GraphX报告

目录

contents

PART

01

研究背景

- 应用场景介绍
- 研究问题描述

研究背景

应用场景介绍

假如我们阅读了一篇论文之后，想要对这篇论文所属的领域做进一步研究，挖掘该领域的其他优秀论文，我们往往会将这篇论文所引用的部分论文找来进行阅读，尤其是在我们初涉某个研究领域的时候。

研究问题描述

基于此，我们研究了如何通过某篇论文，找到与这篇论文相关的其他优秀论文。

PART

02

问题解决

- 数据获取
- 数据预处理
- 解决方案
- 关键代码

数据获取

```
#*Approximating fluid schedules in crossbar pack
#@Michael Rosenblum,Constantine Caramanis,Michel
#t2006
#cIEEE/ACM Transactions on Networking (TON)
#index17
#%357875
#%214023
#%317448
#%319987
#%334185
#%95255
#%294124
#%96319
#%610127
```

01 数据从https://www.aminer.cn/citation下载，选择了其中的v1版本

02 数据集包含了60w+个顶点，60w+条边，每个顶点表示一篇论文，每条边表示一个论文引用关系，因此构成的是一个有向图

03 下载到的数据格式如图所示，index后的数字为这篇论文的ID，%后的数字为这篇论文所引用论文的ID

数据预处理

```
{ "_id" : ObjectId("5dc810ee897c9ec96c13fad9"), "src" : "17", "des" : "319987" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fada"), "src" : "17", "des" : "334185" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fadb"), "src" : "17", "des" : "95255" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fadc"), "src" : "17", "des" : "294124" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fadd"), "src" : "17", "des" : "96319" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fade"), "src" : "17", "des" : "610127" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fadf"), "src" : "24", "des" : "251778" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fae0"), "src" : "24", "des" : "436906" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fae1"), "src" : "24", "des" : "623227" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fae2"), "src" : "24", "des" : "287885" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fae3"), "src" : "35", "des" : "247215" }
{ "_id" : ObjectId("5dc810ee897c9ec96c13fae4"), "src" : "35", "des" : "618899" }
```

01

编写Python文件读取获取到的源数据，将出版年份、出版刊物等无用的信息过滤掉，只保留引用论文ID和被引用论文ID

02

以字典的形式保存每一个引用关系，包括引用论文ID和被引用论文ID，并将其全部存储到MongoDB中，上图是MongoDB中的数据存储格式

数据预处理

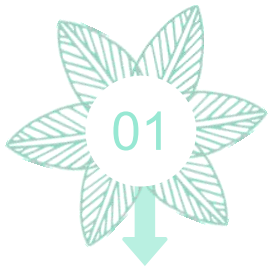
```
12=Webbots, Spiders, and Screen Scrapers
13=Fast k-NN Classification Rule Using Metrics on Space-Filling Curves
14=Making the Digital City: The Early Shaping of Urban Internet Space (Design & the Built Environment S.)
15=Linspire 5.0: The No Nonsense Guide! (No Nonsense Guide! series)
16=Podcasting for Profit: A Proven 10-Step Plan for Generating Income Through Audio and Video Podcasting
17=Approximating fluid schedules in crossbar packet-switches and Banyan networks
18=Federated Identity Management And Web Services Security With IBM Tivoli Security Solutions
19=Start with a Digital Camera (Special Edition) (2nd Edition) (Start with a)
20=Open Process Frameworks: Patterns for the Adaptive e-Enterprise (Practitioners)
21=Fast and Efficient Context-Aware Services (Wiley Series on Communications Networking & Distributed Systems)
22=Multimedia Directory 1997
23=ASIS&T Thesaurus of Information Science, Technology, And Librarianship (Asist Monograph Series)
24=On product covering in 3-tier supply chain models: natural complete problems for W[3] and W[4]
25=Inside SQL Server 2005 Tools (Microsoft Windows Server System Series)
26=Electronic Engineer's Handbook (Core Handbook CD-ROMs)
27=Call of Duty 2: Big Red One(tm) Official Strategy Guide (Official Strategy Guides)
28=Inside Microsoft Dynamics AX 4.0
29=Wiley Plus/Web CT Stand-alone to accompany Java Concepts (Wiley Plus Products)
30=Modeling methodology b: distributed simulation and the high level architecture
31=Beginning Ruby on Rails (Wrox Beginning Guides)
32=Introduction to Information Systems
33=SUSE Linux Enterprise Server Administration (Course 3037)
34=Hyperstat: Macintosh Hypermedia for Analyzing Data and Learning Statistics
35=An Integrative Modelling Approach for Simulation and Analysis of Adaptive Agents
36=Notes from industry
37=A New Quadtree Decomposition Reconstruction Method
38=Computer Accounting with QuickBooks 2006
```

01

02

将论文id与论文标题一一映射

解决方案

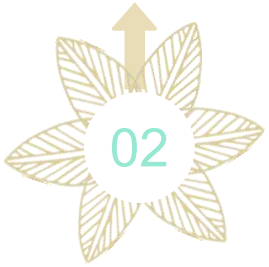


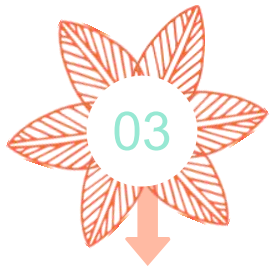
基本思路

我们通过分析一篇论文的一级引用论文和二级引用论文中的优秀论文来做这篇论文的相关论文推荐

步骤一

获取这篇论文的一级引用论文，也就是一级邻居，再通过一级邻居，获取它的二级引用论文，也就是二级邻居



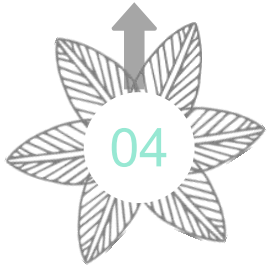


步骤二

使用PageRank算法对整个论文引用关系图中的论文进行等级评估

步骤三

从等级评估结果中选取出一级邻居和二级邻居的等级评估结果，将排名前十的论文推荐给用户



关键代码

```scala
val vertexNum = 629814
val arrInt:Array[(Long, Int)] = new Array[(Long, Int)](vertexNum)
for( a <- 0 until vertexNum){
    arrInt(a) = (a, 1)
}
val users: RDD[(VertexId, Int)] = sc.parallelize(arrInt)
val relationships: RDD[Edge[Int]] = MongoSpark.load(sparkSession).rdd
        .map(row => Edge(row.getString(2).toLong, row.getString(1).toLong, 1))
val graph = Graph(users, relationships)
graph.cache()
```

01

顶点RDD由一个从0到vertexNum-1的数组构造

02

边RDD的构造是先通过调用MongoSpark的load函数将边信息（论文引用信息）加载进来，然后据此创建Edge对象的RDD

关键代码

```scala
def getFirstNeighborIds(id:Long,graph:Graph[Int,Int]):HashSet[Long]={
    //aggregateMessages[Int]发送给每条边的每个顶点Int类型的消息
    val firstNeighbor:VertexRDD[Int]=graph.aggregateMessages[Int](triplet=>{
        if(triplet.srcId==id){
            triplet.sendToDst( msg = 1)
        }
    }, _ + _ )      //聚合相同顶点接收到的消息
    var firstIds=new HashSet[Long]()
    firstNeighbor.collect().foreach(a => firstIds+=a._1)
    firstIds
}
```

01

这个函数的功能是在graph中找到顶点id的一级邻居，它通过调用graph的aggregateMessages函数，在每条边的源顶点为id的时候，将消息1发送给目标顶点，然后在每个顶点上将收到的消息1聚合起来，最后将收到消息的目标顶点放到set集合中

02

当向该函数传入的id是所要查找论文的id时，函数返回的集合是一级引用论文的集合；当传入的是所要查找论文的引用论文id时，函数返回的就是二级引用论文的集合

关键代码

```
val firstNeighbor:VertexRDD[Double]=graph.pageRank( tol = 0.01).vertices

val neighborRank = firstNeighbor.filter(pred=>{
    var flag=false
    secondIds.foreach(id=>if(id == pred._1) flag = true)
    flag
}).sortBy(x=>x._2,ascending = false)          //按照rank从大到小排序
        .map(t=>t._1+" "+t._2)
neighborRank.take( num = 10).foreach(println)
```

01

首先，调用graph的pageRank函数对整个图中的论文进行等级评估，获得一个包含了评估结果的VertexRdd

02

从得到的VertexRdd中过滤出顶点id是一级邻居id或者二级邻居id的元素，对过滤后的VertexRdd按照等级降序排序后，选取前十个元素进行输出

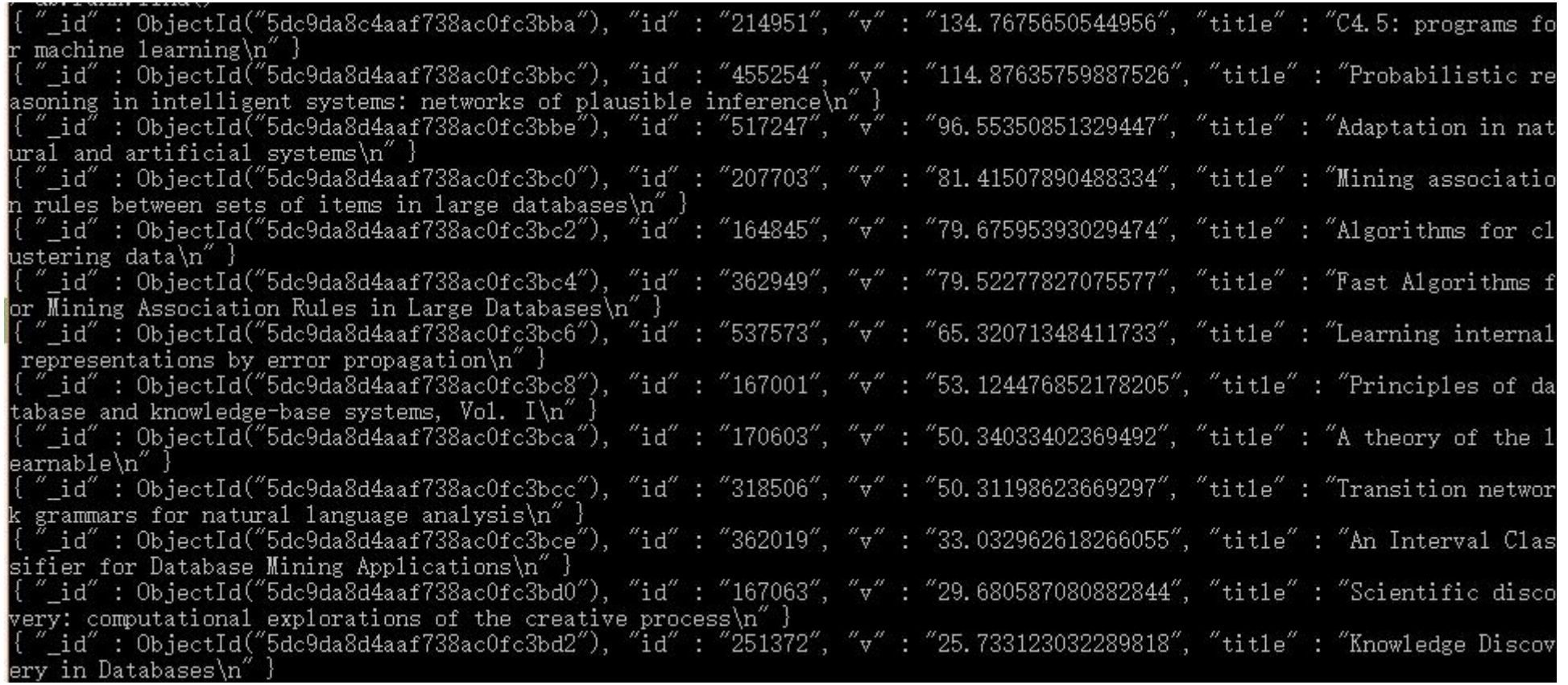
PART

03

总结展示

- 结果展示
- 总结与展望

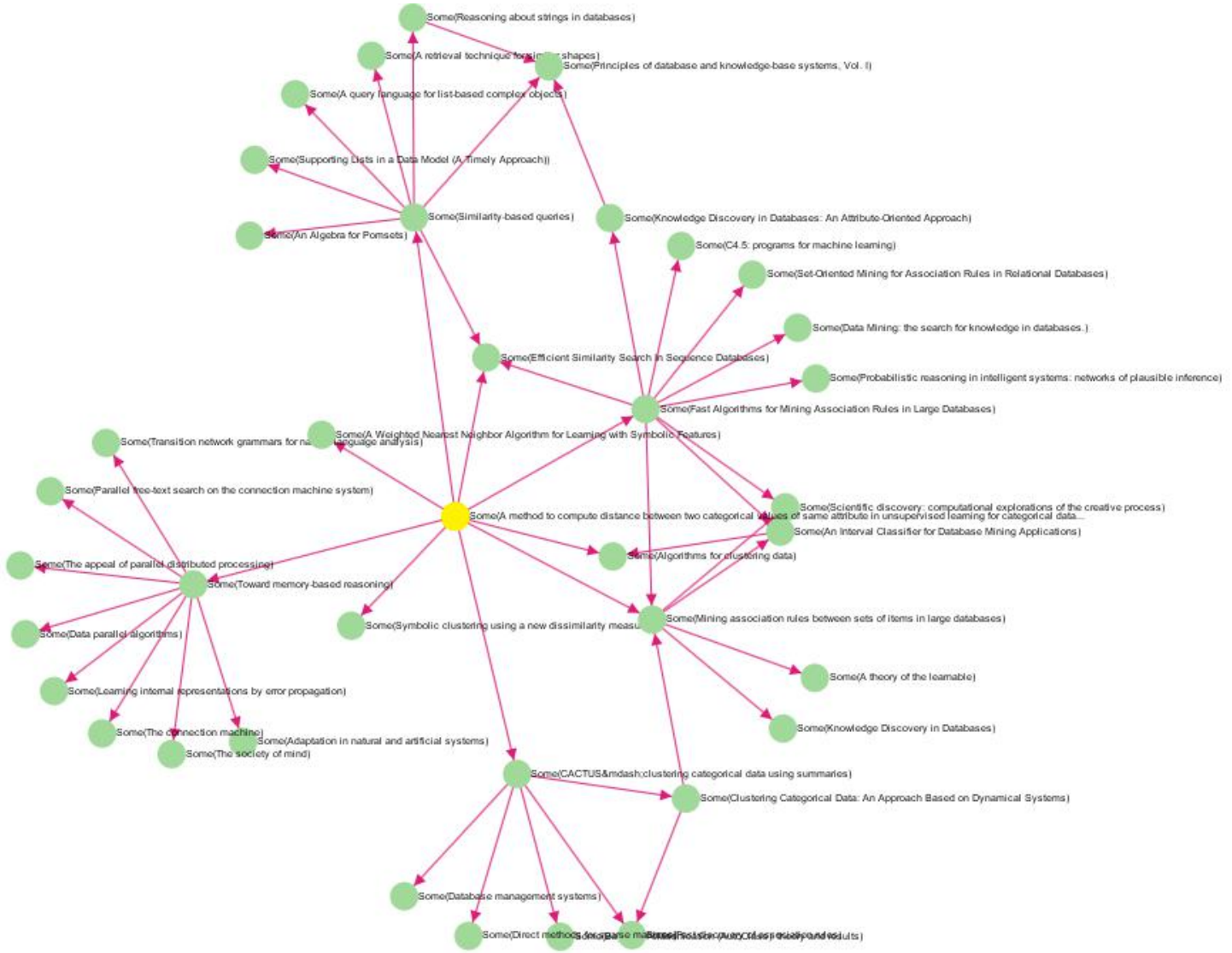
结果展示



将结果传到mongodb中

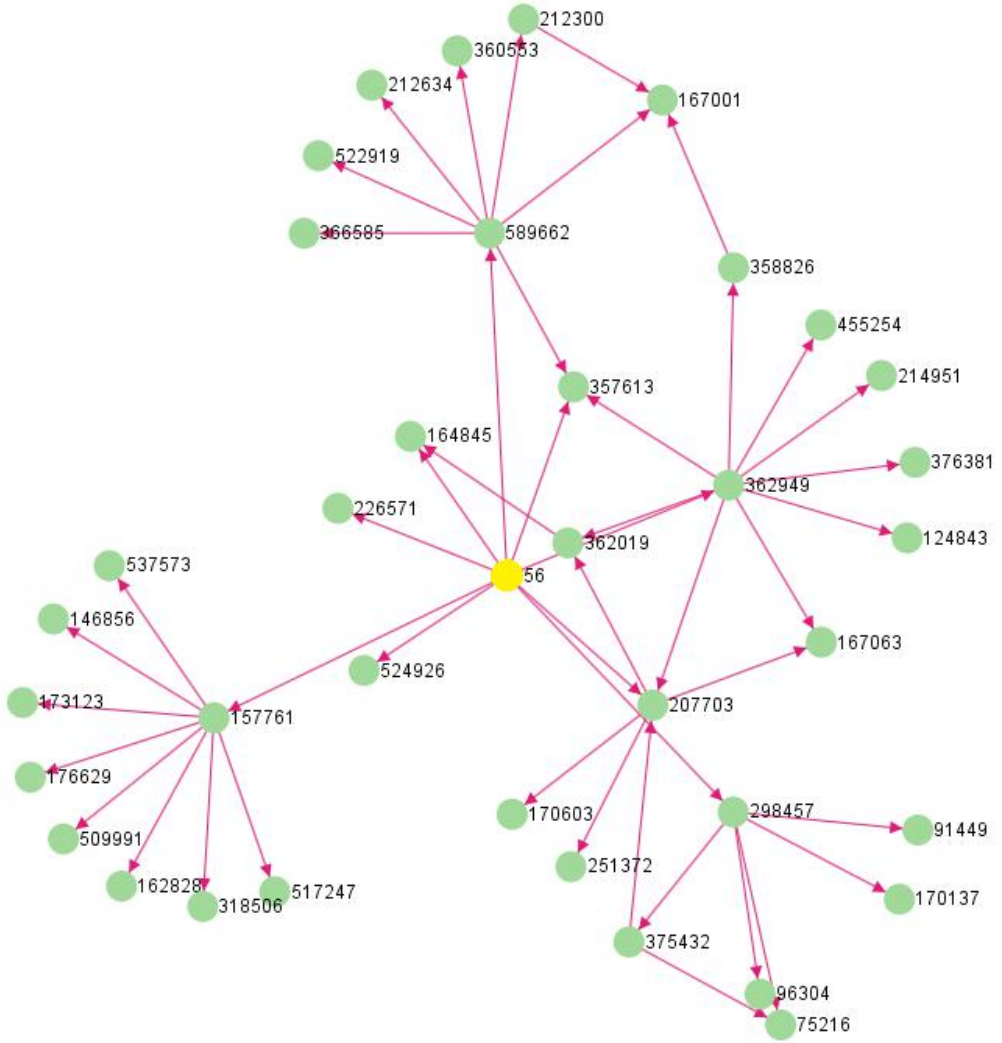
结果展示

调用graphstream jar包对id为56的一级和二级邻居以及他们的调用关系进行展示



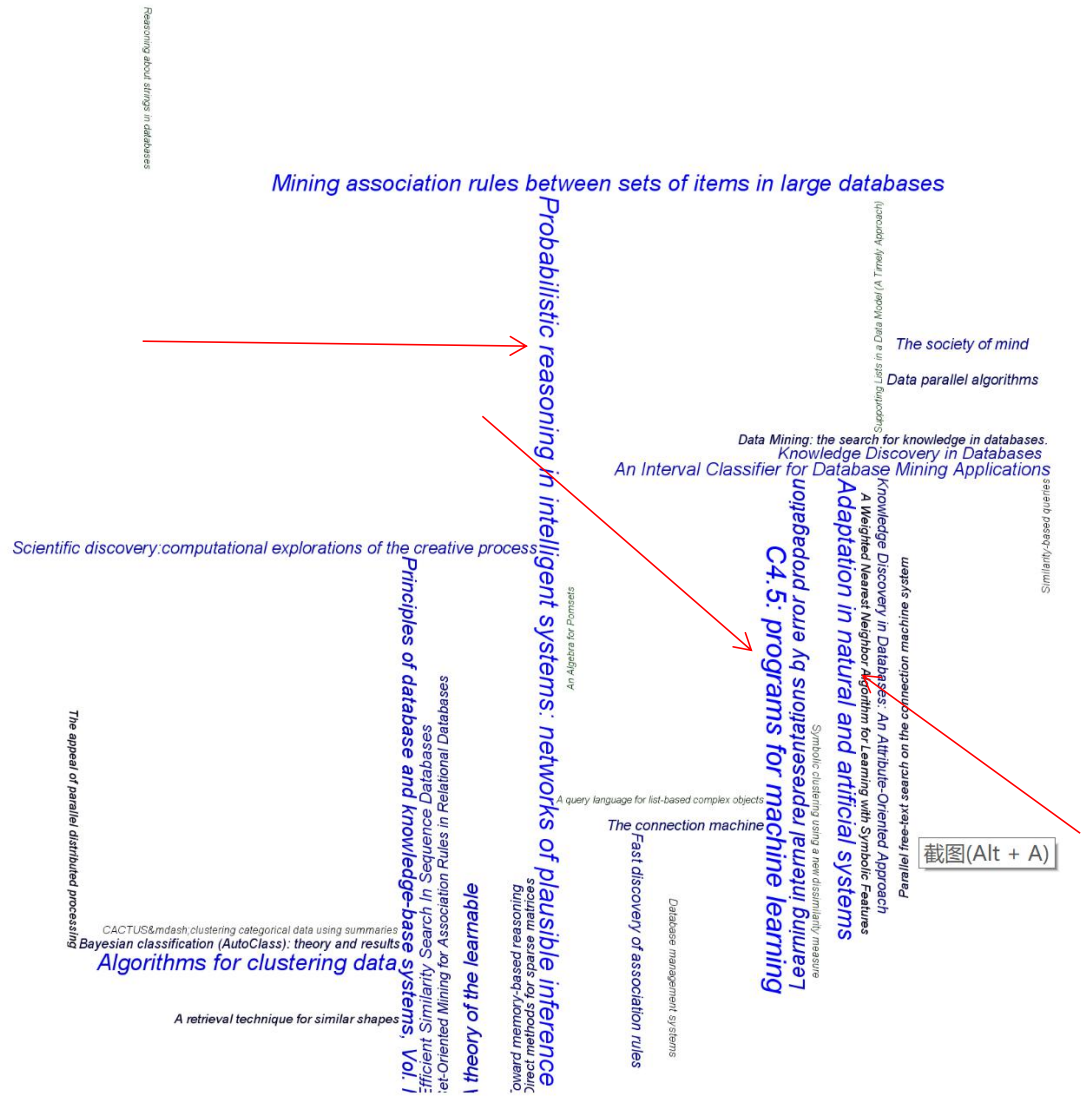
结果展示

由于标题太长了，它们之间的关系不明显，所以这里新增一幅图，用id号来展示互相的引用关系。



结果展示

推荐的论文根据PageRank的权重，展现优先推荐的文章。
其中C4.5: programs for machine learning
Probabilistic reasoning in intelligent systems: networks of plausible inference
Adaptation in natural and artificial systems
是排名前三的，在图中显而易见



总结与展望

总的来说，本次实验是比较成功的，我们实现了用边和点的RDD构造图，总共包含629814个点和632751条边，使用了一次聚合操作aggregateMessages，从MongoDB读图数据，并把结果存回MongoDB中，我们从这次实验中学会了很多。

谷歌学术、百度学术等许多学术网站都有相似文献的推荐，许多时候论文推荐可以帮助我们快速找到适合的论文，减少我们的搜索时间。

谢谢！