MRC: Multi Relational Clustering approach

Majid Rastegar-Mojarad¹, Behrouz Minaei-Bidgoli²
¹Faculty of Engineering, Persian Gulf University, Bushehr 75169, Iran
²Assistant Professor of Computer Engineering, Iran University of Science and Technology
Narmak, Tehran, Iran
²CRCIS (NOOR): Data Mining Research group, Computer Research Center of Islamic Sciences, P.O.
Box 37185-3857, Qom, Iran

Abstract—Clustering is a process of partitioning data objects into groups based on the similarity measures. Most of the existing methods perform clustering within a single table, but most of the real-world databases, however, store information in multiple tables. We propose a new method which is called Multi Relational Clustering (MRC) for clustering a relational database. The MRC approach uses existing clustering algorithms for clustering every table of database. Tables in a database are related to each other based on foreign keys. The MRC approach divides the tables into two categories: dependent and independent tables. A dependent table is a table that includes entities attributes, as well as fields related to the other entities which belong to the other tables. In fact a dependent table includes one or more foreign keys. The MRC approach firstly, clusters independent tables then utilizes these results for clustering dependent tables. The MRC clusters each table by existing clustering algorithm with respect to its fields. An important feature of the MRC approach is ability of clustering several tables in parallel. The proposed approach is very simple and is developed under SQL very efficiently. We offer a version of implementation of k-Means in SQL and use it for clustering a database by MRC approach. Our experiments show that the MRC is efficient for clustering a huge database in a relational environment.

Keyword: Multi Relational Data Mining, clustering, SQL, Relational Database

1. INTRODUCTION

Data mining algorithms look for patterns in data. While most existing data mining approaches look for patterns in a single data table, multi-relational data mining (MRDM) approaches look for patterns that involve multiple tables (relations) from a relational database. Recently, the most common types of patterns and approaches in data mining have been extended to the multi-relational cases [1].

Data clustering refers to the problem of partitioning a data set into homogeneous groups (called clusters) in such a way that patterns within a cluster are more similar to each other than patterns belong to different clusters. Most clustering algorithms proposed in the literature [2]-[7] exploit a similarity (or dissimilarity) measure.

Clustering in multi-relational environments has been studied in [8]-[12]. In [8]-[10] the similarity between two objects is defined based on joinable tuples. These approaches, although provide similarity definitions in multi-relational environments, it is usually very expensive to compute such similarities because an object is often joinable with hundreds or thousands of tuples. Also the cross-relational clustering which is called CROSSCLUS [11], [12] needs very complex computation to determine distance between two records.

Authors of [13], [14] introduce a SQL implementation of the popular K-means clustering algorithm in order to integrate it with a relational DBMS. They compare K-means implementations in SQL and C++ with respect to speed and scalability. They also study the time of export data sets outside of the DBMS. Experiments show that SQL overhead is significant for small data sets, but relatively low for large data sets, whereas export times become a bottleneck for C++.

We propose a new approach clustering in multi relational environment. Our approach is a Multi Relational Clustering which is called MRC, efficiently works for large databases with many tables. The MRC divides tables in two categories. One category is independent tables and another category is dependent. Independent tables include features of only one entity. In fact they do not include any foreign keys. These tables do not need results of clustering of other tables. But dependent tables need results of clustering of other tables because these tables have at least a foreign key and we must replace foreign key with appropriate field for clustering. We first cluster independent tables and then transmit the results to tables that need them, and after that cluster dependent tables. After clustering all tables, a new table is generated which is called the Final table and includes the results of clustering tables. This table is clustered at the final step. Clustering Final table is similar to dependent tables. Figure 1 shows this approach. The MRC approach can cluster independent tables in parallel.
Also we can cluster some dependent tables in parallel which are not dependent to each other. This feature of proposed approach leads to a high improvement in clustering efficiency especially to cluster large databases which include many tables.

We use of K-means algorithm for clustering each table of our database. We present a new version of implementation K-means into SQL. Implementation into SQL has many benefits when have large database. These benefits described in follow.

The MRC has a reasonable accuracy. The MRC can use each existing clustering algorithm for clustering each table. We can select an algorithm for a table with respect to the type of its fields. So, this feature improves database clustering accuracy rather than when we use an algorithm for clustering all of them.

The paper is organized as follow: Section 2 introduces definitions and an overview of multi relational data mining. Also we explain an implementation of data mining algorithms into SQL in this section. Section 3 proposes MRC approach to perform clustering in a database with multi tables. We present implementation of k-means into SQL in section 4. Section 5 Contains experiments to evaluate performance with real database. Section 6 discusses related work. Section 7 provides general conclusions.

2. Multi Relational Data Mining

Multi Relational Data Mining is inspired from the relational model [15]-[17]. This model presents a number of techniques to store, manipulate and retrieve complex and structured data in a database consisting of a collection of tables. It has been the dominant paradigm for industrial database applications during the last decades, and it is at the core of all major commercial database systems, commonly known as relational database management systems (RDBMS). A relational database consists of a collection of named tables, often referred to as relations that individually behave as the single table that is the subject of Propositional Data Mining. Data structures which are more complex than a single record are implemented by relating pairs of tables through so-called foreign key relations. Such a relation specifies how certain columns in one table can be used to look up information in corresponding columns in another table, thus relating sets of records in the two tables [18].

2.1. Clustering in Multi Relationa

Clustering in multi-relational environments has been studied in [8]-[10]. For computing similarity between two tuples, they consider not only the attributes of the two tuples, but also those of the tuples related to them (joinable via a few joins) [8], [9]. However, these approaches suffer from poor efficiency and scalability, because in a large database an object is often joinable with thousands of tuples via a few joins, which makes it very expensive to compute similarity between each pair of objects. Moreover, they use all features indiscriminately and may not generate good clustering results that fit the user's goal [11].

In [9] based on recent advances in the area of multi-relational distance metrics and present RDBC1, a bottom-up agglomerative clustering algorithm that relies on distance information only.

CrossClus [11], [12] is a new algorithm which performs cross-relational clustering with user's guidance. CrossClus is carried out as follows: A user specifies a clustering task and selects one or a small set of features pertinent to the task. This method takes care of both quality in the feature extraction and efficiency in clustering. The CrossClus depends on the user and calculates many complex measures.

But we propose a new approach for clustering a database which use of existing algorithm that perform clustering in a table. We can cluster each table with an appropriate clustering algorithm. Our approach improves clustering's speed and clusters a database with a reasonable accuracy. Our approach has appropriate efficiency for databases that have many tables.

Fig. 1. A schema of the MRC approach

1 Relational Distance-Based Clustering
2.2. Motivation to implement algorithm in SQL

There are several reasons for which implementing algorithm in SQL turned out to be an interesting problem. One of the main points is that this task looked trivial at first sight, but it turned out to be challenging when we faced the problem of handling large data sets with high dimensionality. In [19] the reasons that motivated us to implement our algorithm in SQL are mentioned.

3. The MRC approach

We propose a new approach clustering in multi-relational environment. Our approach is called MRC. The MRC divides tables in two categories. One category is independent tables and another category is dependent. Independent tables include features of only one entity. In fact they don’t include foreign key. These tables do not need results of clustering of other tables. But dependent tables need results of clustering of other tables because these tables have at least one foreign key and we must replace foreign key with appropriate field for clustering. We first cluster independent tables and then transmit the results to tables that need them, and after that cluster dependent tables. After clustering all tables, a new table is generated which is called the Final table and includes the results of clustering tables. This table is clustered at the final step. Clustering Final table is similar to dependent tables. Figure 1 represents this approach.

3.1. Independent table

Independent tables do not need clustering results of any other tables. In relational databases, these tables generate for entities where participate in relationship. These tables include only attributes of one entity and do not have any foreign key or fields belong to other entities. These tables can be clustered with existing methods which perform clustering within a single table. An independent table can be determined with the table’s schema. If there is no any field related to other entities, then this table is an independent table.

For clustering these tables, we first select an appropriate algorithm for each table considering the type of fields, then perform the clustering, and after that applying the generated results for clustering dependent tables with making the Final table. The MRC produces a label of generated cluster and use of this label for clustering dependent tables. This label must represent distance between points. We use center of clusters as the clusters labels. We replace any foreign key in dependent tables with proper label. Foreign key in the dependent table is the primary key in the independent table. We replace foreign keys with cluster’s label. These Cluster’s label generate after clustering tables which include these foreign key as primary key.

After performing clustering an independent table, the MRC creates a table that represents distance between each other centers of clusters. We called this table distance table. We use of distance table for clustering dependent table which includes primary key of that independent table. Some of independent tables don’t use to cluster other tables. These tables belong to entities which have a relationship with target entity. These tables have one field in Final table. Distance between two centers of two clusters can be computed with distance metric that was used by the clustering algorithm.

3.2. Dependent table

We call those tables which need the clustering results of the other tables, dependent tables. In relational database, these tables generate for multi-value attribute, and for fields which represent relationship between entities. These tables include foreign key. These foreign keys should be replaced with appropriate fields which are proper for clustering dependent tables. The MRC replaces foreign keys with other corresponding fields. These corresponding fields are produced after clustering those tables in which the foreign keys have been used as primary keys. Suppose table A has two fields A_F1 and A_F2. First field is foreign key that in table B is primary key. This primary key in B is B_F1. The MRC uses clustering result of table B in clustering table A. To determine distance between two records of table A, we should determine the distance between their fields A_F1. The MRC determines this distance using result of clustering table B.

Existence a value for field A_F1 means that this record has at least related record in table B. Instead of calculating distance between two values of field A_F1, the MRC calculates distance between their related records in table B. The MRC uses results of clustering table B to reduce total time of clustering. We do not calculate again distance between two records of table B. The centers of clusters are used to estimate the distance. Here we utilize distance table that we generated after clustering table B. In relationship with m-n multiplicity between table A and B, one record in table A has relation with several records in table B. In this case, the MRC considers which cluster of B includes most of related records. The MRC uses center of this cluster to estimate distance. Using this method, the accuracy is reduced but clustering’s efficiency is increased. The MRC is a proper method for any databases that include many tables with a high dependency between them.

3.3. Final table

The MRC clusters independent tables firstly, whereas clustering independent tables do not need result of other
tables, so we can cluster all of the independent tables concurrently. Secondly, the MRC clusters dependent tables. It is possible some of them depend on other tables. So we may cluster them in several steps.

Finally, we generate a table which is called the Final table. The Final table fields are results of some independent and dependent tables. Clustering the Final table is the last step of clustering of the database. In fact the MRC cluster each table of the database, and then cluster the clustering results.

4. Implementation

We implement the MRC approach for clustering Iranian Families’ Income-Expense of 2001-2005 national statistics. To cluster each table, we upgrade the proposed algorithm in [13], [14]. There exists an implementation of K-means by SQL. We upgrade that to generate appropriate results after clustering each independent table, for uses to cluster dependent table. After clustering tables, our implementation creates and clusters the Final table that includes clustering results of other tables.

4.1. A new implementation of K-means in SQL

We present a new implementation of K-means into SQL which able to cluster a relational database. We utilize algorithm in [13], [14] and apply it for our purpose. We explain some parts of algorithm which we change or increase it.

Independent tables are clustered by the proposed algorithm in [13], [14]. After clustering a table, we generate a new table. This table is called distance table and includes three fields. This table represents distance between centers of generated clusters. Required query to generate this table shows here. This query calculates distance between two centers and inserts it together identifier of those in Distance table. We generate a distance table for each table. The Distance tables are a small table, because they have three fields and few records. We can cluster all of independent table concurrently.

Insert Into DistanceTable
Select C1.j, C2.j, SQRT (Sum ((C1.val – C2.val) * (C1.val – C2.val)))
From C as C1, C as C2
Where C1.l = C2.l
Group by C1.j, C2.j

The table C includes three fields. J shows cluster center identifier, L shows dimension and Val represents value of center j in dimension l. When we want clustering dependent tables, we utilize appropriate distance table instead of calculate distance between foreign keys. Each record of dependent table may relate to more one record of independent table. Each record of dependent table must relate at least to one cluster center of independent table clusters center. So we generate following query for this purpose. It selects a cluster center which includes most related records. If two clusters have same number of related record, this query selects that cluster with lower identifier. This query uses two tables. One of them is Relationship table which represents relation between records of independent table and dependent table. Another table includes record’s identifier of independent table and its cluster center. We join these tables and extract required result. The Relationship table has two fields, i and j. This table represents a relation between record i of dependent table with record j of independent table. Another table is R84output. That has two fields, k and clus. In this table, k is record identifier of independent table and clus represents its cluster.

Select t1.i, min(t1.clus) as label
From
(Select i, Clus, Count (*) as y3
From R84output, Relationship
Where Relationship.j = R84Output.k
Group by i, Clus
) as t1
,
(Select t3.i, max (t3.y2) as e
From
(Select i, Clus, count(*) as y2
From R84output, Relationship
Where Relationship.j = R84output.k
Group by i, clus) as t3
Group by i
group by i
)
Where t1.i = t2.i and t1.y3 = t2.e
Group by t1.i

Now we can calculate distance between records in dependent tables. To calculate Euclidean distance, at first we calculate distance between fields except foreign keys. After that we add distance between centers of related clusters to calculated distance. Below query performs this task. This query uses three tables, YD, R84output and R84distance. The YD has three fields, i, j and Distance. J and i represent record identifier and the Distance shows distance between two centers. The R84Distance table has three fields, i, j and Distance. The R84Distance table shows distance between two centers of independent table’s clusters. The YD table represents distance between two records of dependent table. The R84output table described before.

Update YD

Select t1.i, min(t1.clus) as label
From
(Select i, Clus, Count (*) as y3
From R84output, Relationship
Where Relationship.j = R84Output.k
Group by i, Clus
) as t1
,
(Select t3.i, max (t3.y2) as e
From
(Select i, Clus, count(*) as y2
From R84output, Relationship
Where Relationship.j = R84output.k
Group by i, clus) as t3
Group by i
group by i
)
Where t1.i = t2.i and t1.y3 = t2.e
Group by t1.i
Set distance = sqrt((Distance * Distance) + (dis.Distance * dis.Distance))

From YD,
(Select R1.k as k1, R2.k as k2,
R84.distance.Distance
From R84.output as R1, R84.output as R2,
R84.distance
Where R1.clus = R84.distance.i and R2.clus =
R84.distance.j
) as dis
Where dis.k1 = YD.i and dis.k2 = YD.j

In our implementation, we must select one of records as cluster center. So we determine cluster center by proposed algorithm in [13], [14], then we find the nearest record to that as cluster’s center in our implementation.

After clustering independent and dependent tables, we generate the Final table. The Final table includes result of tables which don’t use in clustering other tables. Clustering this table performs like dependent table.

In this section, we present the important parts of our implementation and we don’t mention details and other parts which exist in [13], [14].

4.2. Code Generator and used database

We generate a Code Generator which produces the required SQL code to cluster a database. The SQL code generator is programmed in the C# language, which connected to the SQL Server 2000. It allows user to select tables, fields, dependency between tables and target entity. We can cluster a database without the user, but these guidance of user help us to generate more meaningful results. The SQL code for clustering is created by the Code Generator automatically.

Our approach is tested by a real database. This database is Iranian Families’ Income-Expense of 2001-2005 national statistics. It has 42 tables that include families’ income and expense for foodstuffs, wearing stuff, training, traveling and curing.

5. Results

We implement the MRC approach in SQL. All tests were done on a 2GHz Pentium-4 PC with 1GB memory, running Windows XP. We test the algorithm with different size of database and different number of tables. We report result in follow.

In the first experiment, we consider clustering’s time with respect to size of database. So we select 10 relations of database and in each run we change size of database. The time for each clustering shows in table 1.

<table>
<thead>
<tr>
<th>Database size (MB)</th>
<th>Time (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>50</td>
<td>45</td>
</tr>
<tr>
<td>100</td>
<td>102</td>
</tr>
<tr>
<td>200</td>
<td>194</td>
</tr>
<tr>
<td>500</td>
<td>321</td>
</tr>
<tr>
<td>1000</td>
<td>504</td>
</tr>
</tbody>
</table>

We test clustering’s time with respect to number of relation in database. We chose 100 MB for size of database and change number of relation. Table 2 represents the obtained result.

<table>
<thead>
<tr>
<th>Number of tables</th>
<th>Time (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>10</td>
<td>102</td>
</tr>
<tr>
<td>20</td>
<td>142</td>
</tr>
<tr>
<td>30</td>
<td>185</td>
</tr>
<tr>
<td>40</td>
<td>218</td>
</tr>
<tr>
<td>45</td>
<td>251</td>
</tr>
</tbody>
</table>

We change our implementation with optimizing in K-means algorithm [14]. These changes have efficient effect on our results, but we don’t mention those results.

6. Related work

The Clustering has been extensively studied for decades in different disciplines including statistics, pattern recognition, database, and data mining, using probability-based approaches [20], distance based approaches [3], [21], subspace approaches [22], and many other types of approaches.

Multi Relational Data Mining studied in many articles. But clustering in multi relational environment studied in a few articles. In [8]-[13] proposed some algorithms. But none of them proposed an approach. The RDBC [9] is a bottom-up agglomerative clustering algorithm that relies on distance information only. CrossClus [11], [12] is a clustering algorithm depends on user, and it is very complicated, so implementation it in SQL is very difficult.

Research on implementing data mining algorithms in SQL or extending SQL for data mining purposes includes the following: Association rules mining is explored in [23]. SQL extensions to define complex aggregate functions having tables as parameters are proposed in [24]; these extensions are used to tackle the problem of mining...
association rules. The MSQL language [25] provides extensions to query a set of discovered association rules.

Several clustering algorithms implement in SQL. Programming the EM clustering algorithm in SQL is explored in [19]. K-means clustering in SQL appeared in [13], [14], where Standard K-means and some optimizations were introduced.

Implementation data mining algorithm in multi relational environment into SQL is a new topic for research.

7. Conclusion

In this paper, we propose an approach to cluster a relational database. This approach is called MRC. The MRC is very simple and efficient for clustering a huge relational database. Most of existing clustering algorithms work only on single table. We use them for clustering several tables that related to each other. In the MRC can employs different algorithms to cluster tables. Because these algorithms select with respect to type of table’s fields so this feature improves accuracy.

The MRC can cluster several tables in parallel. We implement the MRC into SQL. Implementation in SQL is very important for clustering large databases. Whereas the MRC uses common clustering algorithm to cluster a relational database, so if we can implement these algorithms in SQL, we can cluster a relational database by SQL code. Guidance of users can help the MRC to cluster with high accuracy and generate result with respect to users’ goal.

8. Acknowledgments

We are thankful for any support of Persian Gulf University, Bushehr, Iran, as well as Data Mining Research group at Computer Research Center of Islamic Sciences (CRCIS), NOOR co. P.O. Box 37185-3857, Qom, Iran

References

[12] X. Yin, J. Han, and P. S. Yu, “Cross-Relational Clustering with User’s Guidance," In 11th Int’l Conf. on Knowledge Discovery and Data Mining (KDD’05), Chicago, IL, pp. 344-353, 2005.