Computer Science and Information Systems 18(3):1101–1138 https://doi.org/10.2298/CSIS200131014N

Conflict Resolution Using Relation Classification: High-Level Data Fusion in Data Integration

Zeinab Nakhaei[1], Ali Ahmadi[2], Arash Sharifi[1], and Kambiz Badie[3]

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
{zeinab.nakhaei, a.sharifi}@srbiau.ac.ir
[2] Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran
ahmadi@kntu.ac.ir
[3] Iran Telecommunication Research Center (ITRC), Tehran, Iran
k_badie@itrc.ac.ir

Abstract. The aim of conflict resolution in data integration systems is to identify the true values from among different and conflicting claims about a single entity provided by different data sources. Most data fusion methods for resolving conflicts between entities are based on two estimated parameters: the truthfulness of data and the trustworthiness of sources. The relations between entities are however an additional source of information that can be used in conflict resolution. In this article, we seek to bridge the gap between two important broad areas, relation estimation and truth discovery, and to demonstrate that there is a natural synergistic relationship between machine learning and data fusion. Specifically, we use relational machine learning methods to estimate the relations between entities, and then use these relations to estimate the true value using some fusion functions. An evaluation of the results shows that our proposed approach outperforms existing conflict resolution techniques, especially where there are few reliable sources.

Keywords: conflict resolution, data fusion, relational machine learning, relation estimation, relation classification.

1. Introduction

The main challenge in a data integration system is *conflict resolution*. Conflicts occur at different levels, ranging from schema to value [1]. In this article, we deal mainly with the second of these. Conflict at the value level means that there are multiple sources describing the same real-world entity, providing different values for the same attribute of the entity. To resolve such conflicts, fusion techniques are used. In broad terms, data fusion (DF) is the process of combining multiple sources of data to achieve higher quality data than can be obtained from individual sources [2]. In data integration, the DF process is a combination of values that describe a similar entity from the real world, leading to one value which is closer to the real world. In this article, DF means the process and methods for achieving one accurate single value from multiple values.

In a conflict resolution problem, there are usually many claims about entities provided by a number of different sources. The basis for almost all current DF methods is voting based on two main assumptions: (1) that the claim provided by a reliable source is correct; and (2) that the source that provides the true value is reliable. Existing methods therefore attempt to estimate two parameters: the *truthfulness* of data and the *trustworthiness* of sources [3-7]. However, these methods prove inadequate in some cases. We analyze the causes of this in two respects: the number of reliable sources, and the long-tail phenomenon.

- The number of reliable sources: In some applications, there are few reliable sources, and incorrect information may be copied by multiple unreliable sources. For example, a website publishes fake news tendentiously, and this news is then republished in several weblogs and social networks.
- The long-tail phenomenon: This phenomenon occurs where information on entities is provided by very few sources, as is common in applications [8, 9]. In such cases, it is possible that there are some reliable sources, but these sources may not provide information about certain entities. As a result, the information about such entities is not sufficient to produce a correct value.

In both the above cases, further items of information are needed beyond the attributes of an entity and claims about these. As described in the survey carried out by Li et al. (2016) [10], most truth discovery methods assume that entities are independent, whereas in reality entities may have relations between them and may affect each other. For example, two people who are classmates at university are likely to have the same level of education. Our proposed approach seeks to exploit the additional information deriving from such relationships between entities, and thereby to achieve a higher level of data abstraction.

As discussed in an article by Snidaro et al. (2013) [11], evolving data sources require an entity of interest to be represented by a collection of distinct and complementary pieces of information at multiple levels of abstraction. At lower levels of abstraction, entities are described by low level data (such as information about data sources) and attributes. At higher levels, on the other hand, entities are described by their situation and relationship with respect to other entities: in other words, we are dealing with relations and patterns between entities.

In this article, we use the relations between entities in addition to the attributes of the latter. Drawing inferences about or predicting the relations between entities is one of the challenges in machine learning, as can be seen in problems like *link prediction* and *knowledge graph completion* [12,13]. The main challenge is how to devise a model that can reliably learn relations between new entities. Such models are often trained by supervised methods. These however require a large training dataset comprising both entities and the relations between them.

We need to mention that by relational data model we mean a set of relations in the form of triples (*subject*; *relation type*; *object*), where *subject* and *object* are an entity and *relation type* is a relationship between a *subject* and an *object*. A *relation schema* is a set of relation types. The triples in a relational data model are relation instances.

In order to model relational data in a conflict resolution problem, we need to deal with two basic challenges. First, there are no predefined relation types, and the data sets contain only the entities and values for their attributes. Second, while each row in the input data sets is related to only one entity and its attributes, there may be multiple differing values for each attribute claimed by different sources. To address the first problem, we define a relation schema based on the attributes involved. To deal with the second problem, we create a metadata set containing information about pairs of entities instead of looking at only one entity at a time. This enables us to predict the existence of a relationship between pairs of entities and find relation instances.

The key aim of our efforts is to ensure that all the relations are clearly and reliably defined. To achieve this, our study draws on a combination of two important and widely used areas: *truth discovery* and *relation estimation.* In summary, this article makes the following contributions to knowledge in this area:

1. We consider the problem of conflict resolution at a higher level of abstraction of data, and define new heuristics for using additional items of information about entities, namely the *relations* between them.
2. We define a relation schema based on the attributes of entities, and use this when there is no predefined relation between the entities in question.
3. We introduce a process for assessing the relation between two entities, employing a metadata set obtained from a primary small clean data set and our relation schema.
4. We bridge the gap between the two important broad areas of relation estimation and truth discovery, and demonstrate that there is a natural synergistic relationship between data integration and machine learning.
5. Finally, and most importantly, we demonstrate that using extra information in this way can improve the performance of fusion techniques, particularly in unreliable environments.

The rest of the article is organized as follows. In section 2, we review the existing literature in the two main areas of truth discovery and relation assessment. In section 3, we explain why we use relations to try to solve conflict resolution problem by illustrating how existing methods work that motivates our approach. In section 4, we define the problems surrounding conflict resolution. Section 5 describes our proposed approach in more detail, including the framework, algorithms and required formulations. Finally, the results of our experiments are analyzed in section 6.

2. Related Works

This article bridges the gap between two important areas: data fusion and relational machine learning. Our approach tries to use relational models to estimate relationships between entities, and then to apply this model in order to improve the performance of the data fusion method. In other words, our study is located at the intersection of these two areas. We therefore review in this section articles and existing methods in both areas, beginning with data fusion.

2.1. Data Fusion for Conflict Resolution

The first study that precisely defined the goals of data fusion for the purposes of conflict resolution was provided by Bleiholder and Neumann (2009) [14]. Their survey introduced the problem of data fusion in the larger context of data integration, where data fusion is the final step in a data integration process, schemata have been matched, and duplicate records identified. Data fusion involves merging these duplicate records into a single record, while at the same time resolving data conflicts.

There are two main kinds of data fusion that can be performed at data abstraction levels: low level data fusion and high-level data fusion.

Low level data fusion. All of the methods in the category of low-level data fusion estimate two parameters (the truthfulness of data and the trustworthiness of sources). These methods can be divided into three categories based on the model used to estimate these parameters, namely: iterative models, graphical models and optimizing models.

Iterative model: Early methods of data fusion attempted to estimate the correctness of claims and the reliability of sources, and to determine each of these iteratively. The first such method was truth finder by Yin et al. (2008) [4]. This uses Bayesian analysis, under which the correctness of each claim is calculated as the product of the degrees of reliability of its sources. Truth finder has gained considerable popularity, with a number of methods emerging based on its algorithm. These are reviewed and compared in an article by Li et al. (2012) [15].

Graphical model: There is also a substantial body of work on data fusion that uses the graphical model [3] in order to model the relationship between data correctness and source accuracy. In the proposed method of Zhao et al. (2012) [3], claims are modeled as random variables which depend on the truth of the facts they refer to as well as on the quality of their sources. With the actual claim data, it is then possible to go back and infer the facts most likely to be true and the quality of the relevant sources. More recently, SLiMFast was proposed by Rekadsinas et al. (2017) [16] as a discriminative model that also enables other features of data sources (such as, update date, number of citations) to be taken into account for fusion purposes; where there is sufficient labeled data, SLiMFast uses empirical risk minimization (ERM).

Optimization model: Finally, some further methods model the problem using an optimization framework, where truths and source reliability are defined as two sets of unknown variables like Meng et al. (2015) [6] and Yin et al. (2011) [17].

High level data fusion. Some research goes beyond the above and seeks to estimate additional parameters, including the *correlation between sources* [18] and the *relation between objects* [6, 7 and 17]. The latter relations may be temporal or spatial. These relations are partially addressed by Meng et al. (2015) [6]. However, this work is based on the key assumption that a correlation graph already exists, whereas in our approach the relations between entities are inferred by learning methods. Another study by Yin (2011) [17] features a semi-supervised approach that seeks to find true values with the help of ground truth data. Claims are connected to each other and thus form a graph. Both this work and another similar piece of work by Liu et al. (2018) [7] rely on the similarity of claims and consider this as the relationship between them. Ye et al. (2019) [9] meanwhile propose an algorithm called PatternFinder, that jointly and iteratively learns four variables, i.e.: the latent groups of entities that match to a particular regularity; the group-level representatives that indicate the true value for the attributes of each entity in each latent group; the attribute weights; and the source weights. They also propose an optimized grouping strategy to enhance the efficiency of this approach.

It is important to note that these methods focus only on the apparent characteristics of entities, and use a similarity function to draw inferences about relations. To address this limitation, our previous work (2019) [19] proposes a method for estimating relationships between entities based on clustering them in an embedding space instead of a feature space. In this approach, before clustering, the data points are mapped into an embedding space and are enhanced by creating more informative features. The true values are then determined by defining a confidence score based on the distance between the data and the centers of clusters in the embedding space. Our previous work differs from the work proposed in this article in two main respects. First, in our previous work, we assume that the entities in the same cluster are related, whereas in this article, we create a metadata set and use machine learning methods to infer some rules about the relationships between entities. Second, in order to resolve conflicts between related entities, in our previous article we use the distance of entities from the centroids of clusters. In this article, on the other hand, we define some fusion functions and use these to calculate the confidence score for each entity.

In summary, the methods based on relations between entities can be divided into two groups: those that are aware of relations between entities beforehand, and those that use similarities as relationships between entities. In our current approach, in contrast, there are no prior assumptions about relations, nor are there defined types of relations. Instead, the relations between entities are derived by mining some rules deduced by machine learning methods.

2.2. Relational Machine Learning

In this article, we use relational machine learning to derive relations between entities. Relational machine learning covers a number of methods for the statistical analysis of relational, or graph-structured, data. Nickel et al. (2016) [20] provide a review of how such statistical models can be trained on large knowledge graphs, and then used to predict new facts about the world (equivalent to predicting new edges on the graph). There are two main kinds of statistical relational models that try to predict new relations between entities. The first is based on latent feature models, such as the latent class model [21, 22], the distance model [23], and embedding nets [12, 24, 25 and 26]. The second type of model involves mining observable patterns in graphs.

We look first at three common types of latent feature models.

Latent class model: In this model, each entity is assumed to belong to an unobserved latent class, and a probability distribution describes the relationships between each pair of classes. Kemp et al. (2004) [21] define a generative model in which a particular relationship is obtained between a pair of entities such that their probability depends on the class of each entity. In their article, Airoldi et al. (2005) [22] propose a Bayesian model that uses a hierarchy of probabilistic assumptions about the way entities interact with one another in order to learn latent groups, their typical interaction patterns, and the degree of membership of entities to groups.

Distance Model: This model is based on the idea that entities are likely to be in a relationship if their latent representations are close in terms of distance. Hoff (2008) [23] proposes a model based on the idea of eigenvalue decomposition that represents the relationship between two nodes as the weighted inner-product of node-specific vectors of latent characteristics. Such a model is able to represent datasets with homophily patterns. Homophily provides an explanation for data patterns often seen in social networks, such as transitivity ("a friend of a friend is a friend"), balance ("the enemy of my friend is an enemy"), and the existence of cohesive subgroups of nodes. Note that we use homophily as one of the heuristics in our approach.

Embedding Nets: Recent studies have shown that neural-based representation learning methods are scalable, and are effective at encoding relational knowledge with low dimensional representations of both entities and relations, which means that they can be used to extract unknown relational facts. One of the early works in this area by Bordes et al. (2011) [26] proposes a model in which, for any given type of relation, there is a specific similarity measure that captures the relation in question between entities. This model has the architecture of a neural network. In order to embed entities effectively in this model, it is necessary to define a training objective that learns relationships. Bordes et al. (2013) [25] meanwhile introduce TransE, an energy-based model for learning low-dimensional embedding of entities. In TransE, relationships are represented as translations in the embedding space. Another work by Lin et al. (2015) [12] presents TransR, which embeds entities and relations in a distinct entity space and relation space, and learns to embed better via translations between projected entities.

The problem with all the above latent feature methods is the existence of a large number of entities and relations between them, whereas in problems of conflict resolution there are often no predetermined relation types.

The second main type of statistical relational model – based on mining observable patterns in graphs – seeks to address this problem. This method works on the observed variables of a knowledge graph, extracting rules via mining methods and then using these extracted rules to infer new links. This is the approach adopted in our study: we try to mine some rules for predicting relations. The challenge here is that, in conflict resolution problems, there is usually no training relational data set. We therefore need to create such a data set within our modeling framework. In practice, we use a small clean entity-attribute dataset in order to generate a sufficient metadata set. The proposed model can then mine observable patterns over the metadata set and predict the specific relations between unseen entities.

3. Motivation and Overview

Problems of conflict resolution generally involve dealing with often conflicting claims about an entity. The task of a conflict resolver (or truth finder) is to determine the correctness of each claim. Depending on the level of data abstraction, a conflict resolution problem may engage with several concepts, including entity (or object), attribute, data source, claim, and truth value. The main approach used in most current methods is based on estimating the reliability of data sources. As mentioned earlier, two heuristics are used in this approach: that the claim provided by a reliable source is likely to be correct; and that a source that provides true value will be reliable.

Let us look at an example that illustrates how existing methods work, and what motivates our approach.

Example 1: Suppose the entity about which there are claims from multiple sources is *a person*. Table 1 shows part of the dataset. A data integration system gathers values about the attributes of entities from several sources, that we shall call S_1 to S_3. Each claim vector $\boldsymbol{c}_i^j$ specifies person i described by six attributes – *name*, *workClass*, *education*, *age* and *outcome* – provided byS_j. In order to simplify notation, each claim vector is considered as an observation o. Because of the varying levels of reliability of sources, different values may be published for the attributes of a person. In Table 1, all incorrect values are marked in bold, with the correct values written in brackets after the incorrect ones.

Table 1. Part of dataset including entity, attribute and claim

Source	Observation	Name	WorkClass	Education	Age	Outcome
S_1	o_1	John	Private	Bachelors	32	<=50K
	o_2	Mary	Local-gov	Masters	41	>50K
S_2	o_3	John	Private	Bachelors	32	<=50K
	o_4	Bob	**Local-gov** (Private)	Bachelors	30	<=50K
S_3	o_5	Alice	Local-gov	**Bachelors** (Masters)	45	>50K
	o_6	Mary	**Private** (Local-gov)	Masters	41	>50K

In this example, we can see that two observationso_1 and o_3 describe a person named John, and similarly two observations o_2 and o_6 describe Mary. In contrast, only source S_2 provides data about Bob and only source S_3 provides data about Alice. Current methods work as follows. SinceS_1 and S_2 both provide the same information about John (o_1 ando_3), the trustworthiness of both sources is increased. This also increases the reliability of the information about Bob provided by source S_2 (observation o_4), even although this is the only source of information about Bob. However, in fact source S_2 is not reliable (the workClass information about Bob is incorrect). Similarly, in the absence of other information about Alice, observation o_5 is considered true information and the reliability of S_3 is somewhat increased. But in reality, source S_3 is an unreliable source (it provides two items of erroneous information).The upshot is that current methods will consider observationso_1,o_2,o_3,o_4 ando_5 all to be true; will fail to find the true values ofo_4 ando_5; and will moreover inaccurately estimate the reliability of source S_2 in particular.

The above example indicates clearly how current methods become less effective when they are faced with the long-tail phenomenon and have few reliable sources at the entity level. In such cases, more items of information are needed. Looking at the relations between two entities can provide additional information and so help to describe the entities more effectively than can be achieved at the entity level. When a relation is established between two entities, the value of an attribute belonging to one entity can identify, or at least help to identify, the value of the analogous attribute belonging to the other related entity. Take for example the relation *same age* between two persons: if we know the age of one person, we can determine the age of the other person. Similarly, the presence of a *classmate* relation between two persons can help us to identify the educational level of the two persons in question.

Relation-Based Conflict Resolver (RelBCR): Based on the above observations, we propose a relation-based method for conflict resolution. We investigate the use of further information that can be extracted from a higher level of data abstraction. Such additional information can be inferred by machine learning methods, in the form of a set of *rules* that describe the *relations* between entities. In this context, a relation is a triple that contains two entities and the type of relationship between them; a rule is a set of attributes and their values as an antecedent; and a triple (*subject*; *relation type*; *object*) as a consequent.

Example 2: Consider two relation types $r_1 = < same_{education} >$, $r_2 = < same_{workClass} >$ and the following two rules inferred about the existence of these relations between two entities e_1 ande_2.

Rule 1:workClass(e_1) = workClass(e_2), race(e_1) = race(e_2), outcome(e_1) = outcome(e_2) $\rightarrow$ ($e_1, same_{education}$,e_2).

Rule 2: education(e_1) = education(e_2), outcome(e_1) = outcome(e_2) $\rightarrow$ ($e_1, same_{workClass}$,e_2).

Applying Rule 1 to Table 1 above, the relation type$< same_{education} >$should exist between Alice and Mary. In other words, the education level of Alice should be equal to that of Mary. The value "Bachelors" for the attribute *education* of Alice should therefore be corrected to "Masters". Similarly, the value "Local-gov" should be corrected to "Private" for Bob based on the existence of relation type $< same_{workClass} >$ between Bob and John that is inferred based on Rule 2.

In sum, relations can be very informative and can help in estimating the correctness of values claimed about attributes. With proper and reliable rules, we can extract relations and which can then be used in the DF process.

However, discovering and using proper rules raises some challenges:

- In relational machine learning applications like link prediction or knowledge completion, relation types are predefined. For example, in social networks the relation type "friendship" is defined and entities are described by both attributes and relations. However, in a conflict resolution problem, the initial data is described only by attributes. This begs the questions: what are the relation types, and how can we define them?
- We use relations as additional information to increase the accuracy of the DF process. However, although there are a number of methods for estimating relations between entities used in applications like link prediction and ontology completion [12, 13, 24], in such problems there is usually a large training dataset of entities and relations. The challenge is how to draw inferences about relations when there is no training set containing entities and relations.
- After finding relations between entities, the issue is how to use these relations to estimate the correct values for each attribute.

In this article, we address these challenges. To meet the first challenge, we define a relation schema based on attributes. A relation schema is a set containing relation types in the form of $< same_{attribute} >$ and$< bigger_{attribute} >$. This means that it is always possible to define at least one relation type for each attribute. For example, for the attribute age, a relation type$< same_{age} >$ can be defined. Details related to the definition of relation types are given in section 4, Definition 1. For the second challenge (drawing inferences without a training set), we create metadata sets containing attributes of a pair of entities and one binary attribute which determines whether there is a relationship between the pair of entities. We next apply learning methods like classification and association rule mining across these metadata sets. Such classifiers or association rule miners serve as inference engines that can be used to identify appropriate rules about relations. We can then decide about new pairs of entities, and the relations between them, through these inferred rules. Finally, for the third challenge, we define some fusion functions and use these to select the correct values from among multiple values about entities. In summary, our approach uses *relations* as a new concept in conflict resolution problems at a high level of fusion.

4. Problem Definition

In this section, we first define concepts in the DF process for conflict resolution. We then define problems of particular interest in this article.

The problem of conflict resolution involves a range of general concepts. An *entity* is a real-world object of interest, like a person, book or film. An attribute is a feature of an entity that describes it at the entity level, such as the name, age, gender and race of a person. A *data source* is a resource that provides the values of attributes: for example, websites. These values may be correct or incorrect, and so are called *claims.*

Suppose there are N_s data sources providing claims about the attributes of entities. Let $O = \{e_1, \dots, e_{N_o}\}$be the set of all $N_o$entities and let $A = \{att_1, \dots, att_M\}$ be the set of all M attributes. Each attribute can be continuous or categorical. So let $AT = \{t_1, \dots, t_M\}$be the set of attribute types. For the j-th attribute type, t_j = 1 if att_j is categorical, and t_j = 2 if att_j is continuous. The claim about j-th attribute of the i-th entity provided by the k-th source is denoted as c_{ij}^k.All claims are collected in a $\{N_o \times M \times N_s\}$ third-order tensor$\boldsymbol{C} = \{c_{ij}^k\}$. We denote the k-th frontal slice of the tensor $\boldsymbol{C}$ by $\boldsymbol{C}_k$(which is a matrix of size $\{N_o \times M\}$), representing all of the claims provided by the k-th data source. The claims about the i-th entity provided by the k-th data source are a vector denoted by $\boldsymbol{c}_i^k = \{v_j\}_{j=1\dots M}$, where $v_j \in D_j$ is the value of the attribute subject to att_j in the domain attributeD_j. A relation is in the form of triple (*subject*; *relation type*; *object)*,where *subject* and *object* are an entity and *relation type*is a relationship between a *subject* and an *object*.

Definition 1 (relation schema): Given the set of attributes A and attribute type AT, we can define a *similarity relation type* according to both continuous and categorical attributes. For continuous attributes, a *comparative relation type* can also be defined. In this article, we use only one relation type for the categorical attribute att, $< same_{att} >$, and two relation types for the continuous attributes att,$< bigger_{att} >$ and $< same_{att} >$. Note that, as a general rule, the number of relation types can be more depending on the given data set and the application. A collection of these relation types forms the relation schema Rel_Sc. The number of defined relation types in Rel_Sc is$N_r = \sum_{j=1}^{M} t_j$,and the k-th relation type in Rel_Sc is denoted by r_k.

Example 3: Let us take the attribute set $A = \{age, education, race, outcome\}$ and $AT = \{2,1,1,2\}$. If for instance $att_1 = age$ and $t_1 = 2$, this attribute is a continuous attribute and two relation types$< same_{age} >$ and $< bigger_{age} >$ can be defined. The relation schema is $Rel_Sc = \{< same_{age} >, < bigger_{age} >, < same_{education} >, < same_{race} >, < same_{outcome} >, < bigger_{outcome} >\}$and total number of defined relation types is six.

Definition 2 (metadata set): Given a clean dataset, including entities $O = \{e_1, \dots, e_{N_o}\}$ and true values about each attribute, for each relation type r in the relation schema, a metadata set is created. Note that, each relation type is constructed based on an attribute. We indicate these attributes by k. Each row in the metadata set is a pair of entities (e_i, e_j) and the columns are all attributes of e_i and e_j except k. The last column is a binary attribute that indicates the existence of relation $(e_i; r; e_j)$. When we have clean data on N_o entities with M attributes, the number of instances in the metadata set is $N_o \times N_o$, and the number of columns is $2 \times (M - 1) + 1$.

Example 4: Let the set of entities be $O = \{John, Mary, Bob\}$ and the attribute set be $A = \{age, job, marital\}$. Given relation type $r =< same_{age} >$, a metadata MD set is created. The rows of MD be $(John, Mary)$, $(John, Bob)$,$(Mary, Bob)$ and the columns are lhd_job, rhd_job, $lhd\_marital$, $rhd\_marital$ and $same_{age}$. The term of lhd means left-hand entity and rhd means right-hand entity. For example, for the pair $(John, Mary)$, the values of columns are John's job, Mary's job, John's marital status, Mary's marital status and 1 if John and Mary are the same-aged and 0 otherwise.

In section 5.2, Example 7 illustrates the details of metadata creation process.

Definition 3 (Relation tensor): Given a set of entities O, and a set of relation types in relation schema Rel_Sc, all possible triples in $O \times O \times Rel\_Sc$ can be grouped naturally in a third-order tensor $\mathcal{RT} \in \{0,1\}^{N_o \times N_o \times N_r}$, whose entries are set such that

$$\mathcal{RT}_{ijk} = \begin{cases} 1 & , if\ relation\ type\ r_k\ \ exists\ between\ e_i, e_j \\ 0 & , otherwise \end{cases}$$

The k-th frontal slice of tensor $\mathcal{RT}$ denoted by $\boldsymbol{RT}_k$ is a matrix $\{N_o \times N_o\}$, that indicates the relation instances of the k-th relation type in Rel_Sc.

Example 5: Let the set of entities be $O = \{John, Mary, Bob, Alice\}$. Given the relation schema in Example 3, the third relation type $<same_{education}\ \ >$,

$$\boldsymbol{RT}_3 = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$\boldsymbol{RT}_3$ shows that both John and Bob on the one hand, and Mary and Alice on the other hand, have the same education level.

Definition 4 (Confidence tensor): Let $\boldsymbol{C} = \{c_{ij}^k\}$ be the claims about the j-th attribute of the i-th entity provided by the k-th source. Confidence score of the claims are indexed by a third-order tensor $\mathcal{CT} \in [0,1]^{N_o \times M \times N_s}$,such that for the j-th attributeof the i-th entity $\sum_{k=1}^{N_s} \mathcal{CT}[i][j][k] = 1$. A higher confidence score indicates that the claim is closer to the real world and the probability of its correctness is high.

Example 6: In Table 1, confidence score for the value "Bachelors" about attribute *education* of Alice that is provided by S_3 must be few because it is an incorrect claim.

Problem definition: Given a collection of claims $\boldsymbol{C}$ about a set of entities O from N_s sources, we attempt to accurately infer the relation tensor $\mathcal{RT}$ and confidence tensor $\mathcal{CT}$ such that correct values have higher confidence score than other incorrect claims based on their relations.

5. Methodology

The main purpose of our proposed RelBCR is to improve the performance of DF using relations between entities. This approach contains two main parts: estimating relations by calculating $\mathcal{RT}$ and truth finding by calculating $\mathcal{CT}$.This section of the article first gives a broad perspective of our approach by introducing a framework in section 5.1.

Thereafter, in section 5.2, we talk in more detail about relation extraction, and explain the assumptions and requirements needed to infer relations. Finally, we discuss how to compute a confidence score based on the relations.

5.1. Framework

In conflict resolution problems where there is insufficient data at the entity level, we can gain additional information by drawing inferences from relations between entities, and using such additional information at a higher level to select true values. So, in RelBCR, there are two main parts. We have called the first part, drawing inferences about the existence of relations, the G-model. The second part, called the F-model, obtains related entities as an input and then calculates the accuracy of claims about entities.

Figure 1 contains illustrations of the F-model and G-model of our RelBCR.

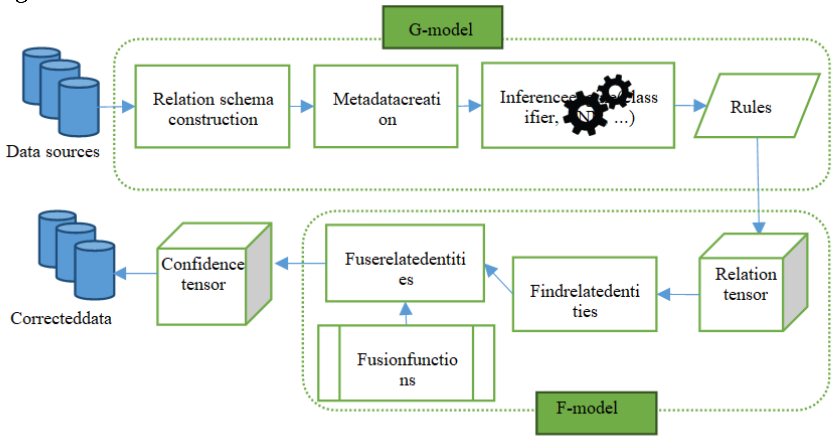


Fig. 1. Framework of proposed approach RelBCR

Data sources provide some claims about attributes of entities. All claims are collected to the tensor $\boldsymbol{C}$. The G-model contains three modules. Below, we explain the meaning of each module and the inputs and outputs involved.

Relation schema construction: According to Definition 1 given earlier, the task of this module is constructing relation types. The inputs for this module are a subset of attribute set A contains m attributes and related attribute type set AT, and the output is relation schema Rel_Sc. The relation schema is created as follows:

- For each attribute att create one relation in the form of $< same_{att} >$.
- For each continuous attribute att create one relation in the form of $< bigger_{att} >$.

The relation schema construction method is shown in Algorithm 1.

Algorithm 1: m is the number of attributes. The relation schema is defined as an array of the size of $\sum_{j=1}^{m} t_j$, where t_j indicates the type of attribute j. If the attribute is continuous, t_j is equal to two, indicating that two relation types must be added to the relation schema. If on the other hand the attribute is categorical, t_j is equal to one. For

ease of understanding relation types, these are added to the array Rel_Sc in the forms of $< bigger_i >$ and $< same_i >$, with subscript i for the i-th attribute.

Time complexity: In Algorithm 1, first the number of attributes is assigned to m. At most two relation types are added to the relation schema. So, the time complexity is $O(m)$.

Algorithm 1: Relation Schema Construction

```
Input: Attribute set $A = \{att_1, att_2, \dots, att_m\}$ and attribute type set $AT = \{t_1, \dots, t_m\}$.
Output: $Rel\_Sc$ ,the array in the size of $\sum_{j=1}^{m} t_j$ .
1: $m \leftarrow$ length($A$)
2: $r \leftarrow 1$ // counter for relation types
3: for $i \leftarrow 1$ to $m$ do{
4:   if $AT[i] = 2$ then{// attribute $i$ is continuous
5:     $Rel\_Sc[r] \leftarrow < bigger_i >$
6:     $r \leftarrow r + 1$
7:   }
8:   $Rel\_Sc[r] \leftarrow < same_i >$
9:   $r \leftarrow r + 1$
10: }
11: return $Rel\_Sc$
```

Metadata creation: To assess relations between entities, we create a metadata set as a training set, to be used as a training relation classifier. This training dataset needs to include sufficient negative and positive instances for each relation type. For this reason, we need a clean dataset, including entities and true values about each attribute. On the face of it, this may seem in contradiction with the main purpose of this article, which is to find true values for the attributes of entities. However, this dataset serves to provide *ground truth* data which, even on a very small scale, can greatly help us to create an appropriate metadata set (see section 6.6). In section 5.2, we explain the metadata creation process in more detail.

Inference engine: The input for this module is metadata, while the outputs are rules that can be used to indicate the existence of relations between two entities. The inference engine can be a learning method, such as classification, clustering or association rule mining. The types of rules about relations deduced by the inference engine depend on the learning method used. For example, if we use association rule mining, we will obtain association rules.

We can then decide about new entities, and the relations between them, through what we call a relation tensor $\boldsymbol{\mathcal{RT}}$ – which is an output of the G-model and an input for the F-model. The F-model has two modules, described below.

Find related entities: Using $\boldsymbol{\mathcal{RT}}$ for each entity and each relation type, related entities can be found. One entity can be in a relation with several entities. At the same time, there can be multiple relations for each entity. Because each relation is defined based on a specific attribute, this relation is used to compute the confidence score of claims related to that attribute.

Fuse related entities: The confidence scores for each claim for all of the entities are then calculated using the *fusion functions* set out in Table 4. The output of this module is confidence tensor $\boldsymbol{\mathcal{CT}}$.

After calculating $\boldsymbol{\mathcal{CT}}$, we select the value with the highest confidence score as the correct value.

5.2. RelBCRAlgorithm

Within the framework of our proposed RelBCR, there are two main phases for calculating $\mathcal{RT}$ and $\mathcal{CT}$ – the outputs of the G-model and F-model respectively. In this section, the processes and algorithms of these phases are explained and the time complexity in each phase is analyzed.

G-model: relation assessment phase. In this article, each relation type is denoted by $r = < same_{att} >$ or $< bigger_{att} >$. The entities and relations between them are represented by a third-order tensor, with each entry showing the existence of a relation between two entities. Using tensor representation for relations makes it relatively easy to obtain additional information by tensor manipulation. For example, the k-th *frontal slice* of a tensor $\mathcal{RT}$ of size $N_o \times N_o \times N_r$ representing a relation is a matrix of size $N_o \times N_o$, which represents the existence of a k-th relation between the entities in question.

Figure 2 shows a schematic image of tensor $\mathcal{RT}$.

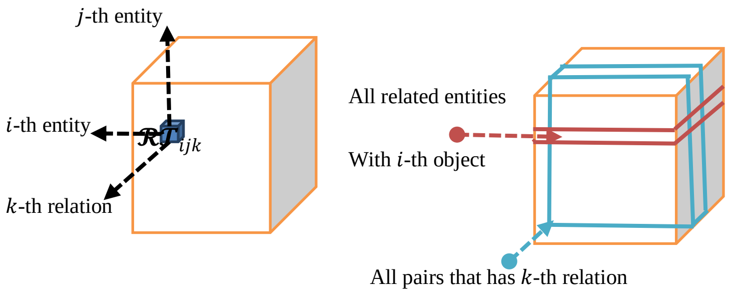


Fig. 2. Tensor representation of relations (left), frontal and horizontal slices of relation tensor (right)

Instances of a relation schema are gained by learning methods like classification. One such method, which we use, is known as triple classification. This seeks to judge whether a given triple (subject; relation type; object) is correct or not. We use metadata as a training set of classifiers, produced using a primary entity-attribute data set.

Example 7: Table 2 is part of an adult data set, in which each row is related to *one entity* that has several attributes like age, sex, education, and so on. This data set is considered as ground truth data. For each relation type in the relation schema, one metadata set is created. For a relation type in the form of $<same_{att}>$ the entity orders is not important, but for a relation type in the form of $<bigger_{att}>$ two orders $(e_1 e_2)$and $(e_2 e_1)$ are different from each other. Take for example the relation type $< same_{workClass} >$. A metadata set is created in which each row is related to one pair of entities, and includes the attributes of both the subject (left-hand entity) and object (right-hand entity), except $workClass$. Thus, the class label becomes binary: if the attribute of $workClass$ is the same for both entities, the class label is 1; otherwise, it is 0. Table 3 shows part of the metadata set produced using Table 2.

We can later run all the classifiers or clustering methods over the new metadata set and thus predict the existence of relation type $< same_{workClass} >$for new pairs of entities.

Table 2. Part of original entity-attribute data (adult data set) as ground truth

Entity	Sex	Education	Age	WorkClass
$\boldsymbol{e_1}$	Male	Bachelor	32	Private
$\boldsymbol{e_2}$	Male	Hs-grad	47	Private
$\boldsymbol{e_3}$	Female	Masters	35	Exec-managerial
$\boldsymbol{e_4}$	Male	Hs-grad	52	Private

Table 3. Example of metadata related to $< same_{workClass} >$ relation

Entity Pair	lhd_sex	lhd_edu	…	rhd_sex	rhd_edu	…	$\boldsymbol{same_{workClass}}$
$\boldsymbol{e_1 e_2}$	Male	Bachelor		Male	Hs-grad		1
$\boldsymbol{e_1 e_3}$	Male	Bachelor		Female	Masters		0
$\boldsymbol{e_1 e_4}$	Male	Bachelor		Male	Hs-grad		1
$\boldsymbol{e_2 e_1}$	Male	Hs-grad		Male	Bachelor		1
$\boldsymbol{e_2 e_3}$	Male	Hs-grad		Female	Masters		0
$\boldsymbol{e_2 e_4}$	Male	Hs-grad		Male	Hs-grad		1
$\boldsymbol{e_3 e_1}$	Female	Masters		Male	Bachelor		0
…							

The metadata creation module is summarized in Algorithm 2.

Algorithm 2: The number of entities in ground truth is n and the number of columns (attributes) is m. In the metadata set we consider all the pairs of an entity. So, if we have n^2 pairs of an entity, the row size of the metadata set is similarlyn^2. The columns of the metadata set contain attributes of both entities of each pair, except attribute k which represents the corresponding attribute of the relation; and one attribute as a class that indicates the existence of$k$-th relation type between these entities. The column size is therefore $(m-1)+(m-1)+1 = 2\times m-1$. The time complexity for metadata creation is thus $O(mn^2)$. Note that for the k-th relation type, we call the metadata creation algorithm by two inputs, the ground truth data set D and the index of attribute k.

```
Algorithm 2: Metadata Creation
Input: ground truth data set $D_{n\times m}$, index of attribute $k$ .
Output: metadata set $MD_{n1\times n2}$
1: $n1 \leftarrow n\times n$ // number of rows in metadata set
2: $n2 \leftarrow 2\times m-1$ // number of columns in metadata set
3: $inx \leftarrow 0$
4: for$i \leftarrow 1$ to $n$do{
5:   for $j \leftarrow 1$ to $n$do{
6:     $inx \leftarrow inx + 1$
7:     $MD[inx][1..m-1] \leftarrow D[i][1..k-1, k+1..m]$
       //all attribute values of the first entity except $k$
8:     $MD[inx][m..2\times m-2] \leftarrow D[j][1..k-1, k+1..m]$
       //all attributes of the second entity except $k$
9:     if $(D[i][k] = D[j][k])$ then$MD[inx][2\times m-1] \leftarrow 1$
10:      else$MD[inx][2\times m-1] \leftarrow 0$
11:  }
12: }
13: return$MD$
```

After the creation of metadata for each relation type, these data sets are used as training sets for a classifier. Running the training classifier for each relation type allows us to infer models, which can then be used to construct the relation tensor $\boldsymbol{\mathcal{RT}}$. The k-th frontal slice of the relation tensor is populated for the k-th relation type and for each pair of entities, thus showing the relation instances for the k-th relation types. A complete illustrative example is provided at the end of this section.

F-model: fusion phase. In this section, we explain the process of calculating confidence tensor $\boldsymbol{\mathcal{CT}}$, and introduce fusion functions used to infer true values. The claims provided by multi-sources are also represented by a third-order tensor, with each entry showing the existence of a certain claim about the given attribute of an entity. The goal of the F-model is to estimate the truthfulness of each claim when it is indexed by the third-order tensor$\boldsymbol{\mathcal{CT}}$.

Let $\boldsymbol{C} = \{c_{ij}^{k}\}$be the set of all claims about the j-th attributeof the i-th entity provided by the k-th source. We calculate the confidence score of claims as follows. Let relation types about attribute $j$be in the set $Rel^{j} = \{r_n\}_{n=1,2}$. Note that, for some attributes this set contains only one relation. For such attributes the first sigma in equation (1) is eliminated. Using$\boldsymbol{\mathcal{RT}}$, entities related to the entity i are recognized and added to the set $RO^{i} = \{e_{i'}\}$, such that $\boldsymbol{\mathcal{RT}}_{ii'n} = 1$. Based on the claims about i' provided by the set of sources $DS = \{S_{k'}\}$, we calculate the confidence score using the following equations:

$$\boldsymbol{\mathcal{CT}}[i][j][k] = \frac{1}{Z} \sum_{n=1,2} \sum_{i' \in RO^{i}} \sum_{k' \in DS} F\left(c_{ij}^{k}, c_{i'j}^{k'}, r_n\right) \tag{1}$$

where F is a fusion function and Z is a normalization factor that is:

$$Z = \sum_{k} \sum_{n} \sum_{i'} \sum_{k'} F\left(c_{ij}^{k}, c_{i'j}^{k'}, r_n\right), \tag{2}$$

where k is the index of sources provided claims about entity i.

A variety of different fusion functions are possible, depending on the nature of relation type r_n. This function can be a similarity function if r_n is a relation in the form of $< same_{att} >$ or $< bigger_{att} >$, with function F equal to a simple subtraction function. We define three types of fusion function F, which are listed in Table 4.

Table 4. Fusion functions

Form of relation type r	*Attribute*	*Fusion function F*
$< same_{att} >$	categorical	$F\left(c_{ij}^{k}, c_{i'j}^{k'}, r_n\right) = sim(c_{ij}^{k}, c_{i'j}^{k'})$
$< same_{att} >$	continuous	$F\left(c_{ij}^{k}, c_{i'j}^{k'}, r_n\right) = \frac{1}{\lvert c_{ij}^{k} - c_{i'j}^{k'} \rvert + \varepsilon}$
$< bigger_{att} >$	continuous	$F\left(c_{ij}^{k}, c_{i'j}^{k'}, r_n\right) = c_{ij}^{k} - c_{i'j}^{k'}$

Two modules of the F-model, which identify related entities and fused related entities respectively, are shown in Algorithm 3 and Algorithm 4.

The inputs for Algorithm 3 come from relation tensor $\boldsymbol{\mathcal{RT}}$, entity i and relation r. The output is a vector RO a list of entities that are related to i-th entity based on r-th relation type.The time complexity of Algorithm 3 is thus $O(N_o)$.The aim of Algorithm 4 is to calculate the confidence score for each claim based on related entities. The inputs for this algorithm are relation tensor $\mathcal{RT}$ and relation schema Rel_Sc. The output is confidence tensor $\mathcal{CT}$.

```
Algorithm 3: find related entities
Input: relation tensor $RT_{N_o \times N_o \times N_r}$, entity $i$, relation type $r$.
Output: $RO$, a list of entities that are related to $i$ based on relation type $r$
1: for $j \leftarrow 1$ to $N_o$ do{
2:     if $RT[i][j][r] = 1$ then $RO$.add($j$)
3: }
4: return $RO$
```

```
Algorithm 4: Fuse related entities
Input: relation tensor $RT_{N_o \times N_o \times N_r}$, relation schema $Rel\_Sc$
Output: confidence tensor $CT_{N_o \times M \times N_s}$
1: for $i \leftarrow 1$ to $N_o$ do{
2:   for $j \leftarrow 1$ to $M$ do{
3:     $R \leftarrow$ the indices of relation types about attribute $j$
         //size of $R$ is 1 or 2 based on attribute type $j$
4:     for $k \leftarrow 1$ to $N_s$ do{
5:       $RO \leftarrow find\_related\_entities(RT, i, Rel\_Sc[R])$
         // $RO$ is the list of all related entities to entity $i$ based on the relation
         $Rel\_Sc[R]$
6:       DS $\leftarrow find\_sources(RO)$
         // a procedure that finds the list of all sources providing value about
          entities in $RO$
7:       calculate $CT[i][j][k]$ according to Eq. (1)
8:     }
9:   }
10: }
11: return $CT$
```

Algorithm 4: In this algorithm, for each entity i, each attribute j, and each data source k, $CT[i][j][k]$ is calculated. First, in line 3 the indices of relation types about attribute j are stored in array R. Then in line 5, according to the relation types in R, all entities that are related to entity i are stored in the array RO using Algorithm 3 (Find related entities). Next, in line 6, all the sources producing value about entities in RO stored in the array DS. Finally, $CT[i][j][k]$ is calculated using Equation (1).

Time complexity: There are three loops in Algorithm 4. The time complexity is $O(N_r {N_o}^2 M {N_s}^2)$, where N_r is the number of relation types, N_o is the number of entities, M is the number of attributes and N_s is the number of data sources. Because of $N_o \gg M, N_r, N_s$, Algorithm 4 is a quadratic-time algorithm with respect to N_o, which is validated experimentally in section 6.6.

5.3. Illustrative Example

In this section, an illustrative example is provided for the whole approach step by step, from data claims with conflicts to the resulting fused data.

Example 8: Suppose the entity about which there are claims from multiple sources is *a person.* Attribute set is $A = \{$*age, work-class, sex, education-level, outcome*$\}$. Attribute type set is $AT = \{2, 1, 1, 1, 1\}$. Table 5 below shows clean data set for the persons (entities) that is the ground truth.

Table 5. A sample of the ground truth for the persons

ID	Age	Work-Class	Sex	Education-Level	Outcome
1	32	State-gov	Female	2	<=50k
2	35	State-gov	Male	2	<=50k
3	37	State-gov	Female	2	<=50k
4	46	Self-employed	Male	1	<=50k
5	29	Private	Male	3	>50k
6	53	Private	Male	1	>50k
7	45	Self-employed	Male	3	>50k
8	48	Self-employed	Male	3	>50k
9	35	State-gov	Female	2	<=50k
10	58	Private	Male	3	>50k

For one of the relation type in the relation schema <$same_{age}$ >, the metadata set is created using Algorithm 2. This metadata set has 100 rows and 9 attributesinclude: *lhd_work-class, lhd_sex, lhd_education-level, lhd_outcome, rhd_work-class, rhd_sex, rhd_education-level, rhd_outcome,* and*sameAge*. To calculate the values of attribute *sameAge*, we first discretize the values of attribute *age* into four categories ([20-30), [30-40), [40-50) and [50-60)). If the categories of attribute age for the left-hand entity and the right-hand entity are the same, the value of attribute *sameAge* is equal to 1, otherwise it is 0. Table 6 shows part of the metadata set.

The next step is inferring rules about the existence of relation type $< same_{age} >$ between the entities. We use CAR by Thabtah et al. (2005) [27] as a classifier. The following rules in Table 7 are extracted by this classifier using metadata set as a training set.

Table 8 shows some claims about seven persons with *personID* from 1 to 7. These claims are provided by three sources S_1, S_2 and S_3. Note that although there are some conflicts in the values of only one attribute in this example, but generally the rest of the attributes can also have conflicts in their values. All incorrect values are marked in bold, with the correct values written in brackets after the incorrect ones.

Based on Table 8, the first vertical slice of claim tensor $\boldsymbol{C}$represents all claims about the value of attribute age provided by all sources.

$$\boldsymbol{C}_{\text{age}} = \begin{bmatrix} \mathbf{45} & \mathbf{35} & \times \\ \mathbf{36} & \times & \mathbf{26} \\ \times & \mathbf{32} & \times \\ \times & \mathbf{30} & \times \\ \mathbf{28} & \times & \times \\ \mathbf{57} & \times & \mathbf{47} \\ \times & \times & \mathbf{48} \end{bmatrix}$$

Then, one frontal slice of tensor $\boldsymbol{\mathcal{RT}}$ that is related to relation type <$same_{age}$ > is constructed using the rules extracted by the classifier in Table 7.

Table 6. Part of metadata set related to relation type $< \boldsymbol{same_{age}} >$

Entity pair	lhd _work-class	lhd _sex	lhd _educ.-level	lhd _outcome	rhd _work-class	rhd _sex	rhd _educ.-level	rhd _outcome	SameAge

1,2	State-gov	Female	2	<=50k	State-gov	Male	2	<=50k	1
1,3	State-gov	Female	2	<=50k	State-gov	Female	2	<=50k	1
1,4	State-gov	Female	2	<=50k	Self-employed	Male	1	<=50k	0
…									
2,3	State-gov	Male	2	<=50k	State-gov	Female	2	<=50k	1
2,4	State-gov	Male	2	<=50k	Self-employed	Male	1	<=50k	0
…									
7,8	Self-employed	Male	3	>50k	Self-employed	Male	3	>50k	1
7,9	Self-employed	Male	3	>50k	State-gov	Female	2	<=50k	0
7,10	Self-employed	Male	3	>50k	Private	Male	3	>50k	0
…									

Table 7. Extracted rules related to relation type $< same_{age} >$

Rule 1: *(lhd_work-class = State-gov)Λ(rhd_sex= Male)=>sameAge=0*
Rule 2: *(rhd_work-class=State-gov)Λ(lhd_sex= Male)=>sameAge=0*
Rule 3: *(rhd_work-class= State-gov)Λ(lhd_work-class= State-gov)=>sameAge=1*
Rule 4: *(lhd_work-class= Private)Λ(rhd_outcome=<50k)=>sameAge=0*
Rule 5: *(rhd_work-class= Private)Λ(lhd_outcome =<50k)=>sameAge=0*
Rule 6: *(rhd_work-class = Self-employed)Λ(lhd_work-class = Self-employed)=>sameAge=1*
Rule 7: *(rhd_work-class = Private)Λ(lhd_work-class = Self-employed)=>sameAge=0*
Rule 8:*(rhd_work-class = Private)Λ(lhd_work-class = Private)=>sameAge=0*

$$\boldsymbol{RT}_{<same_{age}>} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Using Algorithm 3, the related entities of each person are found: persons that their *personID* are 1, 2 and 3 are related to each other and persons 4, 6 and 7 are related to each other too.

Table 8. Data claims provided by the sources

Source	PersonID	Age	Work-Class	Sex	Education-Level	Outcome
S_1	1	**45** (35)	State-gov	Female	2	<=50k
	2	36	State-gov	Female	2	<=50k
	5	28	Private	Male	3	>50k

	6	**57** (47)	Self-employed	Male	1	>50k
S_2	1	35	State-gov	Female	3	<=50k
	3	32	State-gov	Female	2	<=50k
	4	**30** (45)	Self-employed	Female	2	<=50k
S_3	2	**26** (36)	State-gov	Female	2	<=50k
	6	47	Self-employed	Male	1	>50k
	7	48	Self-employed	Male	3	>50k

Final step is fusing the values of attribute *age* of entities using algorithm 4. We now calculate the confidence score of two claims about attribute *age* of person 1. The list of related entities to entity 1 is $RO^1 = \{2,3\}$; the set of sources providing claims about $i' \in RO^1$ is$DS = \{S_1, S_2, S_3\}$. All values provided by the sources in DS about the entities in RO^1 is 36, 32 and 26. In this example, we use relation type $r =< same_{age} >$, therefore we apply second fusion function in Table 4. So, for claim $c_{1,1}^1 = 45$ and according to equation (1):

$$\boldsymbol{C\mathcal{T}}[1][1][1] = \frac{1}{Z}\sum_{i'=2,3}\sum_{k'=1,2,3} F\left(c_{1,1}^1, c_{i',1}^{k'}, r\right)$$
$$= \frac{1}{Z}\left(\frac{1}{|45-36|} + \frac{1}{|45-32|} + \frac{1}{|45-26|}\right)$$
$$= \frac{1}{Z}\,0.24$$

In the same way, for $c_{1,1}^2 = 35$, its confidence score is $\boldsymbol{C\mathcal{T}}[1][1][2] = \frac{1}{Z}\left(\frac{1}{|35-36|} + \frac{1}{|35-32|} + \frac{1}{|35-26|}\right) = \frac{1}{Z}1.4$; we calculate normalization parameter Z using equation (2):

$$Z = 0.24 + 1.4 = 1.64$$

Finally, the confidence score of $c_{1,1}^1$ is $\boldsymbol{C\mathcal{T}}[1][1][1] = \frac{0.24}{1.64}$ =0.15, and $c_{1,1}^2$ is $\boldsymbol{C\mathcal{T}}[1][1][2] = \frac{1.4}{1.64}$ =0.85. As a result, the claim provided by S_2 is selected as a correct value. As for person 2, there are two claims, and the value 36 is selected as a correct value, because the probabilities of the correctness for these two claims are 0.8 for $c_{2,1}^1$and 0.2 for $c_{2,1}^3$. The first vertical slice of confidence tensor $\boldsymbol{C\mathcal{T}}$represents all confidence scores of claims about attribute age provided by all sources.

$$\boldsymbol{CT}_{age} = \begin{bmatrix} \mathbf{0.15} & \mathbf{0.85} & \times \\ \mathbf{0.8} & \times & \mathbf{0.2} \\ \times & \mathbf{1} & \times \\ \times & \mathbf{1} & \times \\ \mathbf{1} & \times & \times \\ \mathbf{0.12} & \times & \mathbf{0.88} \\ \times & \times & \mathbf{1} \end{bmatrix}$$

About person 5 there is no related entity so the single value provided by S_1 is considered as the correct value. Although one value is provided about person 4 by S_2 but there are two entities that are related to it, persons 6 and 7. Because of this, there is a tradeoff between the selection of this single value and the average value of the related entities. It depends on the amount of confidence score for this single value and also the precision of the rules. We can decide to select one of these values by defining a threshold.

6. Experiments

In this section, we use two real data sets to evaluate our proposed approach. The aim of our experiments is to answer these questions:

Q1: Can classifiers be used directly to predict the value of entity attributes instead of having to infer the existence of a relationship between entities?

Q2: To what extent can the classifiers make correct and accurate predictions of the relationships in the final output of data fusion?

Q3: How accurate is high-level data fusion in terms of the number of reliable sources, compared to low-level fusion?

1.1. Experimental Setup

Data sets

To demonstrate the advantages of our proposed approach, especially in an environment involving few reliable data sources, we conducted experiments on real data sets generated from UCI machine learning data sets. The basic assumption about the data is that the entities must have one or more relationship(s) between them. In other words, there are several relations between entities in the dataset.We chose the **UCI Adult** (http://archive.ics.uci.edu/ml/datasets/Adult) and **UCI Bank** (http://archive.ics.uci.edu/ml/datasets/Bank+Marketing) datasets because the entities are related to each other by attributes. These two datasets contain raw data. We therefore perform some preprocessing on these in order to clean them up, as follows:

- Deletion of attributes: If the percentage of entities that have an unknown value or the same value for a specific attribute is more than 80%, then this attribute is deleted.For example, in the Bank dataset, 36,959 entities – more than 80% of the entities – have an unknown value. The attributes related to these were therefore deleted from the dataset. In total, in the Bank dataset eight out of 17 attributes, and in the Adult dataset four out of 15 attributes, were deleted by this process.
- Remove instances: If an entity has an unknown value for one or more attributes, this entity is removed from the dataset. For example, in the Adult dataset, 1836 entities have an unknown value related to the "*workClass*" attribute, so these entities are removed. As a result of this process, the number of entities in the Adult dataset was reduced to 30,162, and in the Bank dataset to 43,193.
- Discretization: Since attributes selected for classification must be a categorical attribute, we discretize any continuous attributes. Discretizing these attributes

makes them into ordinal categorical attributes, on which it is then possible to build comparative relationships.

After these three steps to preprocess the initial rough dataset, the resulting dataset is regarded as the ground truths. To answer Q1, we use these datasets as inputs for the classifier in order to train the model to predict the value of attributes. We then create a metadata set as a training set for the classifier to train the model to infer the existence of relationships between entities. Based on attributes selected to create noisy data sources, we then construct a *relation schema*. Table 9 shows the statistics of these datasets, while Table 10 displays the relation schema for each dataset.

Metadata creation: To create metadata sets for each relation type, we implement Algorithm 2 in section 5.2. The ground truth we use to create metadata sets contains 1000 entities, so the Adult and Bank datasets each have 1,000,000 rows of metasets. The number of columns in the Adult dataset is $10 \times 2 + 1 = 21$ and in the Bank dataset is $8 \times 2 + 1 = 17$. (Note that, in section 6.5, we carry out an experiment to investigate the effect of the number of entities used to build the metadata set on the accuracy of our approach.)

In section 6.2, we report the results of our experiments designed to predict the value of attributes vs. the existence of a relation, and hence to answer Q1 (*Can classifiers be used directly to predict the value of entity attributes instead of having to infer the existence of a relationship between entities?*). In the remainder of this section, we explain how to build data sources.

Table 9. Statistics of data sets

	Adult	**Bank**
#entities	30162	43193
#attributes	11	9
#relation types	5	5
#data sources	10	10
#claims provided by data sources	126679	216032

Creating data sources: We generate a data set consisting of multiple conflicting sources, by injecting different levels of noise into the ground truths as the inputs to our approach and baseline methods. We select four attributes in each dataset (The attributes *age*, *education*, *workClass* and *occupation* were selected in the Adult dataset; and the attributes *age*, *job*, *marital* and *education* in the Bank dataset), whose values are then randomly flipped to generate facts that deviate from the ground truths. A parameter α is used to control the degree of reliability of each source. To put it another way, α stands for the percentage of noisy data. In this way, we can generate datasets which contain 10 sources with various degrees of reliability (α= 50, 55, 60, …, 95).

Table 10. Relation schema

Adult	Bank
{$<same_{age}>$,$<bigger_{age}>$, $<same_{education}>$,$<same_{workClass}>$, $<same_{occupation}>$}	{$<same_{age}>$,$<bigger_{age}>$, $<same_{education}>$,$<same_{job}>$, $<same_{marital}>$}

Algorithm Implementation

In section 5, we explained the modules of RelBCR and presented the algorithms related to each module as Algorithm 1 to 4. In this section, we describe in more details about some of the implementation-related issues.

Algorithm 1 is for relation schema construction. In this algorithm, based on the type of each attribute, one or two relation types are created. The type of attributes must be specified as an input to the problem and it is a part of the problem knowledge. We use comma-separated values (CSV) files for input datasets. Therefore, when reading data from the input file, we can specify the type of each column (attribute). In addition, in Algorithm 1 for ease of understanding we show relation types in the form of $< same_{att} >$ and $< bigger_{att} >$. In reality, we use a cell array that is an array of length M (number of attributes) such that each element can be an array of length one or two. The first element of this array represents relation type $< same_{att} >$, and the second of this, if any, represents another relation type $< bigger_{att} >$.

The next point is about metadata sets. Although input data set contains different types of attributes, but after discretization in the preprocessing step, all attributes become categorical and ordinal attributes. So, metadata sets contain only categorical attributes.

Performance Measure

Our proposed framework consists of two main parts. The first part contains the classifier methods for predicting the relations between entities (the G-model), while the second part comprises the fusion functions for finding the truth between conflicting values (F-model). We need to evaluate the performance of the methods used for both parts. Two measures, *precision* and *recall*, are used to evaluate the G-model, while *accuracy* is used to evaluate the F-model:

- **Precision** (or confidence) denotes the proportion of predicted positive instances that are correct real positives.
$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{3}$$
- **Recall** (or sensitivity) is the proportion of real positive instances that are correctly predicted positive.
$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{4}$$
- The **F-measure** is the harmonic mean of precision and recall.
$$F - measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$
- We use **accuracy** as performance measure which is computed as the percentage of the output of the F-model that is the same as the ground truths.

$$accuracy = \frac{1}{n_g}\sum_{i=1}^{n_g} 1\{g_i = f_i\} \tag{6}$$

where n_g is the number of entities in the ground truth, g_i is the entity in the ground truth set and f_i is the fusion output.

Environment

All the experiments in this study were conducted on a workstation with 8GB RAM, an Intel® Core™ i5-4300U CPU @ 1.90GHz 2.50 GHz, and Windows 10 pro. All the algorithms, including those from earlier methods, were implemented in Matlab R2017a. Weka 3.8 data mining tools were used to preprocess the datasets and infer classifiers.

6.1. Relation Estimation

In this section, in order to answer Q1, we need to carry out some experiments to classify each attribute against the relationtype related to this attribute. Two classification methods, a decision tree and classification based on association rules (CAR), are used for this purpose. The aim of this experiment is to investigate the impact of using additional information on the performance of fusion techniques. Such additional information can be extracted either from the original dataset or from the metadata set.

In the first stage, we try to train a classifier that can predict the values of the attributes of entities. These attributes are considered as a *class*. For example, the attribute *occupation* in the Adult data set is a categorical attribute that has 14 values, such as *exec-managerial* and *handlers-cleaners*. Our classifier must therefore be trained to predict 14 classes of *occupation*.

In the second stage, we use relation types as a *class* and train classifiers accordingly. However, relation types are not pre-defined and there is no training set available that contains entities and relations between them (as is also true of applications in relational machine learning such as knowledge graph completion). A metadata set is therefore constructed as described in section 5.2. Each relation type in the relation schema is regarded as a class. This is known as triple classification: it aims to judge whether a given relation instance (*entity1; relation type; entity2*) is correct or not. This is a binary classification task, as explored by Lin et al. (2015) [12] and Socher et al. (2013) [28] in order to evaluate a link prediction task.

We use a decision tree (C4.5) and CAR by Thabtah et al. (2005) [27] as classifiers. C4.5 is a popular algorithm for constructing decision trees. These two classifiers are used to infer the classification models of attributes in our two datasets, the Adult data and the Bank data. Also, we use these classifiers to train models over metadata sets that are considered as training set for each relation type.

For the Adult data set, we use separately the attributes of *age, education, workClass* and *occupation* as classes, and then evaluate the C4.5 classifier. There are a total of 30,162 instances in the Adult dataset, of which 60% are considered as a training set and the rest as a test set. Next, the relation types $< same_{age} >$, $< bigger_{age} >$, $< same_{education} >$,$< same_{workClass} >$and$< same_{occupation} >$ are considered as classes. We construct metadata such that each row includes the attributes of two entities and the relation between them. 1000 entities are used to construct each metadata set, with the same number of positive and negative instances.

For the Bank data, the attributes of *age*, *job*, *marital* and *education* are used as classes, and then to evaluate the C4.5 classifier. There is a total of 43,193 instances in the Bank data set, of which 60% are considered as a training set and the rest as a test set. Next, the relation types $< same_{age} >$, $< bigger_{age} >$, $< same_{job} >$,$< same_{marital} >$ and $< same_{educatio\ n} >$ are considered as classes. Again, we construct metadata such that each row includes the attributes of two entities and the relation between them. 1000 entities are used to construct each metadata set, with the same number of positive and negative instances.

Figure 3 shows the evaluation results of this classifier. The x-axis in the figure indicates the attributes and relation types as a class, while the y-axis presents the percentage of instances classified correctly in all classes by the classifier C4.5. These results show that, when we use the classifier to predict attribute values, the number of instances classified into the correct classes is lower than the number of correctly classified instances when the classes are the existence of relationships. When a specific attribute is considered as a class, the instances are entities and the classes are the different values of this attribute. For example, if the attribute *marital* in the Bank dataset is considered as a class, the instances will be classified into three categories of *married*, *single* and *divorced*. The percentage of all true positives is 67.1%.On the other hand, when one relation type is considered as a class, the instances are pairs of entities and the classes are *yes* or *no*, depending on whether the given relationship exists between a particular pair of entities. For example, the percentage of all true positives in the classification of $< same_{marital} >$is 98.3%.

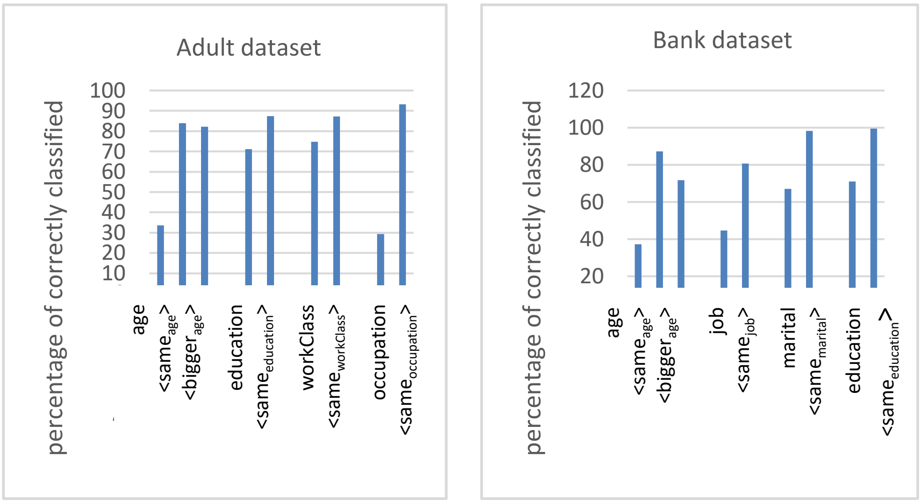


Fig. 3. Percentage of correctly classified instances using methods of attribute as a class and relation as a class

Tables 11 and 12 show the evaluation results of classification through both the original dataset and the metadata set.

Table 11 reports the classification results of two attributes, *occupation* and *age*. *Occupation* is a categorical attribute with 14 values, such as *Exec-managerial*, *Handlers-cleaners* and so on. The classifier therefore classifies the data into 14 classes. The precision and recall of each class of *occupation* are shown on the left-hand side of Table 11. As examples, the class *Other-service* was classified with a precision of 0.279 and a recall of 0.284, while *Armed-Forces* was not classified. The precision and recall of the classification of relation type $< same_{occupation} >$ were 0.928 and 0.955 respectively. The continuous attribute *age* is first divided into 10 categories. The data is then classified into 10 classes. The precision and recall of this classification are reported on the right-hand side of Table 11. Two relation types of the attribute *age* are defined, $< same_{age} >$ and $< bigger_{age} >$. Table 12 contains the results of classifying the attributes *workClass* and *education*, and their related relation types and Table 13 is related to the attributes and relation types of Bank dataset.

Table 11. Comparison of classifier performance measures related to attributes *occupation* and *age* (Adult dataset)

Occupation				Age			
Class	P	R	F	**Class**	P	R	F
Exec-managerial	0.283	0.301	0.292	**cat1 (-inf-24.3]**	0.566	0.733	0.639
Handlers-cleaners	0.129	0.119	0.124	**cat2 (24.3-31.6]**	0.312	0.305	0.308
Prof-specialty	0.47	0.525	0.496	**cat3 (31.6-38.9]**	0.258	0.257	0.257
Other-service	0.279	0.284	0.281	**cat4 (38.9-46.2]**	0.287	0.441	0.348
Adm-clerical	0.313	0.41	0.355	**cat5 (46.2-53.5]**	0.218	0.098	0.136
Sales	0.201	0.167	0.183	**cat6 (53.5-60.8]**	0.188	0.068	0.1
Transport-moving	0.164	0.106	0.129	**cat7 (60.8-68.1]**	0.264	0.187	0.219
Farming-fishing	0.278	0.195	0.229	**cat8 (68.1-75.4]**	0.125	0.019	0.034
Machine-op-inspct	0.185	0.131	0.154	**cat9 (75.4-82.7]**	0.059	0.009	0.015
Tech-support	0.125	0.047	0.069	**cat10 (82.7-inf)**	0	0	0
Craft-repair	0.292	0.368	0.326	$< \boldsymbol{same_{age}} >$ **- yes**	0.835	0.86	0.848
Protective-serv	0.378	0.244	0.297	$< \boldsymbol{same_{age}} >$ **- no**	0.843	0.816	0.829
Armed-Forces	?	?	?	$< \boldsymbol{bigger_{age}} >$**-yes**	0.796	0.793	0.794
Priv-house-serv	0.279	0.119	0.167	$< \boldsymbol{bigger_{age}} >$ **- no**	0.841	0.843	0.842
$< \boldsymbol{same_{occupation}} >$- yes	0.928	0.955	0.941				
$< \boldsymbol{same_{occupation}} >$- no	0.943	0.91	0.927				

Table 12. Comparison of classifier performance measures related to attributes work class and education (Adult dataset)

WorkClass				Education			
Class	P	R	F	**Class**	P	R	F

State-gov	0.213	0.039	0.066	**(-inf-4.75]**	0.2	0.001	0.002
Private	0.771	0.97	0.859	**(4.75-8.5]**	0.067	0	0.001
Federal-gov	0.279	0.013	0.024	**(8.5-12.25]**	0.715	0.904	0.798
Local-gov	0.452	0.19	0.268	**(12.25-inf)**	0.703	0.586	0.639
Self-emp-inc	0.303	0.028	0.051	$< \boldsymbol{same_{education}} >$ - **yes**	0.894	0.873	0.884
Never-worked	?	?	?	$< \boldsymbol{same_{education}} >$ - **no**	0.849	0.874	0.862
$< \boldsymbol{same_{workClass}} >$- yes	0.866	0.857	0.861				
$< \boldsymbol{same_{workClass}} >$ - yes	0.878	0.886	0.882				

Table 13. Comparison of classifier performance measures related to attributes of *age*, *job*, *marital* and *education* (Bank dataset)

Age				**Job**			
Class	P	R	F	**Class**	P	R	F
cat1 (-inf-25.7]	0.478	0.18	0.262	**management**	0.585	0.843	0.691
cat2 (25.7-33.4]	0.459	0.569	0.508	**technician**	0.347	0.276	0.307
cat3 (33.4-41.1]	0.354	0.471	0.404	**enterpreter**	0	0	0
cat4 (41.1-48.8]	0.258	0.157	0.196	**blue-colar**	0.438	0.736	0.549
cat5 (48.8-56.5]	0.279	0.203	0.235	**retired**	0.509	0.553	0.53
cat6 (56.5-64.2]	0.417	0.269	0.327	**admin**	0.236	0.164	0.194
cat7 (64.2-71.9]	0.273	0.186	0.221	**services**	0.178	0.078	0.109
cat8 (71.9-79.6]	0.25	0.195	0.219	**self-employed**	0.063	0.002	0.004
cat9 (79.6-87.3]	0.167	0.056	0.083	**unemployed**	0.141	0.014	0.026
cat10 (87.3-inf)	0	0	0	**housemaid**	0.303	0.03	0.055
$< \boldsymbol{same_{age}} >$ **- yes**	0.889	0.912	0.9	**student**	0.501	0.35	0.412
$< \boldsymbol{same_{age}} >$ **- no**	0.842	0.803	0.822	$< \boldsymbol{same_{job}} >$ **- yes**	0.817	0.776	0.796
$< \boldsymbol{bigger_{age}} >$ **- yes**	0.693	0.636	0.663	$< \boldsymbol{same_{job}} >$ **- no**	0.799	0.837	0.817
$< \boldsymbol{bigger_{age}} >$ **- no**	0.734	0.781	0.757				
Education				**Marital**			
Class	P	R	F	**Class**	P	R	F
tertiary	0.788	0.672	0.725	**married**	0.677	0.891	0.769
secondry	0.706	0.847	0.77	**single**	0.648	0.48	0.551
primary	0.536	0.325	0.404	**divorced**	0.333	0.001	0.002
$< \boldsymbol{same_{education}} >$**- yes**	0.995	0.996	0.996	$< \boldsymbol{same_{marital}} >$ **- yes**	0.978	0.998	0.988
$< \boldsymbol{same_{education}} >$ **- no**	0.992	0.989	0.991	$< \boldsymbol{same_{marital}} >$ **- no**	0.995	0.95	0.972

As shown earlier in Figure 3, there are a lot fewer accurate values for attributes than for relations. In addition, for some classes the classifier is unable to construct models; the accuracy of these classes is therefore unknown. That said, while the precision and recall of the classifier are not high for certain relation types like $< same_{occupation} >$, they are of an acceptable level for the F-model, as we demonstrate in the next section.

Let us now look at classification based on association rules (CAR). CAR is a method for extending an efficient frequent pattern mining method, for FP-growth, for constructing a class distribution-associated FP-tree, and for mining large databases efficiently. Moreover, it applies a CR-tree structure to store and retrieve mined association rules efficiently, and prunes rules effectively based on confidence, correlation and database coverage. In effect, this classifier selects the most effective rule(s) from among all the rules mined for classification. Below, we show that using relation classification is more efficient than attribute classification. To demonstrate this, we look at some rules that CAR infers for predictions about attributes. The first three rules are inferred, using the Adult dataset, to predict some values of the *occupation* attribute. We then look at two rules inferred using the metadata set for the relation type $< same_{occupation} >$. These rules show that, for some values of attributes, no rule can be inferred; whereas, for relation types, rules with a high degree of accuracy can be obtained. Finally, we discuss the reasons for these results.

Some of the rules used by CAR for predictions about the attribute *occupation= Prof-specialty, other services* and *Adm-clerical*, and the related level of accuracy, are as follows:

workclass= Local-gov, age=5,hours-per-week=3, education-num=4, outcome= <=50K ==>occupation= Prof-specialty acc:(0.99).

workclass= Local-gov, race= Black, education-num=2, sex= Female, ==>occupation= Other-service acc:(0.97).

workclass= Federal-gov, race= Black, hours-per-week=3, age=3, ==>occupation= Adm-clerical acc:(0.95).

Note that there are no rules for some values of *occupation*.

We next apply CAR to the metadata. For the relation $<same_{occupation}>$,this produces rules such the following:

lhd_workClass= Federal-gov, lhd_education-num='(-inf-2.5]', rhd_age='(3.5-5.5]', lhd_age='(-inf-3.5]', rhd_education-num='(-inf-3.5]' ==>$same_{occupation}$*=1*acc:(0.99).

lhd_workClass= Local-gov, lhd_age='(5.5-6.5]', rhd_outcome =>50K, lhd_education-num='(2.5-3.5]' ==>$same_{occupation}$*=1* acc:(0.99).

The method of relations as a class does not suffer from the problem of lack of rules, because it is a binary classification and so, using the metadata set, relations can be deduced for every pair of entities.

Now we can answer Q1: Can classifiers be used directly to predict the value of entity attributes instead of having to infer the existence of a relationship between entities? The answer is *No*. There are a number of reasons why relations are better than attributes as a class.

1- The method of *relations as a class* deals with binary class prediction rather than the multi-class prediction used in the *attributes as a class* method.Multi-values make the accurate prediction of values difficult, and require extensive training data. In binary classes, in contrast, learning is both faster and more accurate.

2- In multi-classification the classifier may be unable to infer some classes. For example, for the attribute class of *occupation* there are no rules for the values *Armed-Forces* and *Priv-house-serv.* In binary classification, on the other hand, the value of attributes is not important; all that matters is whether the attributes have the same value or not.

3- In our proposed approach, all we need to understand is the relationship between one entity and another entity, not the exact values of the attribute of a given entity.

4- In high level fusion, we have a range of relations and methods, such as embedding nets [12, 24] and MLN [29] that can be applied to extract relations. These are used in our G-model – which is explained further below.

6.2. Effect of G-Model

For the G-model in our framework, we use classifiers trained by metadata, with each class being a relation. In this section, we examine the impact of the performance of the G-model on the accuracy of the F-model in answering Q2 (To what extent can the classifiers make correct and accurate predictions of the relationships in the final output of data fusion?). We use simulated models with different levels of precision and recall, and then evaluate the accuracy of the proposed fusion method. For each value of precision and recall, we repeat the experiment five times, with the average accuracy over these five repetitions shown in Figure 4. In this experiment we use the Adult data set. There are 10 data sources, of which only one is reliable. The accuracy of voting is 0.65.

As expected, the higher precision and recall of the G-model increases the accuracy of the fusion. Figure 4 shows that,for some values of precision and recall, the accuracy of our framework is higher than that of voting (the values that are above of the horizontal dash line). For example, in the G-model precision is 0.8 and recall is 0.7, which leads to the accuracy of the F-model is 0.91. For some values of precision and recall, the performance of this model is the same as (or lower than) voting. We consider such values as a *fail point* of our model. Table 14 reports the accuracy of the F-model for different amounts of precision and recall of the G-model. The empty cells indicate where the precision and recall of the G-model do not lead to appropriate results in the F-model – and so are fail points. For example, Precision =0.8 and recall = 0.4 is a fail point.

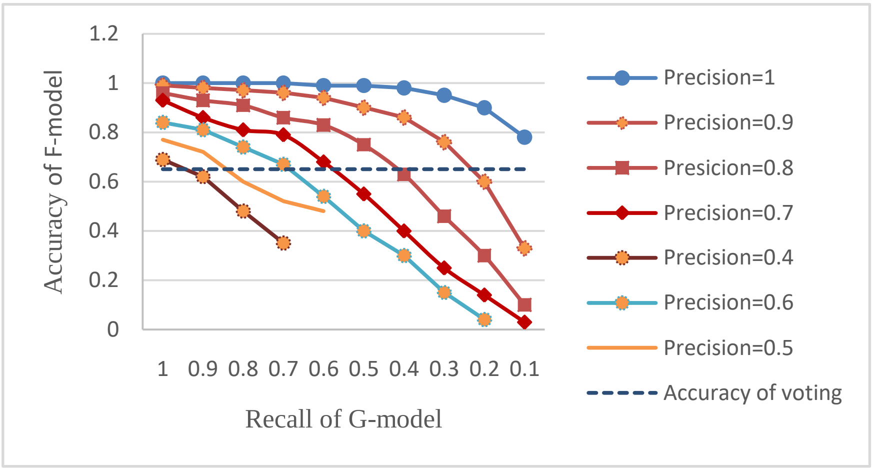


Fig. 4. Effect of G-model on performance of fusion

As can be seen from these tables, precision is more important than recall: with high precision, the framework is robust against low recall. For example, when precision is 0.7, the accuracy is better than voting in order to recall values of more than 0.6. This means that it is very important that our G-model does not mistake wrong relations for true ones. When our G-model is pessimistic about the existence of relations, its precision increases. By using multi-relations for each attribute, we can increase the precision of the G-model.

Table 14. Accuracy of F-model for different amounts of precision and recall

Precision→ **Recall ↓**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
1	0.69	0.77	0.84	0.93	0.96	0.99	1
0.9	×	0.72	0.81	0.86	0.93	0.98	1
0.8	×	×	0.74	0.81	0.91	0.97	1
0.7	×	×	0.67	0.79	0.86	0.96	1
0.6	×	×	×	0.68	0.83	0.94	0.99
0.5	×	×	×	×	0.75	0.90	0.99
0.4	×	×	×	×	×	0.86	0.98
0.3	×	×	×	×	×	0.76	0.95
0.2	×	×	×	×	×	×	0.90
0.1	×	×	×	×	×	×	0.78

6.3. High-Level vs Low-Level Fusion

In order to answer Q3 (*How accurate is high-level data fusion in terms of the number of reliable sources, compared to low-level fusion?*), we now compare our framework with low level fusion techniques including voting, Hub [15] and truth finder [4]. These low-level fusion methods, which we examined in our earlier work [30], contrast with RelBCR, which is a high-level fusion method. In this experiment, we fix the total number of sources as 10, and set the parameter α as the constant number 50%, which corresponds to an unreliable source. We then evaluate the performance of methods with different numbers of reliable sources. The precision of the G-model when used as a decision tree is 0.84 and its recall is 0.8.Figure 5 illustrates each method's accuracy on the dataset for different numbers of reliable sources.

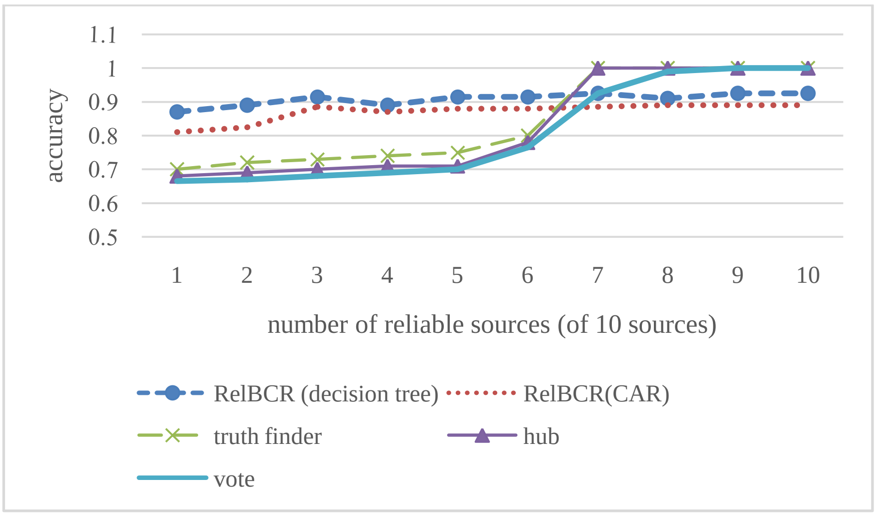


Fig. 5. Performance with respect to number of reliable sources

The following observations can be drawn from our results. First,our proposed approach outperforms existing conflict resolution techniques when there are few sources, because of its use of additional information about relations between entities. Second, when more than 50% of the sources are reliable, the performance of other existing voting models is slightly better than our approach. In general, it is easier to detect truths when we have a larger number of reliable sources, especially when we estimate the reliability of sources. In this experiment, the precision of the G-model – when the model is used as a decision tree – is 0.84 and its recall is 0.8, while the accuracy of the F-model is higher at 0.925. In section 6.3 we show that, if the precision of the G-model is increased, the performance of our overall approach can also be improved. Theoretically, therefore, by using a stronger inference engine we can obtain a higher level of accuracy. At the same time, the advantage of knowing the reliability of sources can be used to increase the accuracy of the F-model. We plan to examine this further in future research. Third and finally,using the G-model to estimate relations between entities and applying it to conflict resolution gives us more scope to estimate the reliability of sources, unlike with low level fusion methods.

6.4. Cost Analysis

As explained earlier, our approach uses additional information to improve the performance of the fusion process. Inevitably, the process of extracting and using such information makes the model more complicated. There are two main procedures in this approach: training the relation classifier in the G-model, and calculating the truthfulness of each claim in F-model. In this section, we discuss the overall costs associated with our approach.

Time Complexity

Here we test the computational complexity of calculating the truthfulness of each claim in the F-model (Algorithm 4, explained earlier). Figure 6 validates our time complexity analysis, confirming that the F-model is a quadratic-time algorithm with respect to the number of entities.

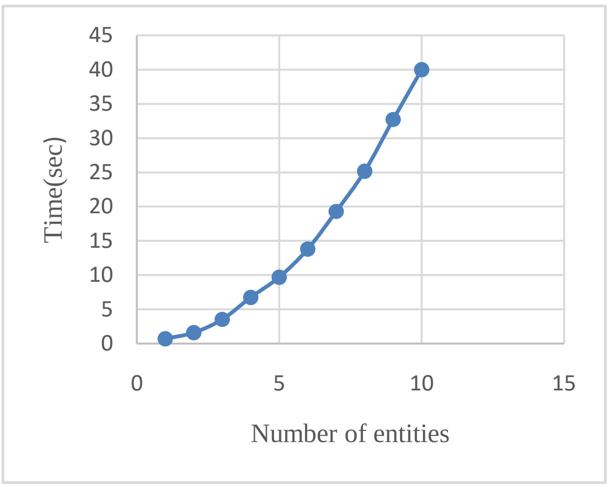
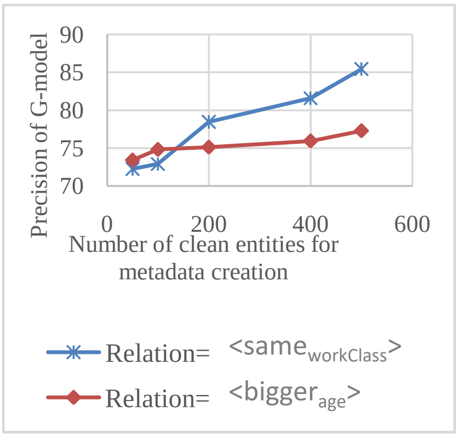


Fig. 6. Running time of confidence score calculation with respect to number of entities

Necessity of Clean Training Dataset

As explained in section 5.2, Figure 7 (left-hand side) shows the precision of the G-model, using various proportions of clean entities for metadata creation. We train the classifier for two relations, $< same_{workclass} >$ and $< bigger_{age} >$. As can be seen, the level of precision improves over iterations most of the time. Figure 7 (right-hand side) shows the accuracy of the F-model based on the G-model trained by various proportions of clean data.



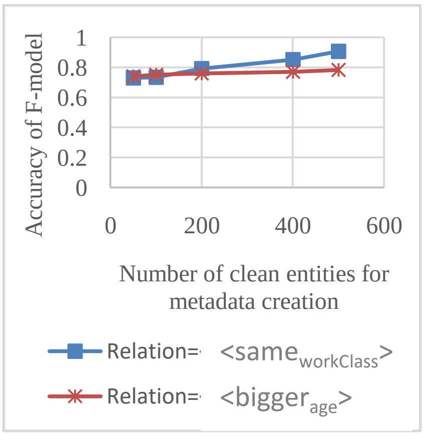


Fig. 7. Changes in precision of G-model (left) and accuracy of F-model (right) with respect to number of clean entities used in metadata creation

Figure 7 clearly illustrates that the precision of the G-model can be improved by increasing the proportion of clean data. However, for some relations like $< bigger_{age} >$, this improvement is very slow; more informative clean data is thus needed to train the classifier of this relation. It seems that using incremental learning to train the classifier, along with the fusing data procedure, can compensate for the lack of sufficient clean data. Hence, after training the classifier with a small clean dataset, we fuse the data and thereby obtain more clean data which can be used to retrain the classifier.

Performance with Respect to the Number of Related Entities

As explained in section 5.2, the truthfulness score of each claim about a specific entity depends on the claims about other entities that are related to it. Figure 8 shows the accuracy of the F-model with respect to the average of entities related to one entity. For this experiment, we use the G-model with a precision of between 0.8 and 0.95.

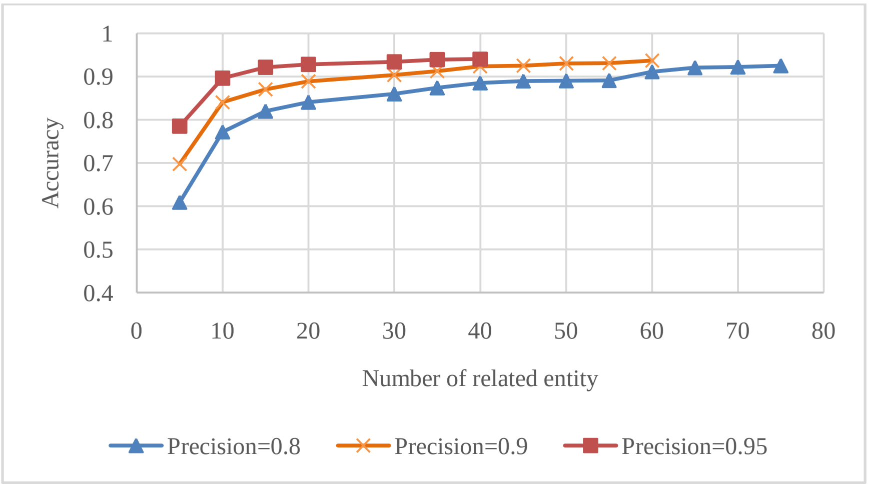


Fig. 8. Accuracy of F-model with respect to number of assessed related entities

This experiment showsthat, when the number of entities related to each entity increases, more evidence is collected for a given claim, and the accuracy consequently increases.This means that, if there are few relevant entities in the dataset for a specific entity, calculating the truthfulness score for this entity becomes very difficult. Figure 9 shows the number of related entities for each category of entities, with the value of the attribute *age* discretized to 10 intervals. As can be seen, in the Adult dataset, the number of related entities for cat2 of the relation $< same_{age} >$ is very small, and it is therefore necessary to obtain additional information from other relations.

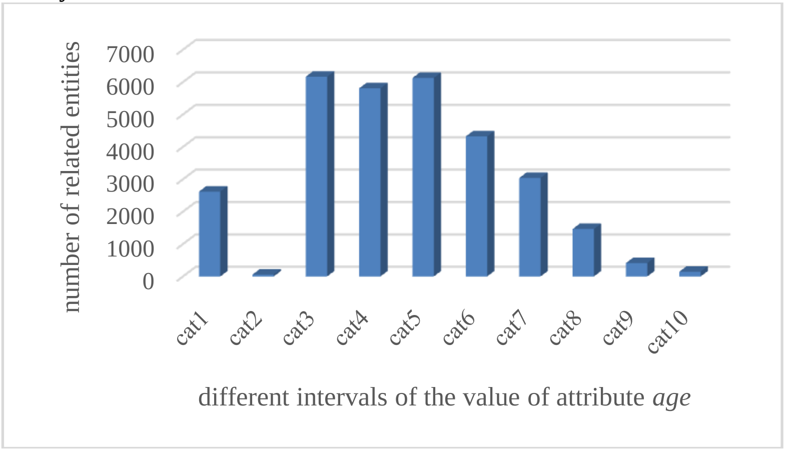


Fig. 9. Number of related entities for each entity with value of attribute *age* discretized into cat1 to cat10 in Adult dataset

7. Conclusion and Future Work

This article proposes a new approach for conflict resolution based on relations between entities, called RelBCR (Relation Based Conflict Resolution). In order to resolve conflicts between entities, fusion methods are used to try to estimate the reliability of data sourcesand then find the true values from among multiple conflicting values. Virtually all previous methods attempt to estimate two parameters: the *truthfulness* of data and the *trustworthiness* of sources. These methods prove however inadequate in some cases: in particular, when there are few reliable sources, and when all data sources don't provide claims about all entities. Using additional information, as our approach does, proves very effective. While previous studies perform only at the entity level or consider the similarity of entities as a relation between entities, our RelBCR approach uses machine learning methods to draw inferences about relations. It consists of two main parts: The G-model and the F-model.

In the G-model, when there are no predefined relations, a usable relation schema is first constructed. Next, a metadata set is created that contains the attributes of two entities and the specific relations between them, instead of only the attributes of one entity. Furthermore, in this phase there is a classifier that learns relations using the metadata. The output of the G-model is a relation tensor. In the F-model, based on defined fusion functions, the true value of the left-hand entity in the relation tensor is estimated.

The results of our experiments contain three major findings. First, relation types can be obtained using datasets that contain only entities and attributes. Second, in order to estimate correct values, using classifiers to infer the existence or absence of relations is more effective than predicting attribute values. Finally, the accuracy of the output data can be improved over other current solutions by using additional information inferred by learning methods. In order to apply this method successfully, there are a few requirements. First, we need to create a metadata set with sufficient positive and negative instances. To achieve this, a clean training data set is necessary. Second, there must be relevant entities for each entity in the dataset. We should also point out to some disadvantages of our approach. First, if the classifier has low negative predictive value (i.e. there is no relationship between the entities but the classifier predicts relationship between them), then the accuracy of our approach will decrease. Because the wrong values in the process of fusion replace the correct values based on the wrong related entities. Second, when almost all data sources provide wrong values about all attributes, the precision of G-model decreases and then the accuracy of our approach decreases.

As regards suggestions for future work, we believe that RelBCR could be strengthened in the following directions:

- **Adaptive entity selection for metadata creation**: We created metadata as a training set for classifiers in order to learn models for relation prediction. If we have 100 entities in the primary data set, we will have 100*100 = 10000 entity pairs in the metadata set. In other words, we have to deal with a very large amount of data. A system for adaptive entity selection that produces a smaller but still informative metadata set could be a useful enhancement.
- **Using latent feature model for relation estimation**:In this article we use classification methods to predict relations, and applied observable patterns to

extract relations. In other words, we used the attributes of pair entities to estimate relations. We can also use some methods that explain relations via latent features of entities (Embedding networks [12, 24] and RBM [31] are examples of such methods).

- **Using weighted multi-relations in the F-model:** To increase the precision of the classifier in relation prediction, we used multi-relations all of which have the same effect. Using a more varied and flexible approach to select the degree of contribution of each relation in truth discovery could produce better results.

References

1. Dong, X. L., Naumann, F. Data fusion: resolving data conflicts for integration. Proceedings of the VLDB Endowment 2, no. 2, 1654-1655. (2009)
2. Foo, P. H., Ng, G. W. High-level information fusion: An overview.J. Adv. Inf. Fusion 8, no. 1, 33-72. (2013)
3. Zhao, B.,Rubinstein,B. IP., Gemmell, J., Han, J.A bayesian approach to discovering truth from conflicting sources for data integration. Proceedings of the VLDB Endowment 5, no. 6, 550-561. (2012)
4. Yin, X., Han, J., Philip, S. Y. Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering 20, no. 6,796-808. (2008)
5. Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., Han, J. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 1187-1198. (2014)
6. Meng, C., Jiang, W., Li, Y., Gao, J., Su, L., Ding, H., Cheng, Y. Truth discovery on crowd sensing of correlated entities. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems,169-182. (2015)
7. Liu, W., Liu, J., Wei, B., Duan, H., Hu, W. A new truth discovery method for resolving object conflicts over Linked Data with scale-free property. Knowledge and Information Systems, 1-31. (2018)
8. Li, Q., Li, Y., Gao, J., Su, L., Zhao, B., Demirbas, M., Fan, W., Han, J. A confidence-aware approach for truth discovery on long-tail data. Proceedings of the VLDB Endowment 8, no. 4, 425-436. (2014)
9. Ye, C., Wang, H., Ma, T., Gao, J., Zhang, H., Li, J.PatternFinder: Pattern discovery for truth discovery. Knowledge-Based Systems 176,97-109. (2019)
10. Li, Y., Gao, J., Meng, C., Li, Q., Su, L., Zhao, B., Fan, W., Han, J. A survey on truth discovery. ACM Sigkdd Explorations Newsletter 17, no. 2, 1-16. (2016)
11. Snidaro, L., Visentini, I., Llinas, J., Foresti, G. L. Context in fusion: some considerations in a JDL perspective. In Information Fusion (FUSION), 2013 16th International Conference on, IEEE, 115-120. (2013)
12. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In AAAI, vol. 15, 2181-2187. (2015)
13. Bordes, A., Glorot, X., Weston, J., Bengio, Y. A semantic matching energy function for learning with multi-relational data. Machine Learning 94, no. 2, 233-259. (2014)
14. Bleiholder, J., Naumann, F.Data fusion.ACM Computing Surveys (CSUR) 41, no. 1, 1.(2009)
15. Li, X., Dong, X. L., Lyons, K., Meng, W., Srivastava, D. Truth finding on the deep web: Is the problem solved? In Proceedings of the VLDB Endowment, vol. 6, no. 2, VLDB Endowment, pp. 97-108. (2012)

16. Rekatsinas, T., Joglekar, M., Garcia-Molina, H., Parameswaran, A.,Ré, C. Slimfast: Guaranteed results for data fusion and source reliability. In Proceedings of the 2017 ACM International Conference on Management of Data,1399-1414. (2017)
17. Yin, X., Tan, W.Semi-supervised truth discovery. In Proceedings of the 20th international conference on World wide web, ACM, 217-226. (2011)
18. Pochampally, R., Das Sarma, A., Dong, X.L., Meliou, A., Srivastava, D. Fusing data with correlations.In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, ACM, 433-444. (2014)
19. Nakhaei, Z., Ahmadi, A. Unsupervised Deep Learning for Conflict Resolution in Big Data Analysis. In International Congress on High-Performance Computing and Big Data Analysis, Springer, Cham, pp. 41-52. (2019)
20. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104, no. 1, 11-33. (2016)
21. Kemp, C., Griffiths, T.L., Tenenbaum, J.B.Discovering Latent Classes in Relational Data.(2004)
22. Airoldi, E., Blei, D., Xing, E., Fienberg, S.A latent mixed membership model for relational data. In Proceedings of the 3rd international workshop on Link discovery, ACM, 82-89. (2005)
23. Hoff, P. Modeling homophily and stochastic equivalence in symmetric relational data. In Advances in neural information processing systems, 657-664. (2008)
24. Yang, B., Yih, W.T., He, X., Gao, J., Deng, L. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575. (2014)
25. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.Translating embeddings for modeling multi-relational data.In Advances in neural information processing systems, 2787-2795. (2013)
26. Bordes, A., Weston, J., Collobert, R., Bengio, Y.Learning Structured Embeddings of Knowledge Bases. In AAAI, vol. 6, no. 1, p. 6. (2011)
27. Thabtah, F., Cowling, P.,Peng, Y. MCAR: multi-class classification based on association rule. In Computer Systems and Applications, The 3rd ACS/IEEE International Conference, p. 33. (2005)
28. Socher, R., Chen, D., Manning, C.D., Ng, A.Reasoning with neural tensor networks for knowledge base completion.In Advances in neural information processing systems, 926-934.(2013)
29. Richardson, M., Domingos, P. Markov logic networks.Machine learning 62, no. 1-2, 107-136. (2006)
30. Nakhaei, Z., Ahmadi, A. Toward high-level data fusion for conflict resolution. In Machine Learning and Cybernetics (ICMLC), 2017 International Conference on, vol. 1, IEEE, 91-97. (2017)
31. Ge, L., Gao, J., Li, X., Zhang, A. Multi-source deep learning for information trustworthiness estimation." In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 766-774. (2013)

Zeinab Nakhaei received the BS degree in software computer engineering from the AmirKabir University of Technology in 2006 and the MS degree in Artificial Intelligence and Robotics from the Iran University of Science andTechnology in 2010. She is currently pursuing the Ph.D. degree with Science and Research Branch ofIslamic Azad University, Tehran, Iran; she is also a lecturer in the Electrical and ComputerEngineering, Islamic Azad University, Tehran, Iran. Her research interests include data integration and data fusion.

Ali Ahmadireceived the Ph.D. in Computer & System Sciences, Majority of Image Processing and Neuralnetworks, University of Osaka Prefecture, Osaka, Japan, March 2004, the MS in Computer & SystemSciences, Majority of Image Processing and Neural Networks, University of Osaka Prefecture, Osaka,Japan, March 2001, and BS in Electrical Engineering, Amirkabir University of Technology, Tehran, Iran,in Sep. 1990. He is currently an associate professor in the Computer Engineering Department, K. N. Toosi University of Technology, Tehran, Iran. His research interests include Semantic data mining andInformation fusion and Interactive learning models.

Arash Sharifi received the Ph.D. in Artificial Intelligence from the Science and Research Branch ofIslamic Azad University, Tehran, Iran, in 2011.He is currently an assistant professor in the Computer and Electronics Department, Science and Research Branch ofIslamic Azad University, Tehran, Iran. His research interests include machine learning, deep learning and data science.

Kambiz Badie received all his degrees from Tokyo Institute of Technology, Japan, majoring in pattern recognition. He has been actively involved in doing research in a variety of issues, such as machine learning, cognitive modeling, knowledge processing & creation in general, and analogical knowledge processing. At present, he is a member of scientific board of IT Research Faculty (Full Professor) at ICT Research Institute, an adjunct professor at Faculty of Engineering Science in the University of Tehran, and in the meantime, the editor-in-chief of International Journal of Information & Communication Technology Research (IJICTR) being published periodically by ICT Research Institute and also an invited member of Iranian Academy of Science.

Received: January 31, 2020; Accepted: April 02, 2021.