示属性。知识图谱包含了许多潜在的关联，因此以知识图谱为基础进行推荐可以使得推荐系统更具精确性，能够发掘出语义网中各个实体之间潜在的、多层的关系；也能使推荐系统更具多样性，能够通过不同的关系发现不同的实体；此外还能使得推荐结果具有可解释性，从用户角度增加对于推荐系统的满意度。

因此，本课题所采用的基于知识图谱的推荐系统，既能提供一个可靠稳定的推荐，同时利用知识图谱作为辅助，对于传统推荐系统所存在的弊端也能有所改善。

本文工作来源于国家重点研发计划项目“基于立体精准画像的学术分类与推荐系统”子课题“实时高精度动态学术画像构建技术”，该子课题的研究内容包括“基于知识图谱的学者画像模型”、“分布式的专家评价模型”以及“基于画像活化模型的学者画像实时更新方案”。本文工作中的学术知识图谱构建是该课题的技术基础之一，基于知识图谱的学术资源推荐是其中的一个应用。

## 1.2 国内外研究现状

作为解决互联网时代信息过载的问题而诞生的推荐系统，在经过二十多年的发展后的今天，已经在许多领域取得突出的成就，包括在线视频、音乐、电影、社交网络及旅游业，在这些领域推荐系统都已证明了其存在的价值和意义，为人们的生活提供了许多便利。目前主流的推荐系统采用的是协同过滤推荐、基于内容推荐以及两者相混合的推荐。

协同过滤算法也就是基于用户的行为进行推荐，它更多的是以用户的角度，以用户的历史数据为基础进行推荐，而不太依赖于物品的特征数据。协同过滤算法是最早的推荐算法，也是目前最广泛使用的推荐算法。协同过滤算法又可以分为基于用户的协同过滤、基于物品的协同过滤。协同过滤算法的基础是相似度算法，相似度算法有包括皮尔逊相似度、杰卡德相似度、余弦相似度等众多算法。通过用户和用户、物品和物品之间的相似度计算可以得到目标用户最有可能感兴趣的物品进行推荐，本课题也是在协同过滤算法的基础上进行推荐。

基于内容推荐则与协同过滤推荐不同，它更多地需要依赖物品的特征数据，将其与用户兴趣相结合进行推荐，而对于用户与物品的历史交互信息没有必要的需求。

目前两种推荐算法的应用都已经比较成熟，但其算法本身会导致效果在某些特定的方面仍然会有不尽如人意的地方。例如基于用户的协同过滤，由于其原理是为用户推荐其他用户感兴趣的物品，因此可以为用户推荐热门的物品，但这也导致推荐结果缺乏个性化。基于物品的协同过滤可以发现物品的长尾，在个性化方面做的更好，但是却导致多样性不足，没有办法为用户发掘潜在的兴趣，从而推荐系统的惊喜度不足。同时协同过滤算法本身天然存在的缺陷导致其存在冷启动问题（新用户的推荐）以及稀疏性问题（用户与物品交互信息过少而导致的推荐结果不理想）。而基于内容进行推荐虽然可以解决协同过滤存在的冷启动和稀疏性问题，但是对于推荐结果却缺乏明确的解释，用户对于结果的满意程度可能会受此影响[4]。

目前，随着推荐系统这项技术日益成熟及成功落地，人们更多地开始考虑如何解决上述问题以完善推荐系统，这些问题都是现在推荐系统领域研究的热点。同时，推荐系统也正在与包括机器学习、深度学习、知识图谱等领域进行结合，以更多的方法来完善推荐系统[5]。

本课题所采用的基于知识图谱进行学术资源推荐，旨在以知识图谱作为一个额外的辅助工具，为目前推荐系统所存在的冷启动、稀疏性等相关问题提供一个解决的途径，同时也能增强推荐系统的精确性、可解释性和多样性。

知识图谱是2012年由Google提出的一种语义关系，其旨在通过各个实体之间的语义关系，为用户发掘出潜在的信息[6]。知识图谱可以融合各个数据源得到的语义信息，然后通过推理得到潜在信息为用户服务。目前，知识图谱的应用已经成为学术界的研究热点。推荐系统也不例外。知识图谱可以将推荐系统中的用户和用户、物品和物品以及用户和物品的关系充分反映和联系起来，能够得到相较于传统推荐系统更多的信息，提高推荐的质量。

基于知识图谱的推荐系统的研究虽然起步不久，但已经有了许多突出的研究成果。目前将知识图谱与推荐系统相融合的主题思路是：通过知识图谱所具有的能够对异构数据进行整合的特性，可以将用户和物品信息更加细化，从而能够更加准确的得出用户、物品之间的相似性。目前，基于知识图谱的推荐方法大体可以分为两种，一种是基于本体的推荐，另一种是基于LOD的推荐[7]。

首先是基于本体推荐，本体可以简单理解为知识图谱中实体与关系的总集合。基于本体的推荐可以对用户和物品的特征更精确的表示，可以发掘出物品、用户之间更具深层次的关联，同时通过细节化实体的特征，使得推荐结果更精确。然后再将知识图谱与传统的推荐算法包括协同过滤推荐、基于内容推荐等相结合，由此得到的推荐系统可以增强用户与物品之间的关联，从而在一定程度上解决了推荐系统稀疏性的问题。但是目前本体的创建还是需要人工创建，创建本体的时间过长是这一方法待解决的问题。

基于LOD的推荐则是另一种思路。LOD即为linked open data，即开放数据连接。其主体的思路为将数据库中的语义网络信息与当前现有的推荐算法相融合。在LOD推荐中，相似度算法是其核心，计算实体之间的相似度从而得出用户的兴趣所在是关键，除了上文中提到的相似度算法外，还有链接数据语义距离、基于隐式语义反馈的路径算法等等。基于LOD的推荐算法可以充分利用语义网络，使语义关联程度与推荐结果精确度形成正比。但是其对于数据库的依赖过强，因此语义网络的建立的完善度和合理程度是影响其相似度计算乃至推荐结果的效果的重要一环。

目前基于知识图谱的推荐的流程总共分为三步：第一步是对于数据获取以及预处理，包括用户信息、物品信息以及它们之间的交互信息。第二步是构建知识图谱，构建知识图谱需要从数据中抽取出实体、关系以及属性。第三步也是最为核心的部分，即基于知识图谱进行推荐。将知识图谱中丰富的语义信息与原有的协同过滤推荐、基于内容推荐等结合。

目前基于知识图谱的推荐在许多领域都得到了成功的应用，包括电影、音乐和旅游业等都有建树。但是目前仍然存在着许多问题急需解决。一是如何准确构建知识图谱，从而反映用户、物品的特征和关系。二是在大数据的情况下，计算的复杂度提高，如何使得推荐更加高效。三是如何充分发挥知识图谱的优点，使得语义网的潜在信息能够被推荐算法充分利用[8]。

虽然基于知识图谱的推荐系统目前还在刚刚起步发展的阶段，但是其呈现出巨大的可能性和多种的发展方向，值得去进一步探索和研究。

## 1.3 本文研究内容

基于上述分析，本课题研究基于知识图谱的学术资源推荐系统，研究内容包括五个部分。一是对于数据的获取和预处理。本课题所采用的数据集为DBLP学术网站的论文数据。获取其xml数据集后，使用python对数据集进行解析，并进行数据实体关系抽取和简单的数据清洗。二是构建知识图谱，知识图谱的构建采用neo4j图数据库进行，采用neo4j-import工具将数据导入数据库，构建实体和关系的知识图谱。三是采用协同过滤推荐算法，并与知识图谱相结合对于论文进行推荐。四是构建系统并实现包括推荐算法、论文查询和用户的基础功能。五是构建图形化界面，对于整个系统实现可视化。

## 1.4 本文组织结构

本文第1章为绪论，介绍课题的研究背景、意义、当前国内外研究现状以及本文的研究内容和组织结构。

第2章为相关技术介绍，介绍系统后端数据库neo4j、前端GUI界面和前后端交互所用的python库py2neo。同时介绍本课题的两大核心部分推荐算法和知识图谱。

第3章分析系统需要最终实现的效果。同时分析系统的开发、运行环境和数据来源，对于系统的功能性需求和非功能性需求进行了介绍。

第4章对于系统的结构进行设计，详细介绍数据库的设计以及各模块需要完成的功能。

第5章介绍系统整个实现的流程，包括数据获取和预处理，通过解析xml数据集获取论文数据，并将数据导入neo4j数据库实现知识图谱的构建。然后实现用户登录和包括论文查询在内的用户基础功能。最后是实现了协同过滤推荐算法。并对于不同的用户（作者和普通用户）分别实现了不同的推荐算法。其中作者端采用了UserCF即基于用户的协同过滤，而用户端则采用ItemCF即基于物品的协同过滤进行推荐。

第6章结论部分是对于整个课题的研究总结和进一步展望，以发现自身在研究中仍存在的不足，以便以后改进。

# 2 相关理论与技术

## 2.1 数据获取

数据集选择的是DBLP官网的xml格式的数据集。DBLP是有关计算机学术领域内对于论文进行整理的一个专业的学术网站，本课题也采用DBLP的数据为基础进行系统的设计。Python对于xml数据集的解析方法有许多，包括SAX、DOM和ElementTree。

本课题采用SAX（Simple API for XML）方法对于xml数据集进行解析。SAX是python自带的标准解析器，通过触发事件并调用回调函数来获得所需要的数据。通过SAX解析xml需要其中的两个功能：一个是解析器，一个是事件处理器。解析器用于获取xml数据内容，并给事件处理器发送触发事件，包括元素开始事件和结束事件，然后由事件处理器对于触发时需要采集的数据进行整理。完成整个功能需要导入xml.sax库中的parse函数，以及xml.sax.handler库中的ContentHandler函数。其中ContentHandler函数包括startElement方法用于处理元素开始事件时需要完成的工作、endElement方法用于处理元素结束事件时需要完成的工作以及characters方法用于获得两个标签（即两个开始或结束标志）之间的数据。

对数据解析完成后需要对于重复数据和脏数据进行处理，由于本课题所采用的数据集规模较大，因此选择Emeditor对于数据进行处理。Emeditor是一款支持大文件处理的文本编辑工具，能够基本满足需要的数据清理工作。

数据获取后以txt格式文件进行保存，以便后续知识图谱构建中的对于文件的处理工作。

## 2.2 Neo4j数据库介绍

Neo4j图数据库属于非关系型数据库（NoSQL）的一种，其表示数据以关系、节点和属性进行表示，采用类似于SQL语言的Cypher语言进行数据库的增删查改工作。其社区版能支持亿级别的数据的查询，同时相较于MYSQL等关系型数据库，其最大的优势在于当进行多层次关系查询或者寻找连接数据时，其效率远远高于一般关系型数据库[9]。目前Neo4j数据库已经进行商用，在稳定性和效率方面都比较不错。本课题选用的Neo4j为3.4.0版本。

Neo4j的数据库通过官网安装社区版，然后由Neo4j自带的浏览器Neo4j Browser对于数据库进行访问和操作。

## 2.3 前后端交互python库py2neo介绍

Python库py2neo是用于实现在python程序中对Neo4j进行操作。Py2neo能够准确解析Cypher语言，并将结果返回python。本课题选用的py2neo使用的是v3版本。安装该版本的py2neo使用命令行pip install py2neo==3.0.0。所使用到的库中函数包括Graph，Node，Relationship和Authenticate。其中Graph用于连接图数据库，Node和Relationship分别实现节点和关系的相关操作，Authenticate函数用于实现链接数据库时的验证工作。

## 2.4 前端界面python库Tkinter介绍

Python库Tkinter是python自带的实现图形化界面，可以实现前端的可视化。Tkinter作为基础的GUI工具优点是对于初学者比较友好，其基本的部件和布局能够完成本课题基本的需求。缺点是界面风格比较简单朴素，使用Tkinter库中的ttk可以进行改进。ttk继承和美化了Tkinter的大多数部件，使GUI界面的整体风格近似于Win7的风格，提高美观性。

## 2.5 知识图谱

知识图谱目前来说仍然是一个处于起步发展阶段的方向，它在2012年由Google提出，本旨在为搜索引擎提供更多、更深层次的语义关系，后来作为一门新兴的技术应用于更多的领域。

知识图谱的核心概念可以理解为组建基于各个实体以及实体之间的关系的语义网络，并以此语义网提供潜在的数据关联，以此提升整个系统对于数据的描述能力。因此构建知识图谱的关键就是实体、关系以及属性。本课题采用Neo4j图数据库完成对于知识图谱的构建工作[10]。

## 2.6 推荐算法

本学术资源推荐系统所采用的推荐算法的主体部分为协同过滤推荐。整个系统将推荐算法分为两大部分，一部分是普通用户端，由于普通用户端考虑到用户与用户之间的联系不多，因此更多的考虑从用户的历史信息出发进行推荐，选择基于物品的协同过滤推荐算法（ItemCF）。而另一部分是作者端，作者端由于作者之间的相互关系较多，因此选择了基于用户的协同过滤推荐算法（UserCF）

下面详细介绍两种算法的原理。

首先是基于物品的协同过滤算法（ItemCF）[11]，ItemCF算法的主题思想是为用户推荐与其历史兴趣相近的物品，因此它可以对于推荐结果通过物品相似度给出合理的解释[12]。其原理核心共分为两步：第一步是计算物品之间的相似度，第二步是将物品相似度与用户的兴趣即用户与物品的历史交互信息相结合进行推荐。

该算法首先需要计算物品之间相似度。如绪论介绍的相似度算法有很多种，如余弦相似度、杰卡德相似度等。综合相似度算法可行性以及相似度算法的效果综合分析，本课题选择了杰卡德相似度（Jaccard相似度）进行推荐[13]。Jaccard相似度算法通过Jaccard系数来衡量两个实体的相似程度。具体算法见公式2.1。

 (2.1)

其中*A*和*B*代表两个集合，*J(A,B)*代表计算两个集合的Jaccard系数。Jaccard系数为0-1中的一个数，计算方法为*A*和*B*集合的交集除以*A*和*B*集合的并集。当两个集合相同时，系数为1；当两个集合没有交集时，系数为0。通过Jaccard相似度算法可以计算两个物品之间的相似度，具体实现将在后文中详细介绍。

计算好相似度后将进行该推荐算法的第二步，即将物品的相似度与用户的历史行为相结合。

此处ItemCF采用的算法见公式2.2。

 (2.2)

其中代表用户*u*对于物品*j*的兴趣度。*N*(*u*)代表用户*u*历史兴趣物品集合，*S*(*j,k*)代表与物品*j*相似度最高的*K*个物品集合，物品*i*为两个集合的交集中的物品。*wji*代表物品*j*和物品*i*的相似度，*rui*代表用户*u*对于物品*i*的感兴趣程度。感兴趣程度在不同的推荐系统中以不同的方式进行衡量，其中以用户评分和用户对于物品点击浏览两种为主流。具体的ItemCF推荐算法在本课题中的实现方法将在后文中详细介绍。

之后介绍基于用户的协同过滤算法（UserCF）[14]。UserCF算法的主题思想是为用户推荐与其兴趣相似的用户所感兴趣的物品[15]。其原理核心也分为两步：第一步是计算用户之间的相似度，得出与用户相似度最高的*K*个用户集合，第二步是将这个用户集合中用户所感兴趣的物品，同时是目标用户没有与之交互的推荐给目标用户。

该算法首先是计算用户之间的相似度。本课题此处依然选用了Jaccard相似度，与之前计算物品相似度唯一不同的地方在于此处实体为用户而不是物品，其余都大体相同不再赘述。具体的相似度算法的实现会在后文中详细介绍。

计算好相似度后，UserCF会为用户推荐与其相似的用户感兴趣的物品。此处物品的排序需要使用UserCF算法所使用的公式，详见公式2.3。

 (2.3)

其中*pui*表示用户*u*对于物品*i*的感兴趣程度。*S*(*u,k*)表示与用户*u*相似度最高的*K*个用户集合，*N*(*i*)表示与物品*i*发生过交互行为的用户集合，*v*是属于两者交集的用户集合。*wuv*代表用户*u*与用户*v*的相似度，*rvi*代表用户*v*对于物品*i*的感兴趣程度。同样的，这里的感兴趣程度可以用用户评分或者用户点击来度量。具体的UserCF算法的实现也会在后文给出。

## 2.7 本章小结

本章介绍本文工作中涉及的相关理论与技术。首先介绍了数据获取的来源，采用DBLP的xml数据集，之后介绍了如何解析数据集并对数据进行预处理，然后介绍了系统的后端neo4j数据库、系统的前端Tkinter图形化界面以及两者交互所用的库py2neo。之后介绍了本课题研究的两大核心知识图谱与推荐算法，其中推荐算法着重介绍了本课题所采用的基于用户的协同过滤推荐（UserCF）和基于物品的协同过滤推荐（ItemCF）。

# 3 系统分析

## 3.1 系统任务概述

目前基于知识图谱的推荐系统的研究仍处于初步发展阶段，学术资源推荐系统作为推荐系统落地的一个领域也是如此。本课题的任务就是将基于知识图谱的学术资源推荐系统进行一个简单但完整的实现。

本推荐系统可以对于DBLP的论文数据通过论文名或者作者名进行模糊查询，同时使用协同过滤推荐算法为用户和作者推荐相关论文。同时本系统能够完成包括知识图谱的查看、用户浏览历史的查看、作者写作信息的查看以及用户登录端的基础功能。这些功能将使用Tkinter集成到图形化界面中，便于用户实现交互。

## 3.2 系统运行环境

系统运行环境如表3.1所示：

表3.1系统运行环境表

|  |  |
| --- | --- |
| 硬件环境 | 软件环境 |
| CPU：Intel Core i5-4200H 3.40GHz | 操作系统：Windows 8.1 |
| 内存：8G | 编辑器：Emeditor |
| 硬盘：1TB | 浏览器：Google Chrome 73.0.3 |

## 3.3 系统开发环境

### 3.3.1 程序语言

本系统采用python语言进行程序开发，其程序生态十分丰富，具有大量封装好的库供研究者使用，本课题使用的python版本为3.7.2。

### 3.3.2 数据库

本系统所使用数据库为Neo4j图数据库。Neo4j图数据库是一个开源的非关系型数据库，其后端技术支持为java语言。Neo4j数据库的数据的存储形式为将数据存储在图中而并不像关系数据库一样将数据存储在表中。其数据具体的存储形式为存储节点、关系和属性。对于Neo4j图数据库的操作采用的是Cypher语言，是一种类sql语言。Neo4j图数据库在本课题中相较于其他数据库优势巨大，体现在其语言简单易学；其自带执行命令的UI：Neo4j Browser；其天然的数据存储格式能够使得知识图谱的构建以及连接数据的查询操作快速高效。

本课题Neo4j数据库采用的版本是neo4j-community-3.4.0-windows。

## 3.4 数据来源

数据集采用的是DBLP的xml数据集。DBLP作为计算机领域论文整合的网站，在学术领域内颇受认可，其论文的质量和全面性都很高，因此选择其作为本系统的数据集。从DBLP官网下载的数据集如图3.1所示。



图3.1 DBLP数据集

获得的数据集中所包含的论文包括article、inproceedings、proceedings等。参考了本系统的运行环境中的内存大小，最终选择了数据集中的article类的数据，共包括1981606篇论文。

论文数据在数据集中具体的表现形式如图3.2所示。

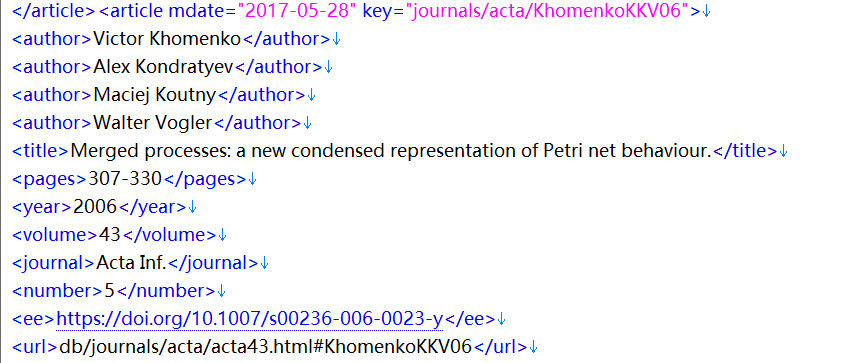


图3.2 DBLP数据集格式

数据集所包含的数据信息包括作者（author）、题目（title）、页码（page）、期刊（journal）年份（year）、doi（ee）等。本系统需要的数据包括作者、题目、期刊、年份和DOI的信息。

## 3.5 功能性需求描述

在进行系统的功能性需求分析时，使用用例图从用户角度对于系统进行分析是最常用的办法。用例图既可以向用户展示系统需要实现的功能，同时系统的开发人员也能对于系统的需求有一个整体把握。如图3.3所示为本系统的用例图。

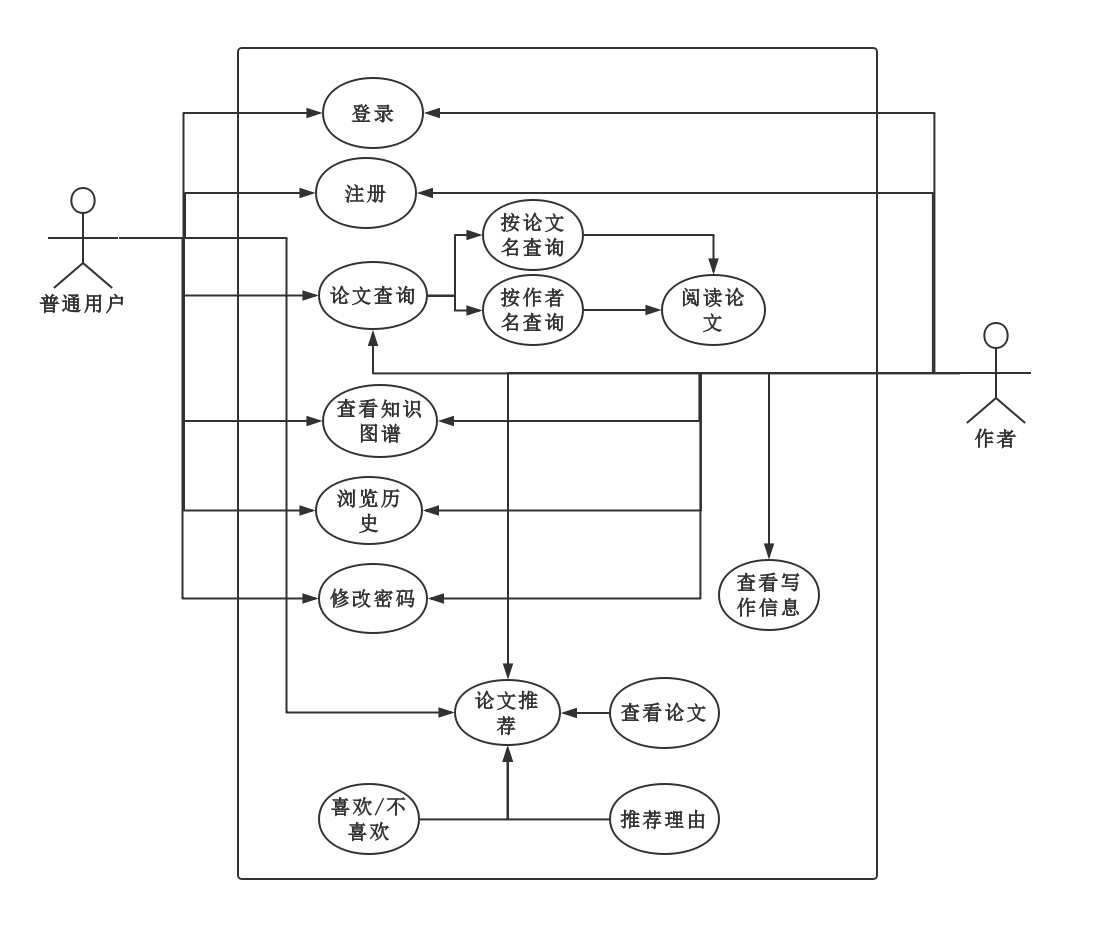


图3.3 基于知识图谱的学术资源推荐系统用例图

本系统的参与者包括用户和作者两种，参与者作者的大多数功能与普通用户相同，唯一增加的一个功能是查看自己写作信息的功能。

以下介绍该用例图相关的用例描述。

1、登录

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例登录系统，验证后可以进入推荐系统。
3. 前置条件：无。
4. 后置条件：获取用户操作权限。
5. 基本事件流：

① 用户点击login登录按钮。

② 系统显示登录界面，要求用户输入用户名和密码。

③ 用户输入用户名和密码。

④ 系统验证用户名和密码。若用户名错误，则弹窗显示是否注册，若密码错误则显示错误信息。若用户名和密码正确则弹窗显示用户信息。

⑤ 进入推荐系统界面，用例结束。

2、注册

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例进行注册，验证后返回主界面并可以选择登录用例。
3. 前置条件：无。
4. 后置条件：用户信息存储进系统。
5. 基本事件流：

① 用户点击sign up注册按钮。

② 系统显示注册界面，要求新用户输入用户名、密码和确认密码。

③ 用户输入用户名、密码和确认密码。

④ 系统验证用户名和密码。若用户名已存在，则弹窗显示错误信息，若密码和确认密码不同，也弹窗显示错误信息，若用户名和密码符合规范，则显示注册成功信息。

⑤ 进入主界面，用例结束。

3、论文查询

1. 参与者：普通用户、作者。
2. 简要说明：用户进入推荐系统后，选择查询框进行查询，查询结果显示在系统中。
3. 前置条件：拥有用户操作权限。
4. 后置条件：查询结果显示在论文信息显示框中。
5. 基本事件流：

① 用户在推荐系统查询框输入想查询的内容。

② 用户选择按论文或是按作者两种查询方式。

③ 用户点击查询按钮进行查询。

④ 系统判断若为按论文查询，则以关键词查询相关论文并将结果显示在查询结果弹窗中，若为按作者查询，则以关键词查询相关作者将结果显示在查询结果弹窗中，并执行其他事件流A1。

⑤ 用户点击相应查询结果，将查询结果以及阅读论文按钮显示在论文信息显示窗口中，用例结束。

1. 其他事件流 A1：显示作者论文。

① 在弹窗中选择要查询的作者。

② 弹窗显示选中作者写作信息。

③ 选择论文并将查询结果以及阅读论文按钮显示在论文信息显示窗口中，用例结束。

4、查看知识图谱

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例查看知识图谱，系统调用Neo4j Browser 查看知识图谱。
3. 前置条件：拥有用户操作权限
4. 后置条件：通过浏览器端查看整个系统的知识图谱
5. 基本事件流：

① 用户点击知识图谱按钮。

② 系统调用谷歌浏览器查看Neo4j数据库中的知识图谱，用例结束

5、浏览历史

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例查看自己之前所浏览的论文信息。
3. 前置条件：拥有用户操作权限。
4. 后置条件：获得用户浏览历史信息。
5. 基本事件流：

① 用户点击浏览历史按钮。

② 系统弹窗显示用户浏览历史。

③ 用户选中弹窗中的论文，并点击查看论文按钮。

④ 系统将查询结果显示在论文信息显示窗口中，用例结束。

6、修改密码

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例可以修改自己的密码。
3. 前置条件：拥有用户操作权限。
4. 后置条件：用户密码被修改。
5. 基本事件流：

① 用户点击修改密码按钮。

② 系统显示密码修改界面，要求用户输入旧密码、新密码和确认密码。

③ 用户输入旧密码、新密码和确认密码。

④ 系统验证。若旧密码错误，则弹窗显示错误信息，若密码与确认密码不同，也显示错误信息。若正确则弹窗显示修改成功。

⑤ 进入推荐系统界面，用例结束。

7、阅读论文

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例进行论文的阅读。
3. 前置条件：论文信息显示在论文信息窗口中。
4. 后置条件：在Chrome浏览器中阅读论文。
5. 基本事件流：

① 用户点击阅读论文按钮。

② 系统调用Chrome浏览器，打开论文的网址供用户阅读，用例结束。

8、论文推荐

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例进行论文的推荐，并将推荐信息显示在推荐窗口。
3. 前置条件：在论文信息显示窗口点击阅读论文按钮。
4. 后置条件：在推荐窗口显示推荐的论文以及相应的功能按钮包括喜欢/不喜欢、推荐理由和查看论文。
5. 基本事件流：

① 用户点击阅读论文按钮。

② 系统调用推荐算法，并将推荐结果显示在推荐窗口。

③ 推荐窗口增加喜欢/不喜欢、推荐理由和查看论文三个按钮供用户后续选择，用例结束。

9、喜欢/不喜欢

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例对于选中的论文进行喜好的选择，并弹窗显示操作结果。
3. 前置条件：获得用户操作权限，在论文推荐窗口操作。
4. 后置条件：对选中论文喜好进行设定。
5. 基本事件流：

① 用户喜欢/不喜欢按钮。

② 系统调用后端函数对于用户与选中论文的喜好进行设置。

③ 若为喜欢，则弹窗显示将增加相似论文推荐，若为不喜欢，则弹窗显示将减少相似论文推荐，用例结束。

10、推荐理由

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例对于推荐论文的推荐理由进行查看。
3. 前置条件：获得用户权限并进入推荐窗口进行操作。
4. 后置条件：获取选中论文的推荐理由。
5. 基本事件流：

① 用户点击推荐理由按钮。

② 系统将根据选中的论文为参数，调用后端函数，并将结果返回。

③ 系统弹窗显示论文推荐理由，用例结束。

11、查看论文

1. 参与者：普通用户、作者。
2. 简要说明：用户用该用例对于推荐论文进行查看。
3. 前置条件：获得用户权限并进入推荐窗口进行操作。
4. 后置条件：将选中论文的相关信息显示在论文信息显示窗口中。
5. 基本事件流：

① 用户点击查看论文按钮。

② 系统将根据选中的论文为参数，调用后端函数。

③ 系统将选中论文信息显示在论文信息显示窗口，用例结束。

12、写作信息

1. 参与者：作者。
2. 简要说明：作者类型用户用该用例对于自己写过的论文进行查看。
3. 前置条件：获得用户权限。
4. 后置条件：获取自己的写作信息。
5. 基本事件流：

① 作者点击写作信息按钮。

② 系统将弹窗显示用户写过的论文，作者可以对于论文使用查看论文用例进一步查看论文信息，用例结束。

## 3.6 非功能性需求描述

本学术资源推荐系统在完成一系列基本需求的基础上，需要达到以下几个目标：

(1) 实时性。因为该系统需要的用户与系统的交互频繁，因此在时间效率方面不能太慢，以免影响用户的操作体验；

(2) 友好性。为了给用户与系统的交互带来友好的体验，选择了Tkinter的GUI界面，为用户提供直观便捷的操作体验。

## 3.7 本章小结

本章首先介绍了系统总体需要完成的目标，然后介绍了系统的运行和开发环境，之后分析了系统所采用的数据集以及数据集的一些数据特征。之后使用用例图对于系统的功能需求进行了分析，最后分析了系统的非功能性需求包括交互的友好性以及系统的效率问题。

# 4 系统设计

## 4.1 系统架构设计

本系统采用的是分层式的结构。采用的三层架构，将整个系统划分为界面层、业务逻辑层和数据访问层，如图4.1所示。

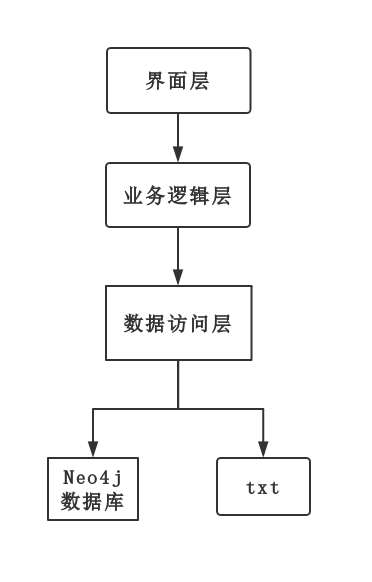


图4.1 系统架构

其中各层的功能包括：

(1) 数据访问层：数据访问层用于实现对于Neo4j数据库的一些基础操作以及系统所使用的txt文本文件的一些数据的增删查改工作；

(2) 业务逻辑层：业务逻辑层用于实现系统的各个具体的功能，包括论文推荐、论文查询、知识图谱的查看等，将各个功能模块化将其实现。与数据访问层的区别在于其重在功能逻辑的实现；

(3) 界面层：本系统界面层使用Tkinter图形化界面使系统和用户可以交互。

## 4.2 数据库设计

### 4.2.1 概念设计

本部分采用E-R图对于数据库进行概念设计，本系统的E-R图如图4.2所示。

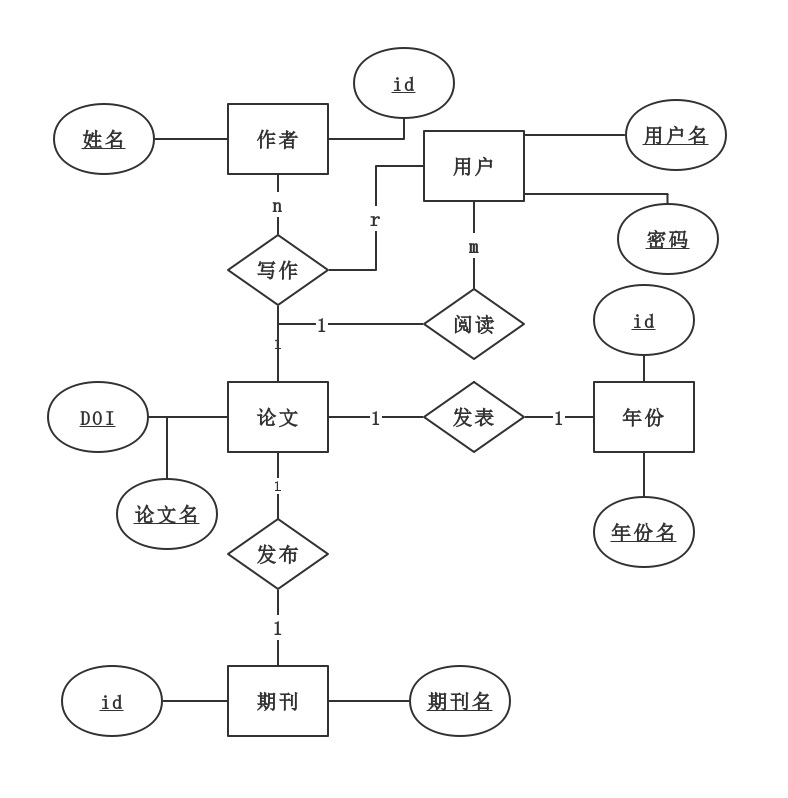


图4.2 E-R图

本数据库以论文名为核心进行设计，每篇论文拥有一个发布年份、每篇论文拥有多个作者，每篇论文拥有多个阅读的用户、每篇论文拥有一个发表的期刊，与此同时用户中的作者也可以与论文有写作关系。

同时每个实体拥有各自的属性：每个作者拥有姓名属性，每个用户拥有用户名和密码属性，每篇论文拥有论文名和DOI号，每个期刊拥有期刊名，每个年份拥有年份名。

### 4.2.2 逻辑设计

本系统数据库的逻辑设计即将概念设计中的E-R图转化为关系模型如下：

* 论文（论文名，DOI号） 实体表；
* 作者（作者名） 实体表；
* 年份（年份名） 实体表；
* 期刊（期刊名） 实体表；
* 用户（用户名，密码） 实体表；
* 写作（论文名，作者名） 关系表；
* 发布（论文名，年份名） 关系表；
* 发表（论文名，期刊名） 关系表；
* 所属（期刊名，作者名） 关系表。

### 4.2.3 物理设计

本系统数据库的物理设计采用MySQL暂存数据表，之后通过Navicat工具将数据表导出成csv文件，为后续导入Neo4j图数据库做准备。本部分主要介绍数据表的格式。

表4.1为论文的实体表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.1 论文表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| name:ID(Article-ID) | varchar | 500 | 否 | 否 | 论文名 |
| doi | varchar | 500 | 否 | 否 | DOI号 |

表4.2为作者的实体表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.2 作者表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| id | int | 11 | 否 | 否 | id号 |
| name:ID(Author-ID) | varchar | 200 | 否 | 否 | 作者名 |

表4.3为期刊的实体表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.3 期刊表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| id | int | 11 | 否 | 否 | id号 |
| name:ID(Journal-ID) | varchar | 200 | 否 | 否 | 期刊名 |

表4.4为年份的实体表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.4 年份表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| id | int | 11 | 否 | 否 | id号 |
| name:ID(Year-ID) | varchar | 45 | 否 | 否 | 年份名 |

表4.5为用户的实体表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.5用户表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| name:ID(User-ID) | varchar | 200 | 否 | 否 | 用户名 |
| password | varchar | 200 | 否 | 否 | 密码 |

表4.6为写作的关系表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.6 写作表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| :START\_ID(Article-ID) | varchar | 500 | 否 | 否 | 论文名 |
| :END\_ID(Author-ID) | varchar | 200 | 否 | 否 | 作者名 |

表4.7为发布的关系表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.7 发布表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| :START\_ID(Article-ID) | varchar | 500 | 否 | 否 | 论文名 |
| :END\_ID(Year-ID) | varchar | 45 | 否 | 否 | 年份名 |

表4.8为发表的关系表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.8 发表表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| :START\_ID(Article-ID) | varchar | 500 | 否 | 否 | 论文名 |
| :END\_ID(Journal-ID) | varchar | 200 | 否 | 否 | 期刊名 |

表4.9为所属的关系表格式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表4.9 所属表 | | | | | |
| 字段 | 类型 | 长度 | 空值 | 主键 | 字段解释 |
| :START\_ID(Journal-ID) | varchar | 200 | 否 | 否 | 期刊名 |
| :END\_ID(User-ID) | varchar | 200 | 否 | 否 | 作者名(用户) |

所有数据表的字符集使用的是utf8mb4，排序规则是utf8mb4\_general\_ci。

## 4.3 系统功能结构

本系统的功能结构图如图4.3所示。

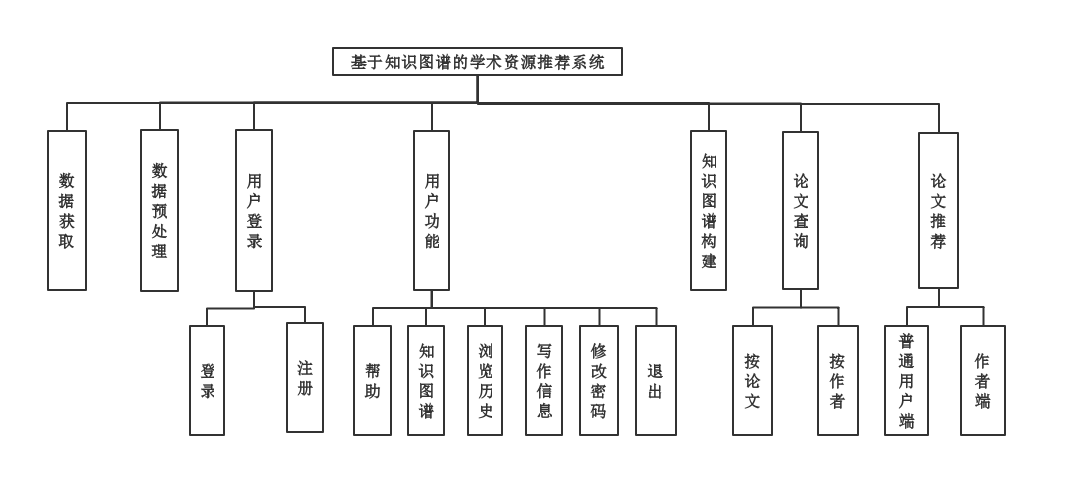


图4.3 系统功能架构图

### 4.3.1 数据获取模块

本系统使用的数据集为xml格式数据集，使用python的SAX方法对数据集进行解析，主要需要完成的功能是获取论文名、作者名、期刊、年份、DOI号等信息。

### 4.3.2 数据预处理模块

由于数据集解析后存在着大量重复、垃圾数据，因此本模块的功能是将这一部分数据剔除掉，使用的工具为Emeditor大文本编辑器，能够完成删除重复行以及特殊字符的替换功能。

### 4.3.3 用户登录模块

本模块需要完成的功能为用户登录并判断用户名密码是否正确；用户的注册两个功能，通过对于后端数据库信息的查找比对完成这一功能的实现。

### 4.3.4 用户功能模块

本模块所完成的功能是当用户登录成功进入推荐界面后，在界面的左侧栏所能使用的功能。包括知识图谱的查看，浏览历史的查看，写作信息的查看（当用户类型为作者时），修改用户密码、帮助和退出这六大功能。查看知识图谱功能的实现使用的是Neo4j Browser工具；浏览历史功能和写作信息功能的实现利用对于数据库的查询实现；修改密码将对数据库进行Updata操作实现；退出功能则需要完成用户退出时用户信息的保存工作。

### 4.3.5 知识图谱构建模块

知识图谱构建模块与此前的数据库设计模块紧密相关。通过MySQL数据库中表的数据，将其导出成csv格式数据，并放入Neo4j数据库的import文件夹下，再使用neo4j-import工具导入数据库，构建出知识图谱。之后再利用Neo4j Browser实现知识图谱的展示。

### 4.3.6 论文查询模块

本模块所要实现的功能是论文或作者的模糊搜索以及显示功能。通过Neo4j数据库的正则表达式功能实现论文或者作者的模糊查询。将查询的结果显示在弹窗中，然后用户选中论文后可在论文信息显示窗口查看论文，该模块主要完成的功能即为这两大部分。

### 4.3.7 论文推荐模块

论文推荐模块总体需要实现普通用户端和作者端的论文推荐。在普通用户端使用基于物品的协同过滤推荐算法，在作者端使用基于用户的协同过滤推荐算法，通过这两种算法返回20个推荐结果，并将结果显示在推荐窗口。同时该模块功能还包括对于论文的喜好选择（喜欢或者不喜欢），系统将根据喜好增加或减少相似论文推荐；对于论文的推荐理由查看，在普通用户端，推荐理由将与用户历史浏览信息有关，在作者端，推荐理由则相似作者有关。该模块也包括对于推荐论文的查看功能，将通过对后端数据库的操作实现并显示在论文信息窗口。

## 4.4 本章小结

本章首先介绍了系统的整体架构，本系统采用三层架构实现。之后介绍了数据库的设计，从概念设计、逻辑设计、物理设计三个层次描述。之后大致介绍了本系统需要实现的七大功能模块包括数据获取模块、数据预处理模块、用户登录模块、用户功能模块、知识图谱构建模块、论文查询模块和论文推荐模块。

# 5 系统实现

## 5.1 DBLP数据获取

上文的系统分析中已经对于DBLP获取的数据集格式进行了着重介绍，本部分将主要实现数据集的解析及后续工作。

对于xml数据集的解析需要导入xml.sax库中的parse方法对xml进行解析，同时还需要导入xml.sax.handler库中的ContentHandler类。首先使用xml.sax的make\_parser方法创建xml解析器，建立article类为ContentHandler的继承类，使用setContentHandler重写ContentHandler的characters、startElement和endElement方法。其中startElement为开始标签时调用，主要用于初始参数的设定。endElement为结束标签时调用，主要用于判定一篇论文所有信息解析结束后将信息写入txt文本文件。Characters函数使用content变量将开始标签和结束标签中的字符串记录下来，并放在对应的变量中。由于SAX方法是逐行对于xml进行解析，因此新解析的数据会对content中的旧数据进行覆盖，因此不用担心数据的混乱。

具体的解析实现以数据库设计中的五张实体表和四张关系表的结构对于数据集进行解析，可以得到九个txt文件，九个txt文件内容对应着九张表的数据。如图5.1所示为论文表的txt格式。

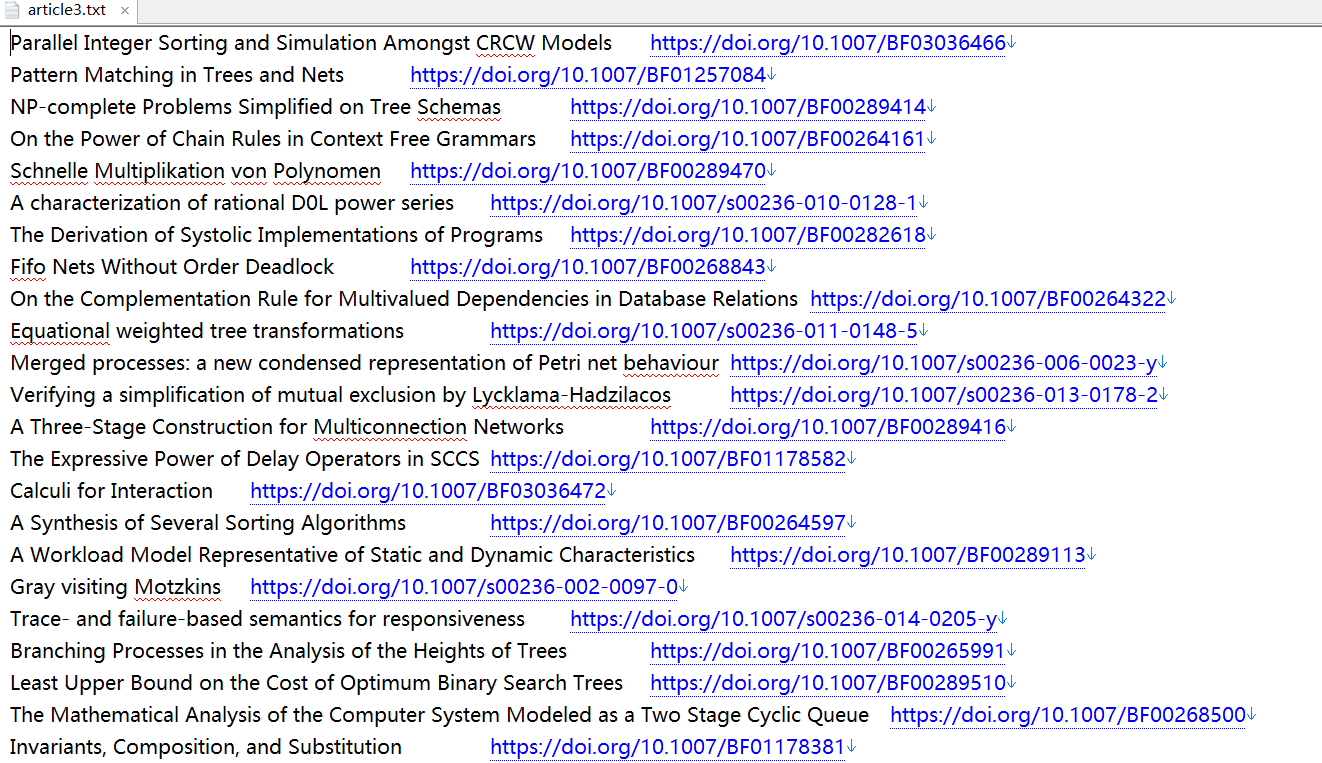


图 5.1 数据获取论文表txt格式

其余表的数据与论文表数据类似，此处不一一展示。

整个数据获取的流程图如图5.2所示。

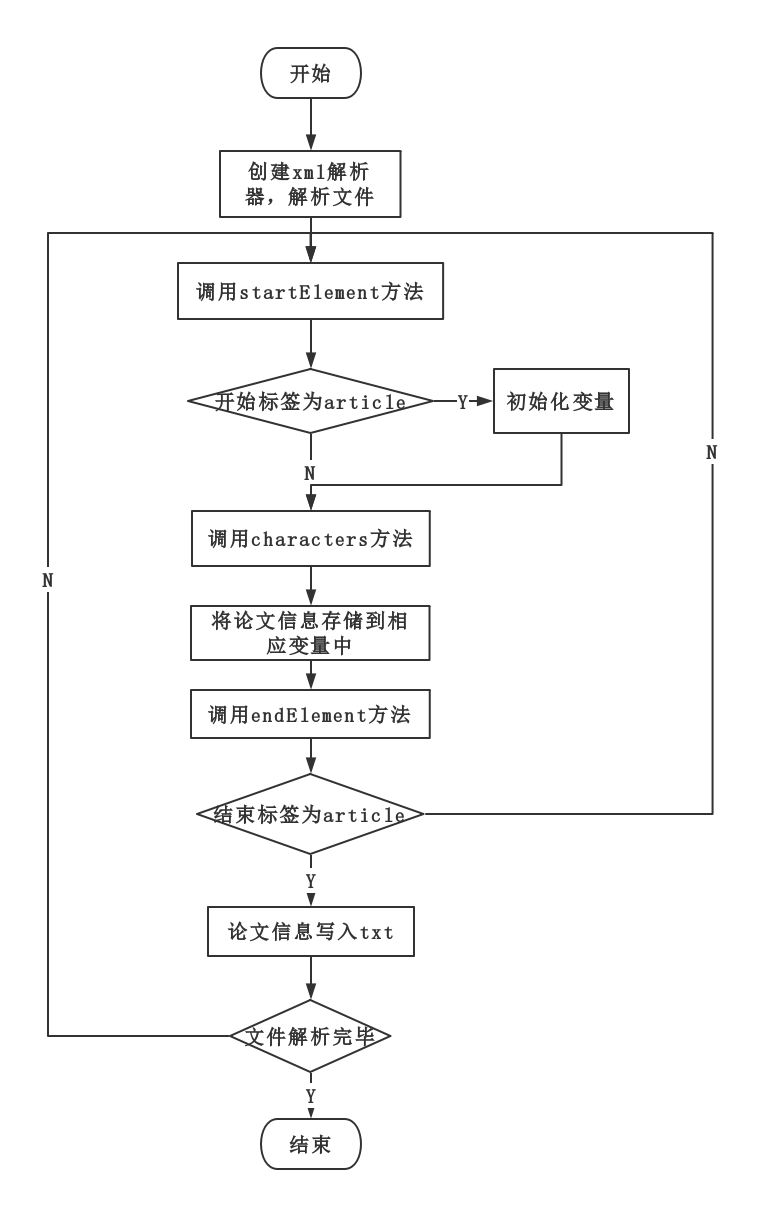


图5.2 数据获取流程图

## 5.2 数据预处理

本部分为对获取得到的txt格式的数据导入至MySQL数据库，并用Navicat工具将数据导出成csv格式，之后对于数据进行简单的数据清洗，为之后导入图数据库构建知识图谱做准备工作。

首先需要完成的工作是将txt格式的数据导入MySQL数据库。由于数据体量较大，因此选择了MySQL的LOAD DATA INFILE工具，其可以使得数据高速导入MySQL数据库，使用LOAD DATA INFILE工具过程如图5.3所示，此为导入期刊作者关系表，其余表的操作类似，此处不一一展示。



图5.3 LOAD DATA INFILE工具

将数据以第4章数据库设计的格式导入MySQL数据库，如图5.4所示为论文表的展示，其余表类似。

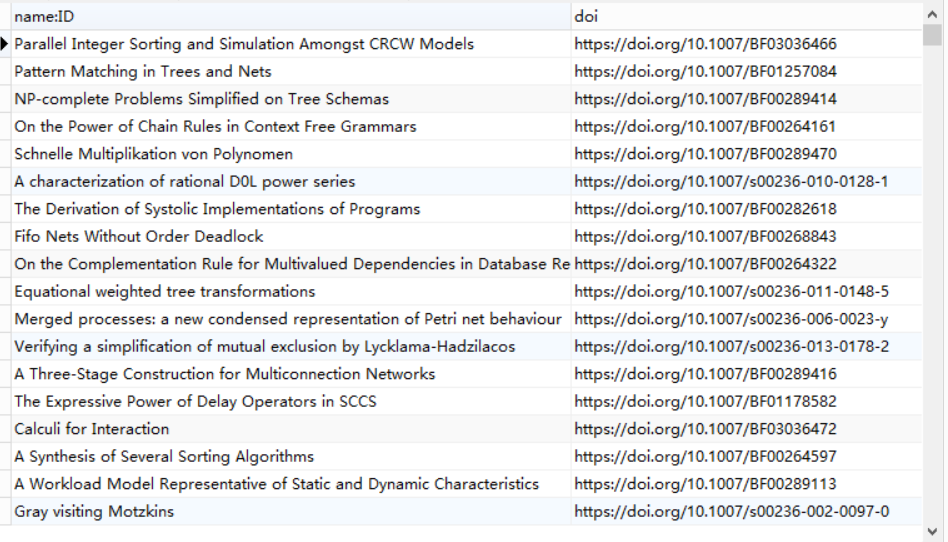


图5.4 论文表

将实体表和关系表都导入数据库后，使用Navicat工具将MySQL数据导出成csv格式数据。导出过程和导出结果如图5.5和图5.6所示。



图5.5 csv数据导出

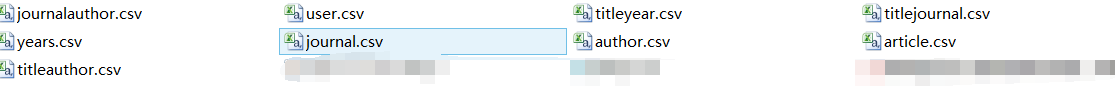


图5.6 csv数据导出结果

生成csv格式数据后需要对于数据进行简单的数据清洗工作。本部分采用Emeditor对于数据进行清洗。进行清洗的工作包括删除重复行数据，使用查找替换功能删除特殊符号如’、”等。

## 5.3 知识图谱构建与展示

本系统知识图谱的构建采用Neo4j图数据库，知识图谱的展示采用Neo4j的UI：Neo4j Browser。

本部分的实现步骤共分为三步：

第一步为将九个csv格式文件放入neo4j-community-3.4.0的import文件下；

第二步使用Neo4j的导入工具Neo4j-import将csv数据导入图数据库。导入工具使用命令行执行相关语句。具体操作如图5.7所示。

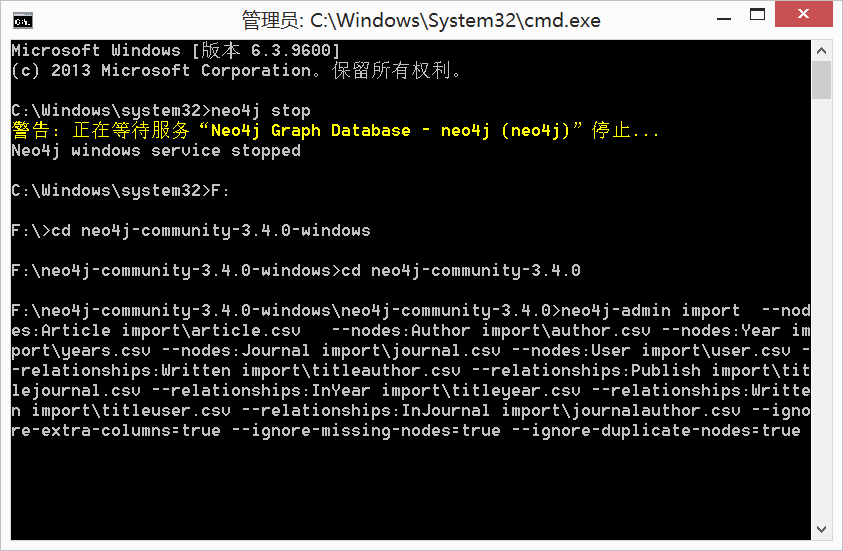


图5.7 Neo4j import工具

其中包括导入节点的语句—nodes、导入关系的语句—relationship、以及对于脏数据的处理语句—ignore-extra-columns忽视多余列、--ignore-missing-nodes忽视缺少属性的节点、--ignore-duplicate-nodes 忽视重复节点。

导入csv数据后结果如图5.8所示。

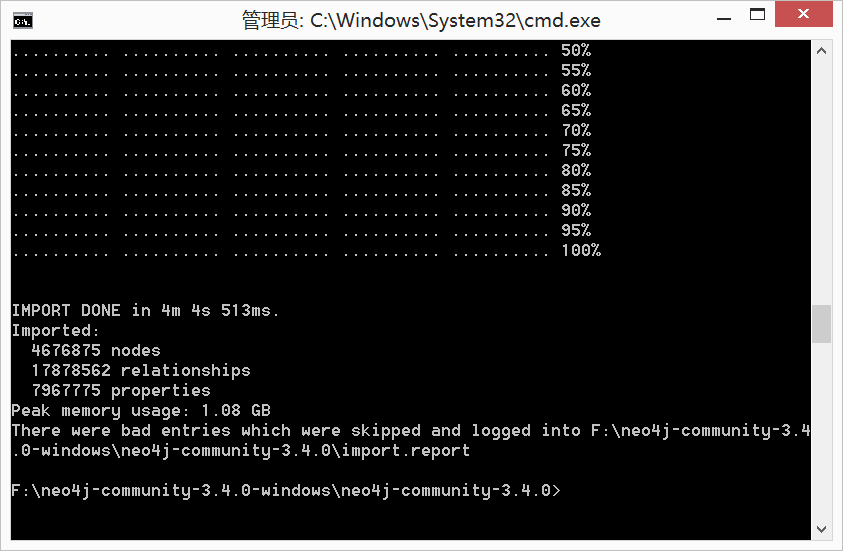


图5.8 导入结果

导入后完成知识图谱的构建。

对于知识图谱的展示，采用Neo4j Browser进行知识图谱的展示。该UI的地址为localhost:7474/browser/。对于Neo4j Browser的版本使用2.0.0，下载neo4j-browser-2.0.0.jar,并将jar包放入neo4j-community-3.4.0下的lib文件中。为了方便在打开UI时即可展示知识图谱，在Neo4j Browser中点击设置，修改Initial Command为MATCH (n:Article) RETURN n LIMIT 25。

之后使用浏览器打开上文所说地址，即可直接展示知识图谱如图5.9所示。

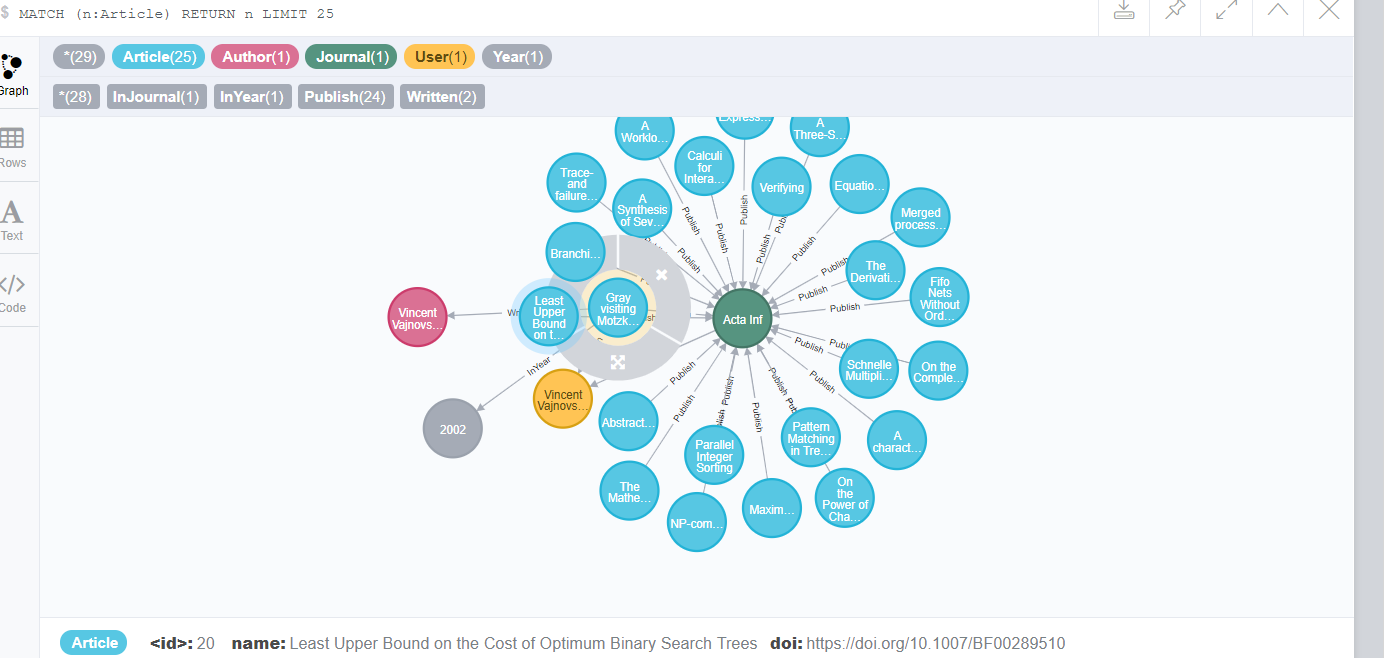


图5.9 知识图谱展示

## 5.4 用户界面编写

用户界面为用户刚使用本系统时所进入的界面。用户界面需要实现两个功能，一个是用户登录、一个是用户注册。

首先是用户登录功能，其将前端获得的用户信息在后端进行查询，存在则进入推荐系统。其实现流程图如图5.10所示。

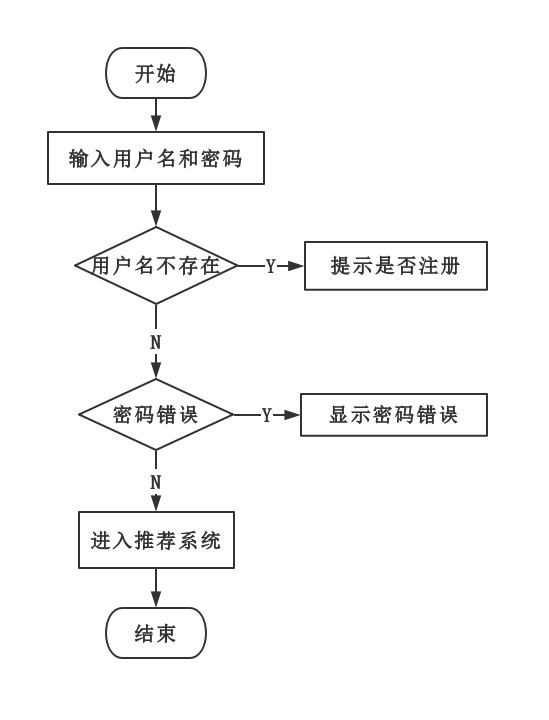


图5.10 用户登录流程图

实现后的用户登录界面效果如图5.11所示。

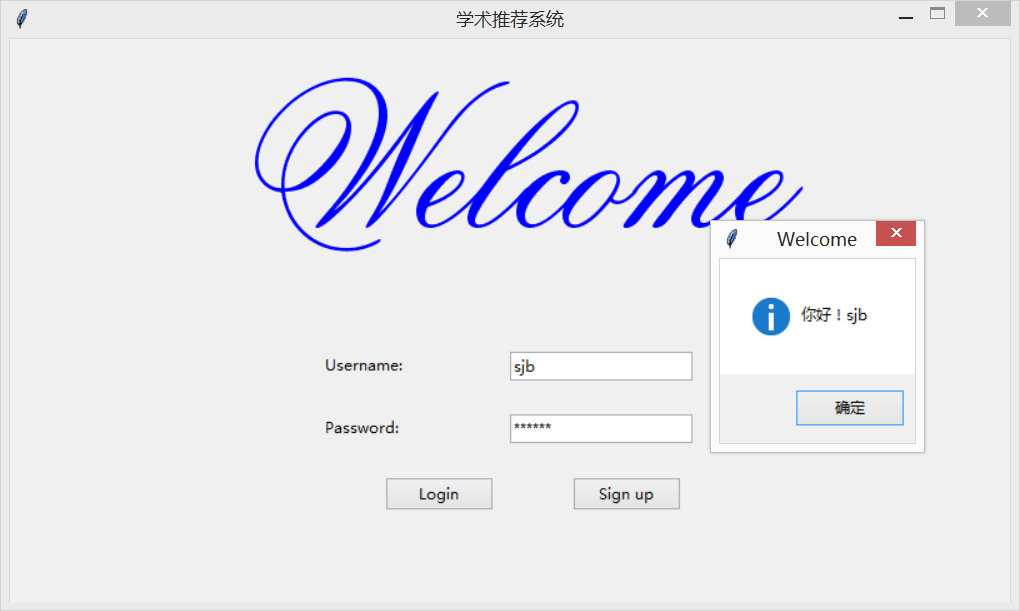


图5.11 用户登录界面

然后是用户注册功能，其将获得的用户信息以用户类型添加到数据库中。其实现流程图如图5.12所示。

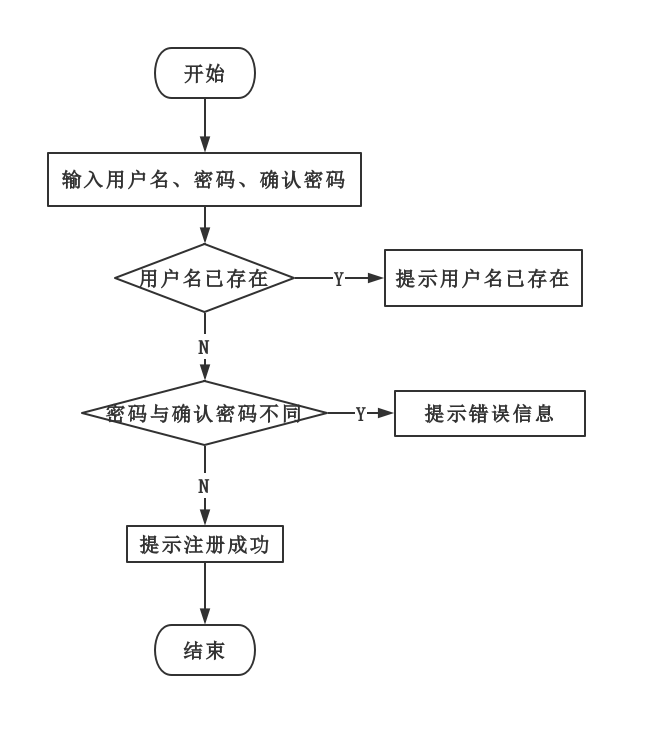


图5.12 用户注册流程图

实现后，用户注册界面效果如图5.13所示。

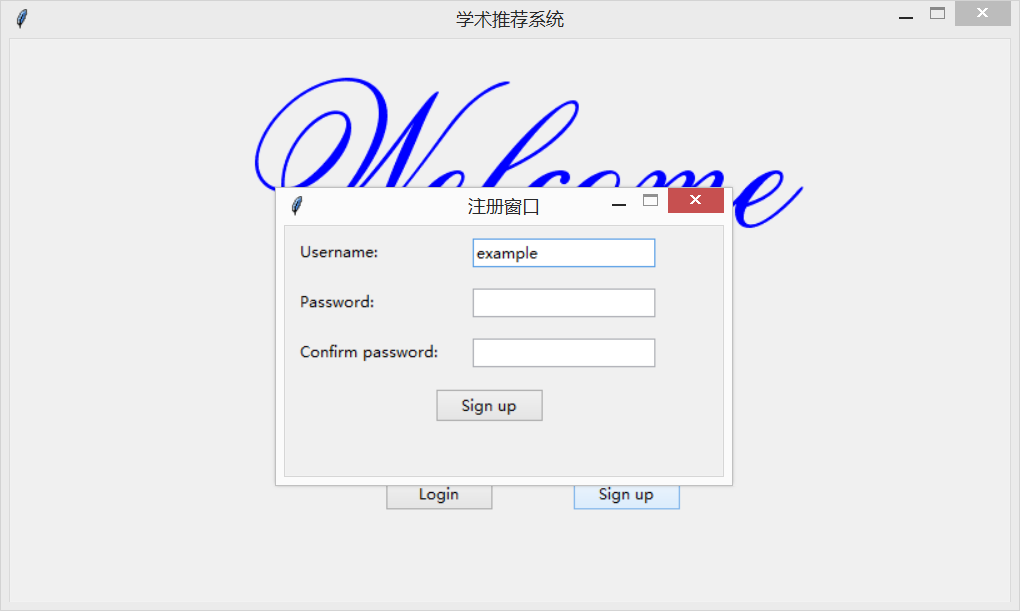


图5.13 用户注册界面

## 5.5 用户功能实现

本部分包括了推荐系统界面左侧栏用户可以操作的六大功能包括：知识图谱、浏览历史、写作信息、修改密码、帮助和退出。

首先，推荐系统界面总体风格如图5.14所示。



图5.14 推荐系统界面

接着依次介绍左侧栏六大功能。

第一个功能是知识图谱的查看，该功能的实现是点击知识图谱按钮，后端通过webbrowser.open()函数实现通过浏览器打开Neo4j Browser。实现效果如上文知识图谱的展示所示，此处不再重复展示。

第二个功能是浏览历史的查看，该功能的实现是点击浏览历史按钮，后端通过提取与当前操作用户存在阅读关系的论文。阅读关系（Read）是当系统运行后才会产生的关系，在数据库创建时不会生成。该关系具有两个属性click和timestamp，分别代表用户点击论文的次数和最后一次点击的时间。浏览历史通过比较该用户所有阅读论文中点击论文时间大小，逆序输出论文名。

浏览历史功能实现效果如图5.15所示。



图5.15 浏览历史

第三个功能为写作信息的查看，此处将首先判断用户是否为作者，如果是则会将该功能按钮添加到左侧栏。该功能的实现通过查找与用户拥有写作关系的论文，将论文名显示。

浏览历史功能实现效果如图5.16所示。



图5.16 写作信息

第四个功能为修改密码。通过修改后端数据库中用户的password属性实现。

其需要输入旧密码、新密码和确认密码。只有旧密码与当前用户的密码属性相同并且新密码与确认密码相同才能成功修改。

修改密码效果如图5.17所示。

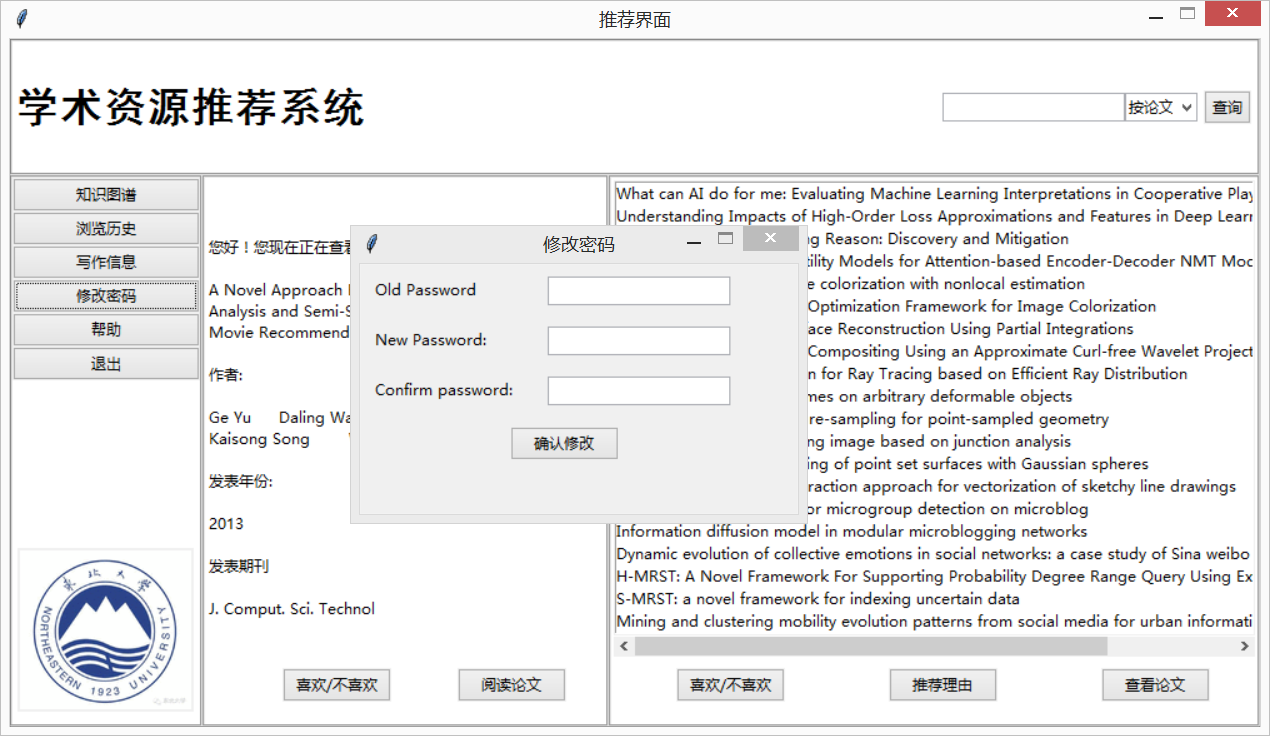


图5.17 修改密码

第五个功能为显示帮助信息。即点击按钮后将系统如何使用信息显示在界面中，此处不再展示。

第六个功能为退出功能，能够实现将用户的推荐论文信息进行保存，从而减少下次登录时系统运行推荐算法所耗时间，并且通过tkinter的destroy函数实现结束推荐系统界面，通过deiconify函数将之前隐藏的用户登录主界面显示功能。

## 5.6 论文查询功能实现

本部分实现的功能是在推荐系统界面上方的搜索框中实现对于论文和作者的模糊搜索。其实现需要依赖Neo4j的正则表达式功能。Neo4j所使用的Cypher语言继承java的正则表达式，=~是其使用的解析符号，(?i)代表忽视大小写，.\*类似于sql语句中的%。因此使用Cypher语句实现模糊搜索的表达式可以写成=~’(?i).\*keyword.\*’。

该功能的实现流程图如图5.18所示。

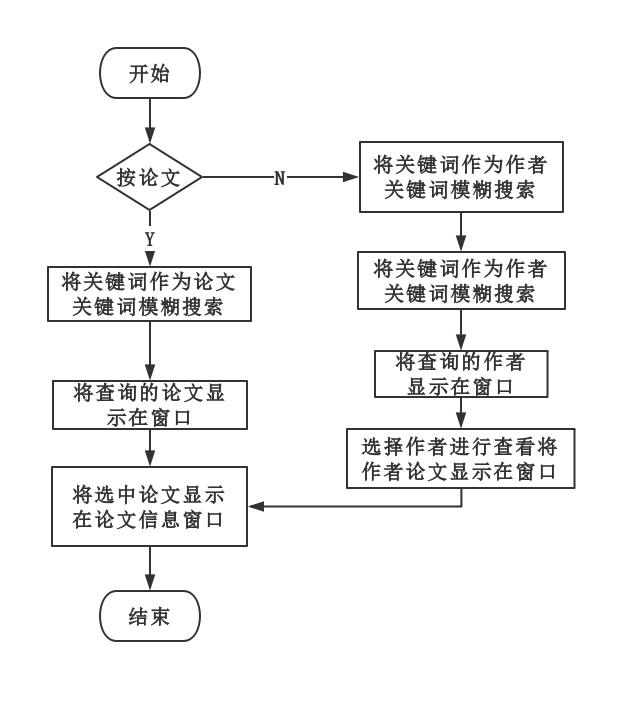


图5.18 论文查询流程图

该功能实现效果如图5.19、5.20和5.21所示。其中图5.19是根据论文查看，图5.20是根据作者查看，图5.21是查看的结果。

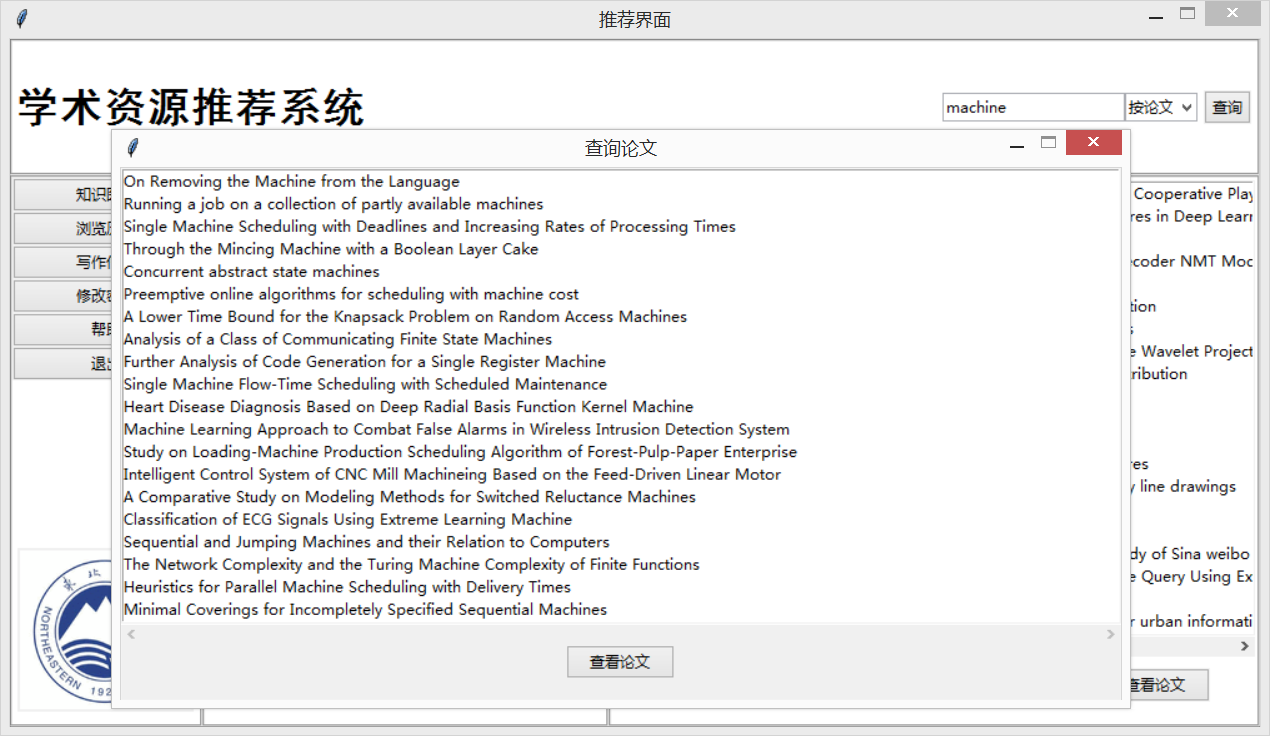


图 5.19 根据论文查询



图 5.20 根据作者查询



图5.21 查询结果

得到查询结果后，会有两个功能键喜欢/不喜欢和阅读论文。阅读论文即获得论文的DOI属性，打开浏览器进行阅读。喜欢/不喜欢功能与推荐模块的喜欢/不喜欢功能一致，将在5.7节详细介绍。

## 5.7 论文推荐功能实现

本部分为本系统的核心部分，其共分为两部分，一部分是普通用户端的推荐，另一部分是作者端的推荐。

### 5.7.1 用户端推荐功能实现

用户端的推荐算法采用的是基于物品（论文）的协同过滤算法（ItemCF），其算法的基础已在上文的关键技术介绍中进行了初步说明。本部分将结合本系统的具体情况进行进一步说明。

首先是对于Jaccard相似度算法在本算法中的实现。有上文的关键技术介绍得到的Jaccard公式为公式5.1。

 (5.1)

本推荐算法需要求得两篇论文之间的相似度，即将两篇论文*PA*和*PB*作为公式中*PA*、*PB*集合。*PA*和*PB*的交集即为两篇论文在期刊、年份和作者当中相同的元素集合，*PA*和*PB*的并集即为两篇论文包括期刊、年份和作者所有元素的集合，计算Jaccard相似度即用*PA*和*PB*的交集除以*PA*和*PB*的并集得到Jaccard系数。

其次是对于基于论文的协同过滤算法（ItemCF），该算法的公式为公式5.2。

 (5.2)

下就该公式如何在本系统中具体应用做进一步分析。首先*N*(*u*)为用户*u*的历史浏览论文，而*S*(*j,k*)为与论文*j*相似度为前*K*个的论文，由于本系统数据集较大，无法完成相似矩阵的实现，因此对于公式进行简化，取*K*为论文数据集的大小。对于*rui*即用户对于论文*i*的感兴趣程度，本系统通过隐式反馈即用户的点击实现。当用户*u*点击过论文*i*即用户*u*查看过论文*i*，则*rui*=1，否则*rui*=0。由于*i*为*N*(*u*)的子集，因此可以得到*rui*=1。经过上述处理过后的公式即为公式5.3。

 (5.3)

因此仅需将用户的所有阅读过的论文作为集合*M*，将其余所有的论文与集合*M*中论文进行相似度的计算，并计算相似度和，取相似度和最高的20篇论文，即为推荐结果。

为了提高用户对于推荐结果的满意度，通过关系Read中的属性click选择了用户点击量前*K*篇论文作为集合*M*，此处取*K*=10。

该算法的实现如算法5.1所示。

|  |
| --- |
| **算法5.1 基于论文协同过滤推荐算法** |
| **Input:** 用户名，*K*值  **Output:** 20篇推荐论文名  **相关的描述：**  **Begin**   1. 在数据库中查询该用户 2. 将用户所有阅读过的论文按点击量排序，取前*K*篇论文作为集合*M* 3. **For each** 其余用户未读且不是不喜欢的论文作为集合other 4. 寻找该论文与集合*M*在期刊、年份、作者中相同元素，作为集合*T*，集合*T*的个数为|*T*|=intersection 5. 将集合*M*在期刊、年份、作者中所有元素作为集合*MT* 6. 将该论文在期刊、年份、作者中的所有元素作为集合*OT* 7. 取集合*MT*和*OT*的并集个数为|*MT*|∪|*OT*|=union 8. 取Jaccard=intersection/union 9. **End for** 10. 以集合other做聚类运算，计算other中每篇论文对于集合*M*论文的相似度和 11. 选择相似度和中最高的20个，保存其论文名   **End** |

获得20篇论文后，将其显示在论文推荐窗口，同时将生成三个功能键为喜欢/不喜欢、推荐理由和查看论文。

推荐理由功能的实现是通过计算选定论文和用户阅读的所有论文进行上文推荐算法所说的Jaccard相似度计算，计算出的Jaccard系数最高的用户阅读论文即为推荐理由。

具体实现效果如图5.22所示。

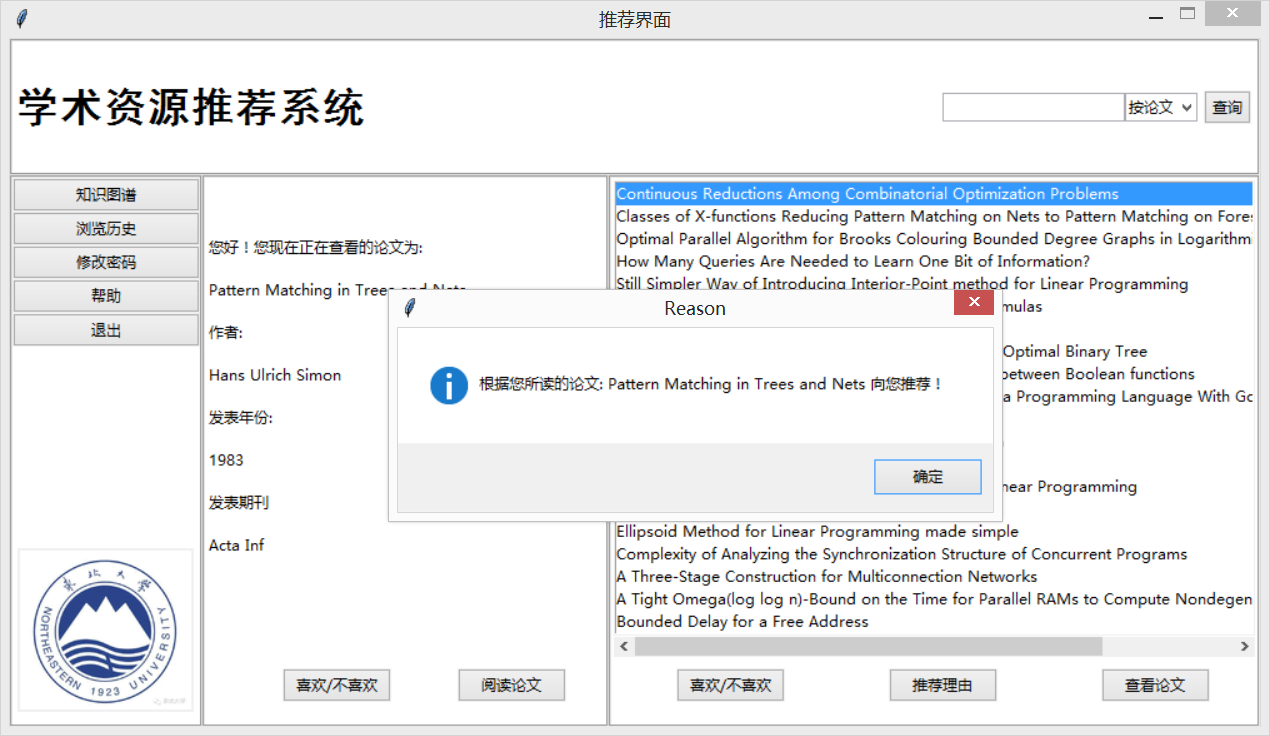


图5.22 推荐理由功能展示

喜欢/不喜欢功能则根据选定的论文，通过Jaccard相似度算法计算，找到Jaccard系数最高的3篇论文。如果目前用户与该论文的关系是Read，则将这几篇论文与用户的关系全部改为Dislike；如果目前用户与论文的关系是Dislike，则将这几篇论文与用户的关系全部改为Read。

具体实现效果如图5.23所示。

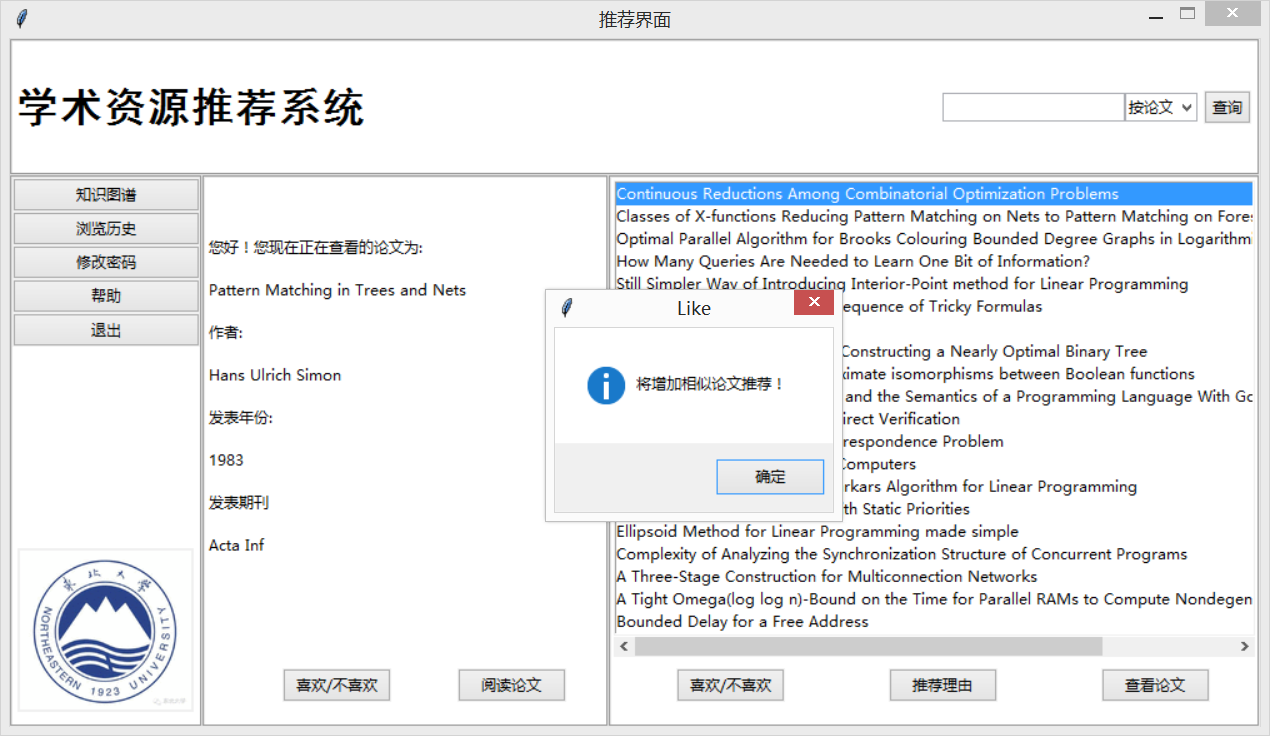


图5.23 喜欢/不喜欢功能展示

查看论文功能在论文查询模块已经进行了介绍，该功能与其一致，也是将选定论文信息显示在论文信息显示窗口。此处不再赘述。

### 5.7.2 作者端推荐功能实现

作者端的推荐算法采用的是基于用户的协同过滤算法（UserCF），该算法的基础介绍亦在上文的关键技术介绍中提及，本部分将结合本系统情况进一步分析。

首先是对于Jaccard相似度算法在本算法中的实现。有上文的关键技术介绍得到的Jaccard公式为公式5.4。

 (5.4)

该算法对于Jaccard算法的使用与ItemCF不同，它无需得到论文之间的相似度，而是需要得到用户之间的相似度。将当前登录推荐系统的用户作为用户*UA*，而将其余用户依次作为用户*UB*，计算*UA*和*UB*的相似度。*UA*和*UB*的交集即为*UA*和*UB*的论文所发表于的期刊和写作论文中相同的元素，*UA*和*UB*的并集则为*UA*和*UB*所有发表的期刊和写作论文元素集合，Jaccard系数为交集的元素个数除以并集的元素个数。取系数最大的前10个用户作为与当前目标用户最相似的用户集合。

其次是基于用户的协同过滤推荐算法在本系统中的应用，该算法的基本介绍在上文关键技术中已进行了介绍，本部分将分析该算法结合本系统情况进行具体实现。该算法的公式为公式5.5。

 (5.5)

本系统将写作关系作为用户反馈，当用户*v*写过论文*i*时，*rvi* =1，否则*rvi*=0。由于*v*为*S*(*u*,*k*)和*N*(*i*)交集的子集，所以用户*v*集合一定是与用户*u*相似度最高的前*K*个用户的子集。因此当论文*i*不属于用户集合*v*所写作的论文时，*rvi*恒等于0。所以可以将论文*i*的集合范围减少到与用户*u*相似度最高的前*K*个用户所写的论文集合。该公式则可理解为对于一篇论文*i*，用户*u*对其的兴趣为所有该论文的作者（在集合*S*(*u*,*k*)中）与用户*u*的相似度和。之后对于*pui*按照相似度系数进行排序，取前20个即为推荐结果。系统具体实现中取*K*=10。

该算法的实现如算法5.2所示。

|  |
| --- |
| **算法5.2 基于用户协同过滤推荐算法** |
| **Input:** 用户名，*K*值  **Output:** 20个推荐论文名  **相关的描述：**  **Begin**   1. 在数据库中查询该用户 2. **For each** 其余用户 3. 寻找目标用户与该用户在期刊和论文中的相同元素，作为集合*T*，集合*T*的个数为|*T*|=intersection 4. 将目标用户在期刊和论文中所有元素作为集合*MT* 5. 将该用户在期刊和论文中的所有元素作为集合*OT* 6. 取集合*MT*和*OT*的并集个数为|*MT*|∪|*OT*|=union 7. 取Jaccard=intersection/union 8. **End for** 9. 选择Jaccard中最高的*K*个作者，作为集合other 10. 寻找集合other中用户所有的写作论文，作为集合*P* 11. 每篇论文的感兴趣度为其所有作者（在集合other中）Jaccard系数和 12. 选择感兴趣度最高的20个结果，保存其论文名   **End** |

获得20篇论文后，将其显示在论文推荐窗口，同时也将生成三个功能键为喜欢/不喜欢、推荐理由和查看论文。

喜欢/不喜欢功能与普通用户端的喜欢/不喜欢功能一致，此处不再赘述。

查看论文功能也与普通用户端查看论文功能一致，此处不再赘述。

推荐理由功能与普通用户端不同，其采用的推荐理由是根据选定的论文的作者进行解释，因此其功能实现的思路是找到选定论文的作者，并保证该作者属于相似作者集合，即为推荐理由。

具体实现如图5.24。

5.8 本章小结

本章主要对系统实现的具体流程进行了介绍，包括数据获取、数据预处理、知识图谱构建、用户界面、用户功能、论文查询和论文推荐七大部分。

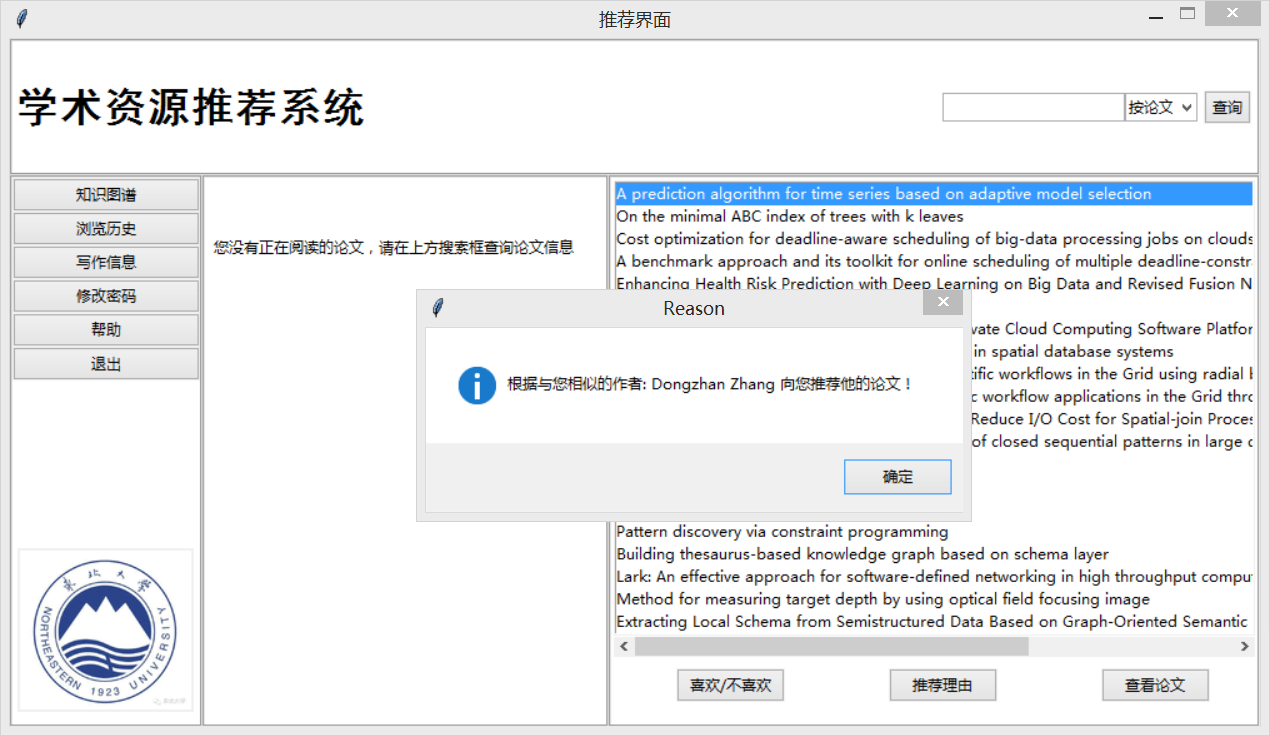


图5.24 作者端推荐理由

# 6 结论

## 6.1 本文工作总结

随着学术资源近年来急速的增长，研究人员感兴趣的学术资源的高效获取成为了一大难题。本文正是基于此现状进行研究，通过推荐系统解决信息过载导致的用户无法高效获得自己需要的学术资源的问题，通过知识图谱提高推荐系统的精确性、多样性和可解释性。

基于此，本文设计并实现了一个基于知识图谱的学术资源推荐系统。其主要工作总结如下：

（1）数据获取。从DBLP学术网站获取xml数据集，并对于数据集通过SAX方法进行解析，获得txt格式数据；

（2）数据预处理。将txt格式数据通过MySQL的load data infile工具导入MySQL，并通过Navicat生成csv格式数据，使用Emeditor工具删除重复数据和垃圾数据；

（3）知识图谱构建。通过Neo4j import工具将csv格式工具导入Neo4j图数据库，生成知识图谱，通过Neo4j Browser展示知识图谱；

（4）推荐算法研究。研究基于物品的协同过滤算法和基于用户的协同过滤算法分别应用于普通用户端和作者端；

（5）用户功能实现。实现用户登录界面以及用户基础功能包括论文的模糊搜索等；

（6）原型系统实现。集上述功能于一体，设计并实现了一个原型系统，并将整个系统使用tkinter进行可视化。

## 6.2 进一步展望

本文工作所开发系统完成了基于知识图谱的学术资源推荐系统的基本功能，但是由于研究工作时间和能力的局限性，该系统仍存在大量的改善空间，主要包括以下几个方面：

（1）作为本系统最为核心的推荐算法，由于数据集整体规模较大，因此在推荐算法的时间效率方面并不是特别尽如人意，本系统的解决方案是在用户打开论文的同时进行推荐算法的计算。这不失为一个解决方案，但是时间效率的局限性仍然使得系统无法实现在用户与系统有任何交互行为的情况下，进行推荐算法，只能满足在有关键交互行为的情况下运行推荐算法，以保证系统整体交互的友好性，此为之后需要改进的方面；

（2）系统的设计基于数据集，因此是相对比较封闭的一个系统。对于数据的更新必须重新下载数据集，并重新对于数据库进行导入。数据库的更新相对不太方便，这也是需要改进的方面；

（3）系统前端界面的美观性不足，虽然能够满足基本的需求，但受Tkinter的局限性，在美化方面仍需改善。

# 参考文献

[1] Resnick P, Varian R H. Editor Recommender Systems[C].Communications Of The ACM,1997,40(3):56-58.

[2] Resnick P, Iacovou N, Suchak M. GroupLens: an open architecture for collaborativefiltering of netnews[C].Proceedings of the 1994 ACM Conference on ComputerSupported Cooperative Work, 1994:175-186.

[3] Adomavicius G, Tuzhilin A. Toward the nextgeneration of recommender systems: a survey of the state-of-the-art and possibleextensions[J]. IEEE Transactions on Knowledge and Data Engineering, 2005,17(6): 734-749.

[4] Aciar S, Zhang D, Simoff S, Debeham J. Informed Recommender: Basing Recommendations on Consumer Product Reviews[M]. Recommender Systems,2007.

[5] Karatzoglou A, Hidasi B. Deep Learning forRecommender Systems[C]. the Eleventh ACM Conference, ACM, 2017:396-397.

[6] 刘峤, 李杨, 段宏. 知识图谱构建技术综述[J]. 计算机研究与发展, 2016, 53(3): 582–600.

[7] Szekely P, Knoblock C A, Slepicka J, et al. Building and using a knowledge graph to combat human trafficking[C] 2015 International Semantic Web Conference on. New York, USA: Springer-Verlag, 2015:205–221.

[8] 常亮. 知识图谱的推荐系统综述[J]. 智能系统学报, 2019(02).

[9] [康杰华](http://www.wanfangdata.com.cn/details/detail.do?_type=perio&id=xxjs201506030), [罗章璇](http://www.wanfangdata.com.cn/details/detail.do?_type=perio&id=xxjs201506030). 基于图形数据库Neo4j的RDF数据存储研究[J]. [信息技术](http://www.wanfangdata.com.cn/details/detail.do?_type=perio&id=xxjs201506030), 2015, (6):115-117.

[10] 徐增林, 盛泳潘, 贺丽荣,等. 知识图谱技术综述[J]. 电子科技大学学报, 2016, 45(4):589-606.

[11] Greg L，Brent S，York J. Amazon.com recommendations: item － to － item collaborative filtering[J]. IEEE Internet Computing, 2003, 7(1): 76－80．

[12] 项亮, 推荐系统实践[M]．北京:人民邮电出版社，2012.

[13] Anand D, Bharadwaj K K. Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities [J]. Expert Systems with Applications, 2011, 38(5):5101-5109.

[14] Bobadilla J, Ortega F, Hernando A. A collaborative filtering similarity measure based on singularities[J]. Information Processing and Management, 2012, 48:204－217.

[15] 许海玲，吴潇，李晓东，等．互联网推荐系统比较研究 [J]．软件学报，2009，20(2):350-362．

# 致谢

本课题的研究工作的顺利完成，我首先要感谢王大玲老师对于我研究工作的悉心指导，包括选题阶段、代码实现阶段以及论文阶段，王大玲老师对于我的疑问都是耐心指导和提出意见。王老师严谨的教学态度和踏实负责的工作态度令我感到钦佩，也再次由衷地感谢王大玲老师。

其次，要感谢东北大学对于我四年的培养，让我学习到了很多东西，这无论对于这次课题的研究或是我今后的学习工作都有巨大的帮助。

然后还要感谢在此次课题研究过程中帮助过我的老师、同学和学长学姐，没有他们的帮助，我的研究工作也无法顺利的完成。

最后需要感谢我的家人，他们对于我在学校的求学提供坚实的后盾，能够让我安心的完成学校的工作。

感谢所有帮助过我的老师、同学、家人，我会继续努力，好好完成以后的工作。