Linking bank clients using graph neural networks powered by rich transactional data

Valentina Shumovskaia¹ · Kirill Fedyanin² · Ivan Sukharev¹ · Dmitry Berestnev¹ · Maxim Panov²

Received: 4 July 2020 / Accepted: 13 January 2021 / Published online: 4 March 2021
© The Author(s) 2021

Abstract

Financial institutions obtain enormous amounts of data about client transactions and money transfers, which can be considered as a large graph dynamically changing in time. In this work, we focus on the task of predicting new interactions in the network of bank clients and treat it as a link prediction problem. We propose a new graph neural network model, which uses not only the topological structure of the network but rich time-series data available for the graph nodes and edges. We evaluate the developed method using the data provided by a large European bank for several years. The proposed model outperforms the existing approaches, including other neural network models, with a significant gap in ROC AUC score on link prediction problem and also allows to improve the quality of credit scoring.

Keywords Link prediction · Graph neural network · Recurrent neural network · Transactional data · Credit scoring

1 Introduction

It is important for the financial institutions to know their client well in order to mitigate credit risks [27], deal with fraud [5, 23] and recommend relevant services [4]. One of the defining properties of a particular bank client is his or her social and financial interactions with other people. It motivates to look on the bank clients as on the network of interconnected agents [4, 28, 32]. Thus, graph-based approaches can help to leverage this kind of data and solve the above mentioned problems more efficiently.

Valentina Shumovskaia and Kirill Fedyanin have equally contributed to this work.

✉ Kirill Fedyanin
k.fedyanin@skoltech.ru
Valentina Shumovskaia
valentinashumovskaya@gmail.com
Ivan Sukharev
ivan7654@gmail.com
Dmitry Berestnev
toberest@gmail.com
Maxim Panov
m.panov@skoltech.ru

Importantly, information about clients and especially about their neighborhood is never complete—market is competitive and we can not expect all the people to use the same bank. Thus, some of the financial interactions are effectively hidden from the bank. That leads to the necessity to uncover hidden connections between clients with limited amount of information which can be done using link prediction approaches [31].

From the other hand, the financial networks have two notable features. First one is the size—the number of clients can be of order of millions and the number of transactions is estimated in billions. The second important feature is the dynamic structure of considered networks—the neighborhood of each client is ever-evolving. The classical link prediction algorithms are only capable of working with graphs of a much smaller size, while the temporal component is usually not considered [31]. Recently, several studies addressed large-scale graphs [30, 34] as well as temporal networks [21]. However, only few works consider the financial networks, see, for example, [4] and [28].

We base our research on the well-developed paradigms of graph mining with neural networks including graph convolutional networks [14, 17], graph attention networks [29] and SEAL framework for link prediction [37]. The considered approaches consistently show state-of-the-art results in many applications but, to the best of our knowledge, were

¹ Risk Modeling and Research, Sberbank, Moscow, Russia

² Skolkovo Institute of Science and Technology, Skoltech, Moscow, Russia

not yet used for financial networks. Our key contributions can be formulated as follows:

- We build a scalable approach to link prediction in temporal graphs with the focus on extensive usage of recurrent neural networks (RNNs) both as feature generators for graph nodes and as a trainable attention mechanism for the graph edges.
- We propose several modifications to graph pooling procedures including the usage of two node convolutions instead of sortpooling [37] and neighborhood prioritization by Weisfeiler–Lehman labeling [36].
- We validate the proposed approaches on the link prediction and credit scoring problems for the real-world financial network with millions of nodes and billions of edges. Our experiments show that our improved models perform significantly better than the standard ones and efficiently exploit rich transactional data available for the edges and nodes while allowing to process large-scale graphs.

The rest of the paper is organized as follows. We start by describing in more detail the problem statement, the considered dataset and the proposed way of validation in Sect. 2. Section 3 provides the detailed description of our approaches and algorithms. We continue with the review of the related work in Sect. 4 and then describe our experiments and show their results in Sect. 5. Finally, Sect. 6 concludes the study and notes some future research directions.

2 Problem and data

From the perspectives of network science and data analysis, the considered problem of linking bank clients is the link prediction problem in graphs with two notable peculiarities. The first one is that the considered graph of clients and transactions between them is very large, having the order of millions of nodes and billions of edges. The second peculiarity is that both nodes and edges have rather complex attributes represented by times series of bank transactions of different types. We want to note that such kind of problem is not limited to banking as graphs with similar structure appear in social networks, telecom companies, and other scenarios, where we consider some objects as nodes and a certain type of communication between them. Thus, the algorithms developed in our work might be applicable beyond banking for any link prediction problem with times-series attributes.

In what follows, we first discuss the dataset studied in our work and then explain some peculiarities of the problem statement.

Table 1 The example of information available for a node (transactions of a client)

Amount	100	45	34
Currency	USD	USD	USD
Date	22/Apr/2019	22/Apr/2019	24/Apr/2019
Time	21:22	23:05	13:17

Table 2 The example of information for an edge (transfers between a pair of clients)

Source	Client 1	Client 1	Client 2
Destination	Client 2	Client 2	Client 1
Amount	41	82	900
Currency	USD	USD	USD
Date	07/May/2020	11/May/2020	23/May/2020
Time	11:15	21:32	09:35

2.1 Dataset

The considered dataset is obtained from one of the large European banks with all the data in the depersonalized format. The data consist of clients' transactions and transfers, each described by timestamps, amount of money, and currency. Based on the data, we construct the graph $G(V, E)$ with a set of vertices V and a set of edges E . Every node is associated with a client and a sequence of transactions (see the example in Table 1). At the same time, every edge is related to a series of transfers between the corresponding pair of clients (see the example in Table 2). Thus, we obtain a huge 86-million nodes graph with about 4 billions of edges.

Such graph size makes its analysis difficult to approach, since the majority of the graph processing methods aimed to solve node classification, graph classification or link prediction problems are suitable for graphs of much smaller size [13]. The time complexity of standard methods typically grows at least as n^2 , where n is number of nodes, limiting the possible graph sizes to several thousands of nodes and up to a one hundred thousand of edges.

As a result, when we work with a particular node or with a particular edge, we are forced to consider certain subgraphs around the target node or the target pair of nodes (for example, see [36]). In this work, we follow this approach and consider the subgraph around target nodes extracting their hop-2 neighbors.

2.2 Problem statement and validation

From the bank perspective, the goal of the study is to determine how strong are ties between its clients. The strength of connection is well correlated with the regularity of transfers between the clients. Thus, our goal is to determine how stable

Fig. 1 Target definition. The target is whether there is at least one transfer between clients is made during the target time segment

is the relationship between nodes. To formulate the particular data analysis problem, we start by describing so-called *out-of-time validation*, see a similar approach in [19]. More specifically, we consider time interval time $[t_0, t_1]$ for $t_0 < t_1$ and use all the information available (e.g., all the transactions and transfers) as an information encoded in a graph. Given the information available for the period $[t_0, t_1]$, we aim to predict the structure of the graph for the time interval $[t_1, t_2]$ with $t_1 < t_2$. In what follows, we say that there is an edge between two nodes in a graph for the certain time period if there was at least one transaction between these nodes during the considered period. Thus, we end up with the link prediction problem where the pair of nodes is described by the graph structure and attributes during the period $[t_0, t_1]$ and the target label corresponds to the existence of the transaction between the pair of nodes during the period $[t_1, t_2]$. In all the experiments below we take $t_1 - t_0$ equal to one year and $t_2 - t_1$ equal to 3 months. The choice of segment length is quite arbitrary; it was dictated by the fact that the same length was applied for other models in the bank. The general consideration is that the year span would be enough to detect all the relevant connections while allowing to fully separate train, validation and test datasets by time. The resulting target definition for the link prediction problem is depicted in Fig. 1.

We note that usually link prediction models are validated in a different way, e.g., by edge sampling [19]. In this approach, the whole set E is considered as positive samples,

while negative samples are constructed by taking $\beta|E|$ node pairs ($\beta > 0$ is a hyperparameter and $|\cdot|$ means the set size), which do not intersect with E . Then, the subgraph is passed to the link prediction algorithm, hiding the link, if it exists. In order to build training, validation and test parts, one divides positive and negative edge sets into three corresponding non-intersecting sets.

However, we think that for the time-evolving graphs in general and banking data in particular the out-of-time validation is more sensible. Thus, in this work, we focus on the out-of-time validation, while still providing a part of the experiments for both settings.

3 Neural network model for link prediction with transactional data

In this section, we describe the proposed neural network for solving a link prediction task powered by rich transactional data. The most challenging part is the work with transactional data itself, which is basically a multi-dimensional time series.

As a base graph neural network, we take SEAL framework for link prediction [37]. Its input parameters are an adjacency matrix of a graph $A \in \mathbb{R}^{n \times n}$ with n nodes, and a node feature matrix $X \in \mathbb{R}^{n \times d}$ with each row containing a feature vector of dimension d for the corresponding node. Then, SEAL considers the neighborhood subgraph for the target pair of nodes and performs several graph convolutions followed by sortpooling operation and fully connected layers, see Fig. 2. We are going to discuss the SEAL architecture in more details in Sect. 3.2.

In our case, the considered network of bank transactions does not have an explicit adjacency matrix or a vector of features as both clients and interactions between them are represented by time series. In the following, we are going to adapt SEAL framework to work with time series data by processing them with RNN, see Sect. 3.1. Moreover, we make a number of specific improvements to the structure of SEAL model making it more efficient, see Sect. 3.2.

3.1 Recurrent neural network powers graph neural network

The question of processing time series corresponding to the graph edges and nodes is challenging. We start by discussing the edges, for the case of nodes see Sect. 3.1.3. The simplest way to deal with edges is just to ignore the whole time series and consider binary adjacency matrix with edges present for pairs of nodes with at least one transfer between them. However, in this case we lose significant amount of important information as the properties of transfers between clients are apparently directly linked with our link prediction objective. In this work, we propose to use recurrent neural network

Fig. 2 SEAL architecture. The input graph is passed to a series of graph convolutional (GC) layers. The obtained nodes features are sorted and pooled with SortPooling layer, and then they are passed to 1-D convolutional layer (1D Conv) and fully connected (FC) layer

(RNN, [8]) as an attention mechanism, which effectively produces weighted adjacency matrix of a graph, see Sect. 3.1.2. We start by discussing the peculiarities of graph convolutions and attention mechanisms.

3.1.1 Graph convolutions and attention mechanisms

We first note that standard graph convolutional architectures (like GCN [17] or SEAL [37]) perform convolution operation by simple averaging over the neighborhood:

$$h'_i = \sigma \left(\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W h_j \right), \quad i = 1, \dots, n, \quad (1)$$

where $(h_1, \dots, h_n)$ are node embedding vectors before the convolution operation, $(h'_1, \dots, h'_n)$ are their counterparts after it, W are learnable weights, $\mathcal{N}_i$ is a set of immediate neighbors of node i and, finally, σ is an activation function. The averaging operation implies that all the neighbors have an equal influence on the considered nodes which is apparently very unnatural in the majority of applications.

Graph attention networks [29] mitigate this problem by introducing attention weights based on node features:

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha(h_i, h_j) W h_j \right), \quad i = 1, \dots, n, \quad (2)$$

where $\alpha(h_i, h_j)$ is some function with learnable parameters. The important peculiarity of the original implementation [29] is that coefficients $\alpha_{ij} = \alpha(h_i, h_j)$ are computed solely basing on node features. Instead, in order to use the full information about the graph, one may consider computation of attention coefficients based also on edge features e_{ij} if available. The resulting convolutional layer can be written as

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha(h_i, h_j, e_{ij}) W h_j \right), \quad i = 1, \dots, n \quad (3)$$

for some function $\alpha(h_i, h_j, e_{ij})$ with possibly learnable parameters. The main challenge in application of equation (3)

Fig. 3 RNN for link prediction architecture

is to design the function α in a way that it can be efficiently estimated from data avoiding overfitting. A particular instantiation of the attention approach was recently considered in the work [12], where edge features were shown to significantly improve the quality of node and graph classification.

We should note that this approach naturally extends to multi-head attention allowing for more expressive models:

$$h'_i = \sigma \left(\parallel_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_k(h_i, h_j, e_{ij}) W_k h_j \right), \quad i = 1, \dots, n \quad (4)$$

where $\parallel$ denotes concatenation, K is the number of channels, and for each channel we have separate matrices W_k and functions $\alpha_k(h_i, h_j, e_{ij})$. However, it comes with the cost of increasing the number of parameters. We now switch to describing the particular way of constructing attention function which is well aligned with the peculiarities of the considered data.

3.1.2 RNN as attention mechanism

In order to get the full use of the data, we first note that one can consider a recurrent neural network predicting the link between two nodes using solely the time series of transfers between them. In this work, we consider a quite simple RNN model, which consists of GRU cell [7], followed by a series of fully connected layers, see Fig. 3. However, such a RNN model does not allow us to detect *new* possible connections since there is no data about interaction between clients in this case. To overcome that drawback, a model based on a graph can be used.

In this work, we propose to use the probabilities of the links between nodes output by RNN model as weights in

the adjacency matrix, which then is passed to graph neural network:

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} p(e_{ij}) W h_j \right), \quad i = 1, \dots, n. \quad (5)$$

This choice of the attention mechanism seems to be natural for the problem of link prediction, as the nodes with a high probability of future interactions are stronger connected and supposedly have a bigger influence on each other. Importantly, RNN model can be further trained together with graph neural network in order to adapt to the graph data and improve the quality.

3.1.3 RNN as feature generator

Now we switch to discussing the feature generation from client's transactions. Since transactions have a time-series structure, similarly to the case of edges, one can construct a recurrent neural network (note that it is a different neural network compared to the previous section). We consider the hidden layers' output of the above network as the meaningful vector representation for the client's transactions. We call such hidden layer outputs *embedded transactions* and use them as node feature vectors X in all the considered graph neural network models. The remaining question is what objective function RNN should target. We suggest to pretrain RNN model on the credit scoring problem similar to [2]. See also additional details in Sect. 5.4.

More formally, let us define the time series of transactions for the client i as z_i . Then, the RNN is trained to solve credit scoring problem based on z_i for $i = 1, \dots, N$, and corresponding target values $y_i \in \{0, 1\}$ (not default–default). As a result for each node i we can compute an embedding vector $x_i = f(z_i) \in \mathbb{R}^m$, where function $g(\cdot)$ outputs the last layer embedding of dimension m for trained RNN and given input.

3.2 Graph neural network for link prediction

Based on the proposed attention mechanism and feature generation approach, we can construct the graph neural network for link prediction powered by RNN. After extracting an enclosing subgraph around the target link, all time series corresponding to edges are processed by RNN and the output probabilities are used to essentially weight the adjacency matrix which together with generated node features X are passed into SEAL model. The resulting model is called SEAL-RNN, see the architecture in Fig. 4. We note that SEAL model contains two non-trivial components: sortpooling approach and node-labeling procedure. In what follows, we are going to discuss them in detail and propose some improvements.

3.2.1 Pooling

The non-standard and rather complicated part of SEAL model is sortpooling operation. The core idea of sortpooling is to compute the embeddings for all the nodes via graph convolutions and then sort them according to some score function. As a result, the fixed number K of embeddings with highest scores are selected and further processed by 1-D convolutions. The purpose of 1-D convolution in SEAL framework is to reduce the size of obtained output, which is $K \times d$, where d is a sum of node features dimension and dimensions of the graph convolution outputs. While delivering the necessary reduction to the fixed dimension and being rather generic, sortpooling apparently increases number of parameters to be learnt.

In this work, we propose another pooling operation instead of sortpooling. We suggest just taking embeddings of two nodes, between which we aim to predict the link. The idea is natural since we want to predict the link between exactly these two nodes, while their embeddings still contain information about the neighboring nodes. Most importantly, it reduces the number of learned parameters in the neural network, and we do not need neither a sorting operation, nor 1-D convolution after pooling. We name the proposed model 2-SEAL, see the schematic representation in Fig. 5.

3.2.2 Modified structural labels

In SEAL framework [37], each node receives a structural label generated by a *double-radius-node-labeling* procedure, which meets the following conditions:

1. two target nodes x and y have label '1';
2. nodes with different distances to both x and y have different labels.

The aim of the labels is to encode some of the topological information about the graph structure. These structural labels are concatenated with initial node features (if exist), and passed to neural network as node features. The labeling function (i is a node index) is computed as follows:

$$f(i) = 1 + \min(d_x, d_y) + (d/2) \left[(d/2) + (d\%2) - 1 \right] \quad (6)$$

where $d_x = d(i, x)$, $d_y = d(i, y)$, $d(i) = d_x + d_y$, $(d/2)$ and $d\%2$ are the integer quotient and remainder of division respectively, while $d(\cdot, \cdot)$ is distance between nodes. Authors of initial paper suggest to take into account all subgraph nodes except y during computing distance d_x , and similarly for d_y .

In this work, we suggest not to hide nodes y and x during finding distances d_x, d_y . That better suits out-of-time validation by allowing to keep in data patterns for all kinds of

Fig. 4 SEAL-RNN model architecture. After extracting an enclosing subgraph around the target link, all time series corresponding to edges are processed by RNN and the output probabilities are used to essen-

tially weight the adjacency matrix which together with generated node features X are passed into SEAL model

Fig. 5 2-SEAL architecture. At first, the input graph is passed to a series of graph convolutional (GC) layers. Then, the obtained nodes features of two target nodes (between which the link is predicted) are passed to fully connected (FC) layer

combinations of link existence in the observed graph and link existence in the future.

4 Related work

The idea to consider bank clients as a large network of interconnected agents was raised in the past several years [4,28,32]. The number of bank clients counts in millions, so we solve the link prediction problem for graphs with millions of nodes, which requires the usage of scalable methods. There are few ways to handle the graphs of such size mentioned in the literature, mostly being the simple heuristics that compute some statistics for the immediate neighborhoods of target nodes, for example, Common Neighbors [20], Adamic–Adar [1] and others [31]. However, these models are not trainable and do not use the information about node features, which limits their performance in real-world applications.

The main challenge in the construction of machine learning models for link prediction is to handle variation in the graph size. One approach is presented in WLN [36]—it is to use Weisfeiler–Lehman structural labels [33] to prioritize

nodes and to leave only the important ones from the immediate neighborhood of evaluated nodes. After that, we can use regular graph neural networks.

The graph convolutional network (GCN, [17]) shows good performance for various graph analysis problems. Original GCN is supposed to use the whole graph, and it is prohibitive for the graph on a scale of millions of nodes. In [37], the SEAL framework was proposed, which extracts enclosing subgraphs around the target link and includes such a pooling layer in the neural network architecture, which holds the fixed number of nodes for every subgraph. This allows using the model on almost arbitrarily large graphs with a linear complexity in a number of nodes in a graph.

The graph attention network (GAT, [29]) allows specifying different weights to different nodes in the neighborhoods. The original approach considers only node features while the extension to edge features was recently considered in [12]. We also follow this line of research but propose an approach to construction of attention coefficients which efficiently leverages sequence information corresponding to the edges.

We also discuss approaches for graphs with dynamic structure, since we can look on our transactional graph in that way. Authors of works [18,35] propose procedures based on the training on a single graph. These approaches cannot be straightforwardly generalized for the subgraphs, while the usage for the whole graph is prohibitive due to scalability issues. [38] proposes the computationally efficient variants of simple heuristics in the case of streaming data, while we evaluate these metrics directly. The work [25] is based on the assumption that well-predicting edge and neighborhood features already exist, which is not the case in our situation. Authors of [24] propose another neural network for working with dynamic graphs but focus on the change of the graph structure over time. Some of these approaches can be modified to be applied in our scenario, but these modifications are non-trivial and require separate research.

5 Experiments

5.1 Dataset preprocessing

We start by dividing the whole considered time period into three parts: first three years, fourth year, and fifth year, which correspond to training, validation, and test time. The splitting into three non-intersecting groups is also done for client IDs. Taking a point in one of the time intervals, we define the base and the target segments. The base segment corresponds to the time segment before the point, while the target segment corresponds to time after. Then, we generate subgraphs and labels according to the out-of-time validation approach discussed in Sect. 2.2. As a result, we generate subgraphs for training, validation and test datasets as depicted on Fig. 6.

The main idea behind such a splitting is to validate that the model can operate in a scenario when the data about new clients are added and the model is not retrained. It is very relevant for the bank as one may not expect frequent model retraining for the model working in production in a large bank. Thus, we split data in that way in order to separate train, test and validation sets as much as possible with no overlaps between sets in time and over clients (thus, blanks in Fig. 6 mean that this part of the data was not used in our pipelines). Moreover, such a split allows us to be completely sure that we do not overfit toward the behavior of particular clients which is relatively easy with large neural network models.

We also separately consider edge sampling validation, for which we observe graph state restricted to the base segment. In this case, the target is whether there was at least one transfer between clients during this time. We consider ROC-AUC measure as a quality metric for the link prediction task in all the experiments.

Fig. 6 Data split for train, validation and test

Table 3 Heuristics approaches and SEAL (with embedded transactions and structural labels as node features) results for banking data (ROC AUC)

Method	Edge sampling	Out of time
Common Neighbors	0.398 ± 0.002	0.629 ± 0.002
Adamic–Adar	0.391 ± 0.002	0.646 ± 0.002
Resource Allocation	0.350 ± 0.002	0.639 ± 0.002
Jaccard Index	0.284 ± 0.002	0.620 ± 0.002
Preferential Attachment	0.746 ± 0.001	0.497 ± 0.002
SEAL	0.850 ± 0.001	0.770 ± 0.001

Best results are marked in bold

5.2 Implementation details

We use PyTorch [22] and PyTorch Geometric [10] to implement all the neural network models. Each model was trained with Adam optimizer [16] using learning rate scheduler and hyperparameter optimization [3] for the number of layers, size of the layers and initial learning rate. We used the server with single GPU (NVIDIA Tesla P100), 32 CPU cores Intel i7 and 512 GB of RAM in all the experiments.

5.3 Link prediction results

5.3.1 Baseline models

We start by describing the experiments with several baselines. Due to the need in the scalability, we consider only simple similarity-based approaches, such as Common Neighbors, Adamic–Adar Index, Resource Allocation, Jaccard Index and Preferential Attachment, as baselines for our task (see [31] for the description of the methods). Also, we take the SEAL model [37] as a baseline (with embedded transactions concatenated with structural labels as node features). The results can be found in Table 3; standard errors were calculated according to [9]. As we may see, the results obtained from simple heuristic methods are beaten by the neural net-

Fig. 7 WL-SEAL model architecture. At first, the input graph is passed to a WL-coloring scheme, where the most significant nodes are extracted. Then, reduced graph is passed to a series of graph convolutional (GC) layers, followed by 1-D convolutional layer (1D Conv) and fully connected (FC) layer

work solution. Importantly, in this experiment the time series of transfers corresponding to edges are completely ignored, and all the methods operate with just a binary adjacency matrices.

Also we should note that there is a significant in the ROC AUC score for different validation settings. It could be explained by the fact that the problem of prediction into the future is a more difficult problem than finding hidden links in the current graph state.

5.3.2 Experiments with pooling

The first improvement of the initial SEAL model is the new pooling operation. The baseline SEAL and the proposed 2-SEAL models are described in Sect. 3. We additionally consider WL-SEAL pooling operation which is based on the idea of the Weisfeiler–Lehman graph isomorphism test. Quiet similarly to the idea described in [36], we propose to color nodes of enclosing subgraphs by the Palette-WL algorithm (Algorithm 3 in [36]) and thereby get nodes ordering. After that, we take only K (hyperparameter) the most significant nodes of the subgraph, as an input of the neural network. Thus, all subgraphs have the same size, so there is no need for a pooling operation after convolutional layers. The architecture is summarized in Fig. 7.

We expect that such pooling is more meaningful in the sense of intuition, but the drawback of such a model is computational expensiveness of coloring algorithm ($O(e^{\sqrt{n \log n}})$). The results can be found in Table 4. We observe that both WL-SEAL and 2-SEAL are superior to SEAL. However, 2-SEAL shows the best results, being less computationally expensive model which motivates us to focus the further studies on this model.

5.3.3 Experiments with node features

Another set of experiments is devoted to the exploration of node features. In the previous experiments with neural networks, we used a concatenation of embedded transactions (the output of an intermediate level of RNN which solves a

Table 4 SEAL (with embedded transactions and structural labels as node features) pooling modifications results for banking data (ROC AUC)

Method	Edge sampling	Out of time
SEAL	0.850 ± 0.001	0.770 ± 0.001
WL-SEAL	0.871 ± 0.001	0.751 ± 0.001
2-SEAL	0.890 ± 0.001	0.780 ± 0.001

Best results are marked in bold

credit scoring task) and structural labels as node features. In this section, we consider different variants of node features: embedded transactions, embedded transactions concatenated with structural labels, structural labels, and modified structural labels. Structural labels and modified structural labels are described in Sect. 3.2.2. Surprisingly, the usage of embedded transactions plays a negative role in the link prediction task, see the results for SEAL model in Table 5. We explain it by the fact that similar purchases do not play a significant role in problems of finding new connections in the network, while network structure and client's connections are a way more important. Also, the proposed modified structural labels give us a better performance.

Note that 2-SEAL-RNN with embedded transactions as node features can be considered as a common graph convolutional network (GCN [17]) baseline: we make a decision based on GCN embeddings of two target nodes. Based on these vectors, the link prediction task can be solved differently: for example, the cosine similarity of the vectors [11] can be considered as a probability of a link, or like in our case, the concatenation of two vectors is fed to the neural network. The significant difference with GCN is that we work with the subgraphs since we cannot work with the whole graph in our scenario.

5.3.4 Experiments with edge features and attention

The final set of experiments is based on the work with data corresponding to edges, where we consider different RNN-

Table 5 SEAL pooling modifications results for banking data with embedded transactions (ET) and structural labels (SL) as node features (ROC AUC)

Method	Embedded transactions (ET)	ET+SL	Structural labels (SL)	Modified SL
SEAL	0.620 ± 0.002	0.747 ± 0.002	0.740 ± 0.002	0.760 ± 0.001
SEAL-RNN	0.610 ± 0.002	0.787 ± 0.001	0.780 ± 0.001	0.794 ± 0.001
2-SEAL	0.700 ± 0.002	0.739 ± 0.001	0.770 ± 0.001	0.787 ± 0.001
2-SEAL-RNN	0.730 ± 0.002	0.804 ± 0.001	0.83 ± 0.001	0.858 ± 0.001

Best results are marked in bold

based models, as described in Sect. 3.1. To generate attention coefficients, we use the RNN-network for each configuration. During the pretraining, we use a single edge transaction sequence between target nodes as the model input. The RNN is trained on a reduced number of subgraphs because we can only use the cases with at least one transaction in the base segment (see Sect. 5.1). Otherwise, there is no historical data about transactions between users available and subsequently no information for the model to make a prediction. We get decent model performance with ROC-AUC 0.81 on the test dataset. However, it can not be directly compared with the results of other methods because the datasets for RNN and graph-based methods are essentially different as discussed above. This pretrained RNN model was frozen, and we use it to generate attention weights $p(e_{ij})$ for each edge, as described in equation (5).

The results of this experiment are summarized in Table 5. We see that in almost all the settings we have a large increase in the ROC AUC score for the proposed 2-SEAL model over the baseline SEAL with sortpooling operation. Even larger benefit is given by considering the transactional data corresponding to the pairs of clients and treating it via the proposed attention-based approach. We conclude that 2-SEAL model with RNN attention is the best model for link prediction for the considered banking dataset.

The summary of the experimental results for the link prediction problem can be found in Table 6. We observe the significant improvement in the ROC AUC score for the proposed 2-SEAL-RNN model compared to the best heuristic approach and SEAL. These results show that it is equally important to fully use the information available in the data and carefully design the architecture of the model.

5.4 Credit scoring results

In this section, we want to show the applicability of the developed link prediction models to other problems relevant for the banking. One of the most important problems in the bank is to control the risks related to working with clients, especially in the process of issuing a loan. This problem is called credit scoring [27], and usually the ensemble of predictive models is used for its solution, which in particular are based

Table 6 Final results on banking data in out-of-time validation setting

Method	Result, ROC AUC
Best heuristic	0.646 ± 0.002
SEAL	0.740 ± 0.001
2-SEAL	0.790 ± 0.001
2-SEAL-RNN	0.858 ± 0.001

Best results are marked in bold

on client transactional data. For example, the RNN model run on time series of transactions has been shown to be very efficient in credit scoring [2].

We consider the credit scoring dataset of approximately one hundred thousand clients which are a part of our initial dataset. As a baseline we take the simple RNN network—GRU cell followed by linear layers and train it on the single node transactions. We get the model of decent quality achieving ROC-AUC score equal to 0.855.

The usage of information available in the network of clients may further improve the prediction quality. Our experiments show the standard graph convolutional network (GCN) [17] trained on these data improves over baseline RNN model by 0.8% Gini, see Table 7. However, GCN model is known to treat all the neighboring nodes equally without any prioritization (see discussion in Sect. 3.1), which is not correct for the bank clients some of which have much more influence on the particular client than the others. This issue was addressed in the literature by introducing graph attention mechanism based on the available node features [29].

In this work, we propose to use the developed link prediction model (2-SEAL-RNN) as an attention mechanism by reweighting the neighboring nodes with coefficients proportional to the probabilities of the connection output by the link prediction model. The idea is similar to the one used above in link prediction, but now the whole 2-SEAL-RNN is used for weighting instead of just RNN in link prediction case. Unlike standard graph attention networks [29], our attention mechanism considers not only node features but also the topology of the graph. Importantly, this approach allows to train the final credit scoring model in an end-to-end fashion. We use the link prediction model from the previous section as a starting point, but then we train it together with the graph neural network solving credit scoring problem. The training

Table 7 Final results on banking data in out-of-time validation setting

Method	Result, Δ in Gini index (%)
Standard GCN	+ 0.8
GCN with LP-based attention	+ 1.4

Best results are marked in bold

on link prediction problem thus becomes a pretraining for the attention further used in credit scoring model.

In Table 7, we compare GCN performance which uses binary adjacency matrices and adjacency matrices weighed by the link prediction model. We note that we use the transaction embeddings obtained by RNN as node features in both models. The results show that the link prediction model used as attention in GCN allows almost to double the effect of considering graph structure in the credit scoring problem. We believe that the further study of the link prediction based attentions in a graph neural network may lead to even better credit scoring models. We also should note that in practice, models based on classical machine learning [6,15] techniques are used. We do not provide a comparison with any of such methods, since our aim is to show the advantage of the graph approach over the approach where every user is scored separately and extensive analysis of credit scoring problem requires separate research.

One of the interesting side effects of end-to-end training is splitting of the attention weights into two modes. We demonstrate it with attention weights histograms before and after end-to-end training. Before the training we observe the single-mode distribution (see Fig. 8a), but after the training, it splits into two modes around 0 and 1, see Fig. 8b. It indicates that attention network becomes able to separate important and non-important neighbors in terms of the specific credit scoring task.

6 Conclusion

In this work, we developed the graph convolutional neural network, which can efficiently solve the link prediction problem in large-scale temporal graphs appearing in banking data. Our study shows that to benefit from the rich transaction data fully, one needs to efficiently represent such data and carefully design the structure of the neural network. Importantly, we show the effectiveness of recurrent neural networks as building blocks of temporal graph neural network, including a non-standard approach to the construction of attention mechanism based on RNNs. We also modify the existing GNN pooling procedures to simplify and robustify them. The developed models significantly improve over baselines and provide high-quality predictions on the existence of stable links between clients, which enables bank with a powerful instrument for the analysis of clients' network. In particular,

(a) Before end-to-end training **(b)** After end-to-end training**Fig. 8** Attention weight distribution, before (a) and after (b) end-to-end training

we show that the usage of the obtained link prediction model as an attention module in the graph convolutional neural network allows to improve the quality of credit scoring.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest. We note that an extended abstract of this paper was published in [26].

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Adamic, L.A., Adar, E.: Friends and neighbors on the web. *Soc. Netw.* **25**, 211–230 (2001)
- Babaev, D., Savchenko, M., Tuzhilin, A., Umerenkov, D.: Et-rnn: applying deep learning to credit loan applications. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2183–2190 (2019)
- Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: *Proceedings of the 30th international conference on machine learning—volume 28, ICML'13*, pp. I-115–I-123 (2013). <https://jmlr.org/>
- Bruss, C.B., Khazane, A., Rider, J., Serpe, R., Gogoglou, A., Hines, K.E.: Deeptrax: embedding graphs of financial transactions. *CoRR* (2019). [arXiv:1907.07225](https://arxiv.org/abs/1907.07225)
- Carcillo, F., Le Borgne, Y.A., Caelen, O., Bontempi, G.: Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. *Int. J. Data Sci. Anal.* **5**(4), 285–300 (2018)
- Chi, B.W., Hsu, C.C.: A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Syst. Appl.* **39**, 2650–2661 (2012). <https://doi.org/10.1016/j.eswa.2011.08.120>
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734 (2014)
- Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent neural networks and robust time series prediction. *IEEE Trans. Neural Netw.* **5**(2), 240–254 (1994)
- Cortes, C., Mohri, M.: Confidence intervals for the area under the roc curve. In: *Advances in neural information processing systems*, pp. 305–312 (2005)
- Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch geometric. In: *ICLR workshop on representation learning on graphs and manifolds* (2019)
- Gao, F., Musial, K., Cooper, C., Tsoka, S.: Link prediction methods and their accuracy for different social networks and network metrics. *Sci. Program.* (2015)
- Gong, L., Cheng, Q.: Exploiting edge features for graph neural networks. In: *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 9203–9211 (2018)
- Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. *IEEE Data Eng. Bull.* (2017)
- Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: *NIPS* (2017)
- Khandani, A., Kim, A., Lo, A.: Consumer credit-risk models via machine-learning algorithms. *J. Bank. Finance* **34**, 2767–2787 (2010). <https://doi.org/10.1016/j.jbankfin.2010.06.001>
- Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: *3rd international conference on learning representations, ICLR* (2014)
- Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: *5th international conference on learning representations, ICLR* (2017)
- Li, X., Du, N., Li, H., Li, K., Gao, J., Zhang, A.: A deep learning approach to link prediction in dynamic networks. In: *Proceedings of the 2014 SIAM international conference on data mining*, pp. 289–297. SIAM (2014)
- Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* (2007)
- Newman, M.E.J.: Clustering and preferential attachment in growing networks. *Phys. Rev. E: Stat., Nonlin, Soft Matter Phys.* **64** **2 Pt 2**, 025102 (2001)
- Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Leiserson, C.E.: Evolvegc: Evolving graph convolutional networks for dynamic graphs. In: *AAAI* (2020)
- Paszke et al., A.: Pytorch: an imperative style, high-performance deep learning library. In: *Advances in neural information processing systems*, vol. 32, pp. 8024–8035 (2019). <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- Phua, C., Lee, V., Smith, K., Gayler, R.: A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119* (2010)
- Ruiz, L., Gama, F., Ribeiro, A.: Gated graph convolutional recurrent neural networks. In: *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5. IEEE (2019)
- Sarkar, P., Chakrabarti, D., Jordan, M.: Nonparametric link prediction in dynamic networks. *arXiv preprint arXiv:1206.6394* (2012)
- Shumovskaia, V., Fedyanin, K., Sukharev, I., Berestnev, D., Panov, M.: Linking bank clients using graph neural networks powered by rich transactional data: Extended abstract. In: *2020 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2020*. IEEE (2020)
- Siddiqi, N.: Credit risk scorecards: developing and implementing intelligent credit scoring, vol. 3. John Wiley & Sons (2012)
- Tran, L., Tran, T., Mai, A.: Solve fraud detection problem by using graph based learning methods. *arXiv preprint arXiv:1908.11708* (2019)
- Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: *6th International Conference on Learning Representations, ICLR* (2018)
- Wachs, J., Fazekas, M., Kertész, J.: Corruption risk in contracting markets: A network science perspective. *International Journal of Data Science and Analytics* pp. 1–16 (2020)
- Wang, P., Xu, B., Wu, Y., Zhou, X.: Link prediction in social networks: the state-of-the-art. *Science China Information Sciences* **58**(1), 1–38 (2015)
- Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C.E., Schardl, T.B.: Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076* (2018)
- Weisfeiler, B.Y., Leman, A.A.: Reduction of a graph to a canonical form and an algebra arising during this reduction (1968)
- Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983. ACM (2018)
- Yu, W., Cheng, W., Aggarwal, C.C., Chen, H., Wang, W.: Link prediction with spatial and temporal consistency in dynamic networks. In: *IJCAI*, pp. 3343–3349 (2017)
- Zhang, M., Chen, Y.: Weisfeiler-lehman neural machine for link prediction. In: *KDD*, pp. 575–583. ACM (2017)
- Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: *Advances in Neural Information Processing Systems*, pp. 5165–5175 (2018)
- Zhao, P., Aggarwal, C., He, G.: Link prediction in graph streams. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 553–564. IEEE (2016)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.