International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2023, 13(1): 27–55, doi: 10.21655/ijsi.1673-7288.00295
©2023 by Institute of Software, Chinese Academy of Sciences. All rights reserved.

Research Article

Cross-source Data Error Detection Approach Based on Federated Learning

Lu Chen (陈璐)[1], Yuxiang Guo (郭宇翔)[1], Congcong Ge (葛丛丛)[2], Baihua Zheng (郑白桦)[3], Yunjun Gao (高云君)[1]

[1] (College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)
[2] (Data Intelligence Innovation Lab, Huawei Cloud Computing Technologies Co. Ltd., Hangzhou 310052, China)
[3] (School of Computing and Information Systems, Singapore Management University, 178902, Singapore)
Corresponding author: Yunjun Gao, gaoyj@zju.edu.cn

Abstract With the emergence and accumulation of massive data, data governance has become an important manner to improve data quality and maximize data value. Specifically, data error detection is a crucial step to improve data quality, which has attracted wide attention from both industry and academia. At present, various detection methods tailored for a single data source have been proposed. However, in many real-world scenarios, data are not centrally stored or managed. Data from different sources but highly correlated can be employed to improve the accuracy of error detection. Unfortunately, due to privacy/security issues, cross-source data are often not allowed to be integrated centrally. To this end, this paper proposes FeLeDetect, a cross-source data error detection method based on federated learning, so as to improve the error detection accuracy by using cross-source data information on the premise of data privacy. First, a Graph-based Error Detection Model, namely GEDM, is presented to capture sufficient data features from each data source. On this basis, the paper then designs a federated co-training algorithm, namely FCTA, to collaboratively train GEDM by using different cross-source data without privacy leakage of data. Furthermore, the paper designs a series of optimization methods to reduce communication costs during federated learning and manual labeling efforts. Finally, extensive experiments on three real-world datasets demonstrate that (1) GEDM achieves an average improvement of 10.3% and 25.2% in terms of the $F1$ score in the local and centralized scenarios, respectively, outperforming all the five existing state-of-the-art methods for error detection; (2) the $F1$ score of the error detection by FeLeDetect is 23.2% on average higher than that by GEDM in the local scenario.

Keywords data governance; data quality; error detection; federated learning

Citation Chen L, Guo YX, Ge CC, Zheng BH, Gao YJ. Cross-source data error detection approach based on federated learning, *International Journal of Software and Informatics*, 2023, 13(1): 27–55. http://www.ijsi.org/1673-7288/295.htm

As mobile devices, Internet of Things devices spread far and wide, and Internet technologies develop rapidly, massive data constantly surge and are accumulated. Analyzing and mining

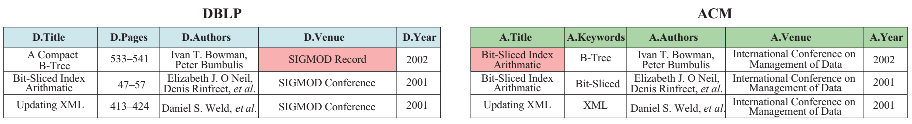
This is the English version of Chinese article "基于联邦学习的跨源数据错误检测方法. 软件学报, 2023, 34(3): 1126–1147. doi: 10.13328/j.cnki.jos.006781"

Funding items: National Key Research and Development Program of China (2021YFC330030); National Natural Science Foundation of China (62025206, 61972338, 62102351)
Received 2022-05-13; Revised 2022-07-29, 2022-09-07; Accepted 2022-09-23; IJSI published online 2023-03-30

massive data are essential to reveal potential data value. Error detection is often the first step[1, 2] of data analysis because data errors will seriously affect data quality[3], mislead downstream decision analysis, and even cause serious economic loss to enterprises[4]. In recent years, data error detection has attracted much attention from academic and industrial circles and has been accordingly studied in depth.

Data errors are attributed to multiple factors, such as typos resulting from manual keying and inconsistencies caused by integrating data from different sources. Common types of errors include typos, missing data, format errors, and violation of data consistency. In real-world scenarios, data errors tend to be heterogeneous and sparse[4], which makes error detection even more difficult. Error detection methods comprise (1) outlier detection[5–7], which statistically detects error values that significantly deviate from the overall distribution of data; (2) duplicate value detection[8, 9], which eliminates duplicate values by identifying tuples pointing to the same entity; (3) format error detection[10], which detects format errors through predefined patterns (such as the date format 2000/01/01); (4) rule violation detection[11, 12], which detects data errors that violate rules through given data rules; and (5) external knowledge violation detection, which employs external knowledge (such as master data[13] and Knowledge-Based systems (KBs)[14]) to detect data errors. The above methods are tailored for a single data source. However, in real-world scenarios, even data describing the same set of entities are seldom centrally stored or managed. Underpinned by cross-source data information, some errors that are difficult to be identified only by relying on single-source information can be more easily detected.

For better understanding, an example is shown in Figure 1. Two tables from different data sources (DBLP and ACM) show the information of some conferences and papers in the field of computer science. Tuples in each table that have the same row describe the information of the same paper. There are two data cells with substitution errors (SEs), highlighted as E_D and E_A. One is in the first tuple in the DBLP table (the paper's correct publication information is "SIGMOD Conference" rather than "SIGMOD Record"), and the other is in the first tuple in the ACM table (the paper's correct title is "A Compact B-Tree" rather than "Bit-Sliced Index Arithmetic"). It is difficult to detect the error E_D by the DBLP table alone. Likewise, it is difficult to identify the error E_A only by the ACM table. In other words, the above error detection methods tailored for single-source data have difficulty in detecting these two errors. (1) Both "SIGMOD Record" and "Bit-Sliced Index Arithmetic" are valid values in the attribute domain of their tables and cannot be regarded as outliers; (2) there are no duplicate records in the two tables, so it is impossible to use duplicate records to detect and identify errors; (3) the values of the two errors are correct in syntax and semantics, and there is no Format Issue (FI); (4) the two errors cannot be captured by data rules in a single table; (5) there is no master data or external knowledge about the two tables, and the building of the master database or external KBs entails substantial labor costs and computing resources. Therefore, it is difficult to use external knowledge to conduct error detection.



DBLP

D.Title	D.Pages	D.Authors	D.Venue	D.Year
A Compact B-Tree	533–541	Ivan T. Bowman, Peter Bumbulis	SIGMOD Record	2002
Bit-Sliced Index Arithmatic	47–57	Elizabeth J. O Neil, Denis Rinfreet, *et al.*	SIGMOD Conference	2001
Updating XML	413–424	Daniel S. Weld, *et al.*	SIGMOD Conference	2001

ACM

A.Title	A.Keywords	A.Authors	A.Venue	A.Year
Bit-Sliced Index Arithmatic	B-Tree	Ivan T. Bowman, Peter Bumbulis	International Conference on Management of Data	2002
Bit-Sliced Index Arithmatic	Bit-Sliced	Elizabeth J. O Neil, Denis Rinfreet, *et al.*	International Conference on Management of Data	2001
Updating XML	XML	Daniel S. Weld, *et al.*	International Conference on Management of Data	2001

Figure 1 Cross-source dataset example

However, if DBLP and ACM are transferred to the same data center and integrated into a centralized dataset (Figure 2), the difficulty of error detection will be greatly reduced. Although these two tables come from different data sources (DBLP and ACM), they describe the information of the same set of entities (academic papers). It is found that for the same

DBLP-ACM

D.Title	D.pages	D.Authors	D.Venue	D.Year	A.Title	A.Keywords	A.Authors	A.Venue	A.Year
A Compact B-Tree	533–541	Ivan T. Bowman, Peter Bumbulis	SIGMOD Record	2002	Bit-Sliced Index Arithmatic	B-Tree	Ivan T. Bowman, Peter Bumbulis	International Conference on Management of Data	2002
Bit-Sliced Index Arithmatic	47–57	Elizabeth J. O Neil, Denis Rinfreet, *et al.*	SIGMOD Conference	2001	Bit-Sliced Index Arithmatic	Bit-Sliced	Elizabeth J. O Neil, Denis Rinfreet, *et al.*	International Conference on Management of Data	2001
Updating XML	413–424	Daniel S. Weld, *et al.*	SIGMOD Conference	2001	Updating XML	XML	Daniel S. Weld, *et al.*	International Conference on Management of Data	2001

Figure 2 Centralized dataset

paper, the value of D.Venue (SIGMOD Conference) and the value of A.Venue (International Conference on Management of Data) frequently co-occur. Therefore, it can be inferred that they refer to the same academic conference. However, the Venue value of the first tuple in DBLP and ACM breaks such high-frequency co-occurrence law. Therefore, it is easy to infer that the value of at least one data cell is erroneous. Since data errors are not randomly distributed[15], the error distribution in the dataset can be learned to predict which data cell is more likely to have an error value, so as to make more accurate inferences. For example, in this dataset, there is a "SIGMOD Conference" that is similar to the valid value of the "SIGMOD Record" in the attribute domain of the D.Venue. However, in the attribute domain of A.Venue, there is no value similar to "International Conference on Management of Data". Therefore, relatively speaking, "SIGMOD Record" and "SIGMOD Conference" are more likely to be mixed up with each other. If the data errors in the training set conform to the above distribution law, the model will learn the knowledge and apply it to the detection set. Based on this, this paper studies the cross-source data error detection method and aims to answer the following three major questions.

(1) **How to represent the features of data with different granularity?** As mentioned above, due to the heterogeneity of data errors in real-world scenarios, it is difficult to represent them in a standard way. An effective way to deal with error heterogeneity is to use machine learning to treat error detection as a binary classification problem. In other words, through a given dataset and some training labels, this paper can predict whether the value of each data cell is erroneous by learning the features of the erroneous data cells and the correct data cells. So, how to effectively represent the features of data cells with different granularity is critical to precise error detection.

(2) **How to ensure data privacy in cross-source error detection?** As shown in Figure 2, cross-source data can effectively improve the quality of error detection. However, for the concern of data privacy and security (such as the *General Data Protection Regulation* promulgated in Europe[16]), in real-world scenarios, data from different sources are often not permitted to be transmitted to a public data center for integration. Therefore, it is urgent to study a cross-source error detection method, which can not only effectively use cross-source data information to improve the quality of error detection but also guarantee the privacy and security of data from different sources.

(3) **How to reduce the communication costs of cross-source data error detection?** During cross-source error detection, different data sources need to exchange information frequently to obtain the necessary cross-source information. The incurred communication costs cannot be ignored. Therefore, how to minimize the communication costs arising in the cross-source error detection process while ensuring the precision of error detection remains a big challenge.

To this end, this paper proposes FeLeDetect, a cross-source detection method based on federated learning. First of all, considering that the graph structure can effectively represent the features of relational data[17], this paper designs a Graph-based Error Detection Model (GEDM) to capture the features of different granularity of each data source (including value-level features, attribute-level features, and tuple-level features), so as to provide precise classification signals for the classifier. GEDM can effectively capture the data features of the input dataset.

Nevertheless, when the input dataset is a vertically distributed subset (such as DBLP in Figure 1), the performance of GEDM is affected due to the inability to access another related dataset (such as ACM in Figure 1). Given that the federated learning technique[18]can collaboratively train the machine learning model while guaranteeing data privacy, this paper proposes an information-lossless federated co-training algorithm based on GEDM, namely FCTA, so as to collaboratively train GEDM from different data sources. While guaranteeing the privacy and security of cross-source data, FCTA is able to capture cross-source data features to achieve an effect close to running GEDM on a joint dataset (Figure 2). Finally, to minimize the communication costs of federated training, this paper further proposes three optimization strategies: (1) data de-duplication, which is to ensure that there is no duplicate information in the exchange of data from different sources; (2) quantization, which is to reduce the number of bytes required for data coding by quantizing continuous data; (3) frequency-reduced transmission, which is to reduce the frequency of data exchange by performing filter optimization according to the similarity threshold.

This paper comes up with the following four major contributions.

(1) This paper proposes FeLeDetect, a cross-source data error detection method based on federated learning. To the best of our knowledge, it is the first attempt to detect errors via different data sources on the premise of data privacy preservation.

(2) A Graph-based Error Detection Model (GEDM) is presented, which is capable of capturing the rich features of data of each source with different granularity for ensuring high-quality error detection results.

(3) This paper proposes an information-lossless federated co-training algorithm, namely FCTA, which collaboratively trains GEDM from different data sources to guarantee the privacy of cross-source data. Furthermore, several optimization methods are presented to reduce communication costs during federated training.

(4) The results of experiments conducted on three real datasets show that GEDM and FeLeDetect can effectively improve the effectiveness of error detection, compared with five existing advanced error detection methods.

1 Related Work

This section outlines related work. Section 1.1 reviews the work on error detection; Section 1.2 describes the work related to federated learning.

1.1 Error detection

Data error detection is a crucial step to improve data quality. Traditional error detection methods can be divided into quantitative and qualitative methods[5–7]. The quantitative method identifies data errors with the expected statistical distribution of data, while the qualitative method detects data errors through rules[11, 12], patterns[10, 19], or external knowledge[13, 14]. However, due to the heterogeneity of errors, it is difficult for these methods to detect multiple types of errors in relational tables, which results in relatively low recall of error detection. In recent years, the error detection method based on machine learning/deep learning technology[2, 4, 20–23] has attracted extensive attention, and it aims to capture the features of data errors through learning. The method uses training labels to guide the classifier to make more precise judgments, thus addressing the challenge of error heterogeneity. Of all these error detection methods, HoloDetect[2] and Raha[20] have shown the best error detection effect. HoloDetect[2] detects erroneous data cells by representing the intrinsic semantic and syntactic features of the data and following the data consistency rules given by humans. Raha[20] systematically generates a set of configuration parameters for an error detection algorithm and encodes the output of the algorithm into an eigenvector for each data cell to learn the features of data errors. Raha achieves error detection results equivalent to those of HoloDetect. However, the process of configuring and running various error detection algorithms by Raha is time-consuming. Compared with Raha, the GEDM proposed in this paper can fully capture the features of data with different granularity and is more efficient. This will be verified in the experimental part in Section 5.4.

1.2 Federated learning

Federated learning[24] is a distributed machine learning technique that allows different data participants to work together to build machine learning models without disclosing their original data. So far, federated learning has been applied to many fields, such as intelligent recommendation[25], intelligent medical care[26], and financial crime detection[27]. In this context, some federated learning frameworks have been proposed, such as FedATT[28], FedProx[29], and FedKD[30]. These computational frameworks are primarily oriented toward horizontal federated learning (i.e., horizontal data distribution). This paper is the first one to apply the vertical federated learning technique to data governance. It uses cross-source data information to improve the precision of error detection, thereby improving data quality. Since federated learning entails frequent data exchange among data participants in the process of model training, how to minimize the high communication cost caused by frequent data exchange during federated training is a research focus of federated learning[31–33]. After proposing FeLeDetect, this paper further designs several effective communication optimization schemes to minimize the high communication traffic in the collaborative training process of the cross-source GEDM at different levels. These optimization strategies greatly reduce the communication costs of FeLeDetect while ensuring the effectiveness of error detection.

2 Basic Knowledge

This section introduces the basic knowledge of the work in this paper. Section 2.1 introduces the fundamental concepts of data error and error detection; Section 2.2 introduces knowledge relevant to federated learning; Section 2.3 gives the specific definitions of the problem studied in this paper.

2.1 Data error and error detection

A relational table containing erroneous data, namely $D = \{t_1, t_2, \cdots, t_n\}$, and its attribute set $A = \{a_1, a_2, \cdots, a_n\}$ are given. Specifically, each tuple (record) $t_i \in D$ is a set of data cells $\{t_i[a_1], t_i[a_1], \cdots, t_i[a_m]\}$. Each tuple describes a real-word entity. The value of a data cell $t_i[a_j]$ in the table D is registered as v_{ij}, which is used to describe the features of entity t_i in a certain feature dimension a_j. The actual value of the data cell $t_i[a_j]$ is denoted as v^*_{ij}. If $v_{ij} \neq v^*_{ij}$, then the data cell is called an erroneous data cell, and v_{ij} is a data error. For example, Figure 1 shows two relational tables, DBLP and ACM (abbreviated as D and D', respectively). Each relational table contains three tuples. Since the actual values of data cells $t_1[a_4]$ and $t'_1[a_1]$ are "SIGMOD Conference" and "A Compact B-Tree", both of which are not equal to their values in D and D', data cells $t_1[a_4]$ and $t'_1[a_1]$ are both erroneous data cells.

The purpose of error detection is to identify all erroneous data in the relational table, namely $t_1[a_4]$ and $t'_1[a_1]$ in Figure 1. In this paper, error detection is considered as a binary classification problem. In other words, a relational table D containing erroneous data and some training labels L are given. The error detection aims to assign a classification label $\hat{y} \in \{0, 1\}$ to each data cell $t_i[a_j]$. Specifically, $\hat{y} = 1$ denotes that the cell is an erroneous data cell, while $\hat{y} = 0$ denotes that the cell is a correct data cell.

2.2 Federated learning

In the real world, data possessed by different organizations/agencies are often defined and managed separately. Data from different sources form isolated islands, making it difficult for data to circulate and be shared. Due to competition among industries, privacy, security, and complex management mechanisms, even data integration between different departments of the same company faces strong resistance[34]. Therefore, it becomes more difficult to integrate data across agencies and organizations. In response to this challenge, Google proposed the concept of federated learning[18]. It is a distributed machine learning technique, which aims to build a machine learning model by using data distributed in different sites while guaranteeing the privacy and security of cross-source data. Therefore, the federated learning technique can be employed to effectively use cross-source data without violating privacy. According to the feature space of data participants and the entity space described, federated learning techniques can be divided into three categories, namely horizontal federated learning, vertical federated learning, and transfer learning[34]. Horizontal federated learning can be applied to scenarios where data participants have the same feature space but different entity spaces (the data are divided horizontally). Vertical federated learning is applicable to the case where data participants have the same entity space but different feature spaces (the data are divided vertically). Transfer learning focuses on scenarios where data participants have both different entity spaces and feature spaces.

The cross-source error detection studied in this paper is highly correlated to the scenario of vertical federated learning. As Figure 1 shows, (1) DBLP and ACM describe the information of the same group of entities (conferences and journal papers); (2) the local datasets DBLP and ACM possessed by each party in Figure 1 can be regarded as those vertically divided from the joint dataset DBLP-ACM in Figure 2. One of the most straightforward methods to carry out the cross source data error detection is to first transfer the data D and D' owned by the data holders F and F' to a public data center and make them integrate as a joint dataset $D_S = D \cup D'$. After that, upon the dataset D_S, a machine learning model M_S is built. However, as mentioned above, in view of privacy and security, such an integrated modeling pattern usually does not work in real-world scenarios. In contrast, cross-source error detection based on vertical federated learning aims to use datasets D and D' to jointly build an error model M_F and ensure no privacy leakage of cross-source data. Therefore, there is no need to transfer and integrate centrally the original data from different sources, so as to ensure the privacy and security of cross-source data. In addition, the method based on federated learning requires that the effect of the federated model M_F should be close to that of the model M_S established in the data's centralized scenario.

2.3 Problem definition

This paper focuses on the error detection problem for relational data. On the one hand, cross-source data can improve the quality of error detection. As shown in Figure 1, it is difficult for an existing error detection method tailored for single-source data to effectively detect the SEs in Figure 1. However, with cross-source data information, the precision of error detection can be improved. Therefore, in line with the general assumptions made by relevant research work[35, 36], this paper assumes that different data holders are willing to participate in cross-source error detection to improve the quality of their own data. On the other hand, data privacy is a global urgent issue to be addressed. Therefore, this paper studies cross-source data error detection considering privacy protection.

Definition 1 (cross-source data error detection under privacy protection)**.** Two relational tables D and D' containing erroneous data are given. They are held by different data participants F and F' separately. (Although the definition only involves two data holders, the method proposed in this paper can extend to multiple data holders. However, in real-world vertical federated scenarios, owing to complicated management and benefit distribution mechanisms, it is rare that data are held by multiple parties[35]). For the sake of data privacy and security, both parties do not accept privacy leakage of original data. A cross-source data error detection method considering privacy protection aims to integrate information of cross-source data D and D' and detect all errors in data D and D' while ensuring the data privacy and security of both parties.

Since cross-source data are not allowed to be integrated centrally, this paper uses the federated learning technique to carry out cross-source data error detection. As mentioned above, the data held by both parties are vertically divided by the joint dataset D_S. Therefore, it is a problem of vertical federated learning. A typical vertical federated learning system mainly consists of two parts, namely encrypted entity alignment and federated learning[34]. Encrypted entity alignment aims to use existing techniques to identify records in two data sources that point to the same entity[35]. This paper follows the general assumption of vertical federated learning research that the records of the two datasets have been completely aligned, or in other words, the encrypted entity alignment has been completed, and only the federated learning phase[36, 37] is to be considered. For example, the relational tables DBLP and ACM in Figure 1 have been completely aligned, and the tuples with the same ID describe the information of the same paper.

3 Graph-based Error Detection Model (GEDM)

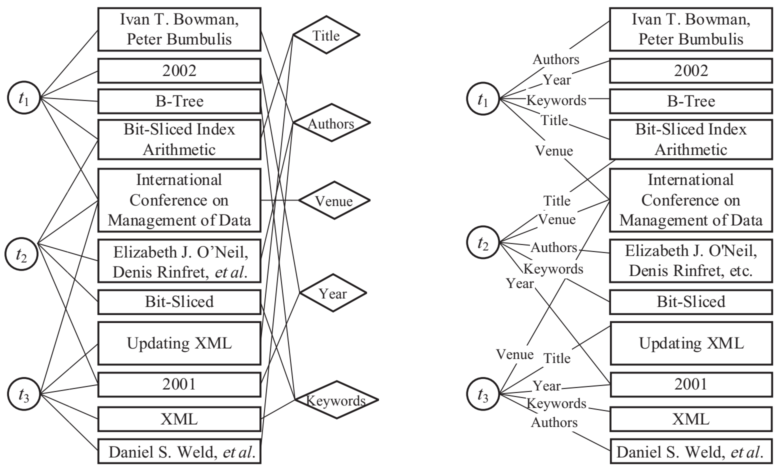
This section describes GEDM. Section 3.1 introduces the construction of a Multi-Relational Graph (MRG) model; Section 3.2 proposes GEDM, which can capture rich features of different granularity in each data source.

3.1 Multi-relational graph construction

Recent studies have shown[17] that representing relational data as a graph structure is conducive to capturing the intrinsic features of data. In view of this, this paper first transforms relational data into a graph and then models it. The EMBDI model proposed by Cappuzzo *et al.*[17] models each tuple (row), each attribute (column), and the value of each data cell in a relational table as three types of nodes in the graph. These nodes are connected according to the logical relationship described by the relational table. Figure 3(a) shows an EMBDI model based on the ACM dataset in Figure 1. Nodes denoted by rectangles (e.g., "Ivan T. Bowman, Peter Bumbulis") correspond to a value of a data cell in the table ACM, and the circular node (e.g., t_1) corresponds to the first tuple in the table ACM, and the diamond-shaped node (e.g., "Title") corresponds to the Title attribute in the table ACM. However, this graph-based model is subjected to two problems.

- First, using this model to transform and model relational tables will produce large-scale graphs with complex structures, which contain a large number of nodes and edges. Storing large graphs requires a lot of storage resources, and it is a great challenge for graph-based training in the follow-up process.
- Second, the graph-based model does not consider the semantic information of edges. For example, an edge connecting a tuple node to an attribute node is not semantically identical to an edge connecting an attribute node to a value node, but the graph-based model does not distinguish these different semantic edges.

In order to overcome these two deficiencies, this paper uses the MRG model[38] to transform a relational table into a graph. The MRG model is a bipartite graph, and there are two types of nodes, namely tuple-level nodes (denoted by t) and value-level nodes (denoted by v). These two types of nodes are the same as the tuple node and the value node defined in the tripartite graph model. In contrast, the MRG model introduces the concept of an attribute-level edge (denoted as a) to connect the t node and the v node. Specifically, in the original relational table, if the value of the tuple t on the attribute a is v, then the node t and the node v are connected by an attribute-level edge a in the corresponding MRG. Figure 3(b) depicts an example of the transformation of the ACM dataset in Figure 1 in accordance with the MRG model. For example, the edge represented by the attribute "year" connects the tuple-level node t_1 and the value node "2002", and its corresponding semantics is as follows: the publication year of the paper described by the first tuple in the ACM dataset is 2002. Through the comparison with Figure 3(a) and Figure 3(b), it can be seen that for the same relational table, the number of edges and nodes of the graph constructed by MRG is reduced compared with the EMBDI model. In particular, the number of edges is reduced from $2mn$ to mn, where m denotes the number of attributes (columns) in the original relational table, and n denotes the number of tuples (rows) in the original relational table. Moreover, each edge in the MRG model has its semantic information (attribute name), which helps represent the semantic relationship between different nodes.


Ivan T. Bowman, Peter Bumbulis
2002
B-Tree
Bit-Sliced Index Arithmetic
International Conference on Management of Data
Elizabeth J. O'Neil, Denis Rinfret, et al.
Bit-Sliced
Updating XML
2001
XML
Daniel S. Weld, et al.
t_1
t_2
t_3
Title
Authors
Venue
Year
Keywords
Ivan T. Bowman, Peter Bumbulis
2002
B-Tree
Bit-Sliced Index Arithmetic
International Conference on Management of Data
Elizabeth J. O'Neil, Denis Rinfret, etc.
Bit-Sliced
Updating XML
2001
XML
Daniel S. Weld, et al.
Authors
Year
Keywords
Title
Venue
Title
Venue
Authors
Keywords
Year
Venue
Title
Year
Keywords
Authors

(a) EMBDI (b) MRG

Figure 3 Example of comparison of EMBDI and MRG model construction

3.2 Model design

Figure 4 shows the GEDM framework. GEDM consists of three parts: (1) graph construction; (2) graph-based feature extraction; (3) binary classifier. Specifically, in the model training (or error detection) phase, GEDM first transforms the training dataset T (or the complete dataset D) into an MRG. Then, the Graph Neural Network (GNN) is used to extract data features of three different dimensions, namely tuple-level features, attribute-level features, and value-level features. Finally, a binary classifier is used to judge whether the value of each data cell is erroneous. Since the process of constructing MRG has been introduced in Section 3.1, only the graph-based feature extraction and binary classifier are introduced below.

3.2.1 *Graph-based feature learning*

After the data table is transformed into a graph, each node and each edge in the graph need to be represented by features to capture the rich data features in the original data table. In recent years, GNNs[39] have received much attention. Because of their powerful feature capture ability, GNNs have received extensive attention in many fields[40–42]. To this end, GEDM uses GNNs to capture features to support high-precision error detection. Given an MRG G, GEDM

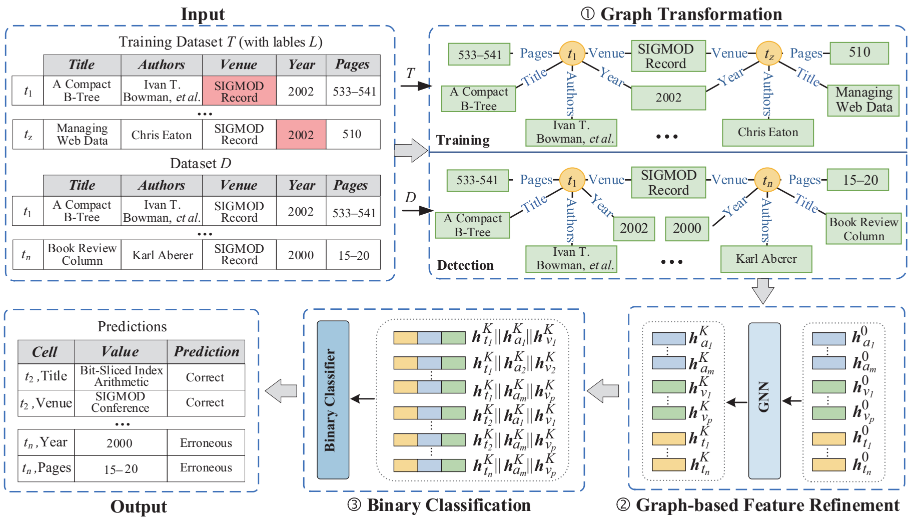


Figure 4 Overview of GEDM

first randomly initializes the initial embeddings of tuple-level nodes, value-level nodes, and attribute-level edges in the graph, and marks them as $\boldsymbol{h}_t^0$, $\boldsymbol{h}_v^0$, and $\boldsymbol{h}_a^0$, respectively. Then, the proposed Multi-Granularity Graph Convolutional Neural network (MGGCN) is used to pdate the embeddings to learn the final representation of the vectors containing data features of different granularity. Each convolution layer of the MGGCN is designed with convolution operations for tuple-level nodes, value-level nodes, and attribute-level edges, so as to aggregate the features of neighbor nodes on the graph and fully capture the graph structure information. Similar to that in Ref. [43], MGGCN adopts inductive learning. In the training process, it uses the training set T instead of the complete dataset D to achieve higher efficiency. Specifically, the purpose of model training is to obtain the optimal model parameters through learning rather than obtain the feature representation of each node and edge on the graph[43]. Once the model training is completed, the final features of each data cell in the entire dataset D can be quickly represented by the trained model in the error detection phase. Next, the convolution operation of each layer in the proposed MGGCN is introduced in detail.

(1) Tuple-level node convolution operation: At the k-th ($k \geq 1$) layer of the MGGCN, the embedding of the tuple-level node t is calculated by the following formula:

$$\boldsymbol{h}_{\mathbb{N}(t)}^k = AGG_{(a,v)\in\mathbb{N}(t)}\left(\boldsymbol{W}_{ta}^k\boldsymbol{h}_a^{k-1} \odot \boldsymbol{W}_{tv}^k\boldsymbol{h}_v^{k-1}\right) \tag{1}$$

$$\boldsymbol{h}_t^k = \sigma\left(\boldsymbol{W}_t^k\left(\boldsymbol{h}_t^{k-1}\|\boldsymbol{h}_{\mathbb{N}(t)}^k\right)\right) \tag{2}$$

where $\boldsymbol{W}_{ta}^k$, $\boldsymbol{W}_{tv}^k$, and $\boldsymbol{W}_t^k$ denote the weight matrices to be learned; $\boldsymbol{h}_t^{k-1}$, $\boldsymbol{h}_v^{k-1}$, and $\boldsymbol{h}_a^{k-1}$ denote the embeddings calculated by the $(k-1)$-th layer network; $(a,v) \in \mathbb{N}(t)$ denotes the set of attribute-level edges and value-level nodes directly connected to the tuple-level node t; $AGG(\cdot)$ denotes an aggregate function, where the mean function is used in this paper; $\odot$ denotes the Hadamard product; $\sigma(\cdot)$ denotes the activation function which is presented by using the tanh function in this paper; and $\|$ denotes the assembly operator.

(2) Value-level node convolution operation: At the k-th ($k \geq 1$) layer of the MGGCN, the embedding of the value-level node v is calculated by the following formula:

$$\boldsymbol{h}_{\mathbb{N}(v)}^k = AGG_{(a,t)\in\mathbb{N}(v)}\left(\boldsymbol{W}_{va}^k\boldsymbol{h}_a^{k-1} \odot \boldsymbol{W}_{vt}^k\boldsymbol{h}_t^{k-1}\right) \tag{3}$$

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_v^k\left(\boldsymbol{h}_v^{k-1} \| \boldsymbol{h}_{\mathbb{N}(v)}^k\right)\right) \tag{4}$$

where $\boldsymbol{W}_{va}^k$, $\boldsymbol{W}_{vt}^k$, and $\boldsymbol{W}_v^k$ denote the weight matrices to be learned; $\boldsymbol{h}_t^{k-1}$, $\boldsymbol{h}_v^{k-1}$, and $\boldsymbol{h}_a^{k-1}$ denote the embeddings calculated by the $(k-1)$-th layer network; and $(a,t) \in \mathbb{N}(v)$ denotes the set of attribute-level edges and tuple-level nodes directly connected to the value-level node v.

(3) Attribute-level edge convolution operation: At the k-th $(k \geq 1)$ layer of the MGGCN, the embedding of the attribute-level node a is calculated by the following formula:

$$\boldsymbol{h}_a^k = \boldsymbol{W}_a^k \boldsymbol{h}_a^{k-1} \tag{5}$$

where $\boldsymbol{W}_a^k$ denotes the weight matrix to be learned and $\boldsymbol{h}_a^{k-1}$ denotes the embedding calculated by the $(k-1)$-th layer network.

3.2.2 *Binary classifiers*

After the stacking of several layers of the MGGCN, the final feature representation of each tuple/value-level node and attribute-level edge is obtained. For a certain data cell $v = t[a]$ in the original relational data table, the final embeddings, namely $\boldsymbol{h}_t$, $\boldsymbol{h}_a$, and $\boldsymbol{h}_v$ of the corresponding nodes and edges are concatenated to obtain a new vector $\boldsymbol{c} = \boldsymbol{h}_t \| \boldsymbol{h}_a \| \boldsymbol{h}_v$, which is used as the input of the binary classifier C. The binary classifier outputs a class label $\hat{y}$ to indicate whether the value of the data cell is correct or not. The binary classifier C consists of a two-layer fully connected neural network and a Softmax layer. The binary classifier uses the ReLU activation function and selects the cross-entropy loss (denoted as $\mathcal{L}$) as the objective function, which is defined as follows.

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}\mathcal{L}_i = -\frac{1}{M}\sum_{i=1}^{M} y_i \log p_i + (1-y_i)\log(1-p_i) \tag{6}$$

where M denotes the number of samples in the training set; y_i denotes the true label value of the i-th training sample ($y_i = 1$ indicates the erroneous data cell; $y_i = 0$ indicates the correct data cell); and p_i denotes the probability that the training sample is predicted to be an erroneous data. It should be pointed out that the MGGCN described above is trained along with the binary classifier C.

4 Federated Error Detection

This section describes FeLeDetect. First, the framework of FeLeDetect is described. Then, the technical details of FeLeDetect are described. Finally, several optimization methods are proposed to reduce communication costs and manual labeling costs of federated training.

4.1 FeLeDetect framework

Based on the GEDM proposed in Section 3, this section proposes FeLeDetect. Here, it is assumed that two data participants F and F' hold data D and D', respectively. The tuples contained by training sets $T \subset D$ and $T' \subset D'$ have been aligned. As mentioned above, F and F' do not allow privacy leakage of original data. FeLeDetect uses FCTA to collaboratively train GEDM on datasets T and T'. The process involves no exchange of any original data, which guarantees the privacy and security of both parties' data. For convenience, only two data participants are described below, but FeLeDetect can still be extended to scenarios with multiple data participants.

FeLeDetect deploys GEDM locally for data holders F and F', and carries out collaborative training in accordance with FCTA. As described in Section 3.2, the training of GEDM involves

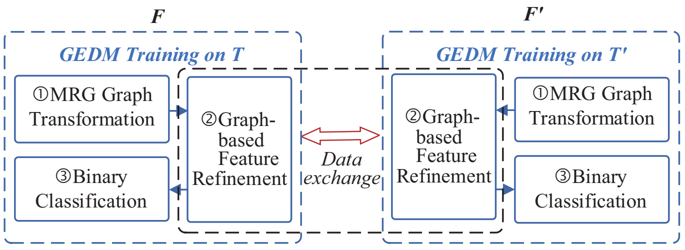

F
F′
GEDM Training on T
GEDM Training on T′
①MRG Graph Transformation
②Graph-based Feature Refinement
③Binary Classification
Data exchange
②Graph-based Feature Refinement
①MRG Graph Transformation
③Binary Classification

Figure 5 Cross-source error detection method FeLeDetect

three steps, namely MRG construction, graph-based feature extraction, and binary classification. The framework of FeLeDetect is shown in Figure 5. First, the data participants transform the relational data into graphs G and G', respectively, by using the MRG model. Then, in the feature learning stage, the two parties follow the FCTA to exchange data and capture the data features of cross-source data so that the features of the nodes and the edges on the graphs G and G' are collaboratively represented. It is worth noting that there is no need for any data/label exchange between the two parties in the binary classification phase. This is because the input of the binary classifier is the feature embeddings that are collaboratively learned. Since the vector has fully captured the features of the dataset of both parties, there is no need to further exchange data.

With the data participant F as an example, Algorithm 1 describes the basic flow in the FeLeDetect training phase.

Algorithm 1. FeLeDetect algorithm (with data participant F as an example).

```
Input: Training set $T$, training set label $L$, depth $K$ of MGGCN, training algebra (Epoch) $N_{ep}$, the
       number of iterations per generation $N_{it}$, label $i$ of data participants responsible for producing
       the seed.
Output: Parameter of the trained model, namely $\Theta_F$.
1.  Use the MRG model to transform the dataset $T$ into a graph $G_F$
2.  if $F$.index $= i$ then
3.      Produce a *seed* and sends it to $F'$
4.  else
5.      Receive the *seed* sent by $F'$
6.  end if
7.  Initialize $\Theta_F$ of the local GEDM
8.  $\boldsymbol{h}_t^0, \boldsymbol{h}_a^0$, and $\boldsymbol{h}_v^0 \leftarrow$ random $(\forall a, v, t \in G_F)$ /* randomly initialize the features of nodes and edges
    in $G_F$ */
9.  for epoch $= 1$ to $N_{ep}$ do
10.     for iter $= 1$ to $N_{it}$ do
11.         for $k = 1$ to $K$ do
12.             Execute the forward propagation of FCTA to obtain the features of the nodes and
                edges in $G_F$, namely $\boldsymbol{h}_t^k$, $\boldsymbol{h}_a^k$, and $\boldsymbol{h}_v^k$
13.         end for
14.         The classifier $C_F$ takes $\boldsymbol{h}_t^k \| \boldsymbol{h}_a^k \| \boldsymbol{h}_v^k$ $(\forall (t, a, v) \in G_F)$ as input and outputs predicted
            label $\hat{y}$
15.         Use the label set $L$ to calculate the loss function and update $\Theta_F$ with back propagation /*
            update the parameter according to the objective function of Eq. (6) */
16.     end for
17. end for
18. return $\Theta_F$
```

The inputs of Algorithm 1 include training dataset T, training label L, depth K of the MGGCN network, maximum training algebra N_{ep}, the number of iterations per generation N_{it} (using the mini-batch training method), and the label i of participants responsible for producing the seed. The algorithm first transforms the dataset T into the corresponding graph G_F (Line 1) and then generates the seed or waits to receive the seed generated by the other party (Lines 2–6), which ensures that both sides use the same *seed* to initialize the parameter Θ_F of their respective local GEDM so that the initialized parameters of both models are the same (Line 7). This is the common setting in federated learning[42]. Θ_F includes MGGCN and parameters to be learned of the binary classifier C_F. Next, the algorithm randomly initializes $\boldsymbol{h}_a^0$ and $\boldsymbol{h}_v^0$, and it randomly initializes $\boldsymbol{h}_t^0$ by using the *seed* (Line 8). It should be noted that different data participants use the same *seed* to initialize $\boldsymbol{h}_t^0$ to ensure that the initial features of the corresponding tuple nodes in each data participant are consistent. This is because the corresponding tuple-level nodes from different data sources represent the same entity (the datasets are aligned). Then, the algorithm performs the training for N_{ep} epochs. The mini-batch training method is used here. In other words, each epoch is composed of N_{it} iterations. Each iteration performs forward propagation calculation on the K-th layer of the MGGCN by using the FCTA to obtain the output of the network (Lines 11–13). Next, $\boldsymbol{h}_t^k \| \boldsymbol{h}_a^k \| \boldsymbol{h}_v^k$ is input into the binary classifier C_F, and the predicted label $\hat{y}$ is output (Line 14). The loss function is calculated and back-propagated to update the model parameters of MGGCN and C_F (Line 15). Finally, the algorithm outputs the trained model parameters (Line 18).

It should be pointed out that collaborative error detection should be carried out after collaborative training is completed. Specifically, both parties transform their respective datasets D and D' into the corresponding graphs and use the trained model to perform forward propagation calculation of the MGGCN and the binary classifier by following the FCTA. The outputs of their classifiers C_F and C_F' are the results of error detection.

4.2 FCTA

In FeLeDetect, data exchange occurs during the feature learning phase. For example, to obtain the embedding output by the k-th layer of the MGGCN, the data holder F first needs to exchange the embeddings of some nodes and edges obtained at the $(k-1)$-th layer with the data participant F'. After completing the data exchange with F', participant F updates the embedding of the tuple-level nodes at the k-th layer of the MGGCN in line with the following formula:

$$\boldsymbol{h}_{\mathcal{N}(t)\cup\mathcal{N}'(t)}^k = AGG_{(a,v)\in\mathcal{N}(t)\cup\mathcal{N}'(t)}\left(\boldsymbol{W}_{ta}^k \boldsymbol{h}_a^{k-1} \odot \boldsymbol{W}_{tv}^k \boldsymbol{h}_v^{k-1}\right) \tag{7}$$

$$\boldsymbol{h}_t^k = \sigma\left(\boldsymbol{W}_t^k \left(\boldsymbol{h}_t^{k-1} \| \boldsymbol{h}_{\mathcal{N}(t)\cup\mathcal{N}'(t)}^k\right)\right) \tag{8}$$

where $\boldsymbol{W}_{ta}^k$, $\boldsymbol{W}_{tv}^k$, and $\boldsymbol{W}_t^k$ denote the weight matrices to be learned; $\boldsymbol{h}_t^{k-1}$, $\boldsymbol{h}_v^{k-1}$, and $\boldsymbol{h}_a^{k-1}$ denote the embeddings calculated by the $(k-1)$-th layer network; $\mathcal{N}(t)$ denotes the set of edges and nodes that are directly connected to the node t in the graph G; and $\mathcal{N}'(t)$ denotes the set of edges and nodes that are directly connected to the node t in the graph G'.

The update method of the embeddings of value-level nodes and attribute-level edges at the k-th layer of MGGCN is the same as that of GEDM in Section 3. Details are listed in Eqs. (3)–(5). Since the training process of both parties is highly symmetrical, the training process is described by taking data participant F as an example. Algorithm 2 gives the flow of the FCTA for F.

Algorithm 2 uses the embeddings $\boldsymbol{h}_a^{k-1}$, $\boldsymbol{h}_t^{k-1}$, and $\boldsymbol{h}_v^{k-1}$ at the $(k-1)$-th layer of the MGGCN as the input and outputs the updated embeddings corresponding to the k-th layer. FCTA first exchanges the embeddings $\boldsymbol{h}_a^{k-1}$ and $\boldsymbol{h}_v^{k-1}$ that correspond to the attribute-level edges and value-level nodes output by the MGGCN network of the previous layer with F' (Lines 1–2) and then updates the embeddings of the tuple-level nodes according to Eqs. (7)–(8) in turn.

Algorithm 2. FCTA (taking data participant F as an example).

```
Input: embeddings of attribute-level edges, tuple-level nodes, and value-level nodes at the $(k-1)$-th
       $(k \geq 1)$ layer of the MGGCN, namely $\boldsymbol{h}_a^{k-1}$, $\boldsymbol{h}_t^{k-1}$, and $\boldsymbol{h}_v^{k-1}$.
Output: embeddings of attribute-level edges, tuple-level nodes, and value-level nodes at the $k$-th
        layer of the MGGCN, namely $\boldsymbol{h}_a^k$, $\boldsymbol{h}_t^k$, and $\boldsymbol{h}_v^k$
1. Send $\boldsymbol{h}_a^{k-1}$ and $\boldsymbol{h}_v^{k-1}$ to participant $F'$ $(\forall(a,v) \in G_F)$
2. Receive $\boldsymbol{h}_a'^{k-1}$ and $\boldsymbol{h}_v'^{k-1}$ from participant $F'$ $(\forall(a,v) \in G_F)$
3. Calculate $\boldsymbol{h}_t^k$ $(\forall t \in G_F)$ /* update the embedding corresponding to node $t$ according to Eqs. (7)–(8)
   */
4. Calculate $\boldsymbol{h}_v^k$ $(\forall t \in G_F)$ /* update the embeddingcorresponding to node $v$ according to Eqs. (3)–(4)
   */
5. Calculate $\boldsymbol{h}_a^k$ $(\forall t \in G_F)$ /* update the embedding corresponding to the edge $a$ according to Eq. (5)
   */
6. return $\boldsymbol{h}_t^k$, $\boldsymbol{h}_v^k$, and $\boldsymbol{h}_a^k$
```

Furthermore, it updates the embeddings of the value-level nodes according to Eqs. (3)–(4) and those of the attribute-level edges according to Eq. (5) (Lines 3–5). Finally, FCTA returns the embeddings of each node and edge at the k-th layer of the MGGCN.

It should be pointed out that, collaborative training can be understood by the fact that both data participants construct a virtual graph $G_v = G \cup G'$, respectively, and collaborative features are represented based on the virtual graph. Therefore, the collaborative training mechanism ensures that the embeddings obtained by both parties are in the same embedded space. So, there is no need to realign the embedded space of both data participants as described in Ref. [17]. In addition, in the process of collaborative training and error detection, both data participants only need to exchange the embeddings of the attribute-level edges and value-level nodes, and they do not have to exchange the embeddings of the tuple-level nodes. The specific reasons are described below from the perspective of data information analysis.

4.2.1 *Data information analysis*

First, the definition of data information is given.

Definition 2 (Data information)**.** The graph-based error detection method aims to learn the embeddings of nodes and edges to capture complex structural relationships. For the tuple-level node t, the data information required in the feature learning process is the embeddings of the attribute-level edge and value-level node $(a,v) \in \mathcal{N}(t)$ directly connected with the tuple-level node t. For the value-level node v, the data information required in the feature learning process is the embedding of the attribute-level edge and tuple-level node $(a,t) \in \mathcal{N}(v)$ directly connected with the value-level node v. For the attribute-level edge a, it only needs its own embedding in the process of feature learning.

In order to facilitate the analysis of data information captured by the FCTA, three error detection scenarios are introduced here:

(1) Local scenario (L), where both data participants only use local data for error detection.

(2) Federated scenario (F), where both data participants use FeLeDetect for federated error detection, and this process does not involve the exchange of original data.

(3) Centralized scenario (C), where both data participants first transmit their data to a data center and then perform error detection on the integrated data.

Definition 3 (Information loss)**.** For a given MRG G in a centralized scenario, we have $(t_i, a_j, v_k) \in G \wedge (t_i, a_m, v_n) \in G$, with $a_j \neq a_m$, $v_k \neq v_n$. In the local scenario, data participants F and F' construct local MRGs G_F and $G_{F'}$, respectively. Due to the longitudinal data splitting, in G_F, we have $(t_i, a_j, v_k) \in G_F \wedge (t_i, a_m, v_n) \notin G_F$; in

$G_{F'}, wehave(t_i, a_m, v_n) \in G_{F'} \wedge (t_i, a_j, v_k) \notin G_{F'}$. Therefore, in the local scenario, the information loss of the data participant F in updating the embedding of t_i is the embedding of (a_m, v_n), and it is the symmetrical case with the information loss of the data participant F'.

Theorem 1. Compared with GEDM in the centralized scenario which directly uses cross-source data information, the FeLeDetect method in the federated scenario indirectly uses cross-source data through FCTA, so as to guarantee no loss of cross-source information.

Proof: First, we need to prove that FCTA triggers no information loss when updating the eigenvectors of tuple-level nodes. In the centralized scenario, based on the integrated dataset $T_S = T \cup T'$, GEDM uses Eq. (1) to obtain the eigenvectors of tuple-level nodes. In the federated scenario, data participants F and F' use Eq. (7) to obtain the eigenvectors of tuple-level nodes on the datasets T and T', respectively. Since $\mathbb{N}(t) = \mathcal{N}(t) \cup \mathcal{N}'(t)$, $(a, v) \in \mathcal{N}(t) \cup \mathcal{N}'(t)$ in Eq. (7) is equal to $(a, v) \in \mathbb{N}(t)$ in Eq. (1). Before FCTA updates the eigenvectors of tuple-level nodes, both parties exchange the eigenvectors of attribute-level edges and value-level nodes. Therefore, the update is based on the data information of both parties, which solves the problem of information loss caused by updating tuple-level nodes with only single-source data in the local scenario. Compared with that in the centralized scenario, the update of the tuple-level node eigenvectors in accordance with FCTA ensures lossless cross-source data information.

Then, we need to prove that FCTA guarantees no information loss when updating the eigenvectors of attribute-level nodes. As described above, the attribute-level edges are updated by Eq. (5). Each update depends only on the result of its last update. Therefore, data exchange is not required in the federated scenario to ensure that the information is lossless.

Finally, we prove that FCTA triggers no information loss when updating the eigenvectors of value-level nodes. In the centralized scenario, based on the integrated dataset $T_S = T \cup T'$, GEDM uses Eq. (3) to obtain the eigenvectors of tuple-level nodes. In the federated scenario, based on the local dataset T or T', FCTA uses Eq. (3) to obtain the eigenvectors of tuple-level nodes. It can be easily found that $(a, t) \in \mathcal{N}(v) = (a, t) \in \mathbb{N}(v)$ and $(a, t) \in \mathcal{N}'(v) = (a, t) \in \mathbb{N}(v)$. For the attribute-level edge, its update only depends on the result of its last update; for tuple-level nodes, the initial embedding of each data participant is generated by the same seed. It has been proved that the updated embedding contains lossless cross-source data information, so no exchange is required. Therefore, $(a, t) \in \mathcal{N}(v)$ in the local data of the data participant F is the same as $(a, t) \in \mathbb{N}(v)$ in the integrated data in the centralized scenario, and the information is complete. Therefore, the data participants in FCTA do not need to exchange the embeddings of the tuple-level nodes, and the update of the value-level nodes can be completed according to Eqs. (3)–(4) by using only local data.

4.2.2 *Privacy analysis*

Since privacy protection is a major challenge to cross-source data error detection, it is critical to ensure that FCTA is subjected to no privacy leakage risk during the data exchange through FCTA. Specifically, FCTA involves two types of data exchange. One is the exchange of the initial eigenvectors between the two parties. Because the initial eigenvectors of the nodes and edges of each party are randomly initialized, they do not contain any information related to the original data. Therefore, it is impossible to infer the original data from the initial embedding. This guarantees that the exchange of the initial embedding suffers from no risk of privacy leakage. The other is the exchange of the embedding updated by the k-th layer network. There is no privacy leakage risk in the exchange of the intermediate layer result of the neural network. Although some studies have shown that the original input can be inferred from the intermediate layer results of the neural network[44], the original input of both data participants is a randomly initialized embedding, or in other words, the original input does not contain any privacy information. To sum up, the data exchange mechanism of FCTA ensures the privacy and security of cross-source data.

4.3 Optimization methods

When FeLeDetect collaboratively trains GEDM, the intermediate layer results of different data participants need to be exchanged frequently, which places great pressure on network communication. In this paper, different optimization techniques are proposed to reduce communication costs in three aspects. At the same time, with regard to the manual labeling cost in monitored learning, this paper also proposes an automatic labeling strategy to reduce the manual labeling cost in the training set.

4.3.1 *Data de-duplication*

In an MRG, some different tuple-level nodes are connected to the same value-level nodes, which will cause some of the same data to be exchanged multiple times. Figure 3(b) shows that tuple-level nodes t_1, t_2, and t_3 are associated with the same value-level node v, namely "International Conference on Management of Data". Therefore, when the node information is transmitted to another data participant, the contents to be transmitted are $\{1 : \boldsymbol{h}_v^k\}$, $\{2 : \boldsymbol{h}_v^k\}$, and $\{3 : \boldsymbol{h}_v^k\}$, which makes the embedding $\boldsymbol{h}_v^k$ of the value-level node "International Conference on Management of Data" transmitted three times. Moreover, many tuple-level nodes and value-level nodes are connected by the same type of attribute-level edges, which will make a large number of eigenvectors $\boldsymbol{h}_a^k$ corresponding to the same type of edges to be transmitted multiple times. In fact, the number of different attribute-level edges in the graph depends on the number of attributes in the dataset. In order to avoid the unnecessary exchange of duplicated data, the embedding of the same value-level node or attribute-level node is transmitted only once. In other words, the data are transmitted after the duplicated data are removed. Still in the example described above, after the de-duplication operation, the data transmitted to the other party is $\{\{1, 2, 3\} : \boldsymbol{h}_v^k\}$. In this paper, the matrix $\boldsymbol{M}_V = [\boldsymbol{h}_{v1}^k, \boldsymbol{h}_{v2}^k, \cdots, \boldsymbol{h}_{vp}^k]$ is used to represent the embedding of the value-level node after de-duplication. The matrix $\boldsymbol{M}_A = [\boldsymbol{h}_{a1}^k, \boldsymbol{h}_{a2}^k, \cdots, \boldsymbol{h}_{am}^k]$ represents the embedding of the attribute-level edge after de-duplication to avoid multiple exchanges of the same embedding. Here, p denotes the number of different attribute values in the dataset and m denotes the number of different attributes in the dataset. With the DBLP dataset in Figure 1 as an example, $p = 13$ and $m = 5$.

4.3.2 *Quantization compression*

Since the matrix $\boldsymbol{M}_A$ is generally much smaller than the matrix $\boldsymbol{M}_V$, the communication cost of FeLeDetect is mainly caused by the transmission matrix $\boldsymbol{M}_V$. Therefore, it is possible to further reduce the communication cost of $\boldsymbol{M}_V$. With the activation function applied, each value in $\boldsymbol{M}_V$ is a real number in the range of $[-1, 1]$. In order to further reduce the communication cost, we discretize these real values to compress their sizes. Specifically, the interval $[-1, 1]$ is divided into 2^η sub-intervals. For example, when $\eta = 2$, the interval $[-1, 1]$ is divided into four sub-intervals, namely $[-1, -0.5)$, $[-0.5, 0)$, $[0, 0.5)$, and $[0.5, 1]$. Thus, only two bits of data are required to represent the four sub-intervals. Each real value in the original matrix falls into one of the sub-intervals, so it can be approximated as the endpoint of the sub-interval. For example, real values falling into the sub-interval $[-1, -0.5)$ are all approximated as -1. The upper limit of the error between the approximated quantized value and the original value is $2^{1-\eta} = 0.5$. In this way, a real value that originally requires 32 bits (4B) of bandwidth only needs η bits of coding after quantization. If $\eta = 16$, the size of $\boldsymbol{M}_V$ will be compressed to half of the original, and the quantization error is only 2^{-15}.

4.3.3 *Low-frequency transmission*

When FeLeDetect collaboratively trains GEDM, multiple epochs of training are required to make the neural network model converge. One epoch means that we need to use all the training data to train the model (including the MGGCN and the binary classifier), and each epoch includes multiple rounds of iteration. In each round, the MGGCN needs to exchange the intermediate results of each layer of the network. However, the intermediate results of a round (such as $\boldsymbol{M}_V$) may be very similar to the results of the previous round, especially after the network parameters tend to be stable. Therefore, it is possible to reduce the frequency of data exchange to further reduce the communication cost of the training process. Specifically, if the intermediate results produced by MGGCN in two adjacent rounds are very similar, both parties can cancel the exchange of the intermediate results. It is assumed that an intermediate result matrix $\boldsymbol{M}_V$ of the current round, an intermediate result matrix $\boldsymbol{M}'_V$ of the previous round, and a transmission threshold τ are given; if $\mathit{Sim}(\boldsymbol{M}_V, \boldsymbol{M}'_V) = \|\boldsymbol{M}_V - \boldsymbol{M}'_V\|_F < \tau$, then the intermediate results $\boldsymbol{M}'_V$ of the previous round of exchange are approximated to those produced by the current round, so as to avoid the exchange of $\boldsymbol{M}_V$. Specifically, $\|\cdot\|_F$ denotes the similarity measurement function. In this paper's experiments, the simplest Euclidean distance is used to measure the similarity.

4.3.4 *Labeling strategy*

As the error detection problem is treated as a binary classification problem in this paper, the binary classifier requires labeled training data to learn how to classify data cells. However, data labeling calls for the participation of experts in this field, which consumes a lot of labor costs, so it often becomes a bottleneck problem in practical applications. HoloDetect[2]uses data enhancement technology to reduce the need for manual labeling. However, it still requires manual labeling for 5%–10% of the entire dataset. For a dataset containing 100,000 tuples and eight attributes, 10% means that 80,000 data cells need to be manually labeled. This amount of manual work cannot be underestimated. Raha[20] combines clustering and label propagation techniques to limit the amount of manual labeling to a constant level, or in other words, it should not increase with the increase in the size of the dataset. However, it cannot be directly applied to cross-source data scenarios. In view of this, this paper designs an automatic labeling strategy to reduce the cost of manual labeling and generate labeled training datasets for cross-source data.

It is assumed that there are two data holders F and F', which own datasets $D = \{t_1, t_2, \cdots, t_n\}$ and $D' = \{t'_1, t'_2, \cdots, t'_n\}$, respectively. The dataset has completed encrypted entity alignment. That is to say, each pair of tuples (t_j, t'_j) $(1 \leq j \leq n)$ represents the same entity. The labeling strategy is carried out in the following steps. First, F randomly samples a subset $S = \{\, t_j \mid t_j \in D \,\}$ of data, where $|S| = n \cdot \alpha\%$. Second, the value of each data cell in each tuple of S is encrypted by the substitution encryption, and the encrypted data subset $E(S)$ is obtained. Finally, F will transmit $E(S)$ and the sampling index set $I = \{j | t_j \in S\}$ to the other data holder F'. Wherein, the index set I indicates the tuple labels sampled in the original dataset D. When the data holder F' receives $E(S)$ and I, it will sample from the D' according to I to obtain $S' = \{t'_j | j \in I\}$ and assemble the tuples pointing to the same entity in S' and $E(S)$ to get an integrated dataset $S_C = \{E(t_j) || t'_j | E(t_j) \in E(S), t'_j \in S'\}$. It should be noted that S_C has $(m_1 + m_2)$ attributes in total. Specifically, m_1 denotes the number of attributes of dataset D and m_2 denotes the number of attributes of dataset D'. F' uses S_C as the input dataset of the error detection tool Raha to obtain the label of each data cell. F' keeps the data labels related to S' and transmits the data labels of $E(S)$ to F. In order to guarantee the quality of the labels, F' only accepts labels returned by Raha with a confidence greater than a certain threshold (90% in this paper's experiments). Then, the data labeling process is completed. Both parties obtain some data labels, and thus the labeled training datasets T and

T' are generated. During this process, both parties do not disclose their original data. As the automatic labeling strategy uses Raha to generate labels, the cost of manual labeling is the same as that of Raha, which is a constant level.

Since the substitution encryption technique is used in the above process, $E(S)$ loses its semantic information of the original data. However, the privacy of the original data is protected. At the same time, Raha can still detect some data errors (such as the dependency violating attribute value and format error). This is because the substitution encryption is equivalent to an anonymous space mapping of the attribute value, and the dependency of the original value is maintained in the encrypted space. Moreover, errors unrelated to semantics (such as format errors) can also be detected. For the sake of convenience, the experiments in this paper employ a substitution cipher to encrypt the sampled data subset $S = \{t_j | t_j \in D\}$, which is a classic code. Although many advanced cryptographic techniques have been proposed and extensively applied, the substitution cipher remains a very important encryption strategy in the field of cryptography. Modern common ciphers such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES) use classical ciphers as their basic building blocks[45]. In practice, both parties can negotiate and choose more complex and advanced encryption technology to encrypt S to achieve higher security.

5 Experimental Analysis

This section presents an experimental evaluation on real datasets, and the main purposes are as follows: (1) prove the effectiveness and sophistication of GEDM; (2) verify that the detection accuracy of FeLeDetect is higher than that of various local detection methods using only single-source data and is equivalent to that of GEDM in the centralized scenario; (3) verify the effectiveness of the communication optimization strategy. In addition, we test the running time of GEDM and FeLeDetect to evaluate their efficiency, verify the effectiveness of FeLeDetect under different error rates and error type distributions, and test the scalability of FeLeDetect. Section 5.1 introduces the experimental data and Section 5.2 the experimental settings and evaluation indicators. Section 5.3 expounds on the details of the experiments and Section 5.4 gives the experimental results and corresponding analysis.

5.1 Experimental data

This paper uses three public real datasets for experimental testing. Table 1 shows the statistics for the datasets used. The types of errors include SE, FIs, Missing Values (MV), and Violation of Dependency rules between Attributes (VAD). As mentioned in the introduction part, it is often difficult to detect SEs in the case of a single source, so cross-source data information is required. The datasets used contain multiple types of errors, which reflects the heterogeneity of data errors.

Table 1 Experimental dataset

Dataset		Size	Number of data errors	Error type
D-A	DBLP	2224×4	444	SE, FI
	ACM	2224×4	444	SE, MV
Flights	Flights1	2445×4	1,879	SE, MV, FI
	Flights2	2445×4	2,972	SE, MV, VAD
Adult	Adult1	97864×4	19,481	SE
	Adult2	97864×4	19,535	SE

The DBLP-ACM dataset (D-A)[46] is an open dataset commonly used in entity alignment research. The dataset contains two data tables (DBLP and ACM), which record the information of some conferences and published papers in the field of computer science. The pattern of the data tables is as follows: title, author, venue, year. The data that have completed the entity alignment are used in the experiment. Since DBLP and ACM belong to different organizations, it is reasonable to assume that both parties do not allow the privacy leakage of original data. This experiment introduces SEs into some data cells of the venue attribute column of DBLP and ACM. This is because many values in the venue attribute domain are very similar, and SEs are likely to occur in real-world scenarios. Specifically, some data cell in the venue attribute column are randomly selected, and the original value of the cell is substituted with another value that is the most similar to it in the venue attribute domain, namely the new value with the smallest editing distance from the original value. At the same time, format errors (illegal characters) and missing errors are introduced into the year attribute column of DBLP and the title attribute column of ACM to simulate the heterogeneity of data errors in real-world scenarios. By default, the proportion of introduced errors reaches 80%. Here, the error type ratio is defined as the ratio of the number of SE data cells to that of all erroneous data cells.

The Flights dataset[47] records the information about more than 1,200 flights collected from different data sources every day in December 2011; and the information collected is recorded in the same table. Due to the different quality of data from different sources, there are many errors in the table, including SEs, MVs, format errors, and VAD. These errors exist in the dataset, so there is no artificially introduced error in this experiment. In this experiment, the data collected on December 1st are randomly selected, and the data attribute set includes source, flight, scheduled departure, actual departure, scheduled arrival, and actual arrival. In addition, in order to simulate the scenario where two data participants are unwilling to integrate the data, the Flights is artificially divided into two sub-tables vertically, that is, Flights1 (source, scheduled departure, and scheduled arrival) and Flights2 (flight, actual departure, and actual arrival), and it is assumed that each sub-table is owned by a data holder.

The Adult dataset comes from the UCI machine learning library (http://archive.ics.uci.edu/ml/), which contains 97,684 records from the U.S. Census of Population (1994). In this experiment, eight attributes are selected and vertically segmented into two disjoint sub-tables, namely Adult1 (age, education, race, and sex) and Adult2 (marital status, relationship, country, and income) and it is assumed that each one of them is owned by a data holder. With a process similar to D-A, SEs are injected into some data cells of age, race, and relationship attributes in thise experiment.

Since datasets with real errors usually have no ground truth, the effectiveness of the error detection algorithm cannot be evaluated[48]. A large number of experiments on error detection have been conducted through artificial error introduction[49, 50]. Related studies[15, 51] show that in the real world, about 5% of the data cells in a dataset are subjected to data errors for various reasons. Therefore, this paper sets the error rate to 5% by default when artificially introducing errors into the D-A and Adult datasets. Here, the error rate is defined as the ratio of the number of erroneous data cells to the total number of data cells in the dataset. Due to the difficulty in detecting SEs, the D-A error type ratio is set to 80% by default. Here, the error type ratio is defined as the ratio of the number of SE data cells to that of all erroneous data units. Unless explicitly stated, the above default error rate of 5% and error type ratio of 80% are used in the following experiments. It is worth noting that although the default error rate and error type ratio are set, this paper also conducts experiments on the change in the error rate and the error type ratio, proving that FeLeDetect is suitable for scenarios with different error rates and error type ratios.

5.2 Experimental setting and evaluation index

Five different types of existing error detection methods are chosen as benchmarks, namely (1) DBoost[52], an exception detection framework that integrates multiple detection models based on machine learning methods; (2) NADEEF[11], a rule-based error detection method that allows users to specify data consistency rules and perform error detection according to the rules; (3) KATARA[14], an error detection method based on external knowledge-based system (KBs); (4) Metadata-driven[22] method (Meta), an integrated error detection method that integrates multiple metadata and (5) Raha[20], a state-of-the-art (SOTA) error detection system based on machine learning[1]. Due to the randomness of Raha's error detection results, the experimental evaluation method of Raha researchers is adopted here, or in other words, the mean value of 10 times of independent running is reported for its related evaluation indicators.

Three different scenarios are set in the experiment. (1) Local scenario (L), where each data holder performs error detection only through local data. The proposed GEDM and five benchmark models are tested in the local scenario. (2) Federated scenario (F), where the data holder does not need to exchange original data, and the error detection models are collaboratively trained. The FeLeDetect error detection method proposed in this paper is tested in the federated scenario, and its superiority is proved. (3) Centralized scenario (C), where all data holders transmit their own data to a data center and then perform error detection based on the integrated dataset. Such a setting is to verify that the detection accuracy of the FeLeDetect method in the federated scenario is equivalent to that of the GEDM method in the centralized scenario. The GEDM and five benchmark models are tested in the centralized scenario.

The experiment comprehensively evaluates the proposed method by using different evaluation indexes and reports the following items. (1) Precision P, namely the probability that the data cell detected as an error is actually an erroneous cell; (2) Recall R, namely the probability that the cell is detected as an error in the actual erroneous data cell; (3) $F1$ score, namely the harmonic mean of the precision and the recall: $F1 = 2 \times (P \times R)/(P + R)$; (4) model training time *TT* and model detection time *DT* (i.e., the error detection time on complete data); (5) the communication cost V of the model training in bytes, which indicates the data communication traffic of both parties in the collaborative training.

5.3 Implementation details

The implementation of the FeLeDetect method entails the use of the PyTorch library[54]. The experiment uses SGD as the optimal algorithm for model training and sets the learning rate to 0.01 and the momentum to 0.9. After some initial experimental evaluation, the batch sizes for DBLP-ACM, Adult, and Flights are set to 16, 128, and 512, respectively. During the training process, all datasets are trained for 300 epochs. In the automatic data labeling process of FeLeDetect, the label sampling rate α of DBLP-ACM and Flights is set to 20%. Due to the large size of the Adult dataset, α of 5% is sufficient to support model training. In the experiment, the labeled dataset is randomly divided into training set (60%) and validation set (40%), and the checkpoint with the highest $F1$ score on the validation set is returned, and the trained model is used for error detection on the complete dataset. By default, the communication optimization parameters are $\eta = 4$ and $\tau = 1.5$. All experiments are carried out on the Dell server with Intel Xeon 4110 CPU, 128 GB RAM, and 4 NVIDIA GeForce RTX2080ti GPUs. All test programs for this evaluation are written in the Python language[2].

[1] HoloDetect[2] and Raha are advanced error detection systems based on machine learning. Since Raha makes the source code publicly available, this paper uses Raha as a benchmark method for comparison. Meanwhile, Ref. [54] confirms that Raha can achieve error detection accuracy at a lower labor cost, comparable to that of HoloDetect.

[2] The experimental code is published in the https://github.com/ZJUDAILY/FeLeDetect.

5.4 Experimental results and analysis

In this section, GEDM and FeLeDetect are comprehensively evaluated on three datasets and compared with the five benchmark methods mentioned above.

5.4.1 *Effectiveness and efficiency of GEDM*

Firstly, the error detection performance of GEDM is compared with that of the other five benchmark methods in the local scenario and the centralized scenario. The precision of the error detection is estimated by using P, R, and $F1$ score; *DT* (in seconds) required by different error detection methods is also given to evaluate the efficiency of error detection. For the sake of fairness, only the error detection time of GEDM (i.e., the detection time of the model) is reported, and the training time *TT* of the model will be given in the supplementary experiment section. Table 2 shows the error detection results of GEDM and the other five benchmark methods in the local scenario and the centralized scenario, and the optimal $F1$ score is marked in bold.

Table 2 Comparison of P, R, and $F1$ of different error detection methods in different detection scenarios

Error detection in the local scenario

Detection method	D-A						Flights						Adult					
	DBLP			ACM			Flights1			Flights2			Adult1			Adult2		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
dBoost	0.17	1.00	0.29	0.33	0.20	0.25	0.78	0.87	0.82	0.72	0.49	0.58	0.63	0.36	0.45	0.23	0.90	0.36
NADEEF	0.44	0.21	0.28	0.50	0.21	0.29	0.08	0.12	0.09	0.40	1.00	0.57	0	0	0	0	0	0
KATARA	0.12	1.00	0.21	0	0	0	0.02	0.13	0.56	0.02	0.14	0.02	0.02	0.29	0.08	0.02	0.10	0.02
Meta	0.45	0.22	0.29	1.00	0.01	0.01	1.00	0.87	**0.93**	0.70	0.68	0.68	1.00	0	0	1.00	0	0
Raha	0.57	0.32	0.42	0.65	0.63	0.64	0.98	0.87	0.92	0.71	0.70	0.70	0.50	0.83	0.62	0.74	0.93	0.82
GEDM	0.68	0.33	**0.45**	0.78	0.80	**0.79**	1.00	0.87	**0.93**	0.71	0.73	**0.72**	0.76	0.75	**0.75**	0.86	0.90	**0.88**

Error detection in the centralized scenario

Detection method	D-A						Flights						Adult					
	DBLP			ACM			Flights1			Flights2			Adult1			Adult2		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
dBoost	0.10	0.20	0.13	0.31	1.00	0.48	0.78	0.87	0.82	0.67	0.30	0.41	0.67	0.36	0.45	0	0	0
NADEEF	0.10	1.00	0.17	0.10	1.00	0.17	0.38	0.41	0.40	0.40	1.00	0.57	0	0	0	0	0	0
KATARA	0.12	1.00	0.21	0	0	0	0.02	0.13	0.06	0.02	0.01	0.01	0.05	0.29	0.08	0.02	0.10	0.02
Meta	1.00	0.21	0.35	1.00	0.01	0.01	1.00	0.87	**0.93**	0.70	0.68	0.69	1.00	0	0	1.00	0	0
Raha	0.67	0.37	0.47	0.68	0.61	0.65	0.97	0.88	0.92	0.66	0.75	0.70	0.67	1.00	0.80	0.87	0.90	0.88
GEDM	1.00	0.72	**0.84**	1.00	0.84	**0.91**	0.98	0.88	**0.93**	0.73	0.74	**0.73**	0.92	0.99	**0.96**	0.99	0.90	**0.95**

Error detection in the federated scenario

Detection method	D-A						Flights						Adult					
	DBLP			ACM			Flights1			Flights2			Adult1			Adult2		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
FeLeDetect	1.00	0.72	**0.84**	1.00	0.84	**0.91**	1.00	0.87	**0.93**	0.73	0.74	**0.73**	0.92	1.00	**0.96**	1.00	0.90	**0.95**

In terms of precision, first of all, the $F1$ score of GEDM is higher than that of the other five benchmark methods in both local and centralized scenarios. Specifically, in the local and centralized scenarios, the $F1$ scores of GEDM are improved by 10.3% and 25.2% on average, respectively, compared with that of the optimal benchmark method. Especially in the centralized scenario, when cross-source data are integrated, GEDM can effectively capture the features of the integrated data, so it can achieve high precision and high recall at the same time. However, other benchmark methods cannot achieve high precision and high recall at the same time. Specifically, dBoost reports low accuracy because it uses heuristic ideas to treat outliers in statistical probability as data errors. NADEEF has low accuracy because rule-based detection methods generally have the problem of coarseness. KATARA inevitably mismatches the relations on some KBs with the attributes on relational data, which leads to its low precision. Moreover, as the limited KBs cannot completely cover the knowledge in relational data, the recall is relatively low. As it is difficult for the error detection methods integrated by Meta to cover all error types, it shows a low recall. Raha is currently the best error detection method, but its accuracy is still not as high as that of GEDM, because it cannot fully capture the rich features of data in the dataset. On the contrary, the GEDM proposed in this paper can not only capture the semantic features of attribute values but also represent the dependencies between different attributes and different tuples. It should be pointed out that since most of the errors in dataset Flights1 (over 85%) are MVs, and the detection of MVs is less difficult, GEDM does not show a clear advantage over Flights1. Second, it should be noted that the error detection performance of dBoost, NADEEF, and KATARA in the centralized scenario is not better than that in the local scenario. This is because these methods cannot effectively capture the correlation between different attributes. Therefore, even in the centralized scenario, the integrated dataset has more attributes (more dimensional data features), and the precision of error detection is not substantially improved. Specifically, dBoost detects errors independently for each attribute column and adjusts the related parameter configuration based on the dataset. Therefore, in the centralized scenario, dBoost performs parameter selection based on the integrated complete dataset, and the global parameter may not be the optimal parameter configuration for both local datasets. So, its detection effect in the centralized scenario is inferior to the error detection in the local scenario. NADEEF performs error detection based on the consistency rule. However, given its coarse-grained detection method, when the number of attributes increases (in the centralized scenario), more misjudgment of detection results may arise. KATARA uses KBs for error detection. Therefore, in the centralized scenario, the probability of a mismatch between the relation in the KBs and the attribute in the data table is higher, and there is no guarantee that the precision of error detection in the centralized scenario will be improved. Since other methods can effectively represent the relationship between different attributes, the detection effect in the centralized scenario is naturally better than that in the local scenario.

In terms of running efficiency, it can be seen from Table 3 that in both local and centralized scenarios GEDM outperforms other benchmark methods. For example, dBoost takes a lot of time to run under different parameter configurations to get the final result; KATARA takes a long time to run because it needs to go through each KB and match it with the attributes in the data table; Raha needs to run different error detection strategies and perform data cell clustering and label propagation, which are extremely time-consuming, so its running time is the longest. In contrast, GEDM's error detection is highly efficient. This is because once the model is trained, the error detection result can be obtained by performing only one forward propagation calculation on the model on the complete dataset.

Table 3 Comparison of *DT* between different error detection methods in different detection scenarios

Detection method	Error detection in the local scenario						Error detection in the centralized scenario			Error detection in the federated scenario		
	D-A		Flights		Adult		D-A	Flights	Adult	D-A	Flights	Adult
	DBLP	ACM	Flights1	Flights2	Adult1	Adult2						
dBoost	2.21	2.37	2.35	2.64	74.29	77.06	5.29	6.06	211.85	—	—	—
NADEEF	0.13	1.05	1.06	1.30	1.76	2.11	1.98	1.42	2.97	—	—	—
KATARA	9.28	9.35	9.62	9.64	40.02	40.02	14.70	15.45	100.72	—	—	—
Meta	2.20	2.16	2.18	2.66	14.69	17.90	4.41	3.10	29.04	—	—	—
Raha	14.97	15.91	14.44	13.73	1,059.06	1,266.30	27.04	31.58	2,397.48	—	—	—
GEDM	**0.10**	**0.09**	**0.23**	**0.22**	**1.29**	**1.27**	**0.12**	**0.36**	**3.12**	—	—	—
FeLeDetect	—	—	—	—	—	—	—	—	—	9.43	9.46	386.9

5.4.2 *Effectiveness and efficiency of FeLeDetect*

For privacy protection, cross-source data are often not allowed to be transferred to a public data center for integration. Therefore, data error detection in the centralized scenario is seriously hindered. In view of this, the error detection method tailored for single-source data can only be performed locally by a data holder, and the information of other data sources related to the data source cannot be obtained. In the federated scenario, the FeLeDetect method proposed in this paper is compared with various benchmark methods (including GEDM) in the local and centralized scenarios. Table 2 shows the error detection results of FeLeDetect in the federated scenario and those of the other five benchmark methods and GEDM, with the best $F1$ score in bold.

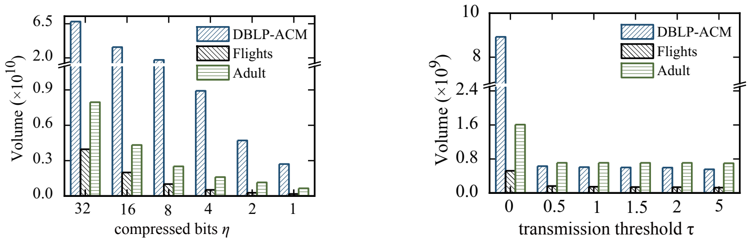
In terms of precision, Table 2 shows that the $F1$ score of FeLeDetect is better than that of other methods (including GEDM) in the local scenario. The $F1$ score of FeLeDetectis further improved by an average of 23.2% compared with that of GEDM which has the best detection precision in the local scenario. This is because in the local scenario, all error detection methods only use local data information and lack the support of data information from other sources. In contrast, FeLeDetect uses the federated learning mechanism to fully capture cross-source data features. This proves that cross-source data can indeed improve the precision of error detection. Since simple errors such as missing and format errors account for more than 85% of all error types in the real erroneous dataset (Flights1), and difficult error types such as SEs and dependency violation account for a relatively small proportion, the proposed method has no significant precision improvement (but it still outperforms or shares the same performance with other methods). The above experiment proves that for the data errors that are easy to be detected, the proposed method can exceed or perform the same as the most advanced method. For data errors that are difficult to be detected, the proposed method can outperform existing methods. In addition, the $F1$ score of FeLeDetect on all three datasets reaches the same level as that of GEDM in the centralized scenario. This also verifies that the FCTA can ensure lossless information from the perspective of experimental results and achieve the same error detection effect as that of GEDM in the centralized scenario.

In terms of running efficiency, Table 3 shows that all detection methods have the shortest running time in the local scenario compared with the federated and centralized scenarios. This is because in the local scenario, the dataset is smaller and has no time cost of federated communication. Nevertheless, FeLeDetect's error detection is still relatively efficient. FeLeDetect can complete the error detection task in about six minutes, especially on a large-scale dataset, such as an Adult dataset with nearly 100,000 tuples, and its running efficiency is improved by more than 60% compared with that of Raha, the benchmark method with the highest precision in the local scenario.

5.4.3 *FeLeDetect communication optimization experiment*

Since data de-duplication can reduce the communication traffic during federated training without affecting the detection effect, only the impact of quantization and low-frequency transmission on federated communication and detection performance will be explored here.

The compression parameter η varies in the range of $\{32, 16, 8, 4, 2, 1\}$ to explore the impact of the number of compressed bits on the communication traffic and error detection accuracy. It should be noted that $\eta = 32$ means no quantization optimization. As shown in Figure 6(a), the communication traffic decreases significantly as η decreases. This is because a smaller number η of bits after compression indicates a shorter coding length of each value in the transmission matrix $\boldsymbol{M}_V$ and a smaller data transmission volume. Also, the error detection precision is less affected by the number of compressed bits. As shown in Figure 7(a)–(f), when η changes, the curves of precision rate, recall rate, and $F1$ score of the detection results are relatively stable. When the number of compressed bits is 1, the precision is significantly reduced. This is because the initial features of nodes and edges in the MRG are randomly initialized, and the final feature representation is obtained by feature aggregation of GNNs. Compressing the intermediate results of neural networks is equivalent to adding an active layer between different layers of GNNs to compress the feature space. As the initial feature is randomly generated, compressing the value of the intermediate embeddings is equivalent to simplifying its feature space. The quantization will lead to a certain error (the upper limit of the error is $2^{1-\eta}$) when the two parties exchange the intermediate results of the neural network. However, such a small intermediate result error only has a slight effect on the precision, but it can greatly reduce the communication cost of federated training.



(a) Communication volume with the number of compressed bits (b) Communication volume with different transmission thresholds

Figure 6 Variation of communication traffic with the number of compressed bits/transmission thresholds

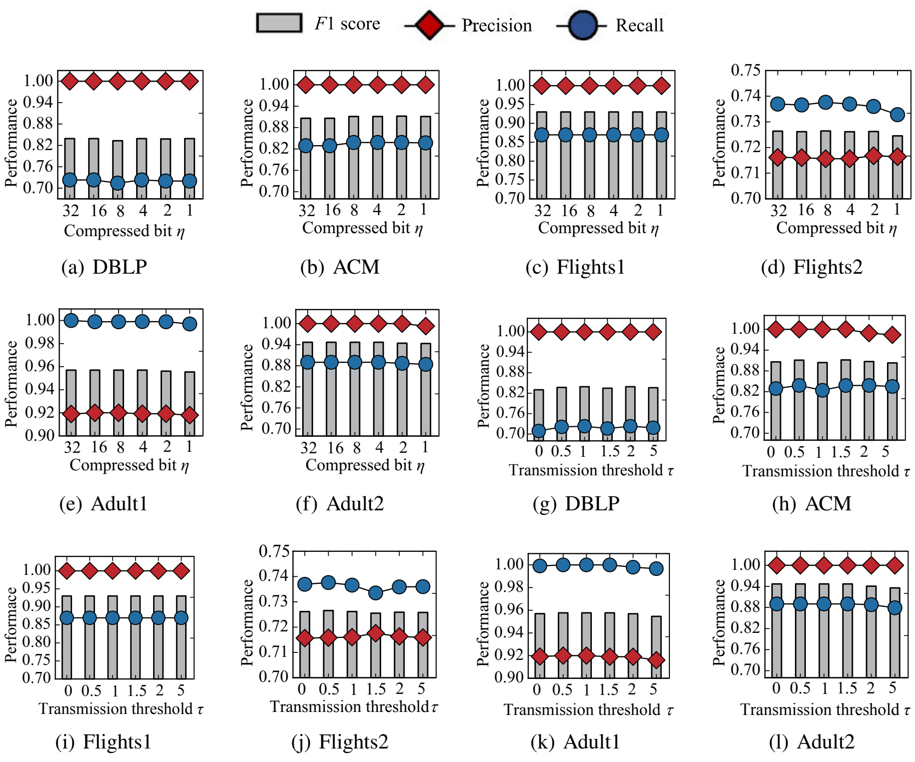


Figure 7 Variation of error detection precision with number of compressed bits/transmission thresholds

Then, the compression parameter η is fixed to 4, and the frequency threshold τ varies in the range of $\{0, 0.5, 1, 1.5, 2, 5\}$ to explore the impact of frequency threshold on communication traffic and error detection accuracy. It should be noted that $\tau = 0$ means that no frequency-reduced optimization is performed. The variation of the communication traffic with τ is shown in Figure 6(b). It can be seen that when the transmission threshold is increased from 0 to 0.5, the communication traffic is significantly reduced. As the transmission threshold increases, more intermediate results are filtered. So, the amount of data that need to be transmitted is significantly reduced. However, as τ continues to increase, the rate of decrease in communication traffic gradually slows down, and the communication traffic tends to be stable. Specifically, when τ increases from 0.5 to 5, the communication traffic is almost stable. This is because in the later stage of training, the model gradually converges, and the update speed of the parameters to be learned in the neural network slows down and tends to be stable, so the difference between the intermediate result matrices generated by two adjacent times is very small, and a very small transmission threshold (such as 0.5) is enough to filter some intermediate results. In the initial stage of training, the model parameters are updated quickly. Even if the transmission threshold is set to 5, the intermediate results generated in the initial stage of training cannot be filtered. As a result, the communication traffic no longer decreases significantly as the threshold increases. As Figure 7(g)–(l) shows, when τ is small, the detection precision is hardly affected. As the transmission threshold increases from 2 to 5, the precision begins to decline. When the transmission threshold is small, the frequency-reduced transmission selectively filters some intermediate results with low information content generated by the neural network and reuses the intermediate results with high information content, so the exchange of key information will not be affected, and the detection results will not be greatly influenced. However, when the threshold value increases to a certain value, some key eigenvectors are filtered (not transmitted), which results in poor model training effect and precision. The smaller threshold can not only ensure the precision of the results but also greatly reduce the communication traffic, which further shows the advantages of frequency-reduced transmission.

5.4.4 *Supplementary experiment*

Next, (1) the training time *TT* of GEDM and FeLeDetecton different datasets is given, (2) the generality of FeLeDetect in datasets with different error rates and error type ratios is verified, and (3) the extensibility of FeLeDetect is verified.

First, Table 4 shows the model training time *TT* of the GEDM in local and centralized scenarios and the collaborative training time *TT* of FeLeDetect in the federated scenario. It should be noted that as described in Section 5.3, model training refers to training for 300 epochs at a given batch size on the training dataset; error detection refers to using a trained model to detect errors on the complete dataset. The results of the comparison between different datasets clearly show that the large dataset Adult takes a longer time to run. Meanwhile, the training time is closely related to the batch size. A dataset with larger batch size, such as Flights, requires less training time. This is because the number of iterations in each epoch is equal to the ratio of the training dataset size to the batch size. When the training dataset size is similar, a larger batch size means fewer iterations, so the neural network conducts less forward propagation and backward propagation, and the training time required is naturally shorter. In addition, according to the error detection time in Table 3, it can be seen that the error detection time of GEDM/FeLeDetect is several orders of magnitude faster than the training time. Even on a large dataset (such as Adult), the error detection process of FeLeDetect takes only 386.97 s, which is much lower than the running time of Raha.

Then, with dataset D-A as an example, the generality of FeLeDetect is verified on datasets with different error rates and error type ratios. As mentioned in Section 1, data errors are sparse in the real world, so the dataset's error rate varies from 3% to 9% under the default error type ratio (80%), and the $F1$ score of FeLeDetect is tested. In addition, under the condition of a default error rate (5%), the error type ratio is changed from 20% to 80%, and the $F1$ score of FeLeDetect is tested. Since the results on each dataset are similar, only the error detection results

Table 4 Training time of GEDM in the local scenario (L) and the centralized scenario (C)/FeLeDetect in the federated scenario (F)

Dataset		GEDM(L)	GEDM(C)	FeLeDetect(F)
D-A	DBLP	270.28	633.75	1443.04
	ACM	286.64		
Flights	Flights1	67.02	65.86	525.84
	Flights2	61.69		
Adult	Adult1	1,023.32	1,711.02	4,474.41
	Adult2	1,005.63		

Table 5 $F1$ score of error detection byFeLeDetect under different error rates/error type ratios

Dataset		Error rate				Error type ratio			
		3%	5%	7%	9%	80%	60%	40%	20%
D-A	DBLP	0.78	0.84	0.83	0.85	0.83	0.88	0.89	0.89
	ACM	0.89	0.91	0.93	0.97	0.90	0.91	0.94	0.97

on dataset D-A are reported here. The experimental results are given in Table 5. As Table 5 shows, the $F1$ score increases slightly as the error rate increases. This is because the sparsity of errors makes error detection more difficult. Second, it can be seen that the $F1$ score slightly increases as the error type ratio decreases. This is because as the error type ratio decreases, the proportion of missing errors or format errors increases, while that of SEs drops. Compared with SEs, missing data and format errors are easier to be detected, so the detection results are more accurate. However, even when SEs account for the highest proportion (80%), FeLeDetect still achieves a high $F1$ score. This is because GEDM can effectively capture the feature of each data source, and the FCTA ensures no loss of cross-source data information. Therefore, FeLeDetect is applicable to datasets with different error rates and different error type ratios.

Finally, the scalability of FeLeDetect is verified. In this experiment, the dataset Adult is selected, and its dataset size is changed. Table 6 shows the training time *TT* and error detection time *DT* of FeLeDetect under different data scales, and the training time and data scale are calculated, respectively. In addition, the Pearson correlation coefficient of error detection time and data size are deduced to assess the linear correlation between them. It can be seen from Table 6 that the training time and error detection time of FeLeDetect increase approximately linearly with the dataset size (the Pearson correlation coefficient is close to 1). This proves that FeLeDetect has great scalability.

Table 6 Comparison between FeLeDetect's *TT* and *DT* with the variation of dataset size

Data size $\|D\|$	*TT*	*DT*
20,000	818.25	39.45
40,000	1,518.05	99.66
60,000	2,237.33	187.97
80,000	3,407.48	299.29
97,864	4,369.12	386.97
Pearson correlation coefficient	0.99	0.99

6 Conclusion

This paper proposed FeLeDetect, a cross-source data error detection method based on federated learning. First, a Graph-based Error Detection Model (GEDM) has been presented to capture sufficient data features from each data source. This paper also proposed an information-lossless federated co-training algorithm, namely FCTA, which collaboratively trains GEDM from different data sources while guaranteeing the privacy and security of cross-source data. Furthermore, the paper designed a series of optimization methods to reduce communication costs during the federated learning and manual labeling efforts. Finally, experiments have been conducted on open datasets and prove that (1) GEDM is superior in both local and centralized scenarios; (2) on the basis of GEDM, FeLeDetect further improves the precision of error detection; (3) the communication optimization algorithm proposed in this paper greatly reduces the communication costs during the federated training process. Since datasets with real errors usually have no ground truth, the effectiveness of the error detection algorithm cannot be evaluated. In the future, we plan to collect, label, and publish benchmark datasets with real errors for subsequent studies on error detection.

References

[1] Ilyas IF, Chu X. Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends in Databases, 2015, 5(4): 281–93.

[2] Heidari A, McGrath J, Ilyas IF, Rekatsinas T. HoloDetect: Few-shot learning for error detection. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 829–846.

[3] Guo ZM, Zhou AY. A survey of data quality and data cleaning research. Ruan Jian Xue Bao/Journal of Software, 2002, 13(11): 2076–2082. http://www.jos.org.cn/jos/article/abstract/20021103?st=search

[4] Wang P, He Y. Uni-Detect: A unified approach to automated error detection in tables. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 811–828.

[5] Dasu T, Loh JM. Statistical distortion: Consequences of data cleaning. Proc. of the VLDB Endowment, 2012, 5(11): 1674–1683.

[6] Wu E, Madden S. Scorpion: Explaining away outliers in aggregate queries. Proc. of the VLDB Endowment, 2013, 6(8): 553–564.

[7] Prokoshyna N, Szlichta J, Chiang F, Miller RJ, Srivastava D. Combining quantitative and logical data cleaning. Proc. of the VLDB Endowment, 2015, 9(4): 300–311.

[8] Elmagarmid AK, Ipeirotis PG, Verykios VS. Duplicate record detection: A survey. IEEE Trans. on Knowledge and Data Engineering, 2006, 19(1): 1–16.

[9] Naumann F, Herschel M. An introduction to duplicate detection. Synthesis Lectures on Data Management, 2010, 2(1): 1–87.

[10] Kandel S, Paepcke A, Hellerstein J, Heer J. Wrangler: Interactive visual specification of data transformation scripts. Proc. of the Int'l Conf. on Human Factors in Computing Systems. Vancouver: ACM, 2011. 3363–3372.

[11] Dallachiesa M, Ebaid A, Eldawy A, Elmagarmid A, Ilyas, IF, Ouzzani M, Tang N. NADEEF: A commodity data cleaning system. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM, 2013. 541–552.

[12] Khayyat Z, Ilyas IF, Jindal A, Madden S, Ouzzani M, Papotti P, Quiané-Ruiz JA, Tang N, Yin S. Bigdansing: A system for big data cleansing. Proc. of the ACM Int'l Conf. on Management of Data. Victoria: ACM, 2015. 1215–1230.

[13] Fan W, Li J, Ma S, Tang N, Yu W. Towards certain fixes with editing rules and master data. The VLDB Journal, 2012, 21(2): 213–238.

[14] Chu X, Morcos J, Ilyas IF, Ouzzani M, Papotti P, Tang N, Ye Y. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Victoria: ACM, 2015. 1247–1261.

[15] Panko RR. What we know about spreadsheet errors. Journal of Organizational and End User Computing, 1998, 10(2): 15–21.

[16] Regulation GDP. Regulation (EU) 2016/679 of the European parliament and of the council. Regulation (EU), No.679, 2016.

[17] Cappuzzo R, Papotti P, Thirumuruganathan S. Creating embeddings of heterogeneous relational datasets for data integration tasks. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 1335–1349.

[18] Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, Kiddon C, Konecny J,

Mazzocchi S, McMahan HB, Overveldt TV, Petrou D, Ramage D, Roselander J. Towards federated learning at scale: System design. Proc. of the Machine Learning and Systems. 2019. 374–388.

[19] Abedjan Z, Morcos J, Ilyas IF, Ouzzani M, Papotti P, Stonebraker M. Dataxformer: A robust transformation discovery system. Proc. of the IEEE Int'l Conf. on Data Engineering. Helsinki: IEEE, 2016. 1134–1145.

[20] Mahdavi M, Abedjan Z, Fernandez RC, Madden S, Ouzzani M, Stonebraker M, Tang N. Raha: A configuration-free error detection system. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 865–882.

[21] Krishnan S, Wang J, Wu E, Franklin MJ, Goldberg K. Activeclean: Interactive data cleaning for statistical modeling. Proc. of the VLDB Endowment, 2016, 9(12): 948–959.

[22] Visengeriyeva L, Abedjan Z. Metadata-driven error detection. Proc. of the 30th Int'l Conf. on Scientific and Statistical Database Management. Bozen-Bolzano: ACM, 2018. 1–12.

[23] Huang Z, He Y. Auto-Detect: Data-driven error detection in tables. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Houston: ACM, 2018. 1377–1392.

[24] McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Federated learning of deep networks using model averaging. arXiv:1602.05629, 2016.

[25] Muhammad K, Wang Q, O'Reilly-Morgan D, Tragos E, Smyth B, Hurley N, Geraci J, Lawlor A. Fedfast: Going beyond average for faster training of federated recommender systems. Proc. of the 26th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. ACM, 2020. 1234–1242.

[26] Chen Y, Qin X, Wang J, Yu C, Gao W. Fedhealth: A federated transfer learning framework for wearable healthcare. IEEE Intelligent Systems, 2020, 35(4): 83–93.

[27] Suzumura T, Zhou Y, Baracaldo N, Ye G, Houck K, Kawahara R, Anwar A, Stavarache LL, Watanabe Y, Loyola P, Klyashtorny D, Ludwig H, Bhaskaran K. Towards federated graph learning for collaborative financial crimes detection. arXiv: 1909.12946, 2019.

[28] Ji S, Pan S, Long G, Li X, Jiang J, Huang Z. Learning private neural language modeling with attentive aggregation. Proc. of the Int'l Joint Conf. on Neural Networks. Budapest: IEEE, 2019. 1–8.

[29] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. Proc. of the Conf. on Machine Learning and Systems. Austin, 2020. 429–450.

[30] Wu C, Wu F, Lyu L, Huang Y, Xie X. Communication-efficient federated learning via knowledge distillation. Nature Communications, 2022, 13(1): 1–8.

[31] Liu Y, Kang Y, Zhang X, Li L, Cheng Y, Chen T, Hong M, Yang Q. A communication efficient collaborative learning framework for distributed features. arXiv:1912.11187, 2019.

[32] Horváth S, Ho C Y, Horvath L, Sahu A N, Canini M, Richtárik P. Natural compression for distributed deep learning. arXiv:1905. 10988, 2019.

[33] Goetz J, Malik K, Bui D, Moon S, Liu H, Kumar A. Active federated learning. arXiv:1909.12641, 2019.

[34] Yang Q, Liu Y, Chen TJ, Tong YX. Federated machine learning: Concept and applications. ACM Trans. on Intelligent Systems and Technology, 2019, 10(2): 1–19.

[35] Liang G, Chawathe SS. Privacy-preserving inter-database operations. Proc. of the Int'l Conf. on Intelligence and Security Informatics. Tucson: Springer, 2004. 66–82.

[36] Hu YC, Niu D, Yang JM, Zhou SP. FDML: A collaborative machine learning framework for distributed features. Proc. of the 25th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Anchorage: ACM, 2019. 2232–2240.

[37] Fu FC, Shao YX, Yu LL, Jiang JW, Xue HR, Tao YY, Cui B. VF2Boost: Very fast vertical federated gradient boosting for cross- enterprise learning. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2021. 563–576.

[38] Ge CC, Wang PF, Chen L, Liu XZ, Zheng BH, Gao YJ. CollaborER: A self-supervised entity resolution framework using multi-features collaboration. arXiv:2108.08090, 2021.

[39] Wu ZH, Pan SR, Chen FW, Long GD, Zhang CQ, Philip SY. A comprehensive survey on graph neural networks. IEEE Trans. on Neural Networks and Learning Systems, 2020, 32(1): 4–24.

[40] Yao L, Mao CS, Luo Y. Graph convolutional networks for text classification. Proc. of the AAAI Conf. on Artificial Intelligence. Honolulu: AAAI, 2019. 7370–7377.

[41] Do K, Tran T, Venkatesh S. Graph transformation policy network for chemical reaction prediction. Proc. of the 25th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Anchorage: ACM, 2019. 750–760.

[42] Wu CH, Wu FZ, Cao Y, Huang YF, Xie X. FedGNN: Federated graph neural network for privacy-preserving recommendation. arXiv:2102.04925, 2021.

[43] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. Proc. of Annual Conf. on Neural Information Processing Systems. Long Beach: NIPS, 2017. 1024–1034.

[44] Zhu L, Liu Z, Han S. Deep leakage from gradients. Proc. of Conf. on Advances in Neural Information Processing Systems. Vancouver: NIPS, 2019. 14747–14756.

[45] Uddin M F, Youssef A M. Cryptanalysis of simple substitution ciphers using particle swarm optimization. Proc. of the IEEE Int'l Conf. on Evolutionary Computation. Vancouver: IEEE, 2006. 677–680.

[46] Mudgal S, Li H, Rekatsinas T, Doan A, Park Y, Krishnan G, Deep R, Arcaute E, Raghavendra V. Deep learning for entity matching: A design space exploration. Proc. of the 2018 ACM SIGMOD Int'l Conf. on Management of Data. Houston: ACM, 2018. 19–34.

[47] Li X, Dong X L, Lyons K, Meng W, Srivastava D. Truth finding on the deep Web: Is the problem solved? Proc. of the VLDB Endowment, 2012, 6(2): 97-108.

[48] Abedjan Z, Chu X, Deng D, Fernandez R C, Ilyas I F, Ouzzani M, Papotti P, Stonebraker M, Tang N. Detecting data errors: Where are we and what needs to be done? Proc. of the VLDB Endowment, 2016, 9(12): 993–1004.

[49] Yan JN, Schulte O, Zhang MH, Wang JN, Cheng R. SCODED: Statistical constraint oriented data error detection. Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 845–860.

[50] Ge CC, Gao YJ, Miao XY, Yao B, Wang HB. A hybrid data cleaning framework using Markov logic networks. IEEE Trans. on Knowledge and Data Engineering, 2022, 34(5): 2048–2062.

[51] Powell SG, Baker KR, Lawson B. Errors in operational spreadsheets: A review of the state of the art. Proc. of the 42nd Hawaii Int'l Conf. on System Sciences. Waikoloa: IEEE, 2009. 1–8.

[52] Mariet Z, Harding R, Madden S. Outlier detection in heterogeneous datasets using automatic tuple expansion. Technical Report, MIT-CSAIL-TR-2016-002, Cambridge: Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2016.

[53] Lahijani MM. Semi-supervised data cleaning. [Ph.D. Thesis]. Berlin: Technical University of Berlin, 2020.

[54] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, ChintalaS. PyTorch: An imperative style, high-performance deep learning library. Proc. of Annual Conf. on Neural Information Processing Systems. Vancouver: NIPS, 2019. 8024–8035.

Lu Chen, Ph.D., professor, doctoral supervisor. Her research interests include metric spatial data management, spatio-temporal data management, and database usability.

Yuxiang Guo, Ph.D. candidate. His research interests include data integration and preparation.

Congcong Ge, Ph.D. Her research interests include data integration and cleaning.

Yunjun Gao, Ph.D., professor, doctoral advisor, senior member of CCF. His research interests include database, big data management and analysis, DB and AI integration.

Baihua Zheng, Ph.D., professor, doctoral supervisor. Her research interests include mobile/pervasive computing, spatial databases, and big data analysis.