**Frequent Pattern Mining and Automated Semantic Pattern Annotation In DBLP**

Abstract

The DBLP (DataBase systems and Logic Programming) computer science bibliography is the online reference for bibliographic information on major computer science publications. It lists the author's scientific research by years, including journal articles, conference papers, and other publications. As of May 2016, DBLP indexes over 3.3 million publications, published by more than 1.7 million authors. The quality of journals and conference papers included in DBLP is very high, and the speed of literature renewal is very fast, which reflects the forefront of international academic research. This paper mainly uses DBLP database to do the following analysis: First, analyze the authors that have coauthor relationships, predict advisor-advisee relationships and the approximate period for such advisory supervision based on the frequent pattern mining method; Second, generate the semantic annotation of the author’s field and other semantic features.

*Keywords*: data mining, DBLP, frequent pattern, automated semantic annotation

Contents

1 Introduction

2 Data Preprocessing

2.1 Data Extraction

2.2 Data Input

2.3 Database Filter

3 Frequent Pattern Analysis of Coauthor Relationships

3.1 Mining Frequent Coauthor Relationships

3.2 Frequent Pattern Evaluation

* 1. Mining Advisor-Advisee Relationships

4 Automated Semantic Pattern Annotation and its Application

4.1 Feature Words Extraction

4.2 Semantic Feature Vector Calculation

* 1. Cluster Analysis

4.4 Application of Automated Semantic Pattern Annotation

1. **Introduction**

Data mining generally refers to the process to search hidden information in enormous numbers of data by using algorithms. The DBLP (DataBase systems and Logic Programming) computer science bibliography is the online reference for bibliographic information on major computer science publications. It lists the author's scientific research by years, including journal articles, conference papers, and other publications. As of May 2016, DBLP indexes over 3.3 million publications, published by more than 1.7 million authors. The quality of journals and conference papers included in DBLP is very high, and the speed of literature renewal is very fast, which reflects the forefront of international academic research. This paper mainly uses DBLP database to do the following analysis: First, analyze the authors that have coauthor relationships, predict advisor-advisee relationships and the approximate period for such advisory supervision based on the frequent pattern mining method; Second, generate the semantic annotation of the author’s field and other semantic features.

The main steps are:

* Data preprocessing.

The data in DBLP is stored as an XML (Extensible Markup Language) file. XML files can be used to markup and describe data. XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is very suitable for web transmission, providing a unified way to describe and exchange structured data that is independent of the application or vendor. XML is a cross-platform, context-dependent technology in the Internet environment and an effective tool for today's distributed architecture information. But it also has a lot of problems, for instance: compared with the database method, its query, insert, modify, delete and other operations are more time-consuming; it is more difficult than relational database to connect, filter, computing and other operations ; And the text representation of XML, symbolization of the mark will lead to increase the size of XML data larger than binary data representation, especially when the amount of data is large, the efficiency becomes a big problem; the biggest problem is data members in the each record of the DBLP data set are not uniform, and some attributes do not exist in every record, which leads to some problem in the data analysis. Therefore, data preprocessing is required. Because the downloaded XML file is quite large (1.67GB), most of the existing XML parse libraries cannot directly read the complete file for parsing, hence consider using JAVA's SAX method to segment the data set and then filter through keywords, integrate, process, and finally input the data to ACCESS database.

* Coauthor relationship analysis.

Firstly, Apriori algorithm is used in mining frequent authors and papers, find out the frequent authors’ and coauthors’ papers, and obtain the quantity of papers that they published both individually or in groups, then the support and the confidence can be calculated. Evaluate the frequent pattern by calculating the lift, χ2, all confidence, max confidence, Kulczynski, cosine, and the imbalance ratio (IR). These can evaluate the closeness of coauthor relationship. By using the above methods, the imbalance ratio, and the statistical order of the authors’ papers publication, it is possible to roughly analyze the possible advisor-advisee relationship and the time when they start coauthoring paper together based on the publication time.

* Automated semantic annotation.

Mainly analyze the semantic links between papers, keywords, authors and so on, and cluster the data. First, the title of all papers will be extracted and the occurrences of each word will be counted, and some meaningless adverbs will be deleted. And then extract the fixed number (such as 100) of the most frequent words as the semantic coordinate axis, count the occurrences of the rest of the remaining words that appear together with these coordinate axis words in the title as the coordinates value on these coordinate systems. In this way, we can get the coordinate vector of each word in the semantic coordinate system, and can be clustered according to the clustering method such as k-means to get the combination of semantic similar words.

1. **Data Preprocessing**

Since the data in the DBLP is stored as an XML file, it is more time-consuming to query, insert, modify, delete; and difficult to connect, filter, and operate other operations than a relational database. Besides, when the data size is extremely large, efficiency becomes a big problem; the biggest problem is the data members in each record of the DBLP data set are not uniform, and some attributes do not exist in every record, which leads to some problem in the data analysis. Therefore, data preprocessing is required. In the XML file parsing process, there are three tools to be chosen: DOM, SAX and DOM4J.

DOM is the official W3C standard for representing XML documents in a platform-independent and language-independent manner. A DOM is a collection of nodes or pieces of information organized in a hierarchical structure. This hierarchy allows developers to find specific information in the tree. Analyzing the structure usually requires loading the entire document and constructing the hierarchy before any work can be done. Since it is based on the information hierarchy, the DOM is considered to be tree-based or object-based. DOM and generalized tree-based processing have several advantages. First, since the tree is persistent in memory, it can be modified so that the application can make changes to the data and structure. It can also navigate up and down at any time in the tree, rather than disposing like SAX. DOM is much simpler to use. However, there is a fatal problem in the current application: the DOM tool will read the entire XML file into memory and then parse the data, and will use large numbers number of memory. The DBLP database XML file is 1.67GB, DOM simply cannot do the job.

The advantages of SAX processing are very similar to the advantages of streaming media. The analysis can start immediately, rather than waiting for all the data to be processed. Besides, since the application will only check the data when reading data, there is no need to store the data in memory. This is a huge advantage for large documents. In fact, an application does not even have to parse the entire document; it can stop parsing when a condition is met. The SAX parser uses an event-based model that triggers a series of events when parsing XML documents. When a specific tag is found, it can activate a callback function, pass the information that the tag. SAX memory requirements are usually relatively low, because it allows developers to decide the tag they want to process. In particular, when the developer only need to deal with some of the data in the document, this extended ability is better reflected. In general, SAX is much faster than its replacement, DOM. But SAX also has the problem, because it uses the sequential read method, when encounters certain types of specific nodes, it triggers some events, conducts the corresponding process. Therefore, it can only deal with the current XML node data, cannot access the entire tree hierarchy of the file at one-time access as DOM does, it is also difficult to backtracking.

DOM4J has the advantages of DOM and SAX. DOM4J and SAX to take the same method to sequentially read the file to reduce the used memory; However, it provides the node backtracking function, it is possible to obtain the current node of the parent node information, and the hierarchy between nodes. It also supports supporting Xpath method to access nodes, the operation is more convenient than SAX. Therefore, this paper uses the DOM4J method to deal with DBLP XML data files.

* 1. **Data Extraction**

The data labels in dblp.xml include *articles, inproceedings, proceedings, book, incollection, phdthesis, mastersthesis, www*, and so on. Therefore, when defining xmlSAXReader to read nodes, all of the above nodes need to be included.

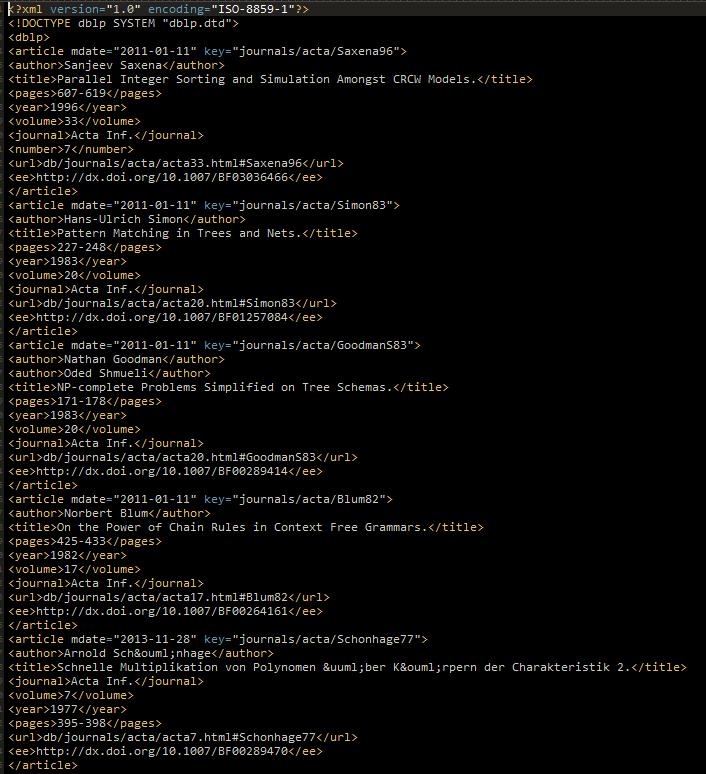


Figure 1 dblp.xml data fragment

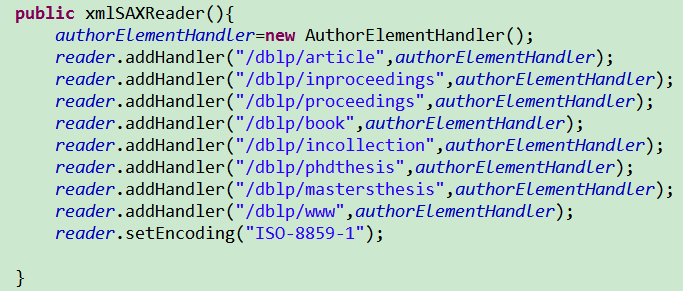


Figure 2 xmlSAXReader defined to read the nodes

Each paper record may *contain title, Author, key, mdate, type, journal, booktitle, publisher, isbn, year, month, crossref, volume, pages, ee, url, cdrom, number, address, cite, note, series, school, chapter* and other fields. Therefore, when reading the nodes, it is necessary to determine whether each node contains these fields to input to the database. The concrete way is to save all of its child nodes in a node queue whenever a record is read. Then the appropriate data is inserted to the database.

**2.2 Data Input**

Microsoft Access 2016 is used as the database; it supports SQL language operations. The data input process mainly uses database connection and insert operations.

The first step is to design the database. Considering the complexity of the data in the DBLP dataset, a paper is often written by several authors, and if the data is separated to several fields, each DBLP record will be spread out across several database records, such as a paper with three authors, the record will be separated to three records, each record in the other fields are the same, only the author field is different. It will not only waste memory space, but also unconducive for the data filtering, connection, query and other operations. Hence, it is necessary to consider designing the database into two tables: Articles and Article\_Author tables.

Articles table contains all fields except the *Author* field, including *title, key, mdate, type, journal, booktitle, publisher, isbn, year, month, crossref, volume, pages, ee, url , Cdrom, number, address, cite, note, series, school, chapte*r. The title is the key value, and some of the fields in some records are empty, so the dataset is sparse.

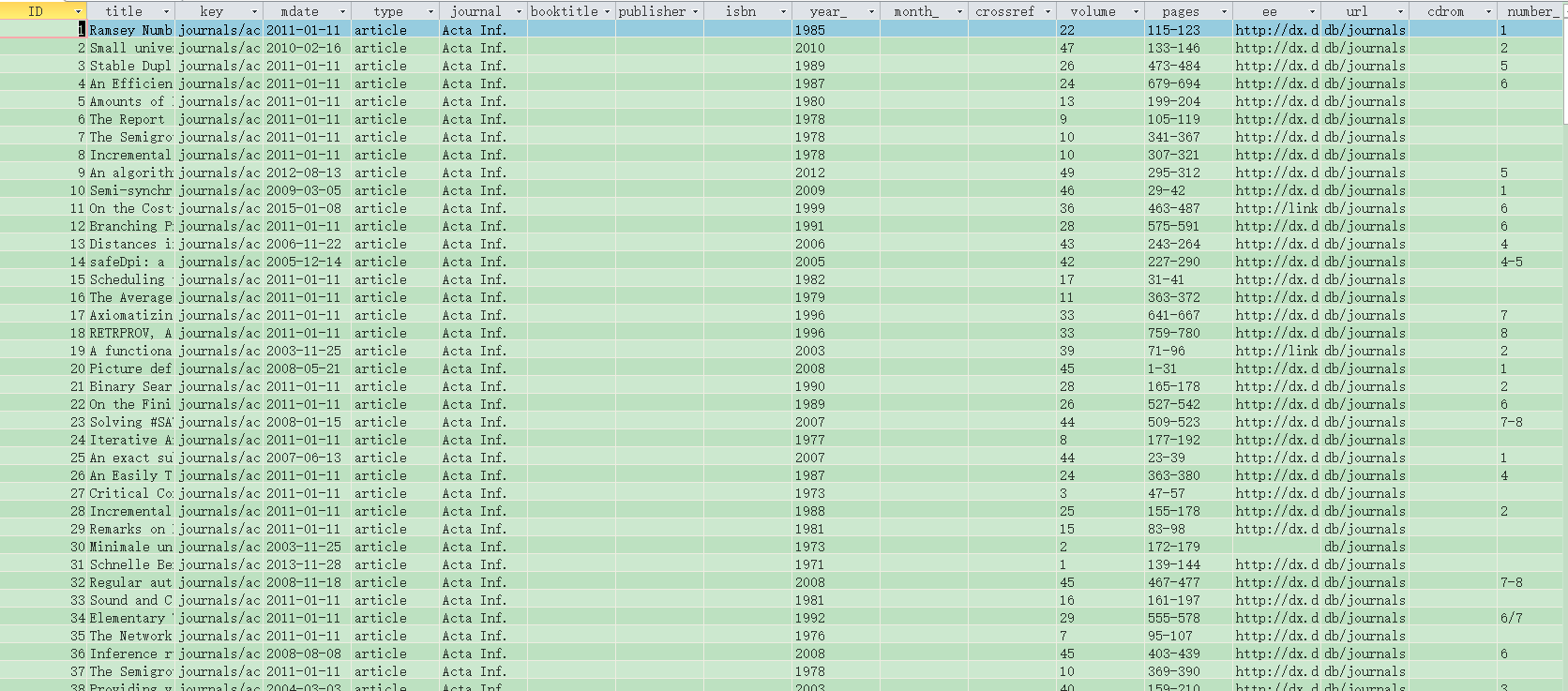


Figure 3 Articles table

The Article\_Author table stores the relationship records of the article and the author, and each record contains the *title* and *Author* fields, representing an article and its author.

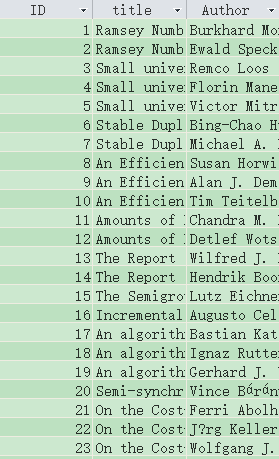


Figure 4 Article\_Author table

**2.3 Database Filter**

After the above steps, a complete database includes the article and the author information are obtained, but because the data records are extremely large, and not all articles are important in this research, hence the keywords are used to filter the title of the article. In this section, the study of image processing related to the literature and the author is conducted, so the filter keywords are: vision, image, graphics imaging, opencv, video.

In the Access database, the SQL query statement:

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

INTO Article\_Author

FROM Article\_Author

("(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 \*');

Articles table filtering:

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 \*');

The filter results: Article\_Author table’s total number of records is 307384; Articles table’s total number of records is 96388.



Figure 5 Article\_Author table after filtered

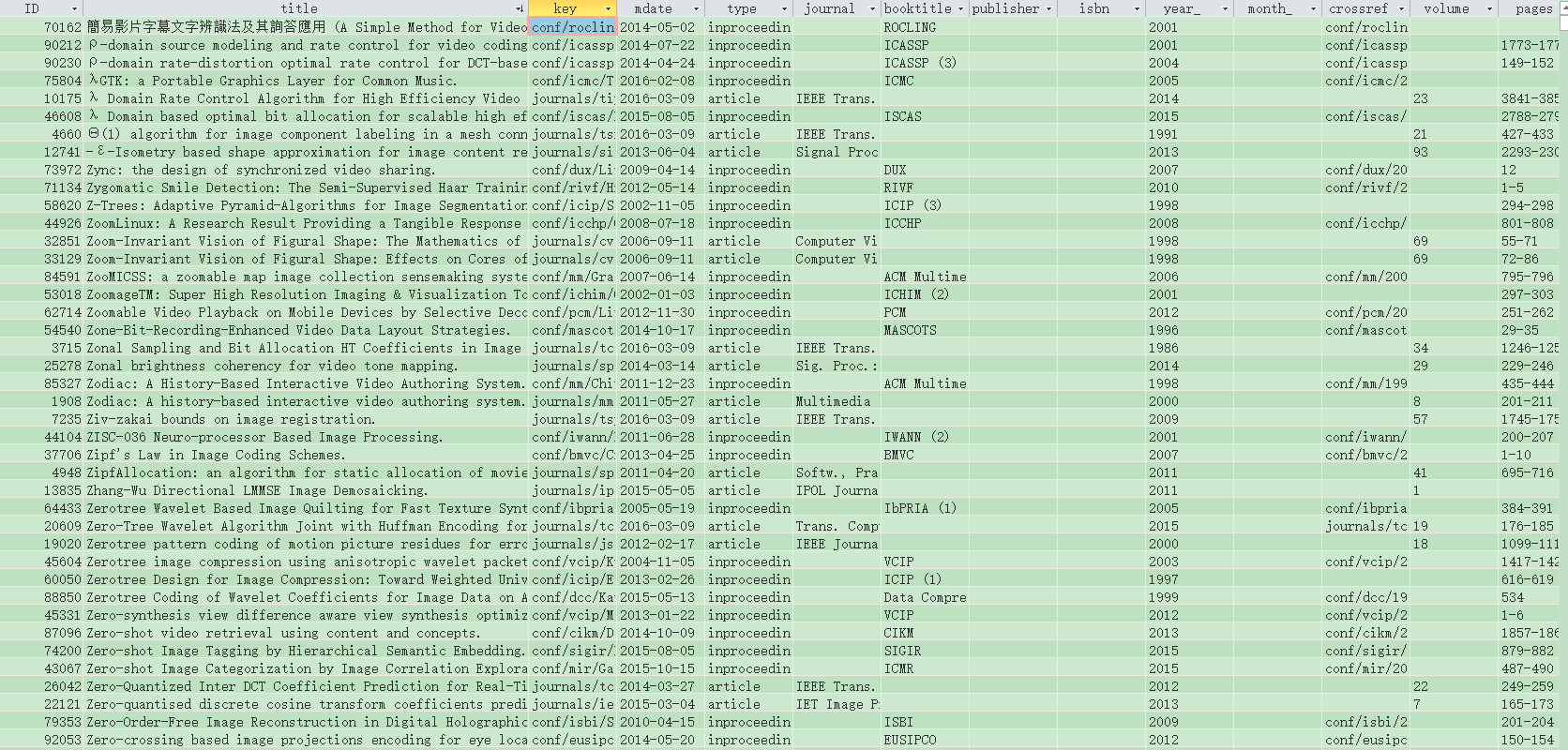


Figure 6 Articles table after filtered

1. **Analysis of Frequent Patterns of Coauthor Relationship**

In this chapter, the coauthor relationship mining will be conducted from the processed data. The main steps are: 1. Find the authors in the frequent coauthor relationship, in other words, the authors that frequently publish papers together or the frequent itemsets. 2. Using a series of evaluation mentioned in the textbook to analyze the frequent patterns of the coauthor relationship; 3. According to the results of the analysis, add some information of the papers, find the advisor-advisee relationships.

**3.1 Frequent Coauthor Relationships Mining**

The first step is frequent coauthor mining. Consider the use of Apriori algorithm that depend mainly on the Apriori property, which states the fact that all nonempty subsets of a frequent itemset must also be frequent. If two authors published several articles together, so they are frequent 2-itemsets; each corresponding author must publish several articles as a frequent 1-itemset. Therefore, to obtain high frequency itemset, the low frequency itemset should be mined first, hence filtering the group of frequent authors candidates by using this method, the data to be mined decreases as well as time and space consumed.



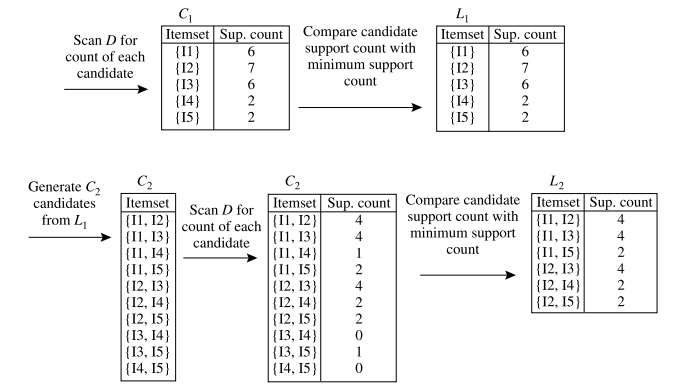


Figure 7 Apriori algorithm (pseudo-code) and flow chart

Firstly, mine one item frequent itemset by counting the number of authors and articles, let the number of articles as its support count, and sort according to the number of articles to get Authors table.

Code:

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

INTO Authors

FROM Article\_Author

GROUP BY Author

ORDER BY COUNT (\*) DESC;

Results: the total number of records is 130516

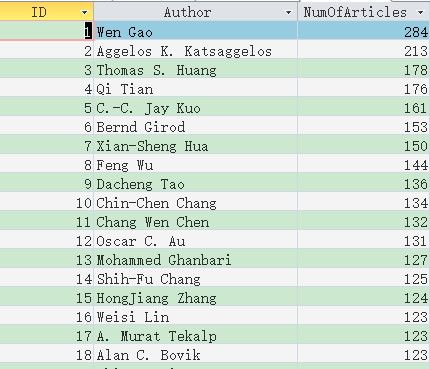


Figure 8 The Authors table

And then set the minimum support count threshold is 3, removing authors that published article less than 3, get Authors\_Frequent\_1 (sup> 2) table (one author frequent set, the minimum support count is 3):

Code:

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

INTO [Authors\_Frequent\_1 (sup> 2)]

FROM Authors

WHERE NumOfArticles> 2

ORDER BY NumOfArticles DESC;

Results: the total number of records is 25662



Figure 9 Authors\_Frequent\_1 (sup> 2) table

Connect Authors\_Frequent\_1 (sup> 2) table and the table Article\_Author to get Article\_Author\_Frequent\_1 table as the candidates of frequent 2-itemset mining:

Code:

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)]

ORDER BY title DESC;

Results: the total number of records is182351



Figure 10 Article\_Author\_Frequent\_1 table

From the Article\_Author\_Frequent\_1 table, the set of 2-frequent itemset of Authors\_Frequent\_2 (sup> 2) tables can be obtained by connecting two Article\_Author\_Frequent\_1 tables, where the connection rule is the same title field of the articles, and you can get the two authors of articles: Author1 and Author2. At the same time, COUNT (\*) can be used to count the number of 2-author articles in descending order, as the NumOfCoArticles field of the new table, and use the HAVING clause to remove the number of articles that less than two.

Code:

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;

Results: A total of 24633 records.

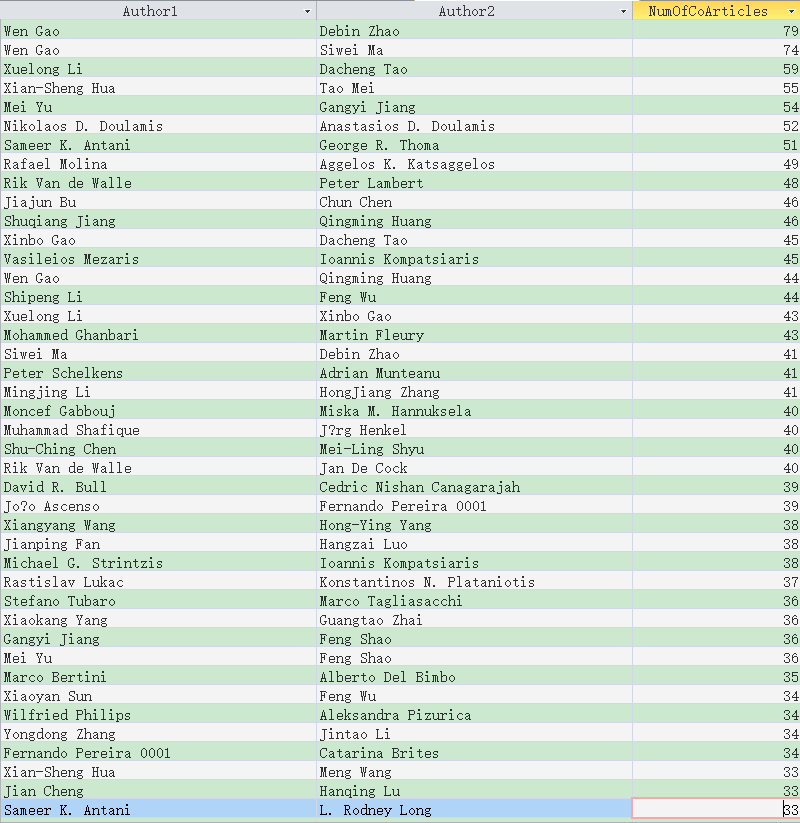


Figure 11 Authors\_Frequent\_2 (sup> 2) table

The frequent authors, coauthors, the number of articles mining have been completed. The number of articles can be used as the support count; these data can be used for the frequent pattern evaluation.

**3.2** **Frequent Pattern Evaluation Measures**

According to the Authors\_Frequent\_2 (sup> 2) table, Authors\_Frequent\_1sup> 2) table, the number of frequent multi-authored articles NumOfCoArticles (AB), the number of individual articles NumOfArticles1 (A) and NumOfArticles2 (B), the total number of articles, and other information. Integrate these data into the EXCEL spreadsheet, and use formula to calculate the other table fields:

* AB ̅ = A-AB;
* A ̅ B = B - AB;
* A ̅ B ̅ = All-A-B + AB;
* Support (AB): sup\_AB = AB / All;
* Confidence (A => B): conf\_A => B = AB / A;
* Confidence (B => A): conf\_B => A = AB / B;
* Chi-square value: x^2=(AB ̅-AVG (AB)) ^ 2 / AVG (AB ̅) + (A ̅ B -AVG) (A ̅ B)) ^ 2 / AVG (A ̅ B) + (A ̅ B ̅ -AVG (A ̅ B ̅)) ^ 2 / AVG (A ̅ B ̅);
* Lift = AB \* All / (A \* B);
* All confidence: all\_conf = AB / MAX (A, B);
* Maximum confidence: max\_conf = AB / MIN (A, B);
* Kulczynski measure: kulc = (AB / A + AB / B) / 2;
* Cosine = SQRT (AB \* AB / (A \* B));
* imbalance ratio: IR = ABS (A-B) / (A + B-AB).

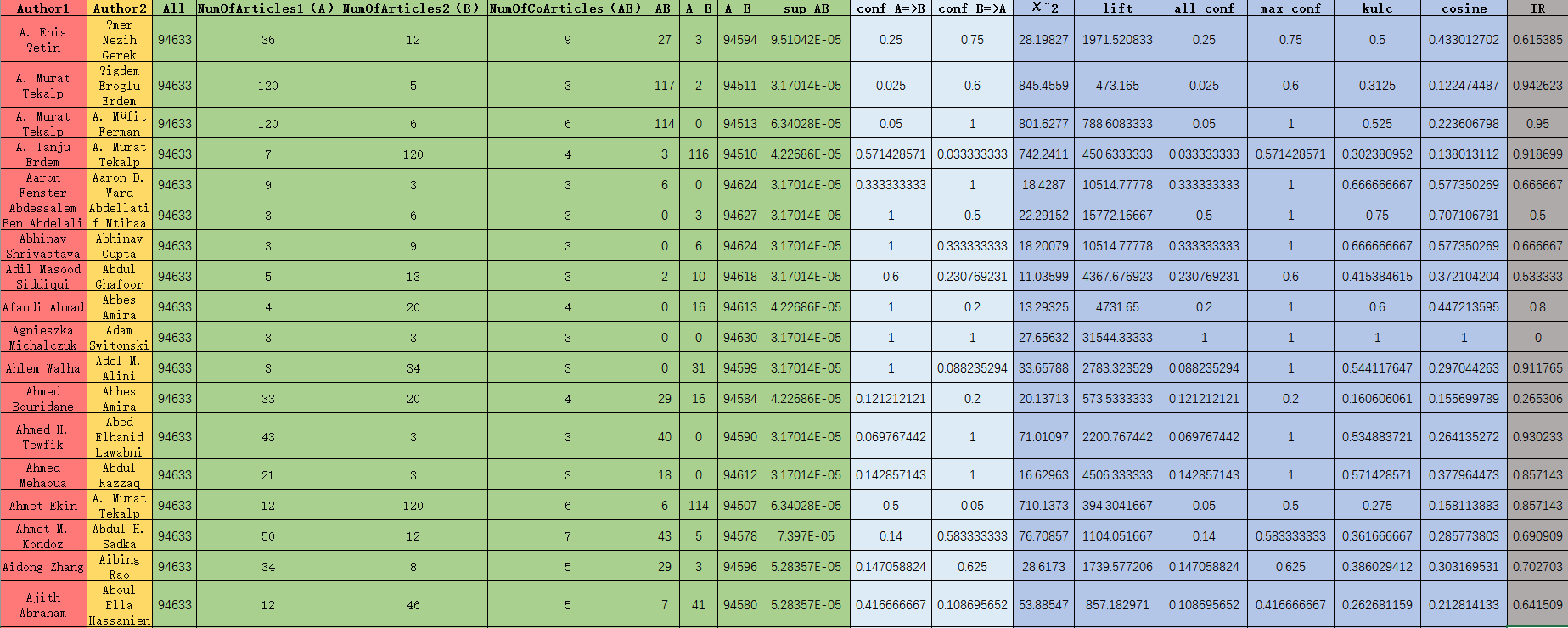


Figure 12 Frequent pattern evaluation table

It can be seen from the chi-square value and the lift measure are influenced by the number of total records ‘All’, the values are sometimes large and sometimes small, hence they cannot be good evaluation measures. The total confidence, maximum confidence, Kulc and cosine measures are not free from the influence of the total number of records; are null-invariant measures, and the range is between 0 to 1, may intuitively reflect the relationship between the coauthors.

If we only want to study the closeness between coauthors, the Kulc and cosine measures better because they are the arithmetic mean and the average of the two types of confidence respectively, which can reflect the coauthor relationships between two authors.

And if we want to study Advisor-Advisee Relationship among the authors, only considering the closeness coauthors relationship is not enough, it is necessary to consider the identity difference between them. Therefore, it is more effective to comprehensively consider of the total confidence, the maximum confidence and imbalance ratio. Because in general, the advisor published more articles than the advisee; and the proportion of the articles that are coauthored with the advisee is generally small. The number of articles published by the advisee is also relatively few; and the proportion of articles that are coauthored with the advisor is large. Therefore, these will lead to a smaller all confidence, greater the maximum confidence, and larger imbalance ratio. These characteristics will play important roles in the next section of this paper.

**3.3 Advisor-Advisee Relationship Mining**

From the analysis in Section 3.2, the frequent coauthor relationships and the frequent pattern measures are obtained. Besides, the advisor-advisee relationship mining needs to consider the all confidence, maximum confidence and the imbalance ratio. But these measures can only explain the coauthors relationships are quite close; they are not at the same level and the imbalance in the relationship is also obvious. But it is not enough to determine whether the coauthor relationships are advisor-advisee relationship.

In order to solve this problem, we need to consider the publication time of the article and the number of coauthors per author. Generally, the advisor’s age is at least 5 years older than the advisee’s age. Therefore, the advisor should have published a paper more than 5 years earlier than the advisee for the first time. In addition, the number of advisor’s coauthors will be more than the advisee’s. In the mining process these two conditions will be added. Hence, the first step is to obtain each author’s first publication time and the number of coauthors.

First, NumOfCoAuthors table, connect two Article\_Author table, the rules are the same article name, the author name is different; and then count the number of coauthor per author:

Code:

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;

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

NumOfCoAuthors

INTO NumOfCoAuthors

FROM [query NumOfCoAuthors]

GROUP BY [query NumOfCoAuthors] .Author

Results: : the total number of records is 127758.

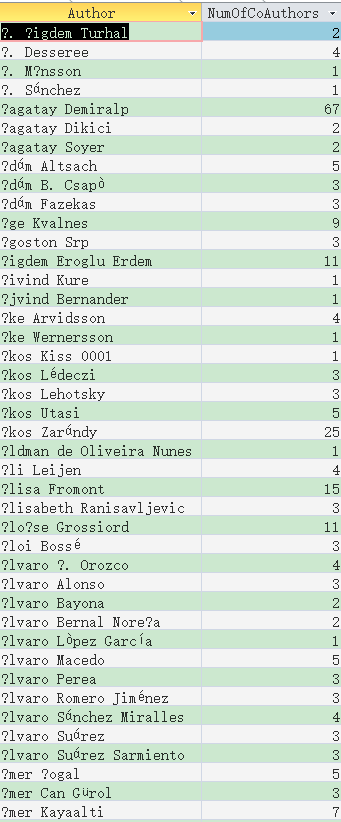
d

Figure 13 NumOfCoAuthors table

Then to obtain the FirstPubYear table: connect the Article\_Author and Articles table, select each author’s earliest publication year as FirstPubYear:

Code:

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;

Results: Total 130516 records.

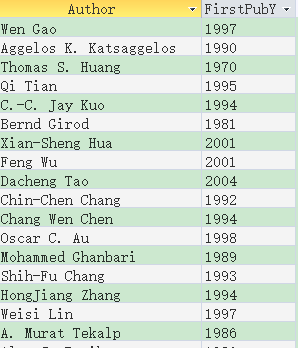


Figure 14 FirstPubYear table

Finally, connect Article\_Author, Articles and other tables to get Advisor\_Advisee table, where the filter conditions are: maximum confidence> 0.5; maximum confidence is 50% greater than the all confidence; both FirstPubYear difference is more than 5 years; NumOfCoAuthors difference is greater than 2:

Code:

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

INTO Advisor\_Advisee

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

WHERE [Query Advisor\_Advisee].Advisor=Article\_Author1.Author AND [Query 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

Results: 5951 records.

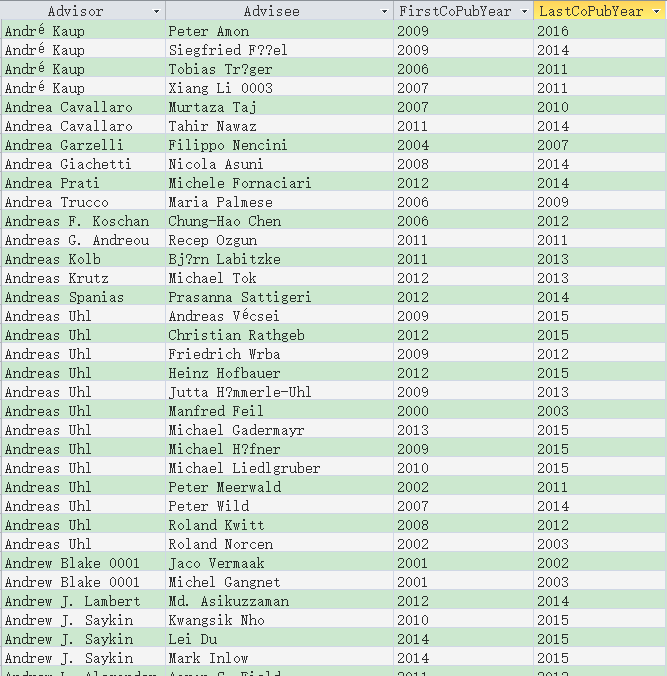


Figure 15 Advisor\_Advisee table

The table shows the earliest and latest year of the advisor-advisee’s publication articles. According to experience, the advisee usually begins to publish the paper one year after collaborating with the advisor. Therefore, the time of the advisor-advisee relationship is usually from (*FirstCoPubYear*-1) to *LastCoPubYear*. From the data in the table, each advisor has several advisees in different periods. The Advisor\_Advisee relationship histogram from the table above illustrates the problem better.



Figure 16 Advisor\_Advisee relationship histogram

As shown on the figure above, the horizontal axis represents the duration of the Advisor\_Advisee relationship, from 1 year to 17 years. The vertical axis represents the number of years that the Advisor\_Advisee relationship in each year. Most of the authors maintain the relationship between 2 to 4 years, most of them are 3 years, there are 1377 pairs, the others are 1 year, 5 to 8 years; the overall average is 4.3 years. Most of the master's degree students study in university for 2 to 3 years. Ph.D. or postdoctoral students may study for 5 to 8 years; hence the results of mining are quite reasonable.

1. **Automated Semantic Pattern Annotation and its Application**

This section mainly analyzes the semantic relationships between articles, keywords, and authors, and classifies them. First, the title of all papers will be extracted and the occurrences of words will be counted, and some meaningless adverbs will be deleted. And then extract the fixed number (such as 100) of the word with highest occurrences as the semantic coordinate base, the rest of the remaining words with these coordinates of the word at the same time in the title of the number of times as the coordinates of the coordinates of the word. In this way, we can get the coordinate vector of each word in the semantic coordinate system, and can be clustered according to the clustering method such as k-means to get the combination of semantic similar words. Finally, the use of this result to do an application, the user can find a certain author, the program will return to his related field keywords, related areas of the author.

**4.1 Feature Words Extraction**

The core of semantic analysis is the feature words analysis, the feature words are the nouns, adjectives, or other specific terms that frequently appear in the title of papers. In order to get these feature words, firstly, extract all the title of the papers from the Article\_Author table:

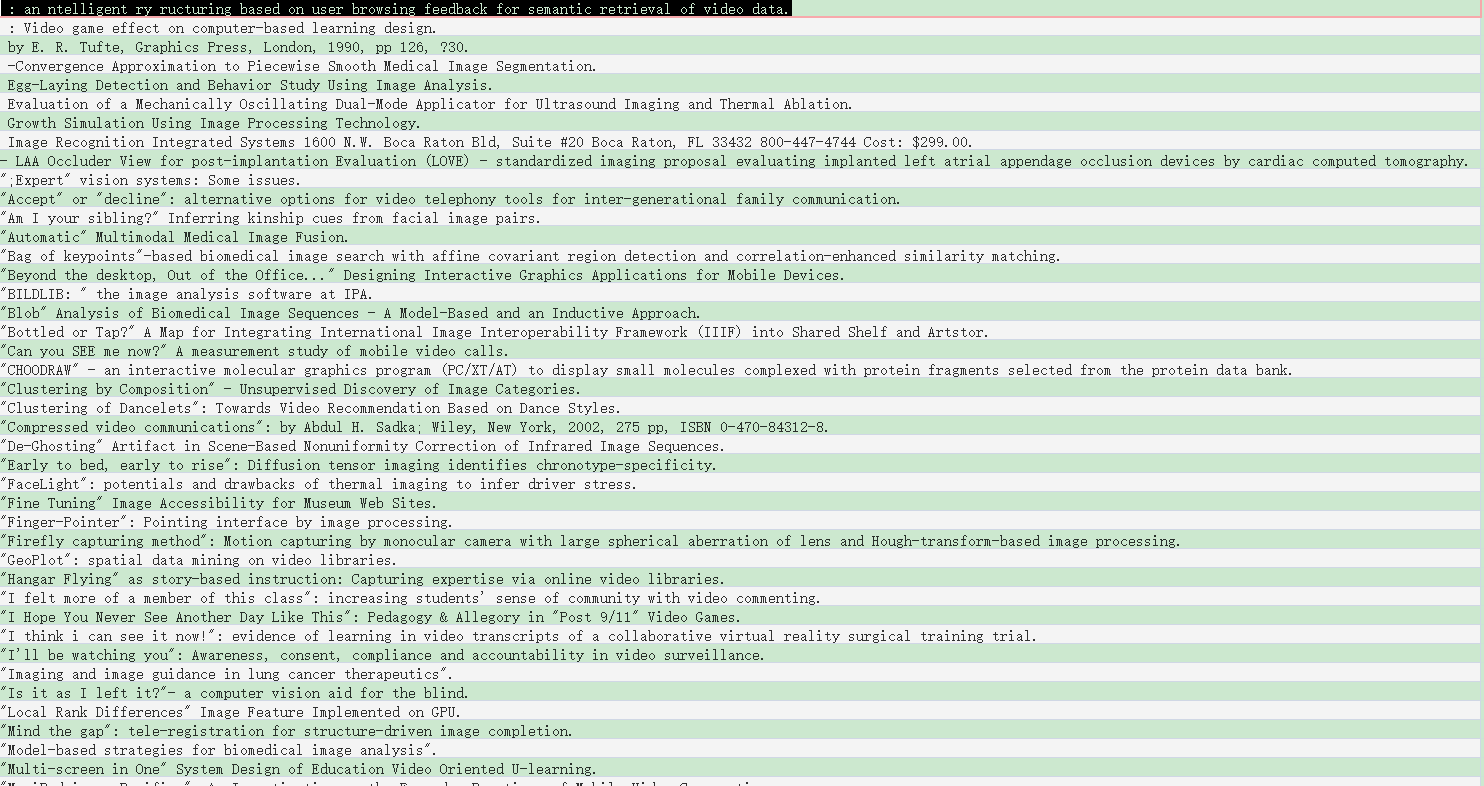


Figure 17 All title of papers in the Article\_Author table

And then use the hash table to filter all the titles appear in the record of all the words, and remove "and", "on", "for" adverbs and the number of occurrences less than 10 times, get frequently appear of academic words and their occurrences, 5061 words in total, saved to the KeywordsAll table.

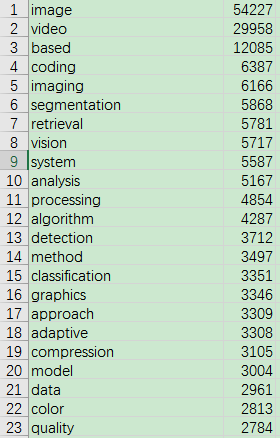


Figure 18 KeywordsAll table

**4.2 Semantic Feature Vector Calculation**

After the keywords obtained, we can calculate and analyze the semantic feature vector of the keywords and the author. The method is to select the first 100 keywords in the KeywordsAll table as the basis vector, and then count the occurrences of each keyword in the KeywordsAll table that appears with the first 100 keywords at the same time in a title. Let this 100 numbers be the semantic vector of this keyword. The author's semantic vector method is also similar, count the occurrences of these 100 keywords in each author's article title, respectively, let these numbers be the semantic vector.

The steps are:

First count the keywords and the title of the paper that the keywords appear: KeywordsAll\_titles Table:

Code:

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

KeywordsAll\_titles

FROM KeywordsAll, titles

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

The result:

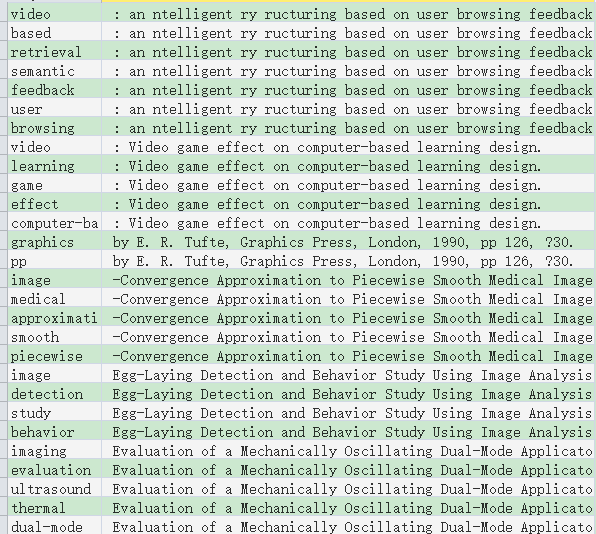


Figure 19 KeywordsAll\_titles table

And then count the occurrences of each keyword and 100 semantic basis vectors that appear in a same article: CountAll\_Vector table:

Code:

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;

The results:

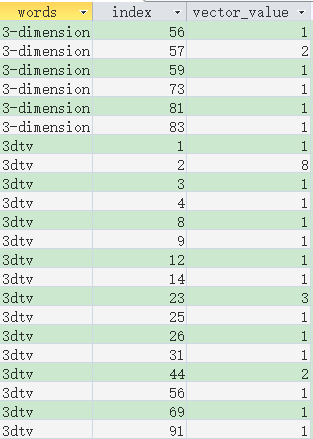


Figure 20 KeywordsAll\_Vector table

Note: words are the keywords, index represents the 100-semantic vector index, vector\_value represents the occurrences, in other words, the vector value; and some of them have no vector value, hence it is 0.

The next step is counting the author's semantic vector: Author\_Vector table:

Code:

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;

The results

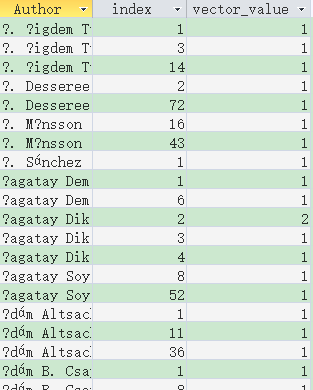


Figure 21 Author\_Vector table

Then MATLAB is used to read and organize the data of the KeywordsAll\_Vector table and the Author\_Vector table, and then normalize the data to get 130516 \* 100 of AuthorID\_Vector matrix and 5061 \* 100 of WordsID\_Vector matrix. Where each row represents each author or keyword, and 100 columns represent the value of normalized vector.



Figure 22 AuthorID\_Vector and WordsID\_Vector matrix

**4.3 Cluster analysis**

Using the k-means algorithm, the keywords are classified into 20 categories by semantic vector. The basic idea of ​​k-means algorithm is to cluster k points as a center in space, and to classify closest objects. The value of each cluster center is updated by iterative method until the best clustering result is obtained.



Figure 23 the corresponding number of each class of keywords and authors after the classification

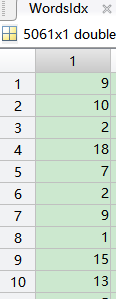
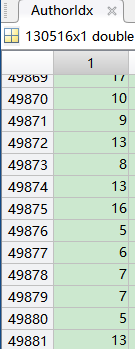
 

Figure 24 Keyword’s & author’s index and category index

**4.4 Application of Automatic Semantic Annotation**

According to the classification index information obtained, it is possible to construct the application of authors’ semantic annotation. Simply by querying the same class (i.e., the same field) keywords and related authors. The main method is based on the classification of the author, connecting the keyword table and the author table, etc., to find keywords and authors with the same classification.

Code:

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

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

[WordsID], [WordsID], [WordsAll], [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

Design a GUI interface, type any author name, click find, get the related fields’ keywords and the author's directory.

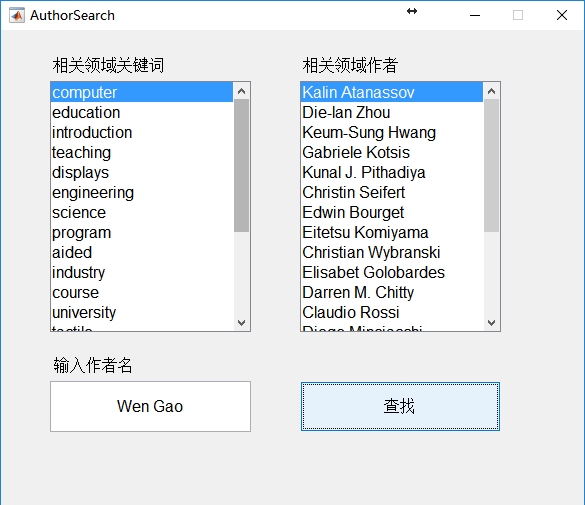


Figure 25 Keywords of author's related fields and related fields’ authors