A Clustering Approach to Generalized Pattern Identification Based on Multi-instanced Objects with DARA

Rayner Alfred[1,2], Dimitar Kazakov[1]

[1] University of York, Computer Science Department, Heslington,
YO105DD York, United Kingdom
{ralfred, kazakov}@cs.york.ac.uk
http://www-users.cs.york.ac.uk/~ralfred
[2] On Study Leave from Universiti Malaysia Sabah,
School of Engineering and Information Technology,
88999, Kota Kinabalu, Sabah, Malaysia
ralfred@ums.edu.my

Abstract. Clustering is an essential data mining task with various types of applications. Traditional clustering algorithms are based on a vector space model representation. A relational database system often contains multi-relational information spread across multiple relations (tables). In order to cluster such data, one would require to restrict the analysis to a single representation, or to construct a feature space comprising all possible representations from the data stored in multiple tables. In this paper, we present a data summarization approach, borrowed from the Information Retrieval theory, to clustering in multi-relational environment. We find that the data summarization technique can be used here to capture the typical high volume of multiple instances and numerous forms of patterns. Our experiments demonstrate a technique to cluster data in a multi-relational environment and show the evaluation results on the mutagenesis dataset. In addition, the effect of varying the number of features considered in clustering on the classification performance is also evaluated.

Keywords: Relational Data Mining, Distance-based, Clustering, Multiple Instance, Relational Database.

1. Introduction

Clustering is a process of grouping data that shares similar characteristics into groups. Despite the increase in volume of datasets stored in relational databases, only few studies handle clustering across multiple relations [16, 17]. In a dataset stored in a relational database with *one-to-many* associations between records, each table record (or *object*) can form numerous patterns of association with records from other tables. For example, Fig. 1 illustrates the scenario in which a single object has multiple instances. In this scenario, relation T1 has a *one-to-many* relationship with relation T2, through the association of fields Id1 and Id2. Similarly, relation T2 has a *one-to-many* relationship with relation T3, through the association of fields Name2 and Id3.

Y. Ioannidis, B. Novikov and B. Rachev (Eds.): Local Proceedings of ADBIS 2007, pp. 38-49

For instance, object Id1=1 with class label A, will have a set of instances, *instance*(id1 = 1) = {(X,111), (X,112), (X,113), (Z,117)}.

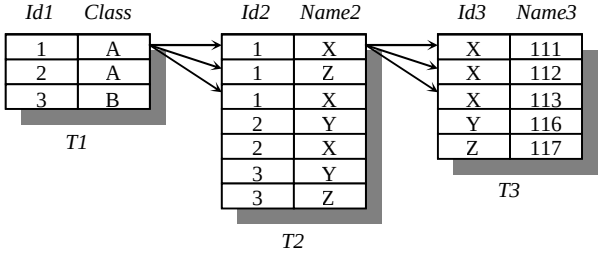

Fig. 1 A dataset stored in a relational database with two levels of *one-to-many* relationship.

Clustering in a multi-relational environment has been studied in Relational Distance-Based Clustering (RDBC) [16,17]. Clustering [12,13] is an unsupervised learning technique, that is, it can operate on unannotated data. However, it can be used as the first step of a supervised learning tool. For instance, a dataset split into classes can be clustered (without making use of the class labels) and then associations between clusters and classes learned using one of many supervised learning tools. This is the case in RDBC, where the role of this tool is performed by a decision tree learner. The approach proposed in this paper, follows the same strategy, combining a novel clustering technique (described in Section 3) with C4.5.

In RDBC, the similarity between two objects is defined on the basis of the tuples that can be joined to each of them. In this way, each of the two objects is expanded into a set of records, and the two sets are compared as follows: for each record in one set, the closest match in the other set is found, and their distance added. The distance between two such records is measured in the usual ways, comparing each pair of attributes in turn, depending on the types of attributes involved, e.g., as differences of numerical values, or a Hamming distance in the case of categorical values. However, the RDBC process of computing the distance between two objects is very expensive, since the process compares repeatedly components of first-order instances where each comparison is eventually reduced to a propositional comparison of elementary features. In addition to that, the RDBC approach only considers the minimum distance measured between instances to differentiate two objects and may not generate good clustering results, which leads to less meaningful clustering results. RDBC approach is also not able to generate interpretable rules. In our approach to clustering in a multi-relational environment, we consider all instances of an object when the distance between two objects is computed. By clustering objects with multiple instances, objects with the same characteristics are grouped together and objects with different characteristics are separated into different groups. Traditional clustering algorithms are based on one representation space, usually a vector space. However, in a relational database system, multiple instances in a non-target table exist for each object in the target table, due to the *one-to-many* association between multiple instances and the object. To cluster multiple-instance data using the established methods would require to restrict the analysis to a single representation or to construct a feature space comprising all representations.

In this paper, we present a data summarization approach, borrowed from the information retrieval theory, to cluster such multi-instance data. We propose a technique that considers all available instances of an object for clustering and we show the evaluation results on the mutagenesis dataset. In addition, the effect of the number of relevant features on the classification performance is also evaluated. The rest of the paper is organized as follows. In Section 2, we present related work on data mining in a multi-relational environment. Section 3 formalizes the problem and introduces our new pre-processing method for the purposes of clustering, called Dynamic Aggregation of Relational Attributes (DARA) [18, 19]. Section 4 provides the experimental evaluation and the last section summarizes the paper and presents some ideas for future research.

2. Learning in a Multi-Relational Environment

The most popular approach to supervised learning in a multi-relational environment is *relational learning*. Relational learning is not a new research area, and has a long history. Muggleton et al. [1] introduce the concept of Inductive Logic Programming (ILP) and its theory, methods and implementations in learning multi-relational domains. ILP methods learn a set of existentially quantified first-order Horn clauses that can be applied as a classifier [9, 11]. In a relational learner based on logic-based propositionalization [2], instead of searching the first-order hypothesis space directly, one uses a transformation module to compute a large number of propositional features and then uses a propositional learner.

Variants of relational learning include distance-based methods [6, 7]. The central idea of distance-based methods is that it is possible to compute the mutual distance [7] for each pair of objects. Relational Instance-Based Learning (RIBL) algorithms extend the idea of instance based learning to relational learning [7]. Instance-based learning (IBL) algorithms [5] are very popular and a well studied choice [3] for propositional learning problems. Probabilistic Relational Models (PRMs) [14] provide another approach to relational data mining that is grounded in a sound statistical framework. In PRMs, a model is introduced that specifies, for each attributes of an object, its (probabilistic) dependence on other attributes of that object and on attributes of related objects. Propescul et al. [15] proposed a combined approach called Structural Logistic Regression (SLR) that combines relational and statistical learning.

Data stored in a multi-relational environment can be considered as multiple instances of an object stored in the target table. As a result, learning multiple instances can be applied in learning data in a multi-relational environment. In multi-instance (MI) learning, instances are organized into bags that are labeled for training, instead of individual instances. Multiple instance learners assume that all instances, in a bag labeled negative, are negative and at least one instance in a bag labeled positive is positive. Several approaches have been designed to solve the multiple instance learning. Dietterich et al. [4] described an algorithm to learn axis-parallel rectangles (APRs) from MI data. Maron et al. introduced a framework called Diverse Density to learn Gaussian concepts [25]. Another approach using lazy learning has been

investigated in this context as well [23]. Unlike the former approaches, a framework for learning rules from multiple data was introduced by Chevaleyre et al. [24]. Most of the approaches [23, 4, 25] are not able to generate interpretable rule sets or decision trees.

In Relational Distance-Based Clustering (RDBC) [16,17], the similarity between two objects are defined based on tuples joinable with them. The distance measure uses the idea of computing distances by recursively comparing the components of first-orders instances, in which it is highly expensive if we have many tables. In addition to that, RDBC approach only considers the minimum distance measured between instances to differentiate two objects and may not generate good clustering results, which leads to less meaningful clustering results. RDBC approach is also not able to generate interpretable rules. In our approach, we transform the data representation in a multi-relational environment into a vector space model (explained in Section 3) suitable or applicable to clustering operation. By clustering them, we could group bags with multiple instances that have similar characteristics that can be extracted, as an interpretable rule to describe the cluster's behaviors.

3. Multi-Relational Setting and DARA

3.1. The Multi-Relational Setting

Let *DB* be a database consisting of *n* objects. Let $R := \{R_1,\ldots,R_m\}$ be the set of different representations existing for objects in *DB* and each object may have zero or more than one representation of each R_i, such that $|R_i| \geq 0$, where i = 1,…,*m*. Each object $O_i \in DB$, where $i = 1,\ldots,n$ can be described by maximally *m* different representations with each representation has its frequency,

$$O_i := \{R_1(O_i):|R_1(O_i)|:|Ob(R_1)|,\ldots,R_m(|O_i):|R_m(O_i)|:|Ob(R_m)|\},$$

with $R_j(O_i)$ represents the *j*-th representation in the *i*-th object and $|R_j(O_i)|$ represents the frequency of the *j*-th representation in the *i*-th object, and finally $|Ob(R_j)|$ represents the frequency of object with *j*-th representation. If all different representations exist for O_i, then the total different representations for O_i is $|O_i| = m$ else $|O_i| < m$.

In relational instance-based learning, the distance measures are defined based on the attribute's type [6] and the distance between two objects is based on the minimum distance between pair of instances from the two objects. In our approach, we apply the vector-space model [8] to represent each object. In this model, each object O_i is considered as a vector in the representation-space. In particular, we employed the *rf-iof* term weighting model borrowed from [8], where in which each object O_i, $i = 1,\ldots,n$ can be represented as

$$(rf_1 \cdot \log(n/of_1),\ rf_2 \cdot \log(n/of_2),\ \ldots,\ rf_m \cdot \log(n/of_m)),$$

where rf_j is the frequency of the j-th representation in the object, of_j is the number of objects that contain the j-th representation and n is the number of objects. To account for objects of different lengths, the length of each object vector is normalized so that it is of unit length ($||o_{rfiof}||= 1$), that is each object is a vector on the unit hypersphere. In this experiment, we will assume that the vector representation for each object has been weighted using *rf-iof* and it has been normalized so that it is of unit length. In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two objects O_i and O_j, $sim(O_i,O_j)$, which is defined as $cos(O_i,O_j) = O_i \cdot O_j/(||O_i|| \cdot ||O_j||)$. The cosine formula can be simplified to $cos(O_i,O_j) = O_i \cdot O_j$, when the record vectors are of unit length. This measure becomes one if the records are identical, and zero if there is nothing in common between them. The idea of our approach is to transform the data representation for all objects in a multi-relational environment into a vector space model and find the similarity distance measures for all objects to cluster them. These objects then are grouped based on the similarity of their characteristics, taking into account all possible representations and the frequency of each representation for all objects. For instance, from Fig. 1 we have O_1 = {(X,111):2:2, (X,112):2:2, (X,113):2:2, (Z,117):1:2}, O_2 = {(X,111):1:2, (X,112):1:2, (X,113):1:2, (Y,116):1:2} and O_3 = {(Y,116):1:2, (Z,117):1:2}, and the feature vectors for O_1, O_2 and O_3 based on these instances, {(X,111), (X,112), (X,113), (Z,117), (Y,116)}, are shown below,

$$O_1 = (2 \cdot \log(3/2), 2 \cdot \log(3/2), 2 \cdot \log(3/2), 1 \cdot \log(3/2), 0 \cdot \log(3/2)) = (0.35, 0.35, 0.35, 0.18, 0.00)$$

$$O_2 = (1 \cdot \log(3/2), 1 \cdot \log(3/2), 1 \cdot \log(3/2), 0 \cdot \log(3/2), 1 \cdot \log(3/2)) = (0.18, 0.18, 0.18, 0.00, 0.18)$$

$$O_3 = (0 \cdot \log(3/2), 0 \cdot \log(3/2), 0 \cdot \log(3/2), 1 \cdot \log(3/2), 1 \cdot \log(3/2)) = (0.00, 0.00, 0.00, 0.18, 0.18)$$

Then, the similarity measure of two objects can be computed by using the cosine distance, where the values for $cos(O_1,O_2)$, $cos(O_1,O_3)$ and $cos(O_2,O_3)$ are as follows;

$$cos(O_1,O_2) = \frac{(0.35*0.18)+(0.35*0.18)+(0.35*0.18)+(0.18*0.00)+(0.00*0.18)}{\sqrt{(0.35^2+0.35^2+0.35^2+0.18^2)*(0.18^2+0.18^2+0.18^2+0.18^2)}} = 0.83$$

$$cos(O_1,O_3) = \frac{(0.35*0.00)+(0.35*0.00)+(0.35*0.00)+(0.18*0.18)+(0.00*0.18)}{\sqrt{(0.35^2+0.35^2+0.35^2+0.18^2)*(0.18^2)}} = 0.28$$

$$cos(O_2,O_3) = \frac{(0.18*0.00)+(0.18*0.00)+(0.18*0.00)+(0.00*0.18)+(0.18*0.18)}{\sqrt{(0.18^2+0.18^2+0.18^2+0.18^2)*(0.18^2+0.18^2)}} = 0.50$$

For cosine distance, the bigger the value is the smaller the distance. So we can conclude that object O_1 is more similar to object O_2 than to object O_3.

3.2. Dynamic Aggregation of Relational Attributes (DARA)

In relational database, records are stored separately in different tables and they are associated through the matching of primary and foreign keys. With a high degree of

one-to-many association, a single record, *O*, stored in a main table is associated with a large volume of records stored in another table. In our algorithm called the Dynamic Aggregation of Relational Attributes (DARA), we convert the data representation from a relational model into a vector space model.

Let *O* denotes a set of *m* records stored in the target table and let *R* denotes a set of *n* records (T_1, T_2, T_3, … , T_n) stored in the non-target table. Let R_i is in the subset of *R*, $R_i \in R$, and is associated with a single record O_a stored in the target table, $O_a \in O$. Thus, the association of these records can be described as $O_a \rightarrow R_i$. Since a record can be characterized based on the bag of term/records that are associated with it, we use the vector space model to cluster these records, as described in the work of Salton *et al.* [8]. In vector space model, a record is represented as a vector or 'bag of terms', i.e., by the terms it contains and their frequency, regardless of their order. These terms are encoded and represent instances stored in the non-target table referred by a record stored in the target table. The non-target table may have a single attribute or multiple attributes and the process of encoding the terms is as follows;

Case I: *Non-target table with a single attribute*

- Step 1) Compute the cardinality of the attribute's domain in the non-target table. For continuous values, discretizes them and take the number of bins as the cardinality of the attribute's domain
- Step 2) To encode values, find the appropriate number of bits, *n*, that can represent different values for the attribute's domain, where $2^{n-1} < |\text{Attribute's Domain}| \leq 2^n$. For example, if a city's attribute has 5 different values (London, New York, Chicago, Paris, Kuala Lumpur), then we just need 3 ($5 < 2^3$) bits to represent each of those value (001, 010, 011, 100, 101).
- Step 3) Each encoded term will be added to the bag of terms that describes the characteristics of the record associated with them.

Case II: *Non-target table with multiple attributes*

- Step 1) Repeat step 1) and step 2) in Case I, for all attributes
- Step 2) For each instance of a record stored in the non-target table, concatenate *p*-number of columns' values, where *p* is less than or equal to the total number of attributes. For example, let $F = (F_1, F_2, F_3, \ldots, F_k)$ denotes *k* attributes in the non-target table. Let $dom(F_i)$ denotes the domain of the *i*-th attribute. So, we can have instances of record in the non-target table with these values ($F_{1,a}$, $F_{2,b}$, $F_{3,c}$, $F_{4,d}$,… , $F_{k-1,b}$, $F_{k,n}$), where $a \in dom(F_1)$, $b \in dom(F_2)$, $c \in dom(F_3)$, $d \in dom(F_4)$,…, $b \in dom(F_{k-1})$, $n \in dom(F_k)$. If $p = 1$, we have $1{:}F_{1a}$, $2{:}F_{2,b}$, $3{:}F_{3,c}$, $4{:}F_{4,d}$,…, $k\text{-}1{:}F_{k-1,b}$, $k{:}F_{k,n}$ as the produced terms. If $p = 2$, then we have $1{:}F_{1,a}F_{2,b}$, $2{:}F_{3,c}F_{4,d}$,…, $(k/2){:}F_{k-1,b}F_{k,n}$ (provided we have even number of fields). Finally, if we have $p = k$, then we have $1{:}F_{1,a}F_{2,b}F_{3,c}F_{4,d}\ldots F_{k-1,b}F_{k,n}$ as a single term produced. We concatenate a number in front of the binary number to indicate the memberships of values to their attributes. For instance, if we have $p = 1$, we have $1{:}F_{1a}$, $2{:}F_{2,b}$, $3{:}F_{3,c}$, $4{:}F_{4,d}$,…, $k\text{-}1{:}F_{k-1,b}$, $k{:}F_{k,n}$, where $i{:}F_{j,k}$, refers to the *i*-th instance of *j*-th attribute and $a \in dom(F_j)$.
- Step 3) For each encoded term, add this term to the bag of terms that describes the characteristics of a record associated with it.

For example, from our previous example, we have the cardinality of *Name2* in *T2* as 3, the cardinality of *Name3* in *T3* as 5. Thus we require 2 digits of binary number to represent the domain of *Name2* in *T2* (01 for X, 10 for Z and 11 for Y), and 3 digits of binary number to represent the domain of for *Name3* in *T3* (001 for 111, 010 for 112, 011 for 113, 100 for 116 and 101 for 117). As a result, when we have p = 1, then we have

O_1 = {1:01, 2:001, 1:01, 2:010, 1:01, 2:011, 1:10, 2:101},
O_2 = {1:01, 2:001, 1:01, 2:010, 1:01, 2:011, 1:11, 2:100} and
O_3 = {1:11, 2:101, 1:10, 2:101}.

And if we have p = 2, then we have

O_1 = {1:01001, 1:01010, 1:01011, 1:10101},
O_2 = {1:01001, 1:01010, 1:01011, 1:11100} and
O_3 = {1:11101, 1:10101}.

The encoding process to transform relational datasets into data represented in a vector-space model has been implemented in DARA [18,19]. Given this data representation, we can use clustering techniques [12,13] to cluster them, as a means of aggregating them. DARA algorithm simply assigns each record in the target table with the cluster number. Each cluster then can generate more information by looking at the most frequent patterns that describe each cluster.

4. Experimental Evaluations

In this experiment, we employ an algorithm, called DARA that converts the dataset representation in relational model into a space vector model and use a distanced-based method to group objects with multiple representations occurrence. With DARA algorithm, all representations of two objects are taken into consideration in measuring the similarity measure between these two objects. The DARA algorithm can also be seen as an aggregation function for multiple instances of an object, and is coupled with the C4.5 classifier (J48 in WEKA) [20], as an induction algorithm that is run on the DARA's transformed data representation. We then evaluate the effectiveness of each data transformation with respect to C4.5. The C4.5 learning algorithm [21] is a state-of-the-art top-down method for inducing decision trees. All experiments with DARA and C4.5 were performed using a *leave-one-out* cross validation estimation with different values of *p*, where *p* denotes the number of attributes being concatenated. We chose well-known dataset, Mutagenesis [22].

The mutagenesis data [22] describes 188 molecules falling in two classes, m*utagenic* (active) and *non-mutagenic* (inactive); 125 of these molecules are mutagenic. The description consists of the atoms and bonds that make up the compound. Thus, a molecule is described by listing its atoms *atom(AtomID, Element, Type, Charge),* and the bonds *bond(Atom1, Atom2, BondType)* between atoms. In this experiment, we use three different sets of background knowledge: B1, B2 and B3.

- B1: The atoms in the molecule are given, as well as the bonds between them; the type of each bond is given as well as the element and type of each atom. The table

for B1 has the schema *Molecule(ID, ATOM1, ATOM2, TYPE_ATOM1, TYPE_ATOM2, BOND_TYPE)*, where each molecule is described over several rows, listing all pairs of atoms with a bond, and the type of each atom, and the type of bond between them.

- B2: Continuous values about the charge of atoms are added to all data in B1.
- B3: Two continuous values describing each molecule are added to all data in B2. These values are the log of compound's octanol/water partition coefficient (*logP*) and energy of the compound's lowest unoccupied molecular orbital ($^{C}LUMO$).

In B1, there are five attributes that describe an individual molecule, namely *first atom*, *second atom*, *first element's type*, *second element's type* and *bondtype*. There are typically several records for each molecule. We performed a *leave-one-out* cross validation estimation using the C4.5 classifier for p = 1, 2, 3, 4, 5 as we have a total of five attributes for dataset B1. Table 1 shows that the predictive accuracy of the decision tree learned is the highest when p is 2 or 5. When we have p = 2, the attributes used for clustering are the following 3 compounds: [*first atom*, *second atom*], [*first element's type*, *second element's type*], and [*bondtype*]. When p = 5, the only attribute used is: [*first atom*, *second atom*, *first element's type*, *second element's type*, *bondtype*].

Table 1. Performance of C4.5 on Mutagenesis datasets B1, B2 and B3

Datasets\p	1	2	3	4	5	6	7	8	9
B1	80.9	81.4	77.7	78.8	81.2	-	-	-	-
B2	79.5	80.0	81.2	80.3	82.8	81.8	79.5	-	-
B3	79.5	81.6	79.1	82.7	80.2	79.1	79.0	82.7	78.6

A test using the correlation-based feature selection (CFS in WEKA) function [20] provides a possible explanation of these results. We find that the two attributes, *first element's type* and *second element's type*, are highly correlated with the class membership, yet uncorrelated with each other. This means that an attribute combining these two would be relevant to the learning task and split the instance space in a suitable manner. The data contains this composite attribute when p = 2, 4 and 5, but not for the cases of p = 1 and 3.

In B2, two attributes are added into B1, which are the charges of both atoms. We performed a *leave-one-out* cross validation estimation using the C4.5 classifier for $p \in \{1, \ldots, 7\}$, as we now have a total of seven attributes for dataset B2. With additional two more attributes, we have a higher prediction accuracy of the decision tree when p = 5, compared to learning from B1 when p = 5, as shown in Table 1 and Fig. 2. When p = 5, we have two compound attributes, [*first atom*, *second atom*, *first element's type*, *second element's type*, *bondtype*], and [*first element's charge*, *second element's charge*]. Fig. 2 shows that the a drop in performance when p = 1, 2 and 7. In contrast, we have higher prediction accuracy when p = 5. We have shown above that in the case of B1, the attributes *first element's type* and *second element's type* are highly correlated with the class membership. For B2, we have used the same technique to find that the *first element's charge* and the *second element's charge* are also highly correlated with the class membership, yet uncorrelated with each other.

This explains the higher prediction accuracy for B2 and $p = 5$, as in this case 2 useful compound attributes are formed: [*first element's type, second element's type*] and [*first element's charge, second element's charge*].

In B3, two more attributes are added to the existing dataset B2, and we now have the following row of attributes: [*first atom, second atom, first element's type, second element's type, bondtype, first element's charge, second element's charge, logP,* ${}^{C}LUMO$]. Fig.2. indicates that the prediction accuracy of a *leave-one-out* cross validation of C4.5 is the highest when $p = 4$ and 8. When $p = 4$, we have the following compound attributes [*first atom, second atom, first element's type, second element's type*], [*bondtype, first element's charge, second element's charge, logP*] and, finally [${}^{C}LUMO$]. Each of the first two subsets of attributes contains a pair of attributes that are highly correlated with the class membership. Again, this can be used to explain the high prediction accuracy for a leave-one-out cross validation of C4.5 when $p = 4$ with dataset B3.

Table 2 shows the DARA+C4.5 performance in the case of the mutagenesis dataset, using leave-one-out cross-validation and the J48 implementation of C4.5 [20]. The results show that (1) *there is no other algorithm that outperformed ours on all datasets*, and (2) *for each of the other algorithms listed in the table, there is a dataset on which our algorithm performed better.* For instance, our approach outperforms RDBC when all available tables are used. Unlike RDBC, our approach computes the distance between two different objects based on the representation of its instances (concatenated attributes). As a result, for each cluster, we can find the representations (by taking the representation with highest weight) that best describe the clusters and these representations can be used as an interpretable rules for clustering or classifying unseen objects with multiple instances.

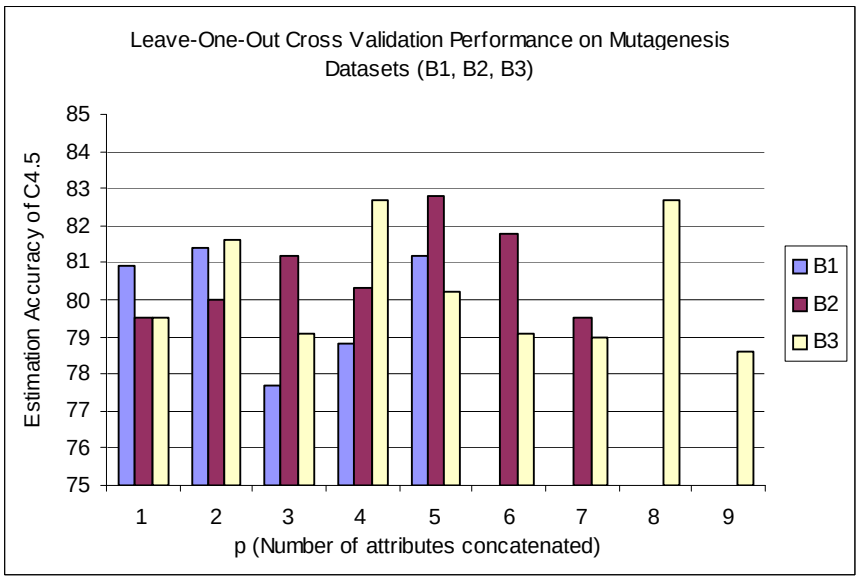

Fig. 2. Leave-One-Out CV Estimation Accuracy on Mutagenesis (B1, B2, B3)

Table 2. Comparison of performance accuracy on Mutagenesis Dataset

Algorithms	Mutagenesis		
	B1	B2	B3
PROGOL [22]	76%	81%	83%
FOIL [10]	83%	75%	83%
TILDE	75%	75%	85%
RDBC [16,17]	83%	84%	82%
DARA [18,19]	**81%**	**83%**	**83%**

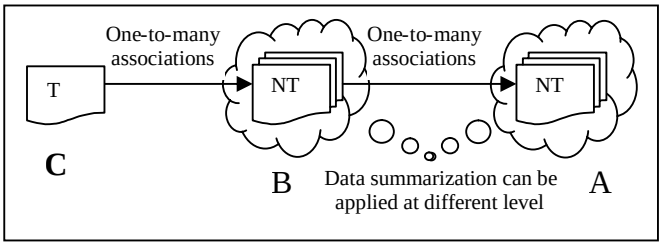

Fig. 3. A target table is associated with multi-level of *one-to-many* relationship of non-target tables (NT: Non target table, T: Target table).

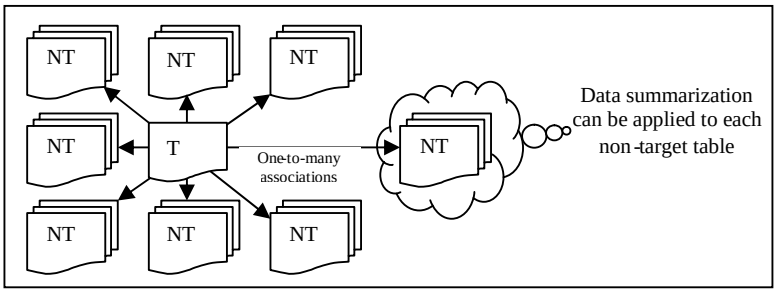

Fig. 4. A target table is associated with a multiple number of *one-to-many* relationships of non-target tables (NT: Non target table, T: Target table).

5. Conclusion

This paper presents an algorithm transforming datasets in a multi-relational setting into a vector space model that is suitable to clustering operations, as a means of aggregating or summarizing multiple instances. We carried out an experiment that clusters the objects in a multi-relational setting based on the patterns formed. The results show that varying the number of concatenated attributes *p* before clustering has an influence on the predictive accuracy of the decision tree learned by the C4.5 classifier. We have found that an increase in accuracy coincides with the cases of grouping together attributes that are highly correlated with the class membership. However, the prediction accuracy is degraded when the number of attributes

concatenated is increased further. The results indicate that limiting the number of attributes may be desirable. At the same time, it is beneficial to combine attributes that are highly correlated with the class membership together. In this study, keeping the number of concatenated attributes n relatively small (e.g. $n \leq 5$), results in the best performance in terms of prediction accuracy as measured by *leave-one-out* cross-validation of the C4.5 decision tree.

Finally, the results show that data summarization performed by DARA, can be beneficial in summarizing datasets in a complex multi-relational environment, in which datasets are stored in a multi-level of *one-to-many* relationships (i.e., tables, see Fig. 3), and also in the case of datasets stored in more than one one-to-many relationships with non-target tables (see Fig. 4). These are some of the areas that need further investigation. Since data summarization can be applied at a different level, this approach is scalable. For instance, in Fig. 3, one may perform the data summarization in B first, so that table A and B can be joined easily, before table B can be summarized in order to learn data stored in table C. Another area that needs further investigation is whether data summarization can be improved by applying a feature selection algorithm (CFS) based on a genetic algorithm to search for good subsets of attributes from multiple tables and create instances from these subsets of attributes in order to improve the prediction accuracy.

References

1. S.H. Muggleton and L. DeRaedt. Inductive Logic programming: Theory and Methods. *The Journal of Logic Programming*, 19 & 20:629-680, May 1994.
2. S. Kramer, N. Lavrač and P. Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors. *Relational Data mining.* Springer-Verlag, 2001. ISBN 3540422897.
3. D. Wettsschereck and T.G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. Machine Learning, 19(1):5-27, 1995.
4. T.G. Dietterich, R.H. Lathrop and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence, 89(1-2): 31-71
5. D.W.Aha, D.Kibler, and M.K. Albert. *Instance-based learning algorithms*. Machine Learning, 6(1):37-66, Jan, 1991.
6. T. Horvath, S. Wrobel, and U. Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1):53-80, 2001.
7. W. Emde and D. Wettschereck. *Relational instance-based learning*. In L.Saitta, editor, Proceedings of the 13th International Conference on Machine Learning, pages 122-130. Morgan Kaufmann, 1996.
8. G. Salton, A. Wong, and C.S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620.
9. S. Džeroski and N. Lavrač, editors. *Relational Data mining*. Springer-Verlag, 2001. ISBN 3540422897.
10. A. Srinivasan, S. Muggleton, R. King. Comparing the use of background knowledge by inductive logic programming systems. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, 1995.
11. A. Srinivasan, R.D. King, D.W. Bristol, An Assessment of ILP-Assisted Models for Toxicology and the PTE-3 Experiment, In *Proceedings of ILP '99*, 1999
12. T. Hofmann, and J.M. Buhnmann, Active data clustering. In *Advance in Neural Information Processing System*, 1998.

13. J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975
14. L. Getoor, N.Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic relational models. In S. Džeroski and N. Lavrač, editors. *Relational Data mining.* Springer-Verlag, 2001.
15. A. Propescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. Structural Logistic Regression: Combining relational and statistical learning. In *Proceedings of the workshop on Multi-Relational Data Mining* (MRDM-2002), pages 130-141. University of Alberta, Edmonton, Canada, July 2002.
16. M. Kirsten, and S. Wrobel, Relational Distance-Based Clustering. In *Proceedings of ILP 2000*, 2000.
17. M. Kirsten, and S. Wrobel, Extending K-Means Clustering to First-order Representations, ILP, 2000.
18. R. Alfred, and D. Kazakov, Data Summarization Approach to Relational Domain Learning Based on Frequent Pattern to Support the Development of Decision Making. In *the Proceedings of The Second International Conference of Advanced Data Mining and Applications, (ADMA 2006)*, pp 889-898, 2006.
19. R. Alfred, and D. Kazakov, Pattern-Based Transformation Approach to Relational Domain Learning Using DARA. In *the Proceedings of The 2006 International Conference on Data Mining (DMIN 2006)*, CSREA Press, eds. S.F.Crone, S. Lessmann, R. Stahlbock, ISBN. 1-60132-004-3, (2006), pp 296-302
20. I. Witten, and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman, 1999.
21. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, Los Altos, California.
22. A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, R.D. King, Theories for mutagenicity: A Study in first-order and feature-based induction. *Artificial Intelligence*, 85, 1996.
23. J. Wang, and J.D. Zucker, Solving the Multiple-Instance Problem: A Lazy Learning Approach. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, Stanford University, Standord, CA, USA. Morgan Kaufmann 2000: 1119-1126.
24. C. Chevaleyre, and J.D. Zucker, A Framework for Learning Rules from Multiple Instance Data. In the *Proceedings of 12th European Conference on Machine Learning,* Freiburg, Germany, 2001: 49-60.
25. O. Maron, and T. Lozano-Pérez, A Framework for Multiple-Instance Learning. *Advances in Neural Information Processing Systems*, 10, The MIT Press 1998.