

DEVICE AND METHOD FOR DETECTING ANOMALIES IN DOUBLE-PARTY INTERACTION DATA

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to devices and methods for detecting anomalies in double-party interaction data.

BACKGROUND

[0002] Identifying behaviours that differ singularly from the majority has become an important area in various applications across many industries. The occurrence of these (usually rare) anomaly behaviours may have serious implications in many domains. In a manufacturing environment, for example, the occurrence of anomaly events may indicate errors in a workflow (e.g. in the interaction of robot devices). In a network security system, anomalous events could mean security breaches or network intrusions and in the e-commerce business, their occurrence could suggest e-commerce frauds such as price/promotion abuse, review/rating gaming, or fraud collusion.

[0003] For this reason, effective approaches for anomaly detection for many domains as mentioned above as well as other domains such as manufacturing, healthcare, insurance, medicine, and many others are desirable.

SUMMARY

[0004] Various embodiments concern a method for detecting anomalies in double-party interaction data, comprising representing interactions between parties of a first group and parties of a second group as a graph, wherein each interaction between a first party of the first group and a second party of the second group is represented by an edge between a respective first node representing the first party and a respective second node representing the second party and wherein information about the first party is assigned to the first node as node attribute information, information about the second party is assigned to the second node as node attribute information and information about the interaction is assigned to the edge as edge attribute information. The method further comprises processing the graph by a graph convolutional neural network having an autoencoder structure, deriving anomaly scores for interactions,

parties of the first group and parties of the second group from a reconstruction loss between the graph and an output of the graph convolutional neural network in response to the graph including at least a loss between the edge attribute information and edge attribute information reconstructed by a decoder of the graph convolutional neural network. Anomalies are detected based on the anomaly scores.

[0005] According to one embodiment, the reconstruction loss includes a loss between node attribute information and node attribute information reconstructed by the decoder.

[0006] According to one embodiment, the reconstruction loss includes a loss between node adjacency information of the graph and node adjacency information reconstructed by the decoder.

[0007] According to one embodiment, the method comprises comparing the anomaly scores with one or more threshold values and detecting an anomaly of an interaction, party of the first group or party of the second group if its anomaly score is above a respective threshold value.

[0008] According to one embodiment, the method comprises training the graph convolutional neural network by, determining, for each training data element of a plurality of training data elements, wherein each training data element comprises a training graph, a reconstruction loss between the training graph and an output of the graph convolutional neural network in response of the training graph and an output of the graph convolutional neural network in response to the graph including at least a loss between the edge attribute information and edge attribute information reconstructed by a decoder of the graph convolutional neural network, and adapting the neural network to reduce an overall loss including the reconstruction losses determined for the training data elements.

[0009] According to one embodiment, the method comprises, for each training data element, sampling sub-graphs of the training graph, for each sampled sub-graph setting a reconstruction target to the sampled sub-graph and expanding the sub-graph by including nodes and edges connected to the sampled sub-graph and computing a reconstruction between the reconstruction target and an output of the graph convolutional neural network in response to the expanded sub-graph, wherein the overall loss includes the reconstruction losses determined for the sub-graphs.

[0010] According to one embodiment, the first group of parties are customers and the second group of parties are service providers.

[0011] According to one embodiment, the interactions are transactions between the first group of parties and the second group of parties.

[0012] According to one embodiment, the method comprises detecting fraud based on the detected anomalies.

[0013] According to one embodiment, the method comprises checking, for each detected anomaly, whether there has been fraud.

[0014] According to one embodiment, the method further comprises utilizing the anomaly score as the input to a human-in-the-loop actioning system and/or an automatic actioning system. These may take corresponding actions if there is an anomaly, such as blocking a customer involved in an anomaly (i.e. an interaction whose anomaly score indicates an anomaly) from certain features.

[0015] According to one embodiment, a server computer is provided including a radio interface, a memory interface and a processing unit configured to perform the method for detecting anomalies in double-party interaction data described above.

[0016] According to one embodiment, a computer program element is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method for detecting anomalies in double-party interaction data described above.

[0017] According to one embodiment, a computer-readable medium is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method for detecting anomalies in double-party interaction data described above.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

- FIG. 1 shows a communication arrangement including a smartphone and a server.
- FIG. 2 illustrates a data processing pipeline according to an embodiment, i.e. implemented by the server.
- FIG. 3 shows a graph anomaly detection neural network according to an embodiment.
- FIG. 4A illustrates the message passing flow for a customer node.

- FIG. 4B illustrates the message passing flow for a service provider node.
- FIG. 4C illustrates the message passing flow for an edge.
- FIG. 5 shows a flow diagram illustrating a method for detecting anomalies in double-party interaction data.
- FIG. 6 shows a server computer according to an embodiment.

DETAILED DESCRIPTION

[0019] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0020] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a vehicle or a method, and vice-versa.

[0021] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0022] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0023] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0024] In the following, embodiments will be described in detail.

[0025] FIG. 1 shows a communication arrangement including a smartphone 100 and a server (computer) 106.

[0026] The smartphone 100 has a screen showing the graphical user interface (GUI) of an app for using one or more of various services, such as ordering food or e-hailing, which the

smartphone's user has previously installed on his smartphone and has opened (i.e. started) to use the service, e.g. to order food.

[0027] The GUI 101 includes graphical user interface elements 102, 103 helping the user to use the service, e.g. a map of a vicinity of the user's position, food available in the user's vicinity (which the app may determine based on a location service, e.g. a GPS-based location service), a button for placing an order, etc.

[0028] When the user has made a selection for a service, e.g. a selection of a restaurant or an online supermarket and/or a selection of food or groceries to order, the app communicates with a server 106 of the respective service via a radio connection. The server 106 (carrying out a corresponding server program by means of a processor 107) may consult a memory 109 or a data storage 108 having information regarding the service (e.g. prices, availability, estimated time for delivery etc.) The server communicates any data relevant or requested by the user (such as estimated time for delivery) back to the smartphone 100 and the smartphone 100 displays this information on the GUI 101. The user may finally accept a service, e.g. order food. In that case, the server 106 informs the service provider 104, e.g. a restaurant or online supermarket accordingly. The server 106 may also communicate earlier with the service provider 104, e.g. for determining the estimated time for delivery.

[0029] It should be noted while the server 106 is described as a single server, its functionality, e.g. for providing a certain service and advertisement data will in practical application typically be provided by an arrangement of multiple server computers (e.g. implementing a cloud service). Accordingly, the functionality described in the following provided by a server (e.g. server 106) may be understood to be provided by an arrangement of servers or server computers.

[0030] The server 106 may store information about interactions between parties, in this example transactions between users like the user of the smartphone 100 and service providers like the service provider 104 in the data storage 108 for analysis.

[0031] These data can thus be modelled as data about double-party interactions. For example, both food order and grocery transactions as well as e-hailing trips (which also can be seen to correspond to a transaction since the passenger pays for the trip) form interactions between customers and merchants (in general service providers including for example also drivers). These interactions are rich with information, for example, information about the orders, food category purchased, prices, drop-off locations, etc. Using these data, it may be

desirable to detect anomalies in these double-party interactions since anomalies in the transactions often indicate the occurrence of fraudulent activities committed by the parties involved. For example, promotion gaming of a customer for a certain merchant where very frequent orders or made, and customer-merchant collusions where the transactions to the merchant are mostly from the colluding customer. Fraudsters tend to repeat the fraudulent behaviours to maximize the gain, thus making the behaviours stand out from others.

[0032] It is desirable that anomaly detection models do not need labelled data such that they can be directly applied to the gathered data without the need to collect labels, which oftentimes are limited and time-consuming to collect. Moreover, fraudsters keep innovating on their fraud modus operandi (MOs), making any model that relies on historical labelled data unable to effectively detect new MOs. However, many classical machine learning algorithms for anomaly detection are unable to model the rich information in the double-party interactions.

[0033] In particular, anomaly models that work on tabular data cannot effectively model the interactions since they force the data to be put in tabular format, which destroys the relational property between two different parties.

[0034] Further, conventional anomaly models that model relational data and work on interaction data are not capable of modelling the rich information attached to the customer-service provider interactions. Typically they focus on the individual property of each entity (e.g., either customer's properties or service provider's properties).

[0035] In view of the above, according to various embodiments, an approach for anomaly detection in double-party interactions is provided which allows the modelling the rich information attached to the interactions.

[0036] FIG. 2 illustrates a data processing pipeline according to an embodiment, i.e. implemented by the server 106.

[0037] First, a data ingestion system 201 gathers raw interaction data between customers (PAX) and service providers (MEX), e.g. in the data storage 108 as described above for food transactions, grocery transactions, e-hailing trips etc.

[0038] The gathered data is then formulated (i.e. represented as an attributed bipartite graph 202, i.e. as a mathematical formulation for representing double-party interactions.

[0039] Specifically, the bipartite graph is a node-and-edge-attributed graph $G = (\mathbf{u}, \mathbf{v}, \mathbf{e}, \mathbf{x}^u, \mathbf{x}^v, \mathbf{x}^e)$. It consists of:

- the first set of nodes $\mathbf{u} = \{u_1, u_2, \dots, u_{n_u}\}$, representing the customer nodes;

- the second set of nodes $\mathcal{V} = \{v_1, v_2, \dots, v_{n_v}\}$, representing the service provider nodes;
- the set of edges E , indexed using $e_{i,j}$ where $i \in \{1, \dots, n_u\}$ and $j \in \{1, \dots, n_v\}$, with n_e represents its cardinality, representing the customer-service provider interactions;
- $X^u \in \mathbb{R}^{n_u \times m_u}$, the attributes/features for every node in $\mathcal{U}$;
- $X^v \in \mathbb{R}^{n_v \times m_v}$, the attributes/features for every node in $\mathcal{V}$;
- $X^e \in \mathbb{R}^{n_e \times m_e}$, the attributes/features for every edge in $\mathcal{E}$.

[0040] Here, n represents the number of nodes/edges in the graph, and m represents the number of node/edge features. The subscripts in n and m indicate which set the numbers represent (e.g., n_u for the set $\mathcal{U}$ and n_e for the set of edges $\mathcal{E}$).

[0041] For example, the data ingestion system 201 gathers data about transactions between customers and service providers (e.g. restaurants and online supermarkets) for the past 7 days. The data may comprise many (e.g. around 30 – 50) features per transaction and/or party (i.e. customer or service provider) for example information about orders, food category purchased, prices, drop-off locations, number of orders, prices, promotions, drop-off location statistics, etc.. From these transactions, the bipartite graph is constructed wherein each node in the bipartite graph represents a customer or a service provider (i.e. e.g. merchant or driver), i.e. each node is either a customer node or a service provider node. An edge between a customer and a service provider exists in the bipartite graph if the customer made at least one transaction with the service provider.

[0042] Each edge is supplied with rich information (i.e. attributes also referred to as features) regarding the transactions between the corresponding customer and service provider (which correspond to the nodes the edge connects). Similarly, each node is also supplied with information about the customer's or service provider's profile, respectively.

[0043] So, for example, the transactions between customers and service providers over multiple services are monitored, e.g. for both food and online supermarket orders and, for example, every day, the data ingestion system 201 extracts the food and online supermarket transactions in the past seven days in the dataset, as well as additional information, including number of orders, prices, promotions, order information, drop-off location statistics, etc. From these transactions, the server 106 constructs a bipartite graph representation of the dataset.

[0044] The bipartite graph 202 constructed by the data ingestion system 201 is then passed to a graph anomaly detection machine learning (ML) model 203, e.g. neural network (e.g. implemented by the server 106). The anomaly ML model 203 is a graph neural network model designed specifically for anomaly detection in bipartite graphs. The input to the ML model 203 is an attributed bipartite graph 202 which comprises features (or attributes) to encode the gathered data as described above. From the output of the graph anomaly detection neural network 203 in response to the bipartite graph 202, an anomaly scoring system 204 determines anomaly scores 204 for each customer-service provider pair, as well as an anomaly score for every customer and every provider pair.

[0045] The anomaly scores may then be evaluated by an actioning system 205 (e.g. by comparing them with one or more thresholds) which may take actions in response to the detection of anomalies. For example, the anomaly score for customer-service provider pairs is used for determining whether the pair is anomalous, such a customer colluding with service provider for abusing promotions.

[0046] The graph anomaly detection neural network 203 is first trained before it is used for anomaly detection. After the graph anomaly detection neural network 203 as been trained, it may be used to generate the anomaly scores. The anomaly score is based on each individual node/edge's reconstruction error from the graph anomaly detection neural network 203 output. This can be seen to be based on the assumption that normal behaviours are common and hence can be easily reconstructed, whereas anomalous behaviours are rare and the model will have problems in reconstructing them. Consequently, the anomalous edges/nodes will have higher reconstruction error, and thus higher anomaly score. The anomaly scores may be in an arbitrary range of non-negative numbers. In practical application, most of the anomaly scores are typically close to 0, but some can be in the tens, or hundreds.

[0047] The anomaly scoring system 204 first calculates the anomaly score for each edge (i.e. each customer-service provider relation) using the respective edge reconstruction error. Then it calculates the anomaly score for the nodes (i.e. customers and service providers). The anomaly score for a node does not only depend on the customer's or service provider's reconstruction error, but also considers all edges that are connected to the node. The anomaly scores for each customer, each service provider, and each customer-service provider interactions for every run day may be stored in the database 108.

[0048] The anomaly scores produced by the anomaly scoring system 204 are then passed to the actioning system 205. The actioning system 205 may for example comprise a human-in-the-loop system, and an automatic actioning system. For the human-in-the-loop system, a set of human expert is provided with sets of most anomalous customers, merchants, and customer-merchant interactions. The human evaluator then manually investigates anomalies, evaluating if a particular modus operandi (old or new) occurred, and decides on what action to perform with regard to the suspected customer or service provider. For the automatic actioning system, the anomaly score is combined with other signals or data to create an automatic actioning task like for example, automatically banning certain promos for users with high anomaly score in the past few days.

[0049] The graph anomaly detection neural network 203 is now described in more detail with reference to FIG: 3.

[0050] FIG. 3 shows a graph anomaly detection neural network 300 according to an embodiment.

[0051] The graph anomaly detection neural network 300 includes an encoder 301 and a decoder which includes a feature decoder 302 and a structure decoder 303. The graph anomaly detection neural network 300 is trained in accordance with a loss function 304 (i.e. in training it is adapted to reduce the value of the loss function for training data).

[0052] The encoder 301 takes the input graph 305 (including the feature/attribute information), and processes it in a series of graph convolution layers. According to various embodiments, a customized convolution operation is used that works on a bipartite graph with both node and edge features. Each convolution layer is followed by a batch normalization and a ReLU Rectified Linear Unit activation function, to produce the layer's representation of node and edge, denoted as H^u , H^v , and H^e . The last layer of the encoder 301 produces a latent representation 306 for each node denoted as Z^u and Z^v , respectively.

[0053] The feature decoder 302 is trained to aim to reconstruct the original input features from the latent representation 306 by passing Z^u and Z^v to a series of graph convolution layers. The final output is the reconstructed node and edge features, denoted as $\hat{X}_u$, $\hat{X}_v$, and $\hat{X}_e$, respectively.

[0054] The structure decoder 303 is trained to aim to reconstruct the original graph structure from the latent representation 306. It works by forming a matrix representation of Z^u and Z^v ,

applying multi-layer perceptron (MLPs) on both Z^u and Z^v , and then using the sigmoid function to produce the probability that node u is connected to node v via an edge.

[0055] For training the graph anomaly detection neural network 300, the loss function 304 includes (for each training data element (i.e. training input graph) of the training data) a reconstruction error which is a combination of the mean squared error of the feature reconstruction (feature decoder output) and the binary cross-entropy of the edge prediction (structure decoder output).

[0056] In the following, the operation of the graph anomaly detection neural network 300 is described in further detail, starting with some notations used in the following.

[0057] The structure of the attributed graph G is represented in by adjacency-like matrix $\mathbf{A} \in \mathbb{R}^{n_u \times n_v}$, where its (i, j) -th item $A_{i,j} = 1$ if there is an edge between u_i and v_j and $A_{i,j} = 0$ else. It should be noted that unlike the adjacency matrix in a regular (homogeneous) graph, the matrix $\mathbf{A}$ does not need not be a square matrix as the number of nodes in $\mathcal{U}$ and $\mathcal{V}$ can be different. Further, $\mathcal{N}$ is defined as the neighbouring operator that returns the list of all of neighbouring nodes. Specifically, for a node u in the first node set (customer nodes),

$$\mathcal{N}(u_i) = \{v_j \in \mathcal{V} \mid A_{i,j} = 1; j \in \{1, \dots, n_v\}\}$$

Similarly, for a node v in the second node set (service provider nodes),

$$\mathcal{N}(v_j) = \{u_i \in \mathcal{U} \mid A_{i,j} = 1; i \in \{1, \dots, n_u\}\}$$

[0058] In addition, another function $\mathcal{M}$ is defined which behaves similarly to $\mathcal{N}$. However, instead of returning neighbouring nodes, it returns the edges connecting the target node to the neighbouring nodes. In particular,

$$\mathcal{M}(u_i) = \{e_{i,j} \in \mathcal{E} \mid A_{i,j} = 1; j \in \{1, \dots, n_v\}\}$$

And

$$\mathcal{M}(v_j) = \{e_{i,j} \in \mathcal{E} \mid A_{i,j} = 1; i \in \{1, \dots, n_u\}\}$$

The anomaly detection task can now be formalized as follows: given a bipartite node-and-edge-attributed graph G , the task is to produce scoring functions applicable to each edge $e \in \mathcal{E}$, as well as each node $u \in \mathcal{U}$ and $v \in \mathcal{V}$, such that the edges/nodes that differ singularly from the majority in terms both the structure and attribute information should be given higher anomaly score.

[0059] Specifically, the task includes providing three scoring functions: $\text{score}_e(\cdot)$, $\text{score}_u(\cdot)$, and $\text{score}_v(\cdot)$ each for scoring the edges, the nodes in $\mathcal{U}$ and the nodes in $\mathcal{V}$, respectively.

[0060] The key components of the graph anomaly detection neural network 300 as described in the following may be seen in the graph convolution operation and the message passing scheme that compute the next (i.e. next layer's) node and edge representations from the current ones (i.e. current layer's).

[0061] In the following, lower case $\mathbf{x}$ is used to denote node/edge features, with a sub-script to enumerate the nodes/edges. Let $\mathbf{x}_i^u$ denote the features for node u_i in $\mathcal{U}$ with $i \in \{1, \dots, n_u\}$, $\mathbf{x}_j^v$ denote the features for node v_j in $\mathcal{V}$ with $j \in \{1, \dots, n_v\}$, whereas $\mathbf{x}_{i,j}^e$ denotes the features for edge $e_{i,j}$ in $\mathcal{E}$.

[0062] The graph anomaly detection neural network 300 comprises several convolution layers. Let K denote the number of convolution layers (even number), where half of are part of the encoder 301 and the other are part of the feature decoder 302. The lower case k is used to enumerate the convolution layers. To represent the intermediate node/edge representations of layer k , we use, lower case $\mathbf{h}$ is used in the following and a super-script $\{.\}^{(k)}$ is added to the notation to specify the layer. Specifically, $\{\mathbf{h}_i^u\}^{(k)}$, $\{\mathbf{h}_j^v\}^{(k)}$, and $\{\mathbf{h}_{i,j}^e\}^{(k)}$ denote the k -th layer representation for node u_i , node v_j , and edge $e_{i,j}$, respectively.

[0063] The convolution operation of layer k takes the node representation of each node of layer $k-1$, i.e. $\{\mathbf{h}_i^u\}^{(k-1)}$ and $\{\mathbf{h}_j^v\}^{(k-1)}$ as well as the edge representations of layer k -

1, i.e. $\{\mathbf{h}_{i,j}^e\}^{(k-1)}$ and performs message passing and outputs new node and edge representations $\{\mathbf{h}_i^u\}^{(k)}$, $\{\mathbf{h}_j^v\}^{(k)}$ and $\{\mathbf{h}_{i,j}^e\}^{(k)}$.

[0064] Instead of training distinct representation for each node, a set of aggregator functions is trained that learns to aggregate feature information from a node's local neighbourhood.

[0065] To compute the new representations $\{\mathbf{h}_i^u\}^{(k)}$, $\{\mathbf{h}_j^v\}^{(k)}$ for node or edge, the k th convolutional layer first collects the messages passed to the particular node/edge. The message to a node u_i or v_j come from the aggregator functions over the neighbouring node representations, the aggregator functions over the edge representations connected to u_i , and its own previous representations. The messages passed to an edge $e_{i,j}$ come from the nodes connected to the edge, i.e., u_i and v_j , and the edges own previous representation.

[0066] FIG. 4A illustrates the message passing flow for a customer node.

[0067] FIG. 4B illustrates the message passing flow for a service provider node.

[0068] FIG. 4C illustrates the message passing flow for an edge.

[0069] The following equations describe formally how the messages passed to u_i , v_j and $e_{i,j}$ are computed:

$$\begin{aligned} \text{Msg}[\rightarrow u_i]^{(k)} = & \left\{ \{\mathbf{h}_i^u\}^{(k-1)} \cup \text{Agg} \left(\left\{ \{\mathbf{h}_j^v\}^{(k-1)} \mid \forall v_j \in \mathcal{N}(u_i) \right\} \right) \right. \\ & \left. \cup \text{Agg} \left(\left\{ \{\mathbf{h}_{i,j}^e\}^{(k-1)} \mid \forall e_{i,j} \in \mathcal{M}(u_i) \right\} \right) \right\} \quad (1) \end{aligned}$$

$$\begin{aligned} \text{Msg}[\rightarrow v_j]^{(k)} = & \left\{ \{\mathbf{h}_j^v\}^{(k-1)} \cup \text{Agg} \left(\left\{ \{\mathbf{h}_i^u\}^{(k-1)} \mid \forall u_i \in \mathcal{N}(v_j) \right\} \right) \right. \\ & \left. \cup \text{Agg} \left(\left\{ \{\mathbf{h}_{i,j}^e\}^{(k-1)} \mid \forall e_{i,j} \in \mathcal{M}(v_j) \right\} \right) \right\} \quad (2) \end{aligned}$$

$$\text{Msg}[\rightarrow e_{i,j}]^{(k)} = \left\{ \{\mathbf{h}_{i,j}^e\}^{(k-1)} \cup \{\mathbf{h}_i^u\}^{(k-1)} \cup \{\mathbf{h}_j^v\}^{(k-1)} \right\}. \quad (3)$$

[0070] Here, $\cup$ represents the concatenation operator, where messages that come from different sources are concatenated. The aggregation function Agg could be a simple aggregation function like Mean and Max, a combination of both, or more complex aggregation functions mentioned like LSTM (Long Short-Term Memory) and Pooling.

[0071] After collecting the messages for each node and edge, the convolutional layer computes the next representations (i.e. its output representations). For this, the messages processed by a linear operation parameterized by parameters W and b . The output of the linear operation is normalized passed to an activation function as follows:

$$\{\mathbf{h}_i^u\}^{(k)} = \text{ReLU} \left(\text{BN} \left(\mathbf{W}_u^{(k)} \cdot \text{Msg}[\rightarrow u_i]^{(k)} + \mathbf{b}_u^{(k)} \right) \right) \quad (4)$$

$$\{\mathbf{h}_j^v\}^{(k)} = \text{ReLU} \left(\text{BN} \left(\mathbf{W}_v^{(k)} \cdot \text{Msg}[\rightarrow v_j]^{(k)} + \mathbf{b}_v^{(k)} \right) \right) \quad (5)$$

$$\{\mathbf{h}_{i,j}^e\}^{(k)} = \text{ReLU} \left(\text{BN} \left(\mathbf{W}_e^{(k)} \cdot \text{Msg}[\rightarrow e_{i,j}]^{(k)} + \mathbf{b}_e^{(k)} \right) \right), \quad (6)$$

where BN represents a batch normalization layer. The weight and bias parameters in the linear operation for each node-set are shared across all nodes in the set. Similarly, the parameters for the edge set are shared across all edges. Hence, in one convolution layer, there are six parameter vectors: W^u , W , W^e , b^u , b^v , and b^e . These parameters are trained when training the neural network 300.

[0072] As mentioned above, the encoder 301 comprises $K/2$ graph convolution layers. The input to the first layer is the original feature for each node and edge of the input graph 305, i.e.

$$\{\mathbf{h}_i^u\}^{(0)} = \mathbf{x}_i^u \quad \{\mathbf{h}_j^v\}^{(0)} = \mathbf{x}_j^v \quad \{\mathbf{h}_{i,j}^e\}^{(0)} = \mathbf{x}_{i,j}^e. \quad (7)$$

[0073] The subsequent layers take the output of the previous layer as the input. Every layer outputs new node and edge representations, $\{\mathbf{h}_i^u\}^{(k)}$, $\{\mathbf{h}_j^v\}^{(k)}$ and $\{\mathbf{h}_{i,j}^e\}^{(k)}$, except the last encoder layer (i.e. the $K/2$ -th layer) which only outputs node representations $\{\mathbf{h}_i^u\}^{(K/2)}$ and $\{\mathbf{h}_j^v\}^{(K/2)}$.

[0074] These node representations become the latent representation 306 (denoted as z) for each node in U and V , i.e.

$$\mathbf{z}_i^u = \{\mathbf{h}_i^u\}^{(K/2)} \quad \mathbf{z}_j^v = \{\mathbf{h}_j^v\}^{(K/2)}. \quad (8)$$

[0075] This construction forces the latent representation of each node to also encode all neighbouring edge features in addition to the local graphical structure and neighbouring node features.

[0076] The feature decoder 302 also consists of $K/2$ convolution layers. The first layer takes the latent representations 306 of the nodes in U and V as the input without accepting edge

representations. Therefore the message collection in the message passing scheme is simplified to:

$$\text{Msg}[\rightarrow u_i]^{(K/2+1)} = \left\{ \mathbf{z}_i^u \cup \text{Agg} \left(\left\{ \mathbf{z}_j^v \mid \forall v_j \in \mathcal{N}(u_i) \right\} \right) \right\} \quad (9)$$

$$\text{Msg}[\rightarrow v_j]^{(K/2+1)} = \left\{ \mathbf{z}_j^v \cup \text{Agg} \left(\left\{ \mathbf{z}_i^u \mid \forall u_i \in \mathcal{N}(v_j) \right\} \right) \right\} \quad (10)$$

$$\text{Msg}[\rightarrow e_{i,j}]^{(K/2+1)} = \left\{ \mathbf{z}_i^u \cup \mathbf{z}_j^v \right\}. \quad (11)$$

[0077] The remaining convolution layers for the feature decoder 302 follow the scheme as described for the encoder 301. The output of the last layer of the decoder 302 becomes the reconstructed node and edge features, i.e.:

$$\hat{\mathbf{x}}_i^u = \{\mathbf{h}_i^u\}^{(K)} \quad \hat{\mathbf{x}}_j^v = \{\mathbf{h}_j^v\}^{(K)} \quad \hat{\mathbf{x}}_{i,j}^e = \{\mathbf{h}_{i,j}^e\}^{(K)}, \quad (12)$$

which are compared to the original input features for calculation of the loss 304 (in training) and the anomaly scores (in inference).

[0078] In the feature decoder 302, the graph structure is given to the model as the basis for passing the messages. Therefore, in order to strengthen the latent variable's encoding of the graph structure, the structure decoder 303 is included in the neural network 300. From the latent representations 306 $\mathbf{z}_i^u$ and $\mathbf{z}_j^v$, the structure decoder 303 tries to predict if a node u_i in $\mathcal{U}$ is connected to a node v_j in $\mathcal{V}$. It works by passing $\mathbf{z}_i^u$ to $K/2$ layers of a first multi layer perceptron 307 (MLP_u) and passing $\mathbf{z}_j^v$ to a similar (second) MLP 308 (MLP_v). It then uses the dot product of the output of the MLPs, and passes it to a sigmoid function 309 to produce the probability of u_i being connected to v_j , i.e.:

$$\Pr(\mathbf{A}_{i,j} = 1) = \text{sigmoid} \left(\text{MLP}_u(\mathbf{z}_i^u)^\top \cdot \text{MLP}_v(\mathbf{z}_j^v) \right). \quad (13)$$

[0079] These probabilities are then compared to the adjacency matrix $\mathbf{A}$ for calculation of the loss 304 (in training) and the anomaly scores (in inference).

[0080] In the following, training of the neural network 300 is explained.

[0081] The edge prediction formula in the structure decoder (Eq. 13) defines the probability for every pair of nodes in $\mathcal{U}$ and $\mathcal{V}$. In practice, the majority of node pairs are not connected. Therefore, to gain more efficiency, the probability is computed for a subset of the pairs without

incurring any performance degradation. Let $\mathcal{E}^+$ denote the set of pairs where u_i and v_j are connected in the graph, whereas $\mathcal{E}^-$ denotes the set of pairs that are not connected, i.e.:

$$\mathcal{E}^+ = \{(i, j) \mid A_{i,j} = 1\}, \quad \mathcal{E}^- = \{(i, j) \mid A_{i,j} = 0\}. \quad (14)$$

[0082] The set $\mathcal{E}^\pm$ is defined as the set of all indices $\mathcal{E}^\pm = \mathcal{E}^+ \cup \mathcal{E}^-$. According to one embodiment, in the training procedure, not all items in $\mathcal{E}^-$ are used. Rather, only a small portion of $\mathcal{E}^-$ is sampled and combined with $\mathcal{E}^+$ to get the set of pairs used in the training $\mathcal{E}^\pm$, i.e.

$$\tilde{\mathcal{E}}^\pm = \mathcal{E}^+ \cup \text{Sample}(\mathcal{E}^-). \quad (15)$$

[0083] Standard uniform random sampling of the negative pairs $\mathcal{E}^-$ with a pre-specified number of samples, e.g., a small multiple of the number of positive pairs ($\mathcal{E}^+$) may be used.

[0084] The loss function 304 is a type of reconstruction loss. Let $\hat{\mathbf{X}}^u$, $\hat{\mathbf{X}}^v$ and $\hat{\mathbf{X}}^e$ be the matrices that contain the network's output for all nodes in $\mathbf{u}$, all nodes in $\mathbf{v}$, and all edges respectively. The training objective function (i.e. loss function) 304 is the combination of the mean squared error (MSE) of node feature reconstruction of the nodes in $\mathbf{u}$, the MSE of node feature reconstruction of the nodes in $\mathbf{v}$, the MSE of edge feature reconstruction and the binary cross entropy (BCE) of the structure decoder's edge prediction:

$$\begin{aligned} \mathcal{L} &= \text{MSE}(\mathbf{X}^u, \hat{\mathbf{X}}^u) + \text{MSE}(\mathbf{X}^v, \hat{\mathbf{X}}^v) + \text{MSE}(\mathbf{X}^e, \hat{\mathbf{X}}^e) + \eta \text{BCE}(\tilde{\mathcal{E}}^\pm) \\ \text{where: } \text{MSE}(\mathbf{X}, \hat{\mathbf{X}}) &= \frac{1}{nm} \sum_{i=1}^n \sum_{l=1}^m (\mathbf{X}_{i,l} - \hat{\mathbf{X}}_{i,l})^2, \text{ and} \\ \text{BCE}(\tilde{\mathcal{E}}^\pm) &= -\frac{1}{|\tilde{\mathcal{E}}^\pm|} \sum_{(i,j) \in \tilde{\mathcal{E}}^\pm} \{A_{i,j} \log(\text{Pr}(A_{i,j} = 1)) \\ &\quad + (1 - A_{i,j}) \log(1 - \text{Pr}(A_{i,j} = 1))\}, \quad (16) \end{aligned}$$

where η denotes a constant for balancing the MSE losses from the feature decoder 302 and the BCE loss from the structure decoder 303.

[0085] In the training process, forward propagation is first performed by feeding the input data (i.e. the training data elements, i.e. training input graphs) to the encoder 301 to produce the latent representations 306 which are then passed to both feature decoder 302 and structure

decoder 303 to produce the reconstructed data. The value of the loss function is then computed as objective of the neural network optimization.

[0086] The training data elements may be input graphs based on historical or simulated transactions, for example from double party interactions between customer and merchant of a food delivery service (e.g. GrabFood) and/or mart transactions (e.g. GrabMart). Depending on the use case, other double-party interactions relevant for the respective use case can also be used such as data on passenger-driver interactions (e.g. from GrabCar and/or GrabBike).

[0087] Algorithm 1 provides the detailed step-by-step process of performing forward propagation, covering all the network components as discussed above.

Data: graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, X^u, X^v, X^e)$; even number of layer K ; parameters: $\mathbf{W}_u^{(k)}, \mathbf{W}_v^{(k)}, \mathbf{W}_e^{(k)}, \mathbf{b}_u^{(k)}, \mathbf{b}_v^{(k)}$, and $\mathbf{b}_e^{(k)}$; structure decoder's MLPs: MLP_u and MLP_v

Result: loss/objective value

```

/* initial representations */
 $\{\mathbf{h}_i^u\}^{(0)} \leftarrow \mathbf{x}_i^u, \forall i \in \{1, \dots, n_u\};$ 
 $\{\mathbf{h}_j^v\}^{(0)} \leftarrow \mathbf{x}_j^v, \forall j \in \{1, \dots, n_v\};$ 
 $\{\mathbf{h}_{i,j}^e\}^{(0)} \leftarrow \mathbf{x}_{i,j}^e, \forall e_{i,j} \in \mathcal{E};$ 
/* encoder */
for  $k \leftarrow 1$  to  $K/2$  do
    compute messages:  $\text{Msg}[\rightarrow u_i]^{(k)}$  and  $\text{Msg}[\rightarrow v_j]^{(k)}$ ,
    for all nodes in  $\mathcal{U}$  and  $\mathcal{V}$  using Eq. (1) and Eq. (2);
    compute all node representations:  $\{\mathbf{h}_i^u\}^{(k)}$  and  $\{\mathbf{h}_j^v\}^{(k)}$ ,
    using Eq. (4) and Eq. (5);
    if  $k \neq K/2$  then
        compute  $\text{Msg}[\rightarrow e_{i,j}]^{(k)}$  for all edges using Eq. (3);
        compute edge repr.:  $\{\mathbf{h}_{i,j}^e\}^{(k)}$  using Eq. (6);

```

```

/* latent representations */
 $\mathbf{z}_i^u \leftarrow \{\mathbf{h}_i^u\}^{(K/2)}, \forall i \in \{1, \dots, n_u\};$ 
 $\mathbf{z}_j^v \leftarrow \{\mathbf{h}_j^v\}^{(K/2)}, \forall j \in \{1, \dots, n_v\};$ 
/* feature decoder */
for  $k \leftarrow K/2 + 1$  to  $K$  do
    if  $k = K/2 + 1$  then
        compute  $\text{Msg}[\rightarrow u_i]^{(k)}$ ,  $\text{Msg}[\rightarrow v_j]^{(k)}$ , and
         $\text{Msg}[\rightarrow e_{i,j}]^{(k)}$  Eq. (9), Eq. (10), and Eq. (11);
    else
        compute  $\text{Msg}[\rightarrow u_i]^{(k)}$ ,  $\text{Msg}[\rightarrow v_j]^{(k)}$ , and
         $\text{Msg}[\rightarrow e_{i,j}]^{(k)}$  Eq. (1), Eq. (2), and Eq. (3);
    compute all representations:  $\{\mathbf{h}_i^u\}^{(k)}$ ,  $\{\mathbf{h}_j^v\}^{(k)}$  and
     $\{\mathbf{h}_{i,j}^e\}^{(k)}$ , using Eq. (4), Eq. (5), and Eq. (6);
/* reconstructed features */
 $\hat{\mathbf{x}}_i^u \leftarrow \{\mathbf{h}_i^u\}^{(K)}, \forall i \in \{1, \dots, n_u\};$ 
 $\hat{\mathbf{x}}_j^v \leftarrow \{\mathbf{h}_j^v\}^{(K)}, \forall j \in \{1, \dots, n_v\};$ 
 $\hat{\mathbf{x}}_{i,j}^e \leftarrow \{\mathbf{h}_{i,j}^e\}^{(K)}, \forall e_{i,j} \in \mathcal{E};$ 
/* structure decoder */
perform negative sampling, combine it with  $\mathcal{E}$  as  $\tilde{\mathcal{E}}^\pm$ ;
compute edge probability  $\Pr(\mathbf{A}_{i,j} = 1), \forall (i, j) \in \tilde{\mathcal{E}}^\pm$ ;
/* loss/objective value */
 $\ell \leftarrow$  compute loss using Eq. (16);
return  $\ell$ 

```

Algorithm 1

[0088] For computing backward propagation to compute the gradients and updating the neural network 300, a deep learning framework may be used.

[0089] After training the network, the anomaly scoring system 204 may use the neural network 300 to determine the anomaly scores as described above and explained in the following in more detail.

[0090] According to one embodiment, the anomaly scoring system 204 generates the values of three anomaly scoring functions for scoring the edges as well as the nodes in $\mathbf{u}$ and $\mathbf{v}$. Given an input graph 305, the scoring functions are computed by first performing a forward

propagation through the encoder 301 and the decoders 303, 304 and then computing individual node/edge reconstruction errors. The edge scoring function is defined as the weighted combination of the reconstruction error of the edge features and the BCE error of predicting if the edge should exist in the graph, i.e.:

$$\text{score}_e(e_{i,j}) = \text{MSE}(\mathbf{x}_{i,j}^e, \hat{\mathbf{x}}_{i,j}^e) + \eta \text{BCE}(e_{i,j}), \quad (17)$$

where the constant η is the same constant used in the training objective. After that, we define the scoring function for the nodes in $\mathbf{u}$ and $\mathbf{v}$. The anomaly score of a node is the combination of the reconstruction error of its node feature and aggregation of the anomaly score of all edges connected to the node, i.e.:

$$\text{score}_u(u_i) = \text{MSE}(\mathbf{x}_i^u, \hat{\mathbf{x}}_i^u) + \text{Agg}_{e_{i,j} \in \mathcal{M}(u_i)} \text{score}_e(e_{i,j}) \quad (18)$$

$$\text{score}_v(v_j) = \text{MSE}(\mathbf{x}_j^v, \hat{\mathbf{x}}_j^v) + \text{Agg}_{e_{i,j} \in \mathcal{M}(v_j)} \text{score}_e(e_{i,j}). \quad (19)$$

[0091] The aggregation function Agg could be Mean or Max depending on the need of the applications. The rationale of the aggregation is that since the entity (customer, service provider or also an item depending on the application) are the actors of the interactions in the edges, an anomaly in the interactions should affect the anomaly score of the entities involved. In addition, an entity is usually involved in many interactions. Some of those interactions may be anomalous, whereas other interactions may be normal. The Max aggregation is more sensitive to a single anomaly in the interaction, whereas the Mean aggregation provides a complete overview of all interactions that involve the entity.

[0092] The training procedure as described above requires access to the full graph structure as well as all node and edge features. This full graph training is, however, not scalable to large size graphs. To be able to scale the learning algorithm to large graph data, stochastic training via minibatching and neighbourhood sampling may be performed.

[0093] Minibatching splits the large graph into minibatches, each consisting of a small sub-graph. For each minibatch, we compute the final output representation for the nodes and edges in the sub-graph (target sub-graph). Computing the output of a particular node/edge in the target sub-graph using graph convolution requires knowing the neighbours of the node/edge. Thus,

given we have K convolution layers, the sub-graph is repeatedly expanded K times by sampling neighbouring nodes connected to the current sub-graph.

[0094] In addition, the flow of the message in the convolution from one layer to the next layer is stored.

[0095] Let U , V , and E (distinct from $\mathbf{u}$, $\mathbf{v}$, $\mathcal{E}$) denote the components of a sub-graph that belong to the node-set $\mathbf{u}$, node-set $\mathbf{v}$, and edge set $\mathcal{E}$, respectively. In the following, $F_{s_1 \rightarrow s_2}$ denotes the message flow from node/edge set s_1 in one layer to node/edge set s_2 in the next layer, where s represents either $\mathbf{u}$, $\mathbf{v}$, or $\mathcal{E}$. A message flow $F_{s_1 \rightarrow s_2}$ contains a tuple of the source and target of the message, e.g., $F_{u \rightarrow v} = (\{u_2, u_3, u_5\}, \{v_4, v_6\})$. Further, $F_{\rightarrow s}$ denotes the a set of all message flows pointing to s .

[0096] To create mini-batches of sub-graph, the nodes in one node-set, e.g., $\mathbf{u}$, are enumerated and group them into mini-batches of u nodes depending on the batch size. Let U be a set that contains a mini-batch of u nodes. The target sub-graph is generated by randomly sampling v nodes in $\mathbf{v}$ that are connected to any nodes in U . The resulting sample consists of a set of v nodes V and a set of edges E that connects V to U . The sets U , V and E form the target sub-graph for this particular mini-batch. This sub-graph is expanded K times by sampling the neighbourhood of U and V , while simultaneously creating the message flow sequence F from layer to layer. It should be noted that the layers are traversed in a backward manner, since the start is a target subgraph that defines the output of the network. Algorithm 2 provides the detailed step-by-step procedure for creating the message flow sequence.

Data: graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathbf{X}^u, \mathbf{X}^v, \mathbf{X}^e)$; even number of layer K ; A batch of u nodes U

Result: message flow sequence F

```

/* create initial sub-graph */
 $V, E \leftarrow$  sample neighboring  $v$  nodes from  $U$ ;
/* iterate every layer in reverse order */
for  $k \leftarrow K$  to 1 do
     $V', E'_1 \leftarrow$  sample neighboring  $v$  nodes of  $U$  from  $\mathcal{G}$ ;
     $U', E'_2 \leftarrow$  sample neighboring  $u$  nodes of  $V$  from  $\mathcal{G}$ ;
     $F_{\rightarrow u}^{(k)} \leftarrow \{F_{v \rightarrow u} : (V \cup V', U), F_{e \rightarrow u} : (E \cup E'_1, U)\}$ ;
     $F_{\rightarrow v}^{(k)} \leftarrow \{F_{u \rightarrow v} : (U \cup U', V), F_{e \rightarrow v} : (E \cup E'_2, V)\}$ ;
     $F_{\rightarrow e}^{(k)} \leftarrow \{F_{u \rightarrow e} : (U \cup U', E \cup E'_2), F_{v \rightarrow e} : (V \cup V', E \cup E'_1)\}$ ;
     $U \leftarrow U \cup U'; \quad V \leftarrow V \cup V';$ 
     $E \leftarrow E \cup E'_1 \cup E'_2;$ 
 $F \leftarrow \{(F_{\rightarrow u}^{(k)}, F_{\rightarrow v}^{(k)}, F_{\rightarrow e}^{(k)}) \mid k \in \{1, \dots, K\}\};$ 
return  $F$ 

```

Algorithm 2

[0097] To perform stochastic training, the forward propagation procedure (Algorithm 1) is adjusted to follow the generated message flow sequence. The gradient of the loss function is then used to perform stochastic updates of the neural network 300.

[0098] So far, an anomaly detection problem in a transductive setting has been discussed, where the model provides reasoning only on the observed data. A more general setting is the inductive setting, where the neural network 300 is also required to come up with a general principle.

[0099] A model that is capable of performing inductive learning can be applied to newly observed nodes/edges/sub-graphs, while a transductive-only model cannot. The inductive anomaly detection task can be defined as follows: given a bipartite node-and-edge-attributed graph $G = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathbf{X}^u, \mathbf{X}^v, \mathbf{X}^e)$ for training and a newly observed (sub)graph $G' = (\mathcal{U}', \mathcal{V}', \mathcal{E}', \mathbf{X}'^u, \mathbf{X}'^v, \mathbf{X}'^e)$ for evaluation, the task is to produce scoring functions applicable to each edge $e \in \mathcal{E}'$, as well as each node $u \in \mathcal{U}'$ and $v \in \mathcal{V}'$, such that the edges/nodes that differ singularly from the majority in terms both the structure and attribute information should be given higher anomaly score.

[00100] In the embodiment described above in model, the inductive capability comes from the convolution operation, message passing and aggregation, as well as neighbourhood sampling. The forward propagation for the encoder 301, feature decoder 302 and structure decoder 303 can be directly applied to the newly observed sub(graph) G' . The anomaly scoring functions discussed are also still applicable to G' .

[00101] In summary, according to various embodiments, a method is provided as illustrated in FIG. 5.

[00102] FIG. 5 shows a flow diagram 500 illustrating a method for detecting anomalies in double-party interaction data.

[00103] In 501 interactions between parties of a first group and parties of a second group are represented as a graph, wherein each interaction between a first party of the first group and a second party of the second group is represented by an edge between a respective first node representing the first party and a respective second node representing the second party and wherein information about the first party is assigned to the first node as node attribute information, information about the second party is assigned to the second node as node attribute information and information about the interaction is assigned to the edge as edge attribute information.

[00104] In 502, the graph is processed by a graph convolutional neural network having an autoencoder structure.

[00105] In 503, anomaly scores for interactions, parties of the first group and parties of the second group are derived from a reconstruction loss between the graph and an output of the graph convolutional neural network in response to the graph including at least a loss between the edge attribute information and edge attribute information reconstructed by a decoder of the graph convolutional neural network.

[00106] In 504, anomalies are based on the anomaly scores.

[00107] According to various embodiments, in other words, anomalies are detected based (at least) on edges (i.e. double-party interactions, i.e. interactions between two parties) whose attribute can only be poorly reconstructed. If the decoder can only poorly reconstruct attribute information about an edge this can usually be an indication that the corresponding information is unusual, i.e. differs from interactions that the neural network has seen in its training and thus, that the interaction represents a behaviour which is not normal.

[00108] It should be noted that the double-party interaction data may in particular include sensor data and the parties may correspond to components of a technical system. In that case, the method may include performing a control action of the technical system in response to a detected anomaly which may hint to a failure in the interaction of components of the technical system like between robot devices in, for example, a manufacturing system comprising a plurality of robot devices or a vehicle in an autonomous driving scenario comprising a plurality of vehicles.

[00109] The method of FIG. 5 is for example carried out by a server computer as illustrated in FIG. 6.

[00110] FIG. 6 shows a server computer 600 according to an embodiment.

[00111] The server computer 600 includes a communication interface 601 (e.g. configured to receive interaction data, i.e. information about interactions). The server computer 600 further includes a processing unit 602 and a memory 603. The memory 603 may be used by the processing unit 602 to store, for example, data to be processed, such as information about interactions and nodes and its graph representation. The server computer is configured to perform the method of FIG. 5.

[00112] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[00113] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A method for detecting anomalies in double-party interaction data, comprising:
Representing interactions between parties of a first group and parties of a second group as a graph, wherein each interaction between a first party of the first group and a second party of the second group is represented by an edge between a respective first node representing the first party and a respective second node representing the second party and wherein information about the first party is assigned to the first node as node attribute information, information about the second party is assigned to the second node as node attribute information and information about the interaction is assigned to the edge as edge attribute information;
Processing the graph by a graph convolutional neural network having an autoencoder structure;
Deriving anomaly scores for interactions, parties of the first group and parties of the second group from a reconstruction loss between the graph and an output of the graph convolutional neural network in response to the graph including at least a loss between the edge attribute information and edge attribute information reconstructed by a decoder of the graph convolutional neural network; and
Detecting anomalies based on the anomaly scores.
2. The method of claim 1, wherein the reconstruction loss includes a loss between node attribute information and node attribute information reconstructed by the decoder.
3. The method of claim 1 or 2, wherein the reconstruction loss includes a loss between node adjacency information of the graph and node adjacency information reconstructed by the decoder.
4. The method of any one of claims 1 to 3, comprising comparing the anomaly scores with one or more threshold values and detecting an anomaly of an interaction, party of the first group or party of the second group if its anomaly score is above a respective threshold value.

5. The method of any one of claims 1 to 4, comprising training the graph convolutional neural network by, determining, for each training data element of a plurality of training data elements,
wherein each training data element comprises a training graph, a reconstruction loss between the training graph and an output of the graph convolutional neural network in response of the training graph and an output of the graph convolutional neural network in response to the graph including at least a loss between the edge attribute information and edge attribute information reconstructed by a decoder of the graph convolutional neural network, and
adapting the neural network to reduce an overall loss including the reconstruction losses determined for the training data elements.
6. The method of claim 5, comprising, for each training data element, sampling sub-graphs of the training graph, for each sampled sub-graph setting a reconstruction target to the sampled sub-graph and expanding the sub-graph by including nodes and edges connected to the sampled sub-graph and computing a reconstruction between the reconstruction target and an output of the graph convolutional neural network in response to the expanded sub-graph, wherein the overall loss includes the reconstruction losses determined for the sub-graphs.
7. The method of any one of claims 1 to 6, wherein the first group of parties are customers and the second group of parties are service providers.
8. The method of any one of claims 1 to 7, wherein the interactions are transactions between the first group of parties and the second group of parties.
9. The method of any one of claims 1 to 8, comprising detecting fraud based on the detected anomalies.
10. The method of claims 9, comprising checking, for each detected anomaly, whether there has been fraud.

11. The method of any one of claims 1 to 10, further comprising utilizing the anomaly score as the input to a human-in-the-loop actioning system and/or an automatic actioning system.
12. A server computer comprising a radio interface, a memory interface and a processing unit configured to perform the method of any one of claims 1 to 11.
13. A computer program element comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 11.
14. A computer-readable medium comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 11.