代码说明

代码主要分为三部分：Java、Matlab和Database。

# Java

主要包括一些预处理程序。

* separateXmlFiles项目是用来拆散大xml文件的，这样方便后续处理。
* UsingAccessDatabase项目是读取xml文件，然后整理数据并将其写入Access数据库的。
* WordCount项目是用来统计在标题中词语出现的频次的。

# Matlab

主要是包括语义向量的计算，以及GUI界面设计的代码，需要用ODBC方式调用数据库，所以在使用之前需要将Database中的cleanData配置到ODBC数据库。

* getAuthorVector.m：计算作者的语义向量。
* getWordsVector.m：计算词语的语义向量。
* getIndex.m：获取索引。
* normalization.m：用来归一化语义特征向量。
* SearchForAuthor.m：查找某个作者的相关领域关键词和作者。
* AuthorSearch.m：GUI界面

# Database

主要是关于数据库整合和查询的代码，是程序的主体，主要采用SQL语言查询。

* Authors表：所有作者及其文章数目

代码：

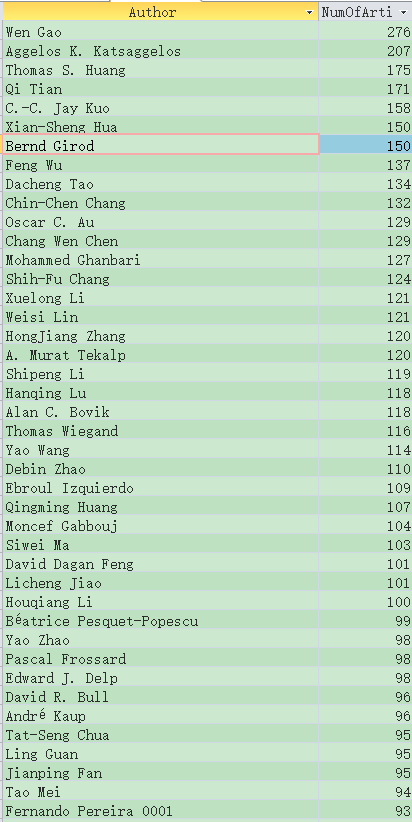
SELECT Article\_Author.[Author] AS Author, COUNT(\*) AS NumOfArticles

FROM Article\_Author

GROUP BY Author

ORDER BY COUNT(\*) DESC;

结果：共130516条记录



* Authors\_Frequent\_1（sup>2）表：支持度计数大于2的作者及其文章数目（一项频繁集）

代码：

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

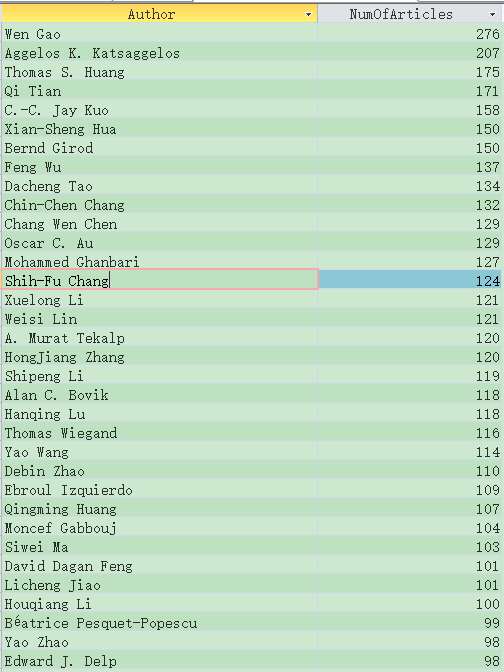
FROM Article\_Author

GROUP BY Author

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

ORDER BY COUNT(\*) DESC;

结果：共25662条记录



* Article\_Author\_Frequent\_1表：Authors\_Frequent\_1（sup>2）表与Article\_Author表连接

代码：

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

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



* Authors\_Frequent\_2（sup>2）表：支持度计数大于2的作者1、作者2及其合作文章数目（二项频繁集）

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

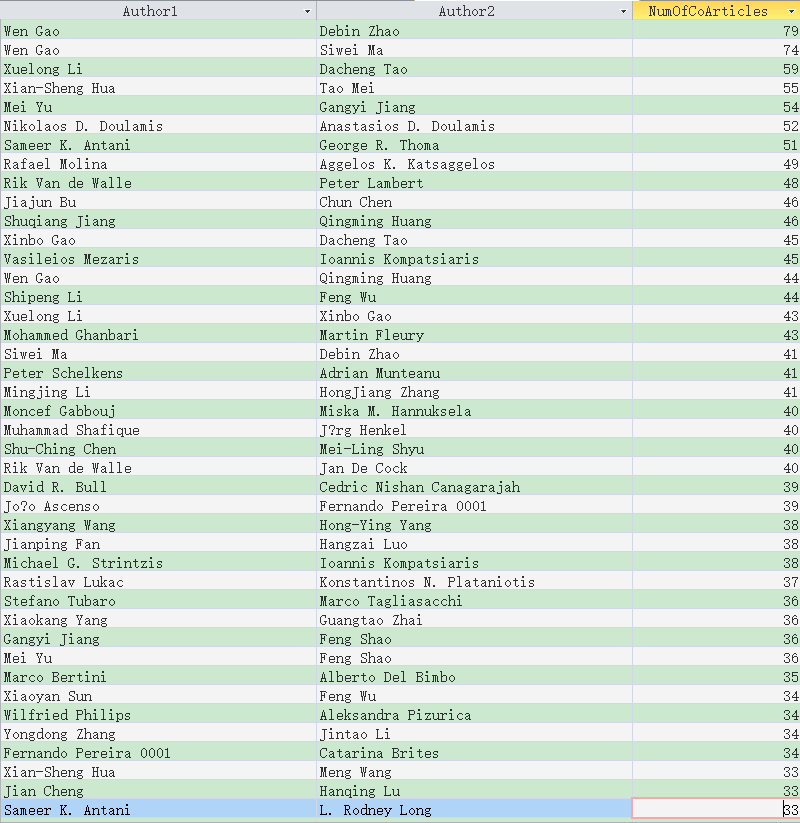
WHERE Article\_Author\_Frequent\_1.title=Article\_Author\_Frequent\_1\_copy.title AND Article\_Author\_Frequent\_1.[Author]>Article\_Author\_Frequent\_1\_copy.[Author]

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

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

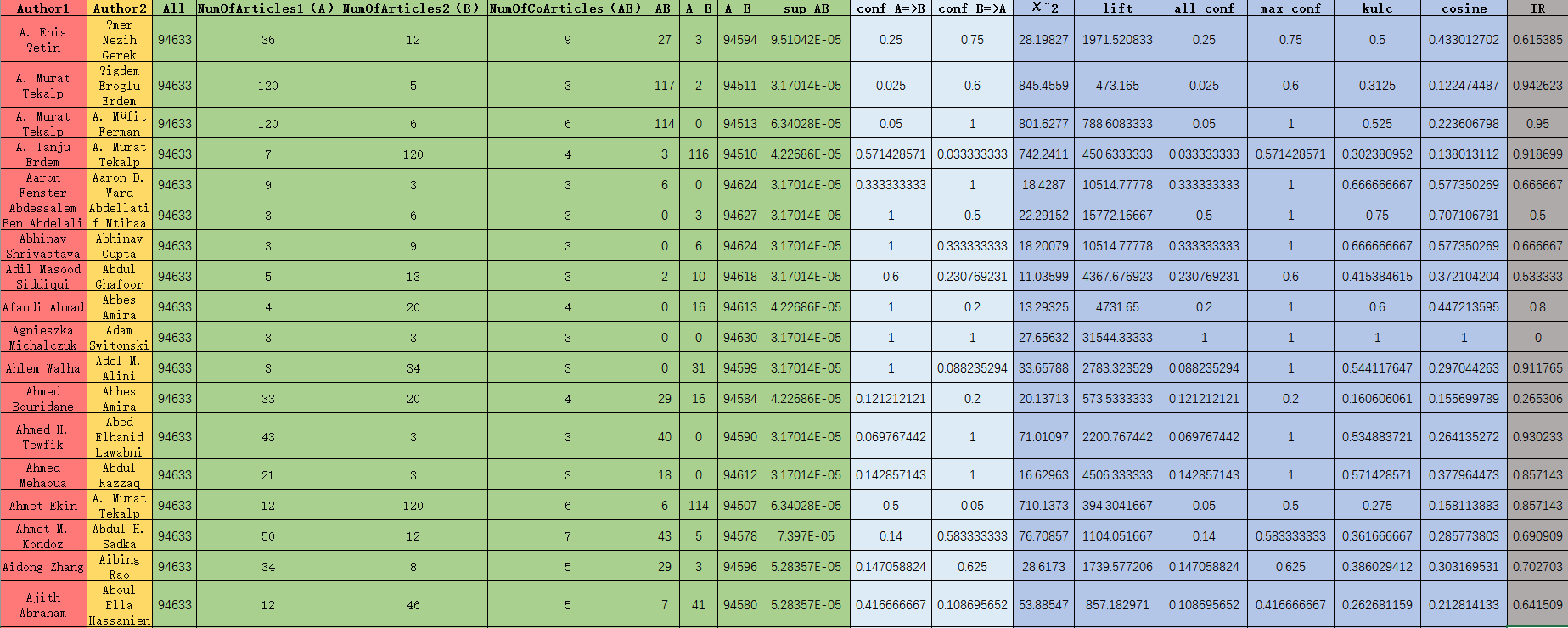
ORDER BY COUNT(\*) DESC;

结果：共24633条记录



* 用6种模式评估方法评估二项频繁集的关系，评价那种关系更好

将统计结果放入excel表格中，根据公式计算出各个模式评估方法



* 评估师生关系

需要求出每个作者第一篇文章的发表时间FirstPubYear

* Author表：

代码：

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

INTO Authors

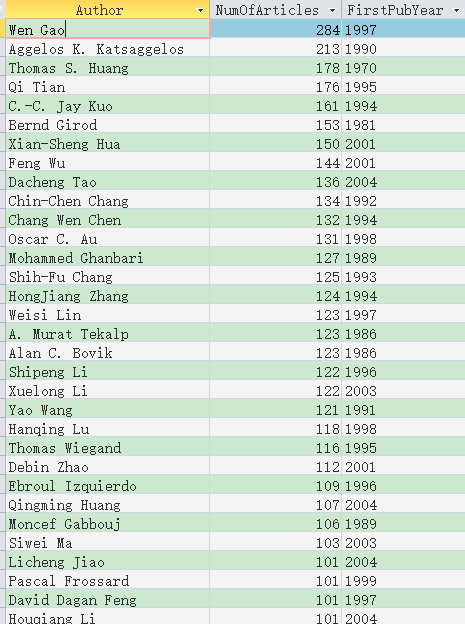
FROM Article\_Author, Articles

WHERE Articles.title=Article\_Author.title

GROUP BY Author

ORDER BY COUNT(\*) DESC;

结果：共130516条记录。



需要求出每个作者的coauthor个数

* NumOfCoAuthors表：

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：

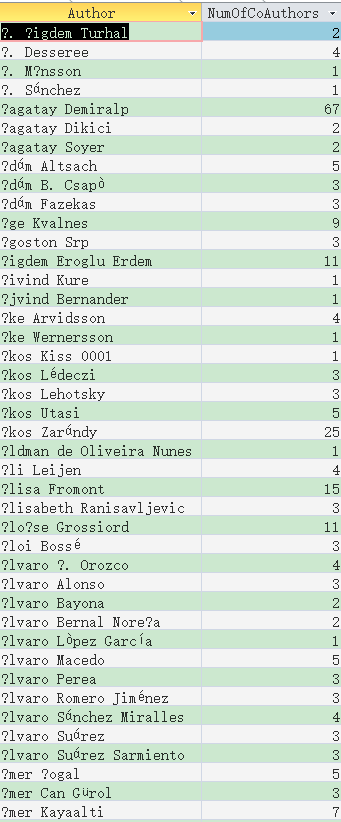
SELECT [查询NumOfCoAuthors].Author AS Author, COUNT(\*) AS NumOfCoAuthors

INTO NumOfCoAuthors

FROM [查询NumOfCoAuthors]

GROUP BY [查询NumOfCoAuthors].Author

结果：共127758条记录。



然后将所有信息，包括Frequent\_Patterns表、FirstPubYear表、NumOfCoAuthors表整合到一张表Complex

* Complex表：

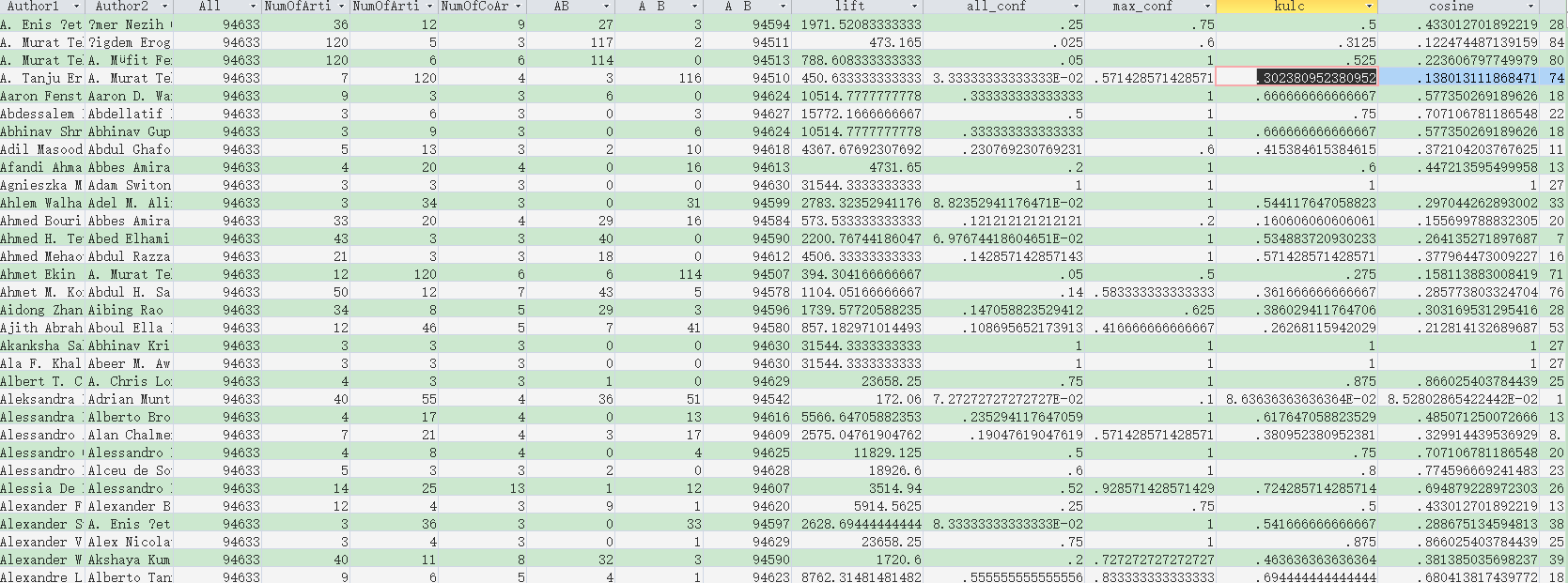
代码：

SELECT Frequent\_Patterns.Author1 AS Author1, Frequent\_Patterns.Author2 AS Author2, Frequent\_Patterns.[All] AS [All], Frequent\_Patterns.[NumOfArticles1（A）] AS NumOfArticles1（A）, Frequent\_Patterns.[NumOfArticles2（B）] AS NumOfArticles2（B）, Frequent\_Patterns.[NumOfCoArticles（AB）] AS NumOfCoArticles（AB）, Frequent\_Patterns.[AB ̅] AS [AB ̅], Frequent\_Patterns.[A ̅ B] AS [A ̅ B], Frequent\_Patterns.[A ̅ B ̅] AS [A ̅ B ̅], Frequent\_Patterns.lift AS lift, Frequent\_Patterns.all\_conf AS all\_conf, Frequent\_Patterns.max\_conf AS max\_conf, Frequent\_Patterns.kulc AS kulc, Frequent\_Patterns.cosine AS cosine, Frequent\_Patterns.[χ^2] AS [χ^2], Frequent\_Patterns.IR AS IR, NumOfCoAuthors1.NumOfCoAuthors AS NumOfCoAuthors1, NumOfCoAuthors2.NumOfCoAuthors AS NumOfCoAuthors2, FirstPubYear1.FirstPubYear AS FirstPubYear1, FirstPubYear2.FirstPubYear AS FirstPubYear2 INTO Complex

FROM Frequent\_Patterns, NumOfCoAuthors AS NumOfCoAuthors1, NumOfCoAuthors AS NumOfCoAuthors2, FirstPubYear AS FirstPubYear1, FirstPubYear AS FirstPubYear2

WHERE Frequent\_Patterns.Author1=NumOfCoAuthors1.Author AND Frequent\_Patterns.Author2=NumOfCoAuthors2.Author AND Frequent\_Patterns.Author1=FirstPubYear1.Author AND Frequent\_Patterns.Author2=FirstPubYear2.Author;

结果：共24633条记录。



最后筛选Complex表得到较为可能的师生关系表Complex\_Filtered

* Complex\_Filtered表：

代码：

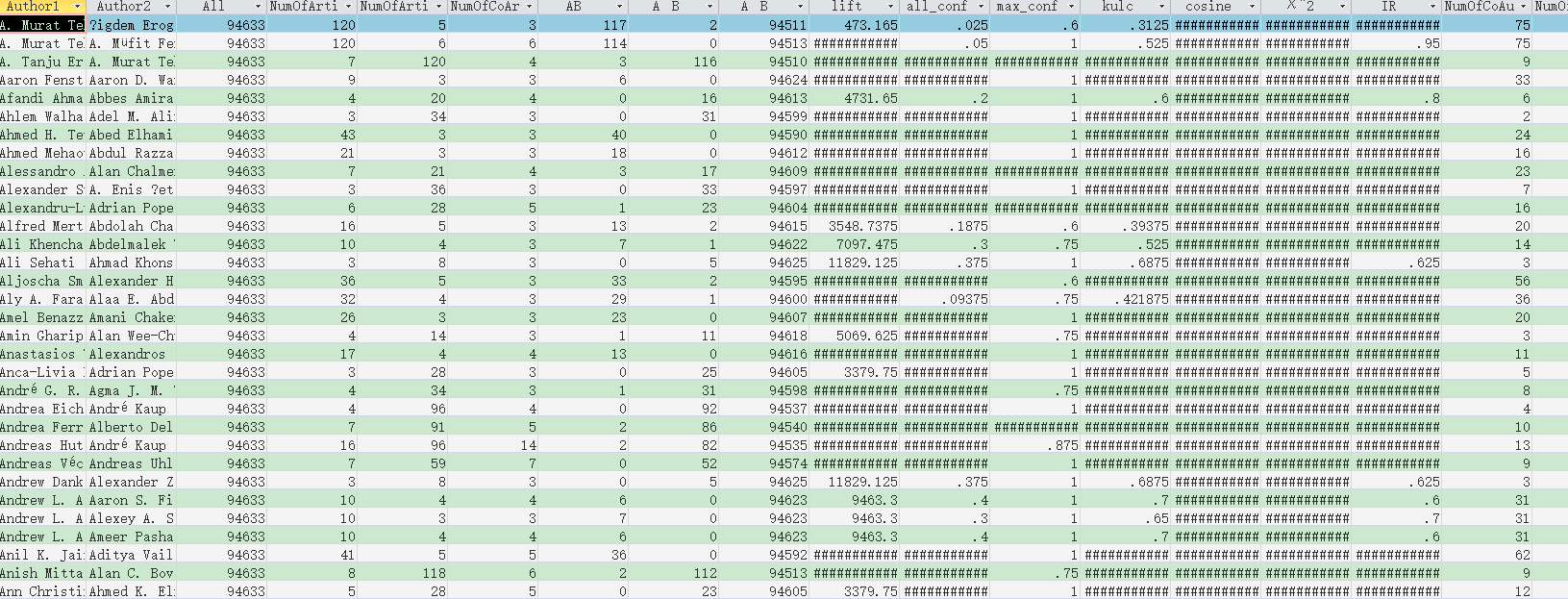
SELECT Frequent\_Patterns.Author1 AS Author1, Frequent\_Patterns.Author2 AS SELECT \*

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

结果：共5951条记录。



连接Complex\_Filtered、Article\_Author 、Articles等表得到Advisor\_Advisee表

* Advisor\_Advisee表：

代码：

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

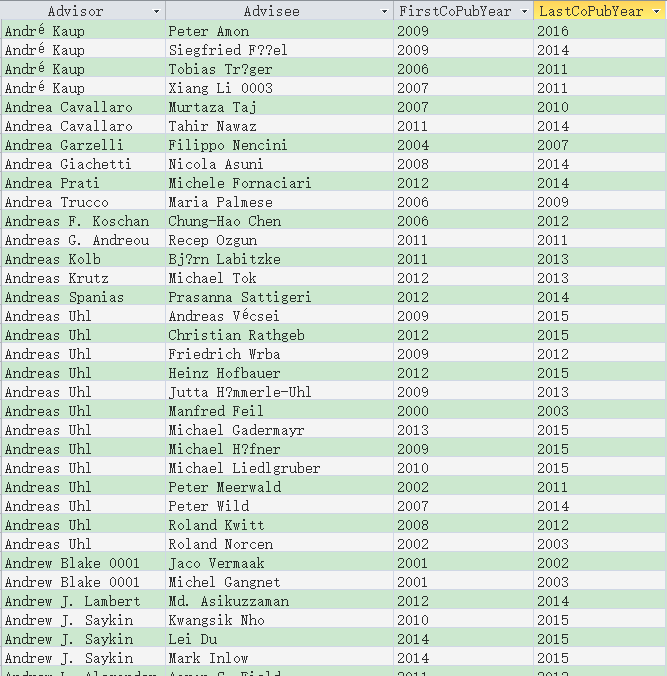
FROM [查询Advisor\_Advisee], Article\_Author AS Article\_Author1, Article\_Author AS Article\_Author2, Articles

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

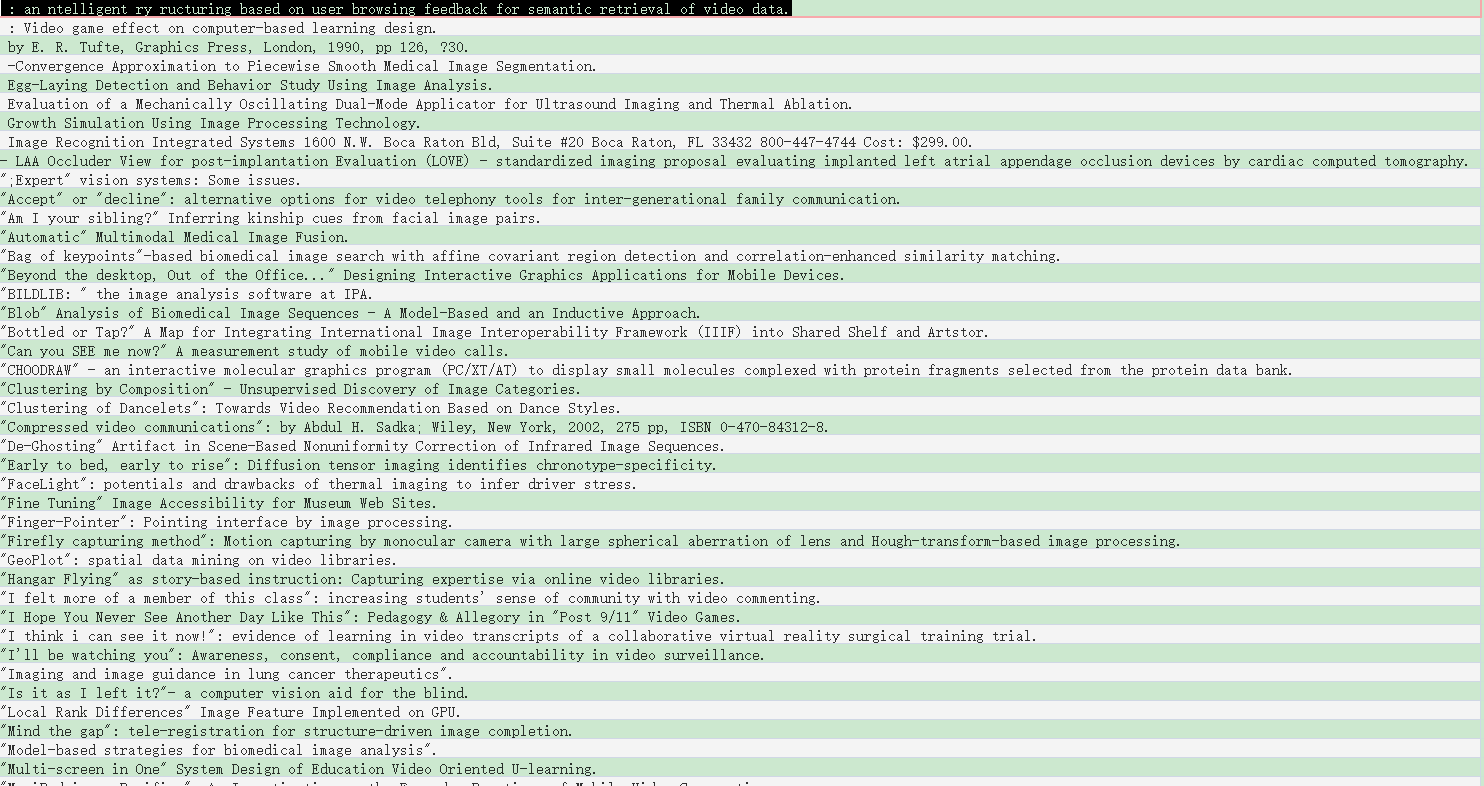
ORDER BY Advisor

结果：共5951条记录。

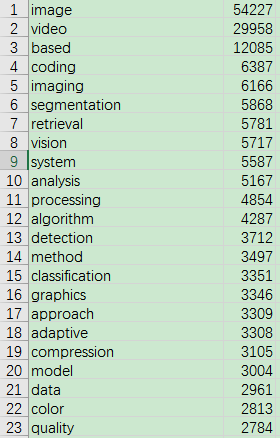


* 高级语义注解

首先从Article\_Author表中提取所有文章的title



然后对文本信息进行解析，并且筛去“and”、“on”、“for”等副词，得到频繁出现的有意义的学术关键词及其计数，共计5061个。



将计数前100的关键词提取出来，作为向量基。

计算每个词、每个作者的语义特征向量。与这100个特征关键词同时出现在某篇文章中，频度与频繁模式