基于DBLP的频繁模式挖掘与自动语义注解

**摘要**

DBLP，即DataBase systems and Logic Programming，是计算机领域内对研究的成果以作者为核心的一个计算机类英文文献的集成数据库系统。它按年代列出了作者的科研成果。包括国际期刊和会议等公开发表的论文。截至2009年7月已经有超过1,200,000文献。DBLP所收录的期刊和会议论文质量较高，文献更新速度也很快，很好地反映了国际学术研究的前沿。本文主要利用DBLP数据库做以下分析：一、用基于频繁模式挖掘的方式分析作者与合作者之间的关系，并寻找其中的师生关系与大概时间范围；二、挖掘作者所述的领域的语义注释单元，并挖掘与其语义相近的作者、文章。

关键词：数据挖掘 DBLP 频繁模式 自动语义注解

**目录**

[第1章 绪论 3](#_Toc458861472)

[第2章 数据预处理 4](#_Toc458861473)

[2.1 数据的提取 4](#_Toc458861474)

[2.2 数据库写入 5](#_Toc458861475)

[2.3 数据库筛选 6](#_Toc458861476)

[第3章 合作者关系的频繁模式分析 8](#_Toc458861477)

[3.1 频繁合作关系挖掘 8](#_Toc458861478)

[3.2 频繁模式评估度量 12](#_Toc458861479)

[3.3 指导者-被指导者关系挖掘 13](#_Toc458861480)

[第4章 自动语义模式注解与应用 18](#_Toc458861481)

[4.1 特征词提取 18](#_Toc458861482)

[4.2 语义特征向量计算 18](#_Toc458861483)

[4.3 聚类分析 18](#_Toc458861484)

[4.4 自动语义注解的应用 18](#_Toc458861485)

1. 绪论

数据挖掘一般是指从大量的数据中通过算法搜索隐藏于其中信息的过程。DBLP，即DataBase systems and Logic Programming，是计算机领域内对研究的成果以作者为核心的一个计算机类英文文献的集成数据库系统。它按年代列出了作者的科研成果。包括国际期刊和会议等公开发表的论文。截至2009年7月已经有超过1,200,000文献。DBLP所收录的期刊和会议论文质量较高，文献更新速度也很快，很好地反映了国际学术研究的前沿。本文主要利用DBLP数据库做以下分析：一、用基于频繁模式挖掘的方式分析作者与合作者之间的关系，并寻找其中的师生关系与大概时间范围；二、挖掘作者所述的领域的语义注释单元，并挖掘与其语义相近的作者、文章。

主要的步骤为：

* 数据预处理。DBLP中的数据是以XML（Extensible Markup Language）文件的形式储存的。XML文件可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。它非常适合万维网传输，提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。是Internet环境中跨平台的、依赖于内容的技术，也是当今处理分布式结构信息的有效工具。但是它也存在很大问题，如：与数据库方式相比，它的查询、插入、修改、删除等操作耗时较多；它也很难像关系数据库一样方便地进行连接、筛选、运算等操作；且XML的文本表现手法、标记的符号化会导致XML数据比二进制表现数据量增加，尤其当数据量很大的时候，效率就成为很大的问题；最大的问题还在于，DBLP数据集中的每一条记录所包含的数据成员是不统一的，有些属性并非每个记录都有，这给分析数据造成了很大的困扰。因此需要进行数据预处理。由于下载的XML文件巨大（1.67GB），大多数现有的XML解析库没法直接读取完整的文件进行解析，所以考虑采用JAVA的SAX方法，分段读取数据集，然后经过关键词筛选、整合与处理，最后逐一写入ACCESS数据库。
* 合作者关系分析。首先用Apriori算法，对文章作者进行频繁挖掘，找出频繁发文章的作者以及作者的组合，并且得到他们写作文章与合作文章的数量，从而可以计算支持度和置信度。然后可以计算出提升度、、全置信度、最大置信度、Kulc和Cosine等模式评估度量，以及用于衡量不平衡度的不平衡比IR，用这些量可以得出各对合作者之间的密切程度。通过对以上模式评估度量、不平衡比IR，以及之前统计的各个作者发表文章的先后顺序，可以粗略地分析出可能的指导者-被指导者关系，并且根据他们合作发表文章的时间，可以大概推断出该关系出现的年份时间段。
* 语义自动注解。主要分析文章、关键词、作者等之间的语义联系，并对其进行聚类。首先是将所有论文的标题提取出来，并对每个单词出现的频率进行统计，得到词频计数，并删去一些无意义副词。然后提取固定数目（如100个）的词频最靠前的词作为语义坐标基，统计其余单词分别与这些坐标基词同时出现在标题中的次数，作为该词在该坐标基下的坐标值。如此可以得到每个词语在语义坐标系下的坐标向量，可以根据Kmeans等聚类方法进行聚类，得到语义相近的词的组合。

1. 数据预处理

由于DBLP中的数据是以XML文件的形式储存的，与数据库方式相比，它的查询、插入、修改、删除等操作耗时较多；它也很难像关系数据库一样方便地进行连接、筛选、运算等操作；且XML的文本表现手法、标记的符号化会导致XML数据比二进制表现数据量增加，尤其当数据量很大的时候，效率就成为很大的问题；最大的问题还在于，DBLP数据集中的每一条记录所包含的数据成员是不统一的，有些属性并非每个记录都有，这给分析数据造成了很大的困扰。因此需要进行数据预处理。在XML文件解析过程中，有三种工具可供选择：DOM、SAX和DOM4J。

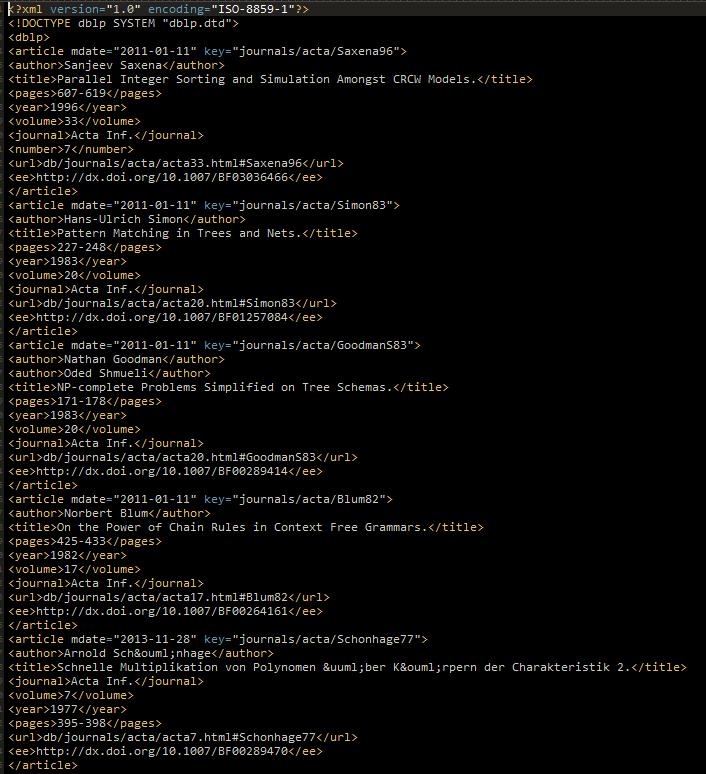
DOM是用与平台和语言无关的方式表示XML文档的官方W3C标准。DOM是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而DOM被认为是基于树或基于对象的。DOM以及广义的基于树的处理具有几个优点。首先，由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构作出更改。它还可以在任何时候在树中上下导航，而不是像SAX那样是一次性的处理。DOM使用起来也要简单得多。然而在当前应用中却存在致命问题，就是DOM工具会把整个XML文件读入内存后再进行解析，会极大地占用内存资源。而DBLP数据库XML文件有1.67GB之大，DOM根本无法胜任。

SAX处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且，由于应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档；它可以在某个条件得到满足时停止解析。SAX解析器采用了基于事件的模型，它在解析XML文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。SAX对内存的要求通常会比较低，因为它让开发人员自己来决定所要处理的tag。特别是当开发人员只需要处理文档中所包含的部分数据时，SAX这种扩展能力得到了更好的体现。一般来说，SAX还比它的替代者DOM快许多。但是SAX也存在问题，就是因为它采用顺序读取的方式，在遇到某类指定结点的时候触发事件，进行相应地处理。因此只能处理当前XML结点的数据，无法像DOM一样一次性获得整个文件的树状层次结构，也很难进行回溯。

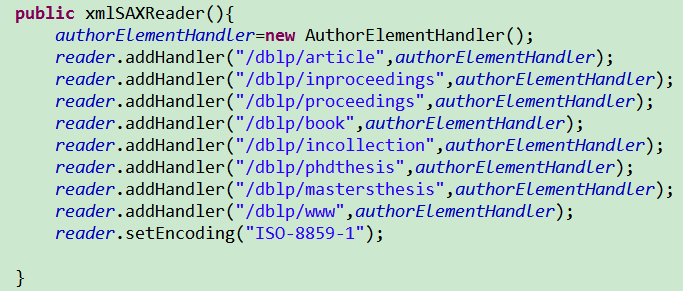
DOM4J则是集合了DOM与SAX的优点。DOM4J和SAX一样采取顺序读取文件的方法，减少了内存的占用；然而它又提供节点的回溯功能，可以得到当前节点的父节点信息，可以得到节点之间的层次关系。它还支持支持Xpath方式访问节点，操作性比SAX更方便。因此，本文采用了DOM4J方式来处理DBLP的XML数据文件。

* 1. 数据的提取

dblp.xml中的数据标签包括article、inproceedings、proceedings、book、incollection、 phdthesis、mastersthesis、www等文章记录，因此在定义xmlSAXReader读取节点时要添加以上所有节点。



1. dblp.xml数据片段



1. xmlSAXReader读取节点定义

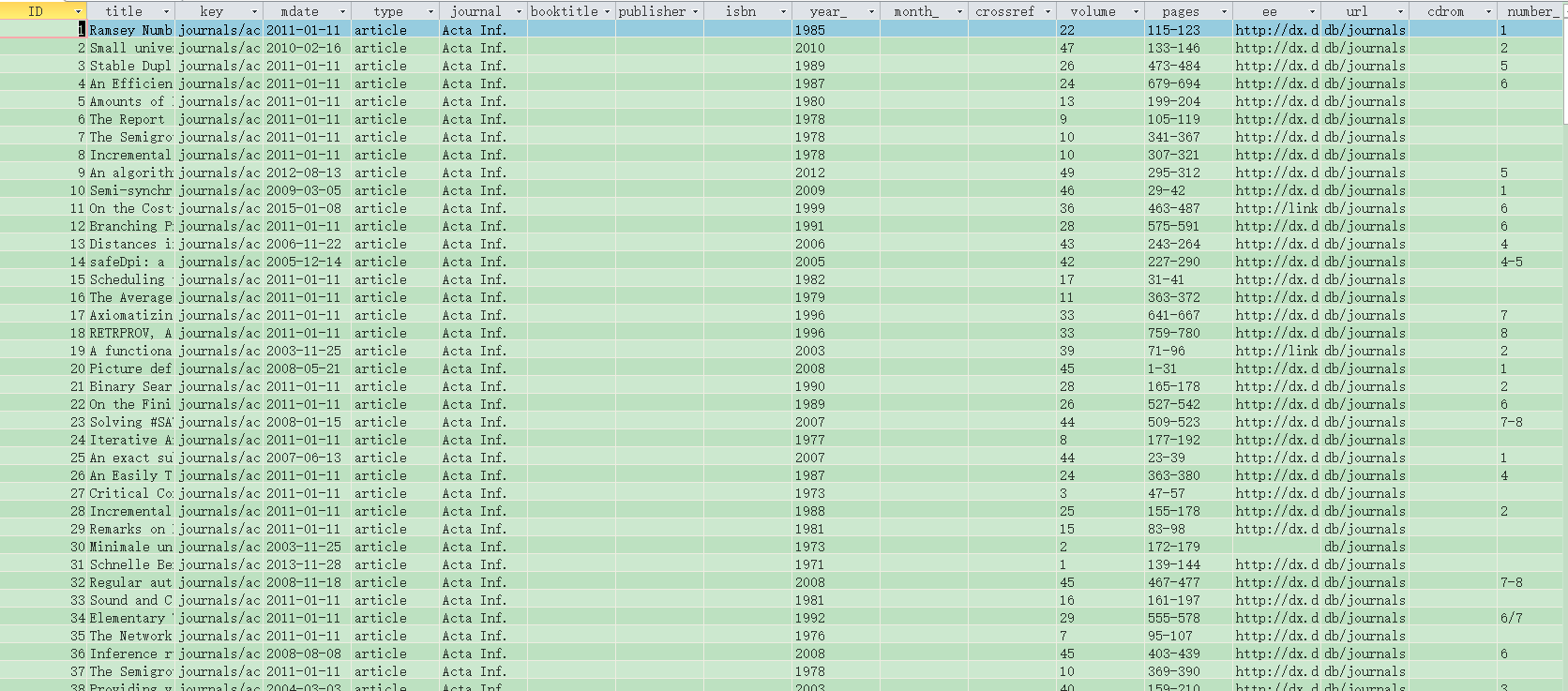
每条文章记录都可能包含title, Author, key, mdate, type, journal, booktitle, publisher, isbn, year, month, crossref, volume, pages, ee, url, cdrom, number, address, cite, note, series, school, chapter等字段，因此在读取节点的时候需要判断每个节点是否包含这些字段，以便写入数据库中。具体的方法是，每当读完一条文章记录时，把它的所有子节点（也就是他的字段）保存在一个节点队列中。然后对每个字段进行判断，在数据库中相应字段下插入相应的数据。

* 1. 数据库写入

本文所采用的是Microsoft Access 2016数据库，支持SQL语言的操作。与处理过程中主要用到的是数据库的连接与插入操作。

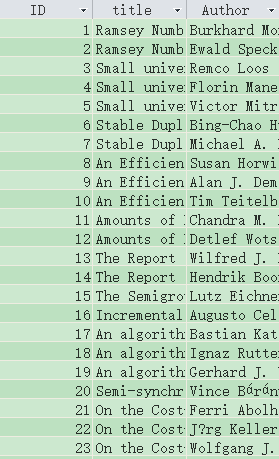
首先要对数据库进行设计。考虑到DBLP数据集中的数据繁杂，一篇文章往往是好几个作者合作的，如果按照对每个字段铺展，则每条DBLP记录会铺展成好几条数据库记录，比如一篇文章有3个作者，则该记录会在数据库中铺展成3条，每条记录中其他字段都相同，仅有作者这个字段不同。因此会造成极大的浪费，而且也不利于后期对数据进行筛选、连接、查询等操作。因此考虑将数据库设计成两张表：Articles表和Article\_Author表。

Articles表存储文章记录，每篇文章的记录包含除了Author字段以外的所有字段，包括title, key, mdate, type, journal, booktitle, publisher, isbn, year, month, crossref, volume, pages, ee, url, cdrom, number, address, cite, note, series, school, chapter。其中把title作为表的键值。可以看到有些记录中的某些字段存在空缺，因此数据集较为稀疏。下文中重点研究的是title、year等字段。



1. Articles表

Article\_Author表存储文章和作者的关系记录，每条记录包含title和Author字段，表示某篇文章及其作者。这是下文重点研究的关系表。



1. Article\_Author表
   1. 数据库筛选

经过以上步骤，已经可以得到完整的包含文章、作者信息的数据库了，但是由于记录数据过于庞大，而并非所有文章都是本文感兴趣的，因此要使用关键词对文章的标题进行筛选。本文研究图像处理相关的文献及作者，因此选用的筛选关键词是：vision、image、graphics imaging、opencv、video。

在Access数据库中新建查询，输入SQL查询语句：

SELECT Article\_Author.[title], Article\_Author.[Author]

INTO Article\_Author

FROM Article\_Author

WHERE (((Article\_Author.title) Like '\* vision \*' Or (Article\_Author.title) Like '\* image \*' Or (Article\_Author.title) Like '\* graphics \*' Or (Article\_Author.title) Like '\* imaging \*' Or (Article\_Author.title) Like '\* opencv \*')) Or (Article\_Author.title Like '\* video \*');

以及对Articles表的筛选：

SELECT Articles.[title], Articles.[key], Articles.[mdate], Articles.[type], Articles.[journal], Articles.[booktitle], Articles.[publisher], Articles.[isbn], Articles.[year\_], Articles.[month\_], Articles.[crossref], Articles.[volume], Articles.[pages], Articles.[ee], Articles.[url], Articles.[cdrom], Articles.[number\_], Articles.[address], Articles.[cite], Articles.[note\_], Articles.[series], Articles.[school], Articles.[chapter]

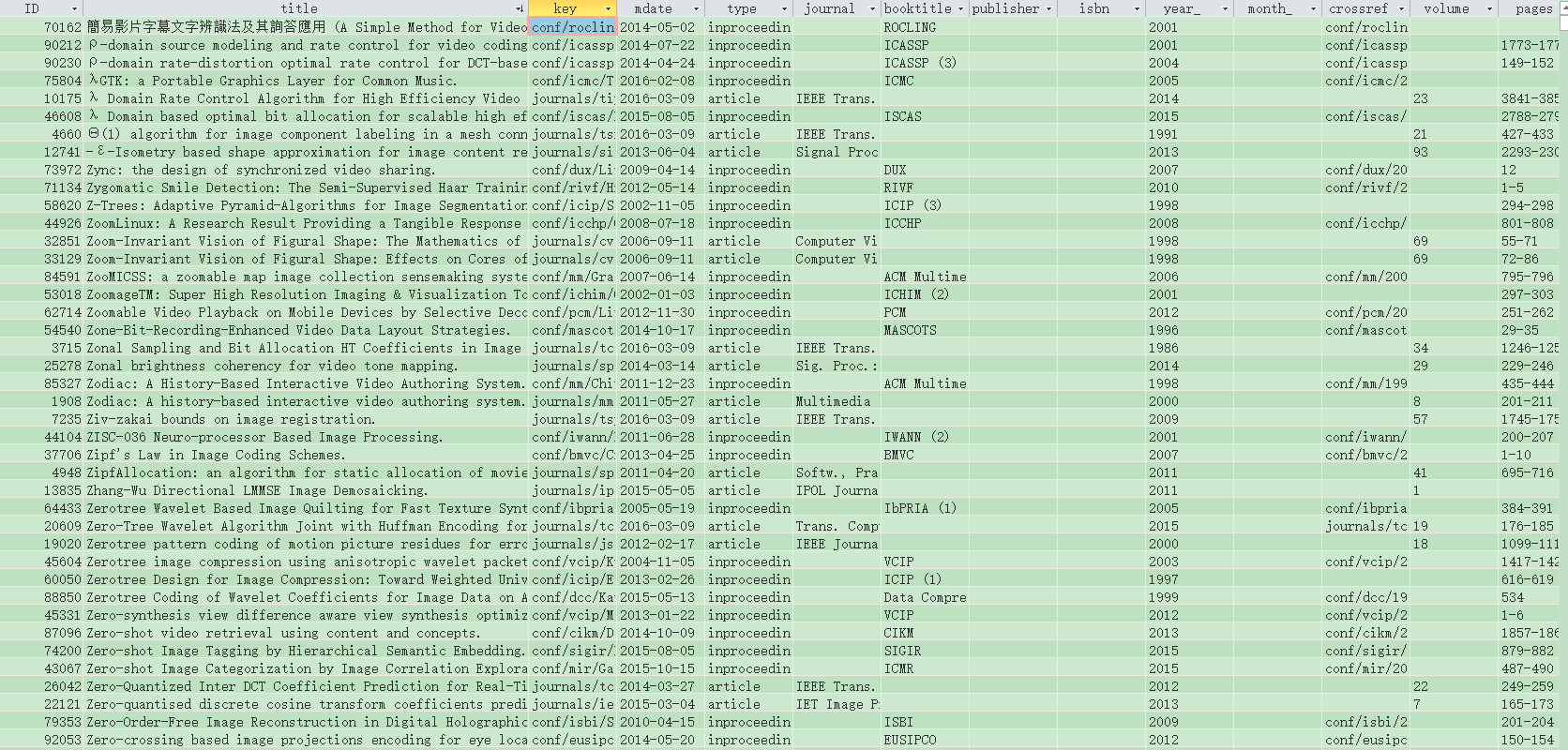
FROM Articles

WHERE (Articles.title Like '\* vision \*') Or (Articles.title Like '\* image \*') Or (Articles.title Like '\* graphics \*') Or (Articles.title Like '\* imaging \*') Or (Articles.title Like '\* opencv \*') Or (Articles.title Like '\* video \*');

得到筛选结果：Article\_Author表共307384条记录；Articles表共96388条记录。



1. 筛选后的Article\_Author表

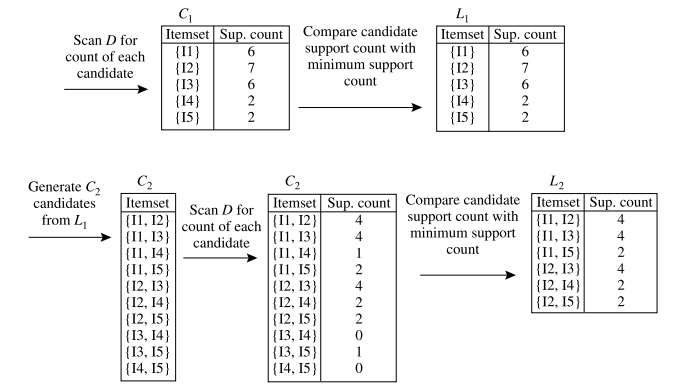


1. 筛选后的Articles表
2. 合作者关系的频繁模式分析

本章对与处理完的数据进行合作者关系的挖掘，主要研究的内容是：1. 寻找频繁合作的作者，也就是多次共同发表论文的作者对，即教材中提到的频繁项集的概念；2. 用教材中提到的一系列模式评估度量对共同发表论文的作者对进行频繁模式的分析；3. 根据分析的结果，外加一些文章的信息，找出合作作者之间存在的指导者-被指导者关系。

* 1. 频繁合作关系挖掘

首先对频繁合作关系进行挖掘。考虑采用Apriori算法，其核心思想是：满足高阶频繁项集条件的作者在低阶必须也是频繁的。如一对作者他们共同发表了多篇文章，因此他们是一个二项频繁的事务；对应到个人，每个作者自身也必然是发表了多篇文章的一项频繁事务。因此，要挖掘高项频繁集，可以先挖掘低项频繁集，这样筛选可选的作者集合，可以大大减少需要挖掘的数据量，节省时间和空间。



1. Apriori算法伪代码与流程图

因此首先挖掘一项频繁集，也就是统计作者及其文章数，把文章数作为其支持度数，并按文章数多少排序，得到Authors表。

代码：

SELECT Article\_Author.[Author] AS Author, COUNT(\*) AS NumOfArticles

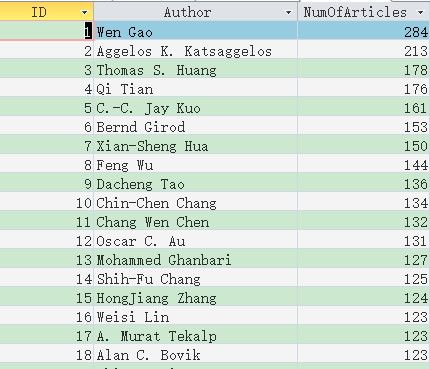
INTO Authors

FROM Article\_Author

GROUP BY Author

ORDER BY COUNT(\*) DESC;

结果：共130516条记录



1. Authors表

然后设置最小支持度计数阈值为3，排除发表文章少于3篇的作者，得到Authors\_Frequent\_1（sup>2）表（一项作者频繁集，最小支持度3）：

代码：

SELECT Authors.Author AS Author, Authors. NumOfArticles AS NumOfArticles

INTO [Authors\_Frequent\_1（sup>2）]

FROM Authors

WHERE NumOfArticles>2

ORDER BY NumOfArticles DESC;

结果：共25662条记录



1. Authors\_Frequent\_1（sup>2）表

然后要将Authors\_Frequent\_1（sup>2）表与Article\_Author表连接，得到Article\_Author\_Frequent\_1表，来作为二项频繁集挖掘的候选作者：

代码：

SELECT Article\_Author.title AS title, Article\_Author.Author AS Author

INTO Article\_Author\_Frequent\_1

FROM Article\_Author, [Authors\_Frequent\_1（sup>2）]

WHERE Article\_Author.Author = [Authors\_Frequent\_1（sup>2）].Author

ORDER BY title DESC;

结果：共182351条记录



1. Article\_Author\_Frequent\_1表

从Article\_Author\_Frequent\_1表中即可挖掘二项频繁集Authors\_Frequent\_2（sup>2）表，其方法是连接两张Article\_Author\_Frequent\_1表，其中连接规则是文章标题title字段相同，即可得到共同合作文章的作者对Author1和Author2。与此同时，用COUNT(\*)可以统计作者对合作文章的数量并降序排列，作为新表的NumOfCoArticles字段，并用HAVING子句筛去合作文章小于两篇的作者对。

代码：

SELECT Article\_Author\_Frequent\_1.[Author] AS Author1, Article\_Author\_Frequent\_1\_copy.[Author] AS Author2, COUNT(\*) AS NumOfCoArticles

FROM Article\_Author\_Frequent\_1,Article\_Author\_Frequent\_1 AS Article\_Author\_Frequent\_1\_copy

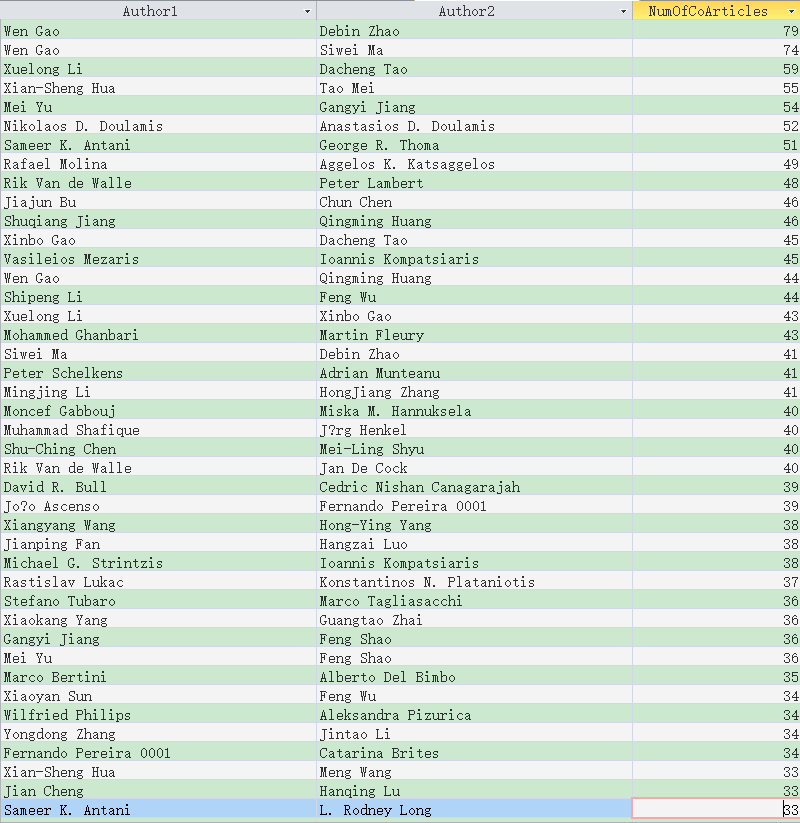
WHERE Article\_Author\_Frequent\_1.title=Article\_Author\_Frequent\_1\_copy.title AND Article\_Author\_Frequent\_1.[Author]>Article\_Author\_Frequent\_1\_copy.[Author]

GROUP BY Article\_Author\_Frequent\_1.[Author], Article\_Author\_Frequent\_1\_copy.[Author]

HAVING (((Count(\*))>2))

ORDER BY COUNT(\*) DESC;

结果：共24633条记录。



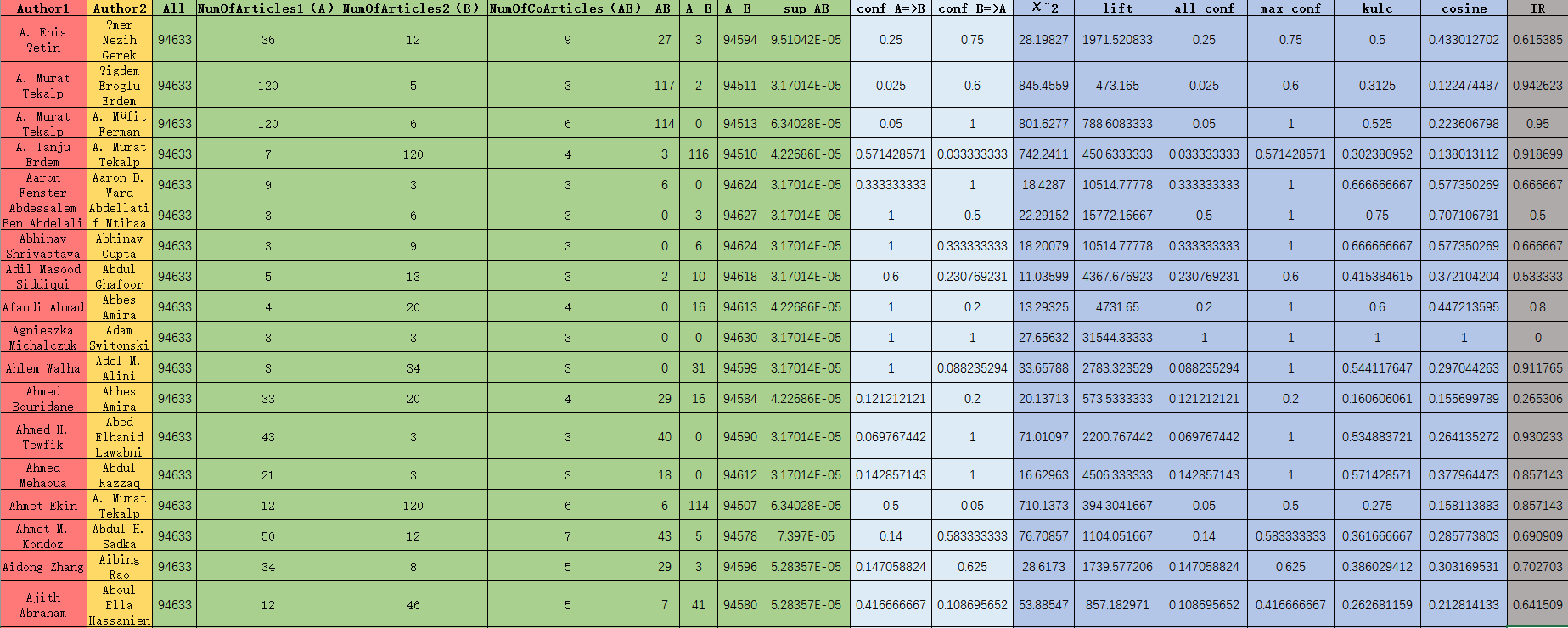
1. Authors\_Frequent\_2（sup>2）表

至此完成了频繁发表文章的作者、合作者及其文章数的挖掘。文章数可以作为支持度数，凭此可以进行频繁模式评估度量的计算。

* 1. 频繁模式评估度量

根据Authors\_Frequent\_2（sup>2）表、Authors\_Frequent\_1sup>2）表中的数据，可以得到频繁作者对的合作文章数NumOfCoArticles（AB）、个人文章数NumOfArticles1（A）和NumOfArticles2（B）、总文章数All等信息，将其整合到EXCEL电子表格中，用公式计算表中所示的其它量。其中：

* AB ̅ = A- AB；
* A ̅ B= B- AB；
* A ̅ B ̅ =All-A-B+AB；
* 支持度（AB）：sup\_AB=AB/All；
* 置信度（A=>B）：conf\_A=>B=AB/A；
* 置信度（B => A）：conf\_B=>A=AB/B；
* 卡方度量：χ^2= (AB-AVG(AB))^2/ AVG(AB) +( AB ̅ -AVG(AB ̅ ))^2/ AVG(AB ̅ )+ (A ̅ B -AVG(A ̅ B))^2/ AVG(A ̅ B)+ (A ̅ B ̅ -AVG(A ̅ B ̅ ))^2/ AVG(A ̅ B ̅ )；
* 提升度：lift=AB\*All/(A\*B)；
* 全置信度：all\_conf= AB /MAX(A,B)；
* 最大置信度：max\_conf= AB /MIN(A,B)；
* Kulc度量：kulc= (AB /A+ AB /B)/2；
* 余弦度量：cosine= SQRT(AB \* AB /(A\*B))；
* 不平衡度：IR= ABS(A-B)/(A+B-AB)。



1. 频繁模式度量计算表

从结果中可以看出，卡方度量和提升度作为度量时，因为受总记录条数All的影响（总记录中大部分并不包含作者A和作者B），其值时大时小，并不能做出很好的度量。而全置信度、最大置信度、Kulc度量和余弦度量都不受总记录数影响，是零不变的度量，且范围都在0~1之间，能相对较直观地反映作者之间的关系密切程度。

如果单纯研究合作者之间互相的密切程度，采用Kulc度量和余弦度量较好，因为它们分别是两种置信度的算术平均与集合平均，可以反映了两作者之间互相关系的综合密切程度。

而如果要研究合作者之间的指导与被指导关系，则光考虑合作者之间的密切程度是不够的，还要考虑两者的身份差别。因此采用综合考虑全置信度、最大置信度和不平衡比较为有效。因为一般而言，指导者发的文章相对比被指导者多；且指导者发的文章中与被指导者合作的文章所占比例一般较小。而被指导者发的文章数相对较少；且一般发的文章中与指导者合作的文章站的比例较大。因此会导致全置信度较小，最大置信度较大，且不平衡比较大的情况。这在下一节指导者-被指导者关系挖掘中将起到关键作用。

* 1. 指导者-被指导者关系挖掘

由3.2节的分析，得到了频繁合作的作者对以及他们的频繁模式度量。并且在考虑指导者-被指导者关系挖掘时，需要采用考虑全置信度、最大置信度和不平衡比来综合评判。但是仅仅以此为判据，只能说明合作者之间合作较为紧密，且两者关系不是平等的同事关系而存在这明显的不平衡性。但这还不足以很好地判定合作者之间是否真的是指导者与被指导者关系。

为了解决这一问题，我们需要考虑发表文章的时间，以及每个作者的合作者数目。根据常识，一般导师的年龄比学生大至少5岁以上，因此一般他们首次在学术圈发表论文的时间应该比学生早5年以上。此外一般导师的合作者数目会比学生多。因此在挖掘过程中需要把这两个条件也加上。因此首先获取每位作者第一篇发表文章的年份，以及每位作者合作者数目。

首先是NumOfCoAuthors表，连接两张Article\_Author表，规则是文章名相同，作者名不同；然后统计每个作者的合作者数目：

代码：

SELECT DISTINCT Article\_Author.Author AS Author, Article\_Author\_copy.Author AS CoAuthor

FROM Article\_Author, Article\_Author AS Article\_Author\_copy

WHERE Article\_Author.title = Article\_Author\_copy.title AND Article\_Author.Author<>Article\_Author\_copy.Author;结果：

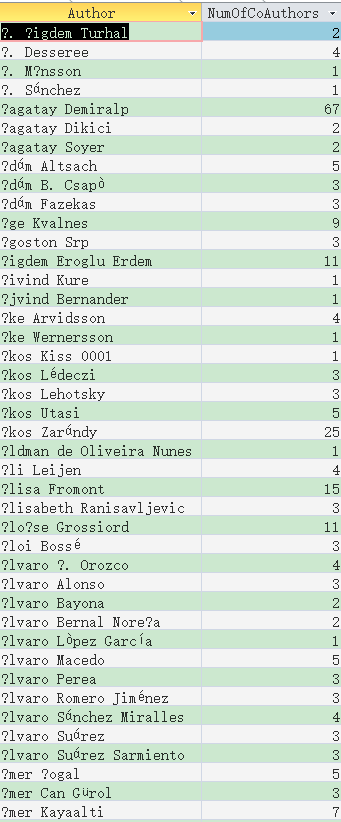
SELECT [查询NumOfCoAuthors].Author AS Author, COUNT(\*) AS NumOfCoAuthors

INTO NumOfCoAuthors

FROM [查询NumOfCoAuthors]

GROUP BY [查询NumOfCoAuthors].Author

结果：共127758条记录。



1. NumOfCoAuthors表

然后是FirstPubYear表，连接Article\_Author和Articles表，选取每位作者发表的文章中year记录最小的，作为FirstPubYear：

代码：

SELECT Article\_Author.[Author] AS Author, COUNT(\*) AS NumOfArticles, MIN(Articles.year\_) AS FirstPubYear

INTO Authors

FROM Article\_Author, Articles

WHERE Articles.title=Article\_Author.title

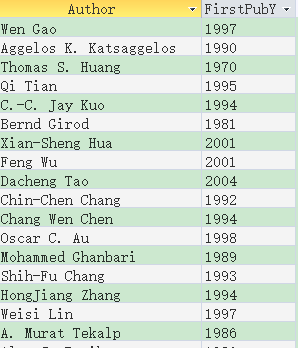
GROUP BY Author

ORDER BY COUNT(\*) DESC;

SELECT Authors.Author AS Author, Authors.FirstPubYear AS FirstPubYear INTO FirstPubYear

FROM Authors;

结果：共130516条记录。



1. FirstPubYear表

最后连接Article\_Author 、Articles等表得到Advisor\_Advisee表，其中筛选的条件是：最大置信度>0.5；最大置信度比全置信度大50%；两者FirstPubYear相差5年以上；NumOfCoAuthors相差2人以上：

代码：

SELECT \* INTO Complex\_Filtered

FROM Complex

WHERE max\_conf>0.5 AND (max\_conf-all\_conf)/all\_conf>0.5 AND IR>0.5 AND ABS(FirstPubYear1-FirstPubYear2)>5 AND ABS(NumOfCoAuthors1-NumOfCoAuthors2)>2;

SELECT IIf(Complex\_Filtered.FirstPubYear1<Complex\_Filtered.FirstPubYear2, Complex\_Filtered.Author1, Complex\_Filtered.Author2) AS Advisor, IIf(Complex\_Filtered.FirstPubYear1<Complex\_Filtered.FirstPubYear2, Complex\_Filtered.Author2, Complex\_Filtered.Author1) AS Advisee

FROM Complex\_Filtered

SELECT [查询Advisor\_Advisee].Advisor AS Advisor, [查询Advisor\_Advisee].Advisee AS Advisee,MIN(Articles.year\_) AS FirstCoPubYear, MAX(Articles.year\_) AS LastCoPubYear

INTO Advisor\_Advisee

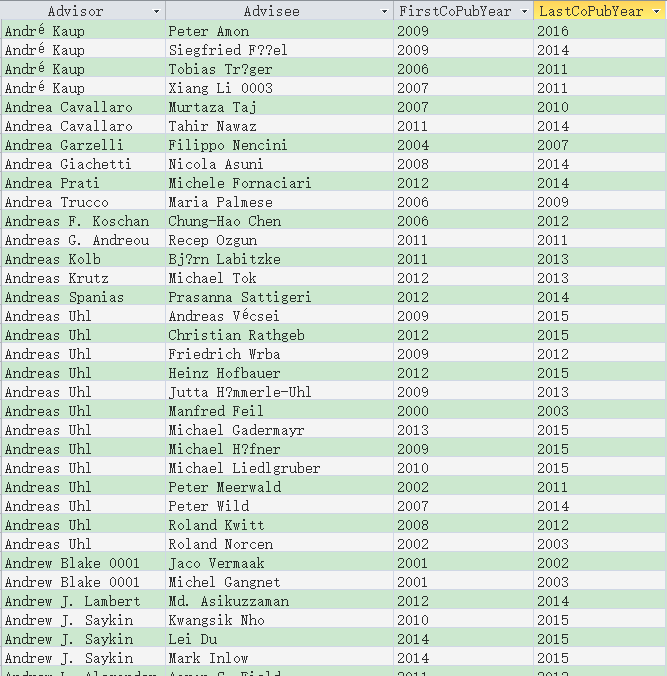
FROM [查询Advisor\_Advisee], Article\_Author AS Article\_Author1, Article\_Author AS Article\_Author2, Articles

WHERE [查询Advisor\_Advisee].Advisor=Article\_Author1.Author AND [查询Advisor\_Advisee].Advisee=Article\_Author2.Author AND Article\_Author1.title=Article\_Author2.title AND Articles.title=Article\_Author1.title

GROUP BY Advisor,Advisee

ORDER BY Advisor

结果：共5951条记录。



1. Advisor\_Advisee表

表中统计了指导者与被指导者最早和最晚的合作文章时间，根据经验，被指导者通常在跟随指导者1年后开始发表论文，因此师生关系维持的时间一般是（FirstCoPubYear-1）~LastCoPubYear年份之间。从表中数据可以看到每位指导者都有若干被指导者，且分散在不同时期。由上表得到的Advisor\_Advisee关系直方图能更好地说明问题。



1. Advisor\_Advisee关系直方图

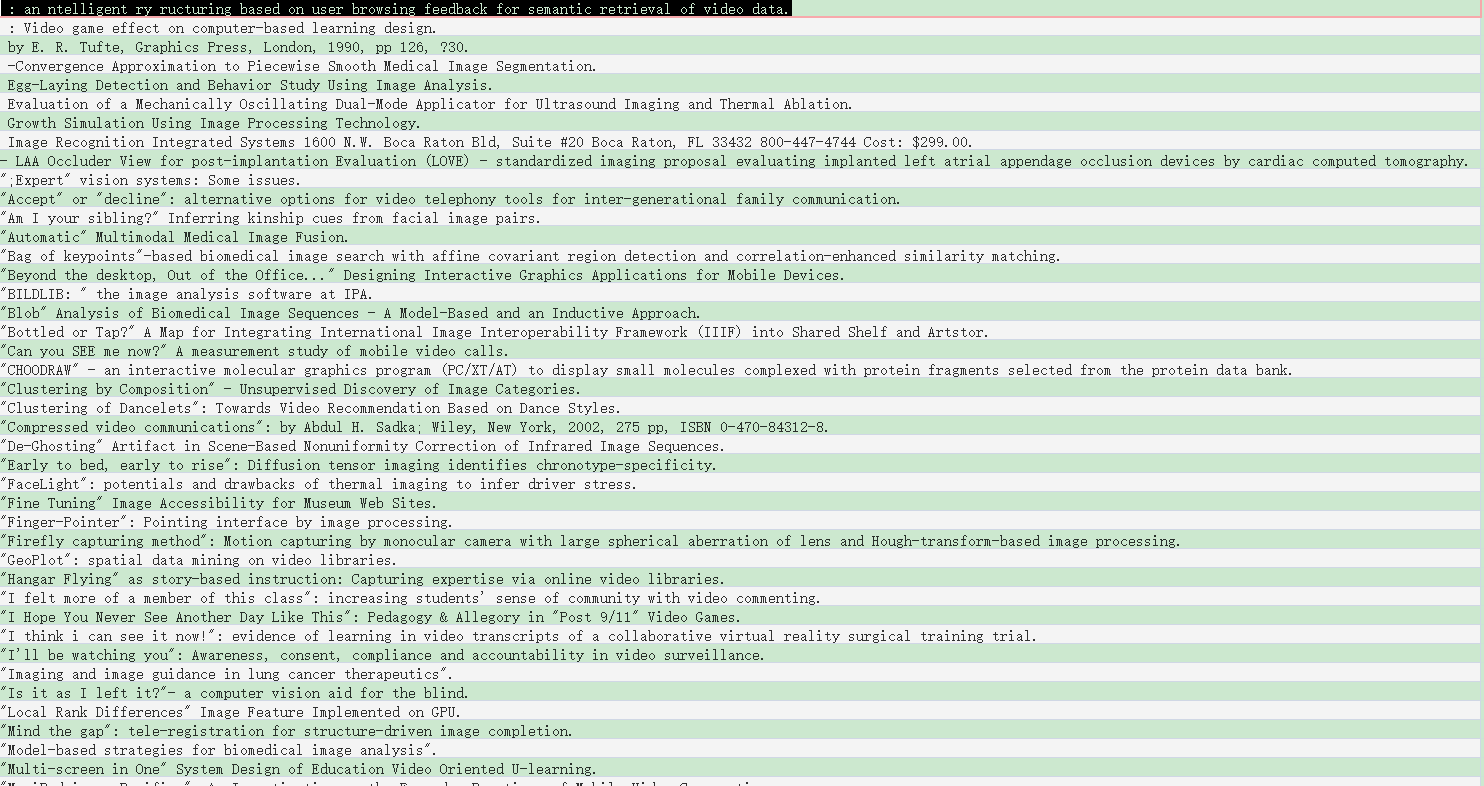
如图所示，横轴表示Advisor\_Advisee关系维持的时间段，从1年到最多17年不等。纵轴表示维持该年数Advisor\_Advisee关系的作者对计数。可以看到绝大多数作者对维持关系在2~4年，其中最多的是3年有1377对，其余的也集中在1年、5~8年；总体的平均年份是4.3年。大多数硕士研究生在校的时间是2~3年。而在校时间稍长的，则是博士研究生、博士后等则可能在5~8年，因此挖掘的结果是合理的。

1. 自动语义模式注解与应用

本节主要分析文章、关键词、作者等之间的语义联系，并对其进行聚类。首先是将所有论文的标题提取出来，并对每个单词出现的频率进行统计，得到词频计数，并删去一些无意义副词。然后提取固定数目（如100个）的词频最靠前的词作为语义坐标基，统计其余单词分别与这些坐标基词同时出现在标题中的次数，作为该词在该坐标基下的坐标值。如此可以得到每个词语在语义坐标系下的坐标向量，可以根据Kmeans等聚类方法进行聚类，得到语义相近的词的组合。最后再利用这个结果做一个应用，用户可以查找某一个作者，程序会返回他的相关领域关键词、相关领域作者。

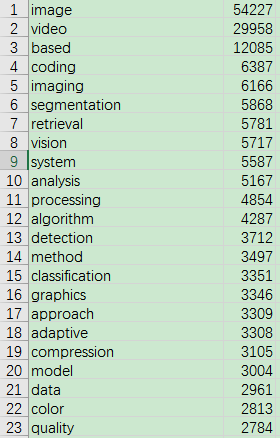
* 1. 特征词提取

本文进行语义分析的核心是特征词，特征词就是频繁出现在文章标题中的名词、形容词等包含领域意义的术语。为了统计得到这些特征词，首先从Article\_Author表中提取所有文章的title：



1. Article\_Author表中的所有文章标题

然后再用哈希表的方式筛选所有标题中出现的所有词的记录，并筛去“and”、“on”、“for”等副词以及出现次数少于10次的词，得到频繁出现的有意义的学术关键词及其计数，共计5061个，保存到KeywordsAll表中。



1. KeywordsAll表
   1. 语义特征向量计算

得到了关键词后则可以对关键词本身以及作者进行语义特征向量的计算与分析。其方法是：选取KeywordsAll表中词频排名前100的关键词作为向量基，然后统计KeywordsAll表中每个关键词分别与这100个关键词同时出现在一个标题中的计数。将这100个数字作为这个关键词的语义向量。作者的语义向量方法也类似，统计出每个作者的文章标题中分别出现这100个关键词的计数，作为语义向量。

具体的步骤包括：

首先统计关键词与其出现的文章的标题的关系表：KeywordsAll\_titles表：

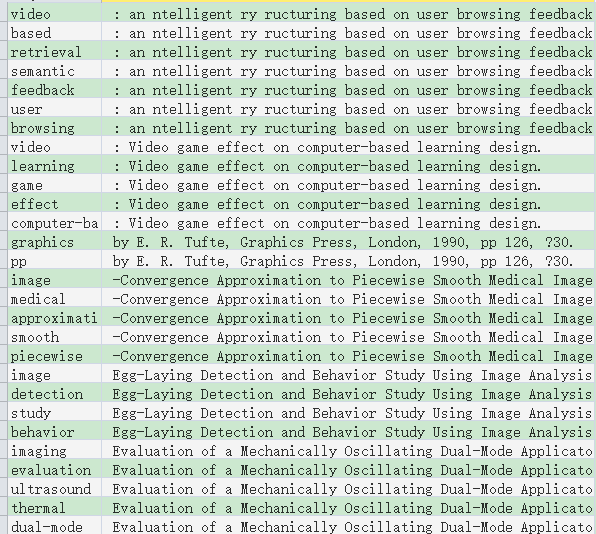
代码：

SELECT KeywordsAll.keywords AS keywords, titles.title AS title INTO KeywordsAll\_titles

FROM KeywordsAll, titles

WHERE titles.title LIKE '\* '+KeywordsAll.keywords+' \*';

结果：



1. KeywordsAll\_titles表

然后统计每个关键词与100个语义向量基共同出现在一篇文章中的计数KeywordsAll\_Vector表：

代码：

SELECT KeywordsAll1.keywords AS words, KeywordsAll.ID AS [index], COUNT(\*) AS vector\_value INTO KeywordsAll\_Vector

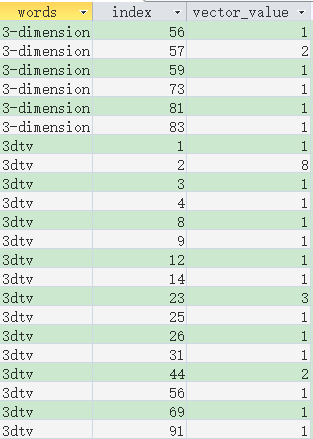
FROM KeywordsAll\_titles, KeywordsAll\_titles AS KeywordsAll\_titles1, KeywordsAll, KeywordsAll AS KeywordsAll1

WHERE KeywordsAll.ID<=100 AND KeywordsAll\_titles.title=KeywordsAll\_titles1.title AND KeywordsAll\_titles.keywords=KeywordsAll.keywords AND KeywordsAll\_titles1.keywords=KeywordsAll1.keywords

GROUP BY KeywordsAll1.keywords, KeywordsAll.ID

ORDER BY KeywordsAll1.keywords;

结果：



1. KeywordsAll\_Vector表

其中words是关键词，index表示100个语义向量的索引，vector\_value表示统计出来的频次，也就是向量值，有的index没有向量值，即为0。

然后统计作者的语义向量Author\_Vector表：

代码：

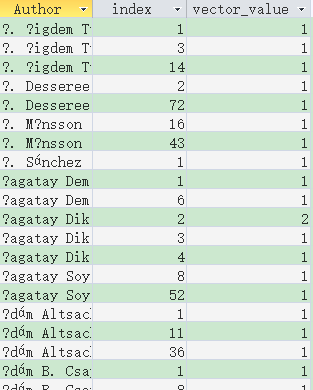
SELECT Authors.Author AS Author, KeywordsAll.ID AS [index], Count(\*) AS vector\_value INTO Author\_Vector

FROM Authors, Article\_Author, KeywordsAll\_titles, KeywordsAll

WHERE (((Authors.Author)=Article\_Author.Author) And ((KeywordsAll\_titles.title)=Article\_Author.title) And ((KeywordsAll.ID)<=100) And ((KeywordsAll.keywords)=KeywordsAll\_titles.keywords))

GROUP BY Authors.Author, KeywordsAll.ID;

结果：



1. Author\_Vector表

然后用MATLAB读取并整理KeywordsAll\_Vector表和Author\_Vector表的信息，然后在进行归一化，即可得到130516\*100的AuthorID\_Vector矩阵和5061\*100的WordsID\_Vector矩阵。其中每行代表了每个作者或关键词，100列代表向量的归一化值。

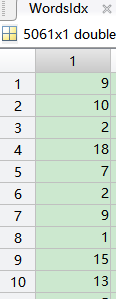
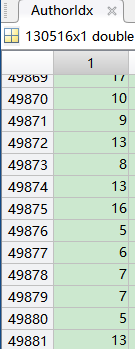


1. AuthorID\_Vector和WordsID\_Vector矩阵
   1. 聚类分析

使用kmeans算法，将关键词按语义向量分为20类。K-means算法的基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。



1. 分类后的关键词和作者以及每类对应的数量

1. 关键词和作者索引以及类别索引
   1. 自动语义注解的应用

根据得到的分类索引信息，可以构建关于作者的语义解释应用。简单地通过查询同一类（即同一领域）的关键词和相关作者即可。主要方法是根据作者所属的分类，连接关键词表和作者表等，找出和他同一分类的关键词和作者。

代码：

function [ relevantKeywords,relevantAuthors ] = SearchForAuthor( AuthorName, numOfData)

conna=database('cleanData','','');

curs=exec(conna,['select KeywordsAll.[keywords] from [AuthorID\_ClassID],[WordsID\_ClassID],[KeywordsAll],[Authors] where [WordsID\_ClassID].[WordsID]=[KeywordsAll].[ID] and [AuthorID\_ClassID].[ClassID]=[WordsID\_ClassID].[ClassID] and [AuthorID\_ClassID].[AuthorID]=[Authors].[ID] and [Authors].[Author]=''',AuthorName,'''']);

curs=fetch(curs,numOfData);

relevantKeywords=curs.Data;

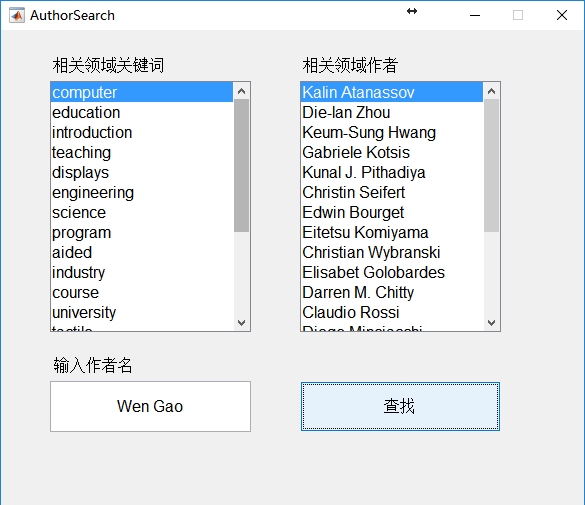
curs=exec(conna,['select Authors2.Author from AuthorID\_ClassID,AuthorID\_ClassID as AuthorID\_ClassID2, Authors,Authors as Authors2 where Authors2.ID=AuthorID\_ClassID2.AuthorID and AuthorID\_ClassID2.ClassID=AuthorID\_ClassID.ClassID and Authors.ID=AuthorID\_ClassID.AuthorID and Authors.Author=''',AuthorName,'''']);

curs=fetch(curs,numOfData);

relevantAuthors=curs.Data;

end

设计一个GUI界面，键入任何作者名称，点查找，集合得到相关领域关键词和作者的名录。



1. 作者的相关领域关键词和相关领域作者