各位老师，上午好!

——————————————————————————

我叫陈小龙。

我的论文题目是《基于Neo4j的研究团体搜索系统设计与实现》。

这篇论文是在祝园园老师的指导下完成的。

——————————————————————————

在这里我要对我的导师表示

和

参加我的论文答辩各位老师表示感谢！

——————————————————————————

下面，我将本论文设计的目的和主要的内容向各位老师进行汇报，恳请各位老师批评指导！

——————————————————————————

我主要从一下几个方面进行介绍：

————————————————

首先第一部分：主要介绍论文的研究背景。以及研究的意义

——————————————————

第二部分介绍论文的研究思路和方法

——————————————————

第三部分主要介绍本论文的研究难点，并对研究成果进行展示

——————————————————

第四部分对论文进行总结

背景：

不同学者之间的合作成果往往以徐文书论文的形式发表。

学术研究的发展，产生了海量的学术论文。学术网络就隐藏在海量的学术文献之中。

——————————————————————————————————————

意义：

使用DBLP文献库构建学术网络

使用Neo4j作为数据存储工具。

集成不同的图论算法：中心度算法、相似度算法、社区检测算；提供了研究团体搜索、相似度查询、中心度查询等功能

首先是国内现状：

1、百度、知网、学者网：都有一个共同的特点：都只能工具关键词进行匹配搜索。

2、这种方式找到的信息都是与关键词直接相关的信息。

3、当要查找的信息与关键词不直接相关的时候，这些工具就无能为力。 （比如，通过作者姓名找到其所有合作者）

清华大学唐杰团队的 Aminer :

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。

——————————————————————————————————————————————————————————————————————

其次是国外现状：

1、谷歌学术搜索、微软学术搜索，也是只能满足通过关键词匹配信息，进行查询

——————————————————————————————————————————————————————————————————————

无论是国内的百度、知网、Aminer 还是国外的 谷歌学术搜索、微软学术搜索都不能满足研究团体搜索、接待相似度查询等功能。

本论文的目的就在于设计开发一个检索学术网络中研究团体、节点关联信息的平台。

关键字：DBLP

1、DBLP包含期刊、会议所发表的论文；也包含部分博士硕士论文。收录的数据丰富，质量很高，被学术界广泛认可。

2、本文只抽取期刊会议发表的论文数据进行分析

————————————————————————————————————————————————————

关键字：Graph

1、将从DBLP文献库中 抽取的数据进行处理，构造成一张学术网络图；在图上进行数据分析与检索

2、图中包含两类节点：一类是作者对应的节点，一类是文章对应的节点

3、图中包含两类边：一类是作者和文章之间的边（表示作者和文章的从属关系）；一类是作者之间的边（表示作者之间的合作关系）

————————————————————————————————————————————————————

关键字：Neo4j

1、Neo4j是一个专用的图数据库

2、Cypher语言受SQL的启发，通过模式匹配，实现对图图数据的增删改查

3、可以扩展其算法库，对图数据进行分析

——————————————————————————————————————————————————————

关键字：社区搜索

1、通过在Neo4j上集成社区搜索算法，实现研究团体搜索功能

2、处理传统的结构密集的社区搜索算法（如：Louvain算法）；还包括结构和属性都密集的社区搜索算法（Equitruss）

————————————————————————————————————————————————————————————

ATC算法：核心是首先找到结构紧密的社区，然后通过判断社区属性是够满足要求，对社区进行过滤；

判断属性是否满足要求：

1、属性编号，每一个属性都由唯一的整数ID

2、频繁项集

这一部分主要从：

数据建模&预处理、算法集成、功能实现

等几个方面介绍本文的研究方法和思路

主要从 数据建模与预处理、算法集成。系统实现架构三个方面 介绍研究方法

数据建模&预处理

在数据建模阶段，需要从文献的 “作者之间的关系” 和 “作者与文章之间的关系”，构建学术网络。

数据建模具体来说：

学术网络图是一个属性图， 并且包含两类节点和两类边

两类节点分别是：

* 作者节点：代表一位作者，节点的属性包括，作者姓名、发表过的文章等信息
* 文章节点：代表一篇文章，节点属性包括文章的标题、引用次数等信息
* 对文章标题使用nltk进行分词，并且给每一个词唯一编号，方便后续算法运行

两类边分别是：

* 作者之间的边：代表作者之间的合作关系，边的权重代表二者的合作次数
* 作者和文章之间的边：代表文章和作者之间的从属关系

数据预处理具体来说：

* 首先要从原始数据中提取节点和边的数据
* 代用流式解析的方式解析DBLP的XML元数据，提取学术期刊文献信息
* 每一个article标签就生成一个文章节点，同时如果文章中的作者则对应生成文章节点
* 一篇文章和它的每一个作者之间都生成一条边
* 同一篇文章的多个作者之间进行两两组合组合，每一个组合生成一条作者之间的边（作者之间有多条边以边的权重表示）
* 节点和边的属性

- 作者节点的属性（作者发表过的全部文章）：用作者名计算一个10位的hashcode，再对一个整数取模（比如：100），把作者散列到100个小文件中分别统计

* 作者之间边的权重：同边的两个端点ID，计算唯一hashcode，把边散列到不同的文件中，分别进行统计

——————————————————————————————————————————————————————————————————

算法集成：

本文所述系统的一个主要功能就是基于学术网络图进行数据分析，包括：

* 节点信息检索
* 节点相似度查询
* 节点中心度查询
* 研究团体搜索

要实现上述功能就必须借助于图的相关算法， 比如：中心性算法、相似度算法、社区检测算法等。

图数据库Nei4j有自带的算法库，但是无法满足本系统的需求；因此需要对Neo4j所支持的算法进行扩展。

本文在图数据库Neo4j的基础上集成了， 计算节点相似度的算法Simrank， 以及搜索属性紧密的社区的算法（Equitruss）

——————————————————————————————————————————————————————————————————

系统实现架构：

本系统功能实现主要分为

> 前端可视化模块：采用MVVM架构，和前端框架Vue实现

- 负责与用户行进交互

- 负责展示系统处理结果

* 后端模块，采用Restful风格API，以及MVC架构

- 负责处理业务逻辑，返回结果到前端

数据预处理可行性：

* XML流式解析，可以在内存受限制的情况下，解析较大的XML文件，不必讲整个XML文件读入内存中。内存中只保留当前在处理的内容
* 采用CSV格式保存数据
* Neo4j 支持CSV文件的直接导入
* ————————————————————————————————
* 算法集成
* 对于C/C++ 实现的算法，需要对代码进行二次开发，保证输入输出和形同调相匹配。采用Jni Java本地接口实现java代码与C/C++代码的相互调用、
* 也完全使用Java实现算法，但是性能相对较低
* ————————————————————————————————————————————————
* 系统架构实现：
* 前端使用Vue框架
* 后端采用RESTful风格开发API，以及MVC数据模式
* Pandas处理CSV文件
* Jni用于Java和C/C++ 的相互调用
* Cypher类似于Sql语言。Neo4j是扩展Neo4j算法扩展的方式
* Vue和Springboot 实现系统功能

数据预处理难点：

* 作者所有文章
* 作者之间的合作次数
* Nltk 文章标题分词
* ————————————————————————————————
* 算法集成：
* - 使用Neo4j的存储过程实现。可以用Cypher直接调用
* Jni实现 Java与C/C++代码相互调用
* 对算法进行适应性二次开发。是之满足集成需求

Simrank是一个利用图的拓扑关系也就是结构信息计算相似度的方法

a和b的相似度等于a的in-neighbors和b的in-neighbors相似度的平均值

或

a和b的相似度等于a的out-neighbors和b的out-neighbors相似度的平均值

https://www.cnblogs.com/zhangchaoyang/articles/4575809.html

属性图的社区搜索：ATC搜索算法

基本思想：

从属性的最大频繁子集开始进行搜索，当找到一个社区后就停止下一个量级的频繁子集搜索。（使用k-truss结构来保证结构的紧密性）。

对于查询顶点V：

* 枚举属性子集的所有非空子集
* 对每一个属性的非空子集，检查是否有包含V的k-truss社区
* 返回关键字集合最大的k-truss社区

优化：

* 在V和其邻居间进行属性的频繁项集挖掘，筛选V属性集合；自上而下，从大的集合开始搜索
* 使用索引：保存truss等价信息和超节点信息的混合索引（使用超图压缩图加快速度，超图中的节点是超节点：连通的、有相同truss值的边的集合聚集成一个超节点。超节点之间有三角形连通就有边。 在超图上进行社区搜索
* ）

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

K-core：节点度数 >= k

K-truss: 每条边至少包含k-2个三角形

K-truss=(k-1)-core=(k-1)-edge

—————————————————————————————————————————————————

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

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。

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。