Graph Representation Learning of Banking Transaction Network with Edge Weight-Enhanced Attention and Textual Information

Naoto Minakawa
The University of Tokyo
Tokyo, Japan
naoto-minakawa@g.ecc.u-tokyo.ac.jp

Kiyoshi Izumi
The University of Tokyo
Tokyo, Japan
izumi@sys.t.u-tokyo.ac.jp

Hiroki Sakaji
The University of Tokyo
Tokyo, Japan
sakaji@sys.t.u-tokyo.ac.jp

Hitomi Sano
The University of Tokyo
Tokyo, Japan
sano-h@g.ecc.u-tokyo.ac.jp

ABSTRACT

In this paper, we propose a novel approach to capture inter-company relationships from banking transaction data using graph neural networks with a special attention mechanism and textual industry or sector information. Transaction data owned by financial institutions can be an alternative source of information to comprehend real-time corporate activities. Such transaction data can be applied to predict stock price and miscellaneous macroeconomic indicators as well as to sophisticate credit and customer relationship management. Although the inter-company relationship is important, traditional methods for extracting information have not captured that enough. With the recent advances in deep learning on graphs, we can expect better extraction of inter-company information from banking transaction data. Especially, we analyze common issues that arise when we represent banking transactions as a network and propose an efficient solution to such problems by introducing a novel edge weight-enhanced attention mechanism, using textual information, and designing an efficient combination of existing graph neural networks.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Graph Attention Network, Graph Isomorphism Network, Graph Autoencoder, Doc2Vec, Graph Representation Learning, Natural Language Processing, Financial Transaction Network

ACM Reference Format:
Naoto Minakawa, Kiyoshi Izumi, Hiroki Sakaji, and Hitomi Sano. 2022. Graph Representation Learning of Banking Transaction Network with Edge Weight-Enhanced Attention and Textual Information. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3487553.3524643

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524643

1 INTRODUCTION

Banking transactions can tell us real-time corporate activities. Large banks possess huge amount of transactions, covering most companies from listed companies to start-ups in almost all industries and different types of transactions. Indeed, the previous work to nowcast GDP using Turkish bank transactions reports that individual-to-firm transactions in Garanti BBVA database account for 6% of Turkish GDP as well as firm-to-firm transactions in the same database account for 36% of Turkish GDP, verifying the usefulness of banking transaction data. [1] We can expect that it is related to stock price and miscellaneous macroeconomic indicators. A bank can also leverage such data to sophisticate credit and customer relationship management. Extracting information from banking transactions is beneficial for financial institutions in many ways.

Building a network from transaction data is effective to extract important information from such data. To highlight the motivation and significance of building a transaction network, let us consider a specific example; If we are to predict stock price or macroeconomic indicators from banking transactions, orthodox statistical or machine learning methods such as regression and gradient boosting [10, 19] can be used. For instance, we predict a company's stock price using features, such as industry or sector information, number of counterparties, and respective transaction amount. However, in these approaches, we cannot sufficiently take into account inter-company relationships that a company has with its counterparties.

To illustrate this, suppose Company A sends 1 million dollars to Company B, 0.5 million dollars to Company C, 0.1 million dollars to Company D, and receives 1.5 million from Company E and 0.4 million from Company F, as illustrated in Figure 1. In this example, Companies B and E are important counterparties to Company A in terms of the transaction amount. In reality, Companies B and E can have their respective counterparties to send and receive huge amount of money. Such counterparties of Companies B and E may influence Company A than Company D, to which Company A sends only 0.1 million. It is difficult to consider this kind of effect with the aforementioned orthodox machine learning models.

To solve this issue, we can represent banking transactions by formulating transaction networks with bank accounts as nodes, transaction occurrence as edges, and transaction amount as edge

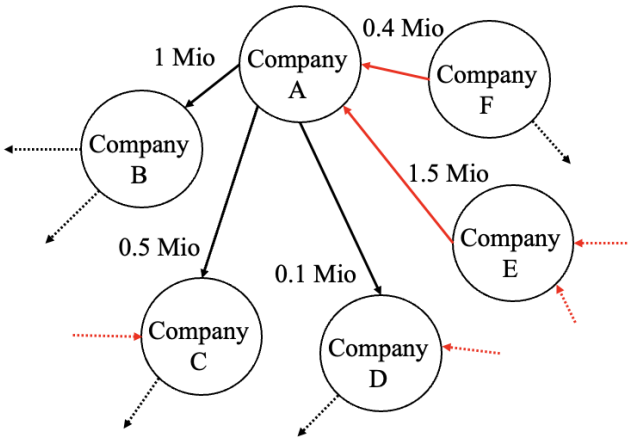

Figure 1: Inter-company relationship appeared in banking transactions

weights, as illustrated in Figure 1. Although complex network analysis can be an effective approach to analyze such network [6], there have been recent advances in the area of deep learning on graphs to capture more complex information [16, 24, 29]. We can obtain embedding of nodes that represents information about each account considering the aforementioned inter-company relationship using graph representation learning techniques. In this paper, we propose a novel approach to capture such information using graph neural networks with a special attention mechanism and textual industry or sector information.

The main contributions of this paper are summarized as follows:

- We proposed a novel efficient attention mechanism for banking transaction networks to consider nodes and edges in a well-balanced fashion when the dimension of edge features is much smaller than that of node features, which can be a common issue when we construct financial networks.
- We adopted textual description of bank accounts as node features when representing banking transaction data as a network. To the best of our knowledge, the combination of textual data and graph neural networks has not been studied for banking transaction data.
- We designed the entire network by combining existing graph neural networks that work well to capture information from the banking transaction network.

The remainder of the paper is organized as follows. Section 2 introduces the related work on the application of graph representation learning to financial transaction data. Section 3 introduces preliminary concepts and proposed design. Section 4 states about the experiment. Section 5 discusses the results. Finally, Section 6 presents the conclusion.

2 RELATED WORK

Recently, the application of graph neural networks and graph representation learning have been widely investigated in a wide range of domains, such as physics, biology, knowledge graph, traffic, social network, language, and image. The specific application includes physical systems modeling, protein interface prediction, side effects prediction, knowledge graph completion, traffic state prediction, user-item interaction prediction, social recommendation, question answering, image classification, among others [29].

In the financial domain, graph neural networks have been applied in the following categories: stock movement prediction, loan default risk prediction, recommender system of e-commerce, fraud detection, and event prediction [16, 24, 29].

The most similar previous works to this paper are which applied various graph embedding methods to banking transactions.

Fujitsuka et al. [7] applied node2vec [8] and traditional network analysis methods, such as Jaccard coefficient [17], to banking transactions and then predicted transaction occurrence between companies. However, neither graph neural networks nor textual information was used in the previous work. Only topological information of the network is considered in this work.

A different approach is to consider sequences of financial transactions as a graph, where a customer engaging in a transaction at two merchants within a specified time window constitutes an edge between those two merchant in the network. [11] With this idea, they formulate bipartite graphs of credit card transactions and learn embeddings of account and merchant entities. This framework is inspired by metapath2vec. [5] However, their work also captures network topology only.

Shumovskaia et al. [22] proposed a graph neural network- based approach, which uses not only the topological structure of the network but rich time-series data available for the graph nodes and edges, evaluating the developed method by link prediction task using the data provided by a large European bank. They leverage concepts from graph convolutional network (GCN) [13], graph attention network (GAT) [23], and SEAL [28]. While their focus is on the temporal characteristics appearing in banking transaction data, our approach is to enrich node features, efficiently capture edge information, and explore different combination of networks from their work.

3 PROPOSED NETWORK DESIGN TO EXTRACT INFORMATION FROM BANKING TRANSACTIONS

In this section, we present the architecture to efficiently extract information from a banking transaction network. Its idea is to use an embedded textual description of banking accounts as node features, encode transaction network by modified graph attention network (GAT) [23] to consider transaction amount as well as graph isomorphism network (GIN) [27] to obtain graph embedding and then to decode using graph embedding to predict transaction occurrence. We aim to capture the importance of linked accounts to a particular account by considering account description and transaction amount using modified GAT [23] in the first layer of the proposed network design and then increase the chance that obtained node features are processed in an injective manner using GIN [27] in the second layer. Figure 3 shows the entire design of the proposed network.

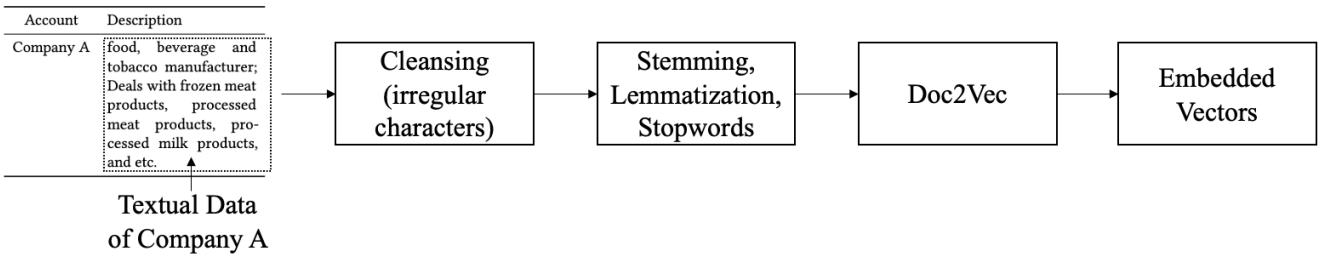

Figure 2: Pipeline to obtain node features

3.1 Obtaining Node Features

First, textual data describing the industry or sector of each account are extracted. Table 1 presents an example of textual data. As shown in Table 1, such texts include information about the industry or sector of the corresponding account. In this example, textual data possess information that Company A, the account holder, deals with processed meat products and is considered to be in the food industry. Then, extracted textual data are cleaned since it includes irregular characters, such as spaces and newline characters. Once the data are cleaned, we apply stemming and lemmatization, remove stopwords (e.g., "a," "the," "of," ...) and finally feed into Doc2Vec [15] model to obtain embedding for accounts. Figure 2 shows the entire pipeline to obtain node features.

Table 1: Illustrative example of account description

Account	Description
Company A	food, beverage, and tobacco manufacturer; Deals with frozen meat products, processed meat products, processed milk products, and etc.

3.2 Doc2Vec

Doc2Vec [15] is a model used to obtain sentence embedding. It is similar to Word2Vec [18]; therefore, we first introduce Word2Vec [18]. Word2Vec [18] consists of two models: continuous bag of words (CBOW) and skip-gram. CBOW considers surrounding words around a current word as context and then predicts the current word using the context words. After the training of the prediction, obtained feature vectors become word vectors. Skip-gram is the opposite of CBOW. It predicts surrounding words around a particular word using the word. Similarly, obtained feature vectors after training become word vectors. Stochastic gradient descent algorithms are used for the training.

Doc2Vec [15] has two variations: distributed memory model of paragraph vectors (PV-DM) and paragraph vector with distributed bag of words (PV-DBOW). PV-DM corresponds to CBOW, whereas PV-DBOW corresponds to skip-gram.

PV-DM takes an additional paragraph vector on top of context words to predict a current word. The paragraph vector is shared across all contexts generated from the same paragraph but not across paragraphs. After being trained to predict a current word, the paragraph vectors can be used as features for the paragraph.

PV-DBOW forces the model to predict words randomly sampled from the paragraph in the output. Since it ignores the context words in the input, PV-DBOW is more computationally efficient but less accurate than PV-DM. After being trained to predict words randomly sampled from the paragraph, the paragraph vectors can be used as features for the paragraph.

3.3 Graph Attention Network

Let us denote a network as $G(V, E)$, node as $u \in V$, edge as $(u, v) \in E$, neighbor nodes of node $u \in V$ as $\mathcal{N}(u)$, $\mathbf{A}$ as the adjacency matrix. We denote $\mathbf{h}_u^{(k)}$ as an embedding of node u in k-th layer in the neural network. $\mathbf{h}_u^{(0)}$ is equivalent to the feature vector of node u, which is a look up of node feature matrix $\mathbf{H} \in \mathbb{R}^{|V| \times d}$. $\mathbf{W}^{(k)} \in \mathbb{R}^{|\text{Hidden dim.}| \times |\text{Input dim.}|}$ is a shared trainable parameter in k-th layer in the network, where |Hidden dim.| and |Input dim.| represents the dimensions of the hidden and input layers, respectively. $\alpha_{u,v}$ is an attention score from node u to node v. σ represents an activation function. We adopted the most widely used **ReLU** as an activation function.

With those notations, GAT [23] is formulated as follows. The message aggregating function $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ from neighbor nodes $\mathcal{N}(u)$ in the k-th layer is given by

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v^{(k)}.$$

Here, the attention score is computed as follows:

$$\alpha_{u,v} = \frac{exp\left(\textbf{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}^{(k)} \mathbf{h}_u^{(k)} || \mathbf{W}^{(k)} \mathbf{h}_v^{(k)}])\right)}{\sum_{v' \in \mathcal{N}(u)} exp\left(\textbf{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}^{(k)} \mathbf{h}_u^{(k)} || \mathbf{W}^{(k)} \mathbf{h}_{v'}^{(k)}])\right)},$$

where $||$ denotes the concatenation operation, and $\mathbf{a}^\top \in \mathbb{R}^{2|\text{Hidden dim.}|}$ is a weight vector. Then, the embedding for node u is updated as follows:

$$\mathbf{h}_u^{(k+1)} = \sigma\left(\mathbf{W}^{(k)} \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right).$$

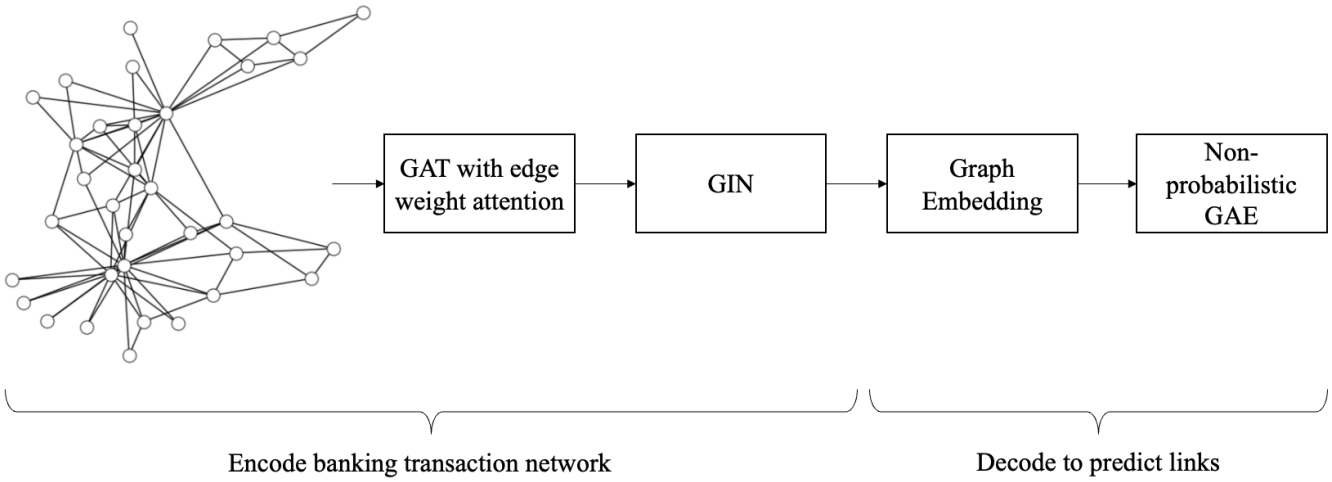

Figure 3: Proposed design of network

3.4 Edge Weight-Enhanced Attention Mechanism for Banking Transactions

In this subsection, we present a novel edge-enhanced mechanism to consider node and edge information in a well-balanced fashion. The most standard formulation to consider edge features when computing attention is to concatenate edge feature vector $\mathbf{e}$, where $\mathbf{e}_{u,v}$ represents transaction amount from node u to node v in this case, to node features $\mathbf{h}_u^{(k)}$ and $\mathbf{h}_v^{(k)}$ when computing attention from node u to node v [2], as shown below:

$$\alpha_{u,v} = \frac{exp\left(\mathbf{LeakyReLU}(\mathbf{a'}^{\top}[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}||\mathbf{W}_e^{(k)}\mathbf{e}_{u,v}])\right)}{\sum_{v' \in \mathcal{N}(u)} exp\left(\mathbf{LeakyReLU}(\mathbf{a'}^{\top}[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_{v'}^{(k)}||\mathbf{W}_e^{(k)}\mathbf{e}_{u,v'}])\right)},$$

where $\mathbf{W}_e^{(k)} \in \mathbb{R}^{|E| \times d_e}$ is a shared weight parameter in k-th layer for edge features; d_e denotes the dimension of edge features and $\mathbf{a'}^{\top} \in \mathbb{R}^{2|\text{Hidden dim.}|+d_e}$ is a weight vector. This type of edge feature- enhanced attention mechanism is implemented in PyTorch Geometric [1] which is a popular library built upon PyTorch [2] for graph neural networks.

The message aggregating function $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ and the update function for node u are the same as the above original GAT. That is to say,

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v}\mathbf{h}_v^{(k)},$$

$$\mathbf{h}_u^{(k+1)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{m}_{\mathcal{N}(u)}^{(k)}\right).$$

However, the edge effect in the above attention mechanism is considered limited in our case. Although a node feature vector obtained by Doc2Vec has d dimensions, and |Hidden dim.| appeared in the weight vector $\mathbf{W}^{(k)}$ is in the order of ten or hundred, an edge feature vector, a transaction amount, is a scalar value. We adopted $d = 300$ and |Hidden dim.| = 128 in our experiments as mentioned in the later section.

In other words, $\mathbf{W}_e^{(k)}\mathbf{e}_{u,v}$ accounts for only 1 dimension out of 2|Hidden dim.| + 1 dimensions, which is the dimension of the concatenated feature $\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}||\mathbf{W}_e^{(k)}\mathbf{e}_{u,v}$, indicating that edge information contributes little in determining the importance of neighbor nodes.

This issue is not a problem in our setting but can appear when we represent financial transactions like banking transactions as a network. More specifically, the dimension of node features can easily be scaled up to a higher dimension by taking different kinds of information about the node, such as textual data about nodes like our formulation. As another example, one can concatenate miscellaneous information about the account, such as the number of employers, revenue, profit, asset, and liabilities.

However, candidates for an edge feature are limited in banking transactions. The transaction amount is the most important information in banking transactions since it indicates how much money Company A sends to Company B. Transaction frequency and descriptive statistics can be another candidate to be concatenated with transaction amount to increase the dimension of edge features. However, the dimension of edge features cannot be easily scaled up as that of node features. Moreover, transaction amount is more important than transaction frequency and descriptive statistics from the practical viewpoint to assess the importance of accounts. Therefore, the dimension of edge features is much smaller than that of node features.

To alleviate this issue, we propose an alternative modification to the aforementioned attention mechanism to consider node features and edge weights in a well-balanced fashion as follows:

$$\alpha_{u,v} = \frac{exp\left(\mathbf{LeakyReLU}\left(\mathbf{a}^{\top}[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}] + \frac{log(e_{u,v})}{\gamma}\right)\right)}{\sum_{v' \in \mathcal{N}(u)} exp\left(\mathbf{LeakyReLU}\left(\mathbf{a}^{\top}[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_{v'}^{(k)}] + \frac{log(e_{u,v'})}{\gamma}\right)\right)}.$$

[1] https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html
[2] https://pytorch.org/

Here, $\mathbf{a}^\top[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}]$ is 1-dimensional since it is a dot product of $1 \times 2|$Hidden dim.$|$- vector $\mathbf{a}^\top$ and $2|$Hidden dim.$|\times 1$-vector $[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}]$. Inside **LeakyReLU**(), we sum 1-dimensional node information and 1-dimensional edge weight. The influence of an edge weight is the same as that of a node feature in terms of dimension.

The reason for taking $log()$ for an edge weight is that the transaction amount varies from the order of 10 (USD) to the order of 100,000,000 (USD). Only a handful of accounts possess such huge transactions; thus, we intend to mitigate the skewness of large values. Additionally, we do not place parameter in front of $log(e_{u,v})$ because we consider transaction amount $log(e_{u,v})$ itself to be a sort of weight parameter, which is the importance of an account to another account.

Furthermore, γ is the parameter used to adjust the influence of the edge weight $log(e_{u,v})$. We can adjust the value range of $log(e_{u,v})$ similar to that of $\mathbf{a}^\top[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}]$, by observing them and controlling the edge influence more flexibly. We adopted $\gamma = 10000$ by observing the value range of $\mathbf{a}^\top[\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}||\mathbf{W}^{(k)}\mathbf{h}_v^{(k)}]$ and $log(e_{u,v})$.

Here, for simplicity, we assume that we are only interested in transaction amount as an 1-dimensional edge feature because it is the most important edge information when formulating a banking transaction network. If we want to consider multi-dimensional features, we can set $e_{u,v} = \sum_i \beta_i \mathbf{e}_{u,v}^{(i)}$, for instance. Note that $\mathbf{e}_{u,v}^{(i)}$ represents i-th component of an edge feature vector $\mathbf{e}_{u,v}$ as well as β_i represents a weight parameter.

3.5 Graph Isomorphism Network

Typical graph neural networks are based on message aggregation from neighbor nodes and update on the aggregated features. Such message passing graph neural networks are proven to be as powerful as the Weisfeiler–Lehman graph isomorphism test in distinguishing different graphs. [27].

Weisfeiler–Lehman graph isomorphism test is used to distinguish different graphs. First, an initial label is assigned to each node in a graph. Then, a new label is iteratively assigned to each node by hashing the multiset of the current labels, which are assigned to neighbor nodes of each node. The iteration halts when relabeling of each node converges [9, 25].

Among message passing graph neural networks, it is proven that the network is as powerful as the Weisfeiler–Lehman test when the neighbor aggregation functions are injective [27].

GIN [27] is a network that satisfies the injectivity of neighbor aggregation function, which is described as follows:

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k)},$$

$$\mathbf{h}_u^{(k+1)} = \mathbf{MLP}^{(k+1)}\left((1+\epsilon^{(k+1)})\mathbf{h}_u^{(k)} + \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right).$$

Here, $\epsilon^{(k)}$ is a learnable parameter or a fixed scaler in the k-th layer.

In previous studies, the authors used one-hot encodings as input node features. Thus, they do not need multilayer perceptrons (MLPs) before summation since the summation of one-hot encodings alone satisfies injectivity. In our case, node features are not necessarily unique since account description is about an industry or a sector of a particular account, which can be common for some accounts. To increase the chance of obtaining injectivity, we applied MLP for node features as follows:

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \sum_{v \in \mathcal{N}(u)} \mathbf{MLP}^{(k)}(\mathbf{h}_v^{(k)}),$$

$$\mathbf{h}_u^{(k+1)} = \mathbf{MLP}^{(k+1)}\left((1+\epsilon^{(k+1)})\mathbf{MLP}^{(k)}(\mathbf{h}_u^{(k)}) + \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right).$$

3.6 Injectivity and Motivation to Use Textual Information

To the best of our knowledge, the use of textual data as node features in graph neural networks has not been studied for banking transaction data. In this subsection, we explain the motivation to use textual data as node features.

The use of industry or sector description is indeed less effective than one-hot encodings in terms of injectivity because the number of industry or sector is less than the number of accounts. However, adding more textual descriptions on industry or sector information easily defines a unique description of a particular account. Moreover, we can introduce the similarity of different nodes using textual information, which is not achieved by one-hot encodings. For example, suppose Company A is a meat processing company, Company B is a milk-producing company, and Company C is an automobile retailer. Companies A, B, and C are represented as $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ with one-hot encodings, respectively, and there is no more information than they are distinguishable. However, they are represented as dense vectors with Doc2Vec, which can capture similarity among companies: Companies A and B should be embedded close, and Company C should be embedded far from Companies A and B. This is because Companies A and B are in the food industry, and Company C is in the automobile industry. Additionally, Doc2Vec can learn the nuance of a slight difference between Companies A and B that they are in the same industry but deal with different products.

3.7 Number of Layers

The over-smoothing problem is well known when using graph neural networks with the aforementioned message passing framework with neighbor node aggregation and update functions. In this framework, adding more layers into the network results in that the node representations converge to indistinguishable features [20].

Although several attempts have been made to alleviate the over-smoothing problem [3, 14], it is not the focus of this paper. We simply avoided the over-smoothing problems by limiting the layer as shallow as two layers.

3.8 Non-probabilistic Graph Autoencoder

Once node features are passed through the GAT layer and succeeding GIN layer, we obtain embedded node features. The obtained embedding is decoded using a non-probabilistic graph autoencoder (GAE) [12].

Let us denote the obtained embedding through the GAT layer and succeeding GIN layer as $\mathbf{Z}$. In this case, $\mathbf{Z}_u$ represents a node embedding for node u.

Then, the adjacency matrix $\mathbf{A}$ of the original network can be reconstructed as follows:

$$\mathbf{A} \approx \mathbf{sigmoid}(\mathbf{Z}\mathbf{Z}^T).$$

For a particular edge, its linkage is reconstructed as follows:

$$\mathbf{A}_{u,v} \approx \mathbf{sigmoid}(\mathbf{Z}_u\mathbf{Z}_v^T).$$

4 EVALUATION

4.1 Dataset

The dataset is provided by one of large Asian banks with all data in the depersonalized format. The data consist of monthly aggregated transactions that companies send or receive money to its counterparties, as shown in Table 2.

Textual data shown in Table 1, which describe industry or sector in English for each account, are originally obtained from Asian central bank. The central bank defines descriptions of different industries or sectors. The above Asian bank mapped the defined industry or sector to each account on their side. Thereafter, the monthly aggregated transactions and textual data were given to us in the depersonalized format.

Table 2: Aggregated monthly banking transaction dataset

Sender	Receiver	Amount
Company A	Company B	1,000,000
Company A	Company C	500,000
Company A	Company D	100,000
Company E	Company A	1,500,000
Company F	Company A	400,000

Based on the transaction data, we construct a directed network with an embedded description of accounts as node features, transaction occurrence for pairs of nodes as edges, and monthly aggregated transaction amount from an account to its counterparties as edge weight. The constructed network consists of 170,094 nodes and 1,244,639 edges, as presented in Table 3. Max degree, which is the number of counterparties an account has, is 15,431. The average degree is 7.2863; median degree is 3; min degree is 1. The average clustering coefficient is 0.0573. As mentioned in the previous section, the edge weights vary from the order of 10 (USD) to 100,000,000 (USD).

4.2 Task

We adopted the link prediction task to learn embeddings since the link prediction problem is interpreted as the prediction of transaction occurrence, which is an inter-company relationship. Additionally, prediction of transaction occurrence is beneficial from the practical viewpoint. For instance, if we can accurately predict the transaction occurrence of a client, a bank can estimate if they are missing potential transactions from a client, implying useful for customer relationship management.

Table 3: Information on the constructed network

Information	Value
Number of Nodes	170,094
Number of Edges	1,244,639
Max Degree	15,431
Average Degree	7.2863
Median Degree	3
Min Degree	1
Average Clustering Coefficient	0.0573

We adopted ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score [19] as evaluation metrics. ROC Curve is a probability curve that plots the true positive rate (TPR) against false positive rate (FPR), where TPR and FPR is computed as follows.

$$TPR(\text{true positive rate}) = \frac{TP}{TP + FN}$$
$$FPR(\text{false positive rate}) = \frac{FP}{TN + FP}$$

ROC-AUC stands for the area under the ROC curve. It is standard practice to evaluate link prediction tasks by ROC-AUC scores as seen in previous works. [7, 22]

4.3 Baseline Methods

As baselines, we adopted several popular graph representation learning methods: node2vec [8], GCN [13], GAT [23] and GIN [27]. Complex network analysis methods, such as Jaccard coefficient [17], are also widely used for link prediction; however, they have been tested in previous work and did not outperform node2vec [7]. Therefore, we did not use them in our experiment.

node2vec [8] is one of the most popular approaches among graph representation learning techniques called shallow embeddings [9]. Shallow embedding owes its name to the fact that the encoder model to map nodes to embeddings is simply a lookup of embeddings. Compared to graph neural networks, the following three drawbacks are known [9].

First, it does not share parameters between nodes when encoded, resulting in being a statistically and computationally inefficient method. Second, it does not leverage node features, being potentially less informative. Third, it can generate embeddings for nodes present during the training. This is called *transductive*. In contrast, graph neural networks can generate embeddings for unseen nodes that were not present during the training. This is called *inductive*.

node2vec [8] is similar to DeepWalk [21], and both of them use a random walk to obtain node embeddings. DeepWalk [21] runs a random walk for each node in a graph to generate random sequences of nodes. It generates embeddings for each node by training a skip-gram algorithm of Word2Vec [18]. node2vec [8] is different from DeepWalk [21] in the sense that node2vec uses a second-order random walk with two parameters p and q to control focuses of a random walk if it explores the close area or far area from the current node.

Table 4: Comparison of different approaches

Approach	ROC-AUC Score (Test)
node2vec [8]	0.7778
Graph Convolutional Network (GCN) [13] with textual information	0.8694
Graph Attention Network (GAT) [23] with textual information	0.8522
Graph Isomorphism Network (GIN) [27] with textual information	0.8836
Proposed Network	**0.9497**

GCN [13] is one of the most popular models among graph neural networks. The model structure of GCN is relatively simple and often shows good performance. Therefore, it is commonly used as a benchmark for comparison. It is categorized as a spectral-based method since it has a solid mathematical foundation in graph signal processing [26]. GCN [13] is formulated as follows:

$$\mathbf{H}^{(k+1)} = \sigma\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(k)}\mathbf{W}^{(k)}\right).$$

Here, $\mathbf{H} \in \mathbb{R}^{|V|\times d}$ is a node feature matrix; k is the number of layers; $\mathbf{H}^{(k)} \in \mathbb{R}^{|\text{Hidden dim.}|\times|\text{Input dim.}|}$ as a node embedding in k-th layer in the network; $\tilde{\mathbf{A}}$ is the adjacency matrix with added self-connections; $\mathbf{W}^{(k)} \in \mathbb{R}^{|\text{Hidden dim.}|\times|\text{Input dim.}|}$ is a shared trainable parameter in k-th layer in the network; σ is an activation function. We used **ReLU** as an activation function and $k = 2$ as the number of layers. Note that $\mathbf{H}^{(0)} = \mathbf{H}$. $\tilde{\mathbf{D}}$ is a degree matrix represented as follows:

$$\tilde{\mathbf{D}}_{i,i} = \sum_j \tilde{\mathbf{A}}_{i,j}.$$

GAT [23] and GIN [27] are explained in detail in the previous section.

4.4 Experiment

We conducted an experiment to evaluate the performance of the proposed network design and aforementioned baseline models.

For the implementation, we used nltk [3] for preprocessing text data, gensim [4] for Doc2Vec, PyTorch geometric [5] for implementation of different baseline methods as well as our proposed design of the network, and scikit-learn [6] for computing evaluation metrics.

For node2vec [8], the bias parameter of the random walk is $p = q = 1$; the walk length of the random walk is 15; the context size is 10. The dimension of embedding is 128, and the window size is 10.

For graph neural network baseline models and our proposed network, we adopted the following settings. The dimension of node features d obtained from Doc2Vec is 300. We adopted 128 as the dimension of the output of the GAT layer and 64 as the output of the GIN layer. More specifically, for node $u \in V$, the dimension of $\mathbf{h}_u^{(0)}$ is 300; the dimension of $\mathbf{h}_u^{(1)}$ is 128; the dimension of $\mathbf{h}_u^{(2)}$ is 64.

[3]https://www.nltk.org/
[4]https://pypi.org/project/gensim/
[5]https://pytorch-geometric.readthedocs.io/en/latest/index.html
[6]https://scikit-learn.org/stable/

We adopted cross entropy as loss function, 0.001 as learning rate, 100 as the number of epochs, Adam as an optimizer. We use 70% of the entire edges as train edges, 10% of the entire edges as validation edges, and 20% of the entire edges as test edges. Regarding the train-test split, the split was conducted based on normal random sampling, not on stratified sampling. We record ROC-AUC score for test edges when the validation score is the best through running epochs.

The results of the experiment is shown in Table 4.

5 DISCUSSION

As presented in Table 4, the test ROC-AUC score of the proposed design is the highest as 0.9497. Comparing baseline methods with the proposed network, we analyze this is due to the expected effect of considering textual information, edge weight-enhanced attention mechanism, and choice of combination of networks to capture importance of neighbor nodes in the first GAT layer and to increase representation power in the succeeding GIN layer.

When we compare node2vec with other baseline models, such as GCN, GAT, and GIN, one of the key differences is whether we leverage node features, which is account description in this case, in addition to network topology. Hence, we can confirm the effect of considering textual information. As explained in the previous section, GIN has a higher score than GCN and GAT because of its representation power.

The key difference between the proposed method and standard graph neural networks such as GCN, GAT, and GIN is the use of the edge-enhanced attention mechanism. In the proposed network, information of transaction amount is considered on top of textual industry information when determining the importance of neighbor nodes. Hence, we can confirm the efficiency of the novel edge weight-enhanced attention mechanism.

In future studies, first, we will include more expressive power of the proposed network by adding more textual information for each account to make it unique, as discussed in Subsection 3.6. Due to the limitation of data availability, we tested with the industry or sector description of each account.

Additionally, we will conduct more experiments to demonstrate the detailed comparison between different methods, including comparison with SEAL [28] used in previous work [22] and comparison with the existing edge-enhanced attention mechanism [2]. Also, use of modern language models such as BERT [4] instead of Doc2Vec [15] should be tested.

Finally, we will use the obtained embedding for downstream tasks, such as stock price and macroeconomic indicator predictions. Since banking transactions possessed by large banks cover

most companies, from listed companies to start-ups in almost all industries, we can expect it is closely related to stock price and macroeconomic indicators.

6 CONCLUSIONS

In this paper, we propose a graph neural network-based method to obtain the representation of banking transaction networks. To capture information of each banking account, we use Doc2Vec to obtain an embedded vector of the textual industry or sector description for each account, enabling consideration of similarity among different nodes and scalable node representation to a more detailed unique description of accounts. Additionally, we use embedded descriptions of accounts as node features, transaction occurrence as edges, and monthly aggregated transaction amount from an account to its counterparties as edge weight to model banking transaction networks. Consequently, we achieved a better ROC-AUC score for predicting transaction occurrence using the novel edge weight-enhanced attention mechanism in GAT and GIN. We can expect to increase expressive power of the proposed network by adding more textual information for each account to make it unique as well as to apply obtained embeddings to prediction of stock price and macroeconomic indicator predictions.

ACKNOWLEDGEMENT

This work was supported by JST-Mirai Program Grant Number JPMJMI20B1, Japan.

REFERENCES

[1] Ali B. Barlas, Seda Guler Mert, Berk Orkun Isa, Alvaro Ortiz, Tomasa Rodrigo, Baris Soybilgen, and Ege Yazgan. 2021. Big Data Information and Nowcasting: Consumption and Investment from Bank Transactions in Turkey. (2021). http://arxiv.org/abs/2107.03299

[2] Jun Chen and Haopeng Chen. 2021. Edge-Featured Graph Attention Network. (2021). http://arxiv.org/abs/2101.07671

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 1725–1735. https://proceedings.mlr.press/v119/chen20v.html

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[5] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/3097983.3098036

[6] Janina Engel, Michela Nardo, and Michela Rancan. 2021. *Network Analysis for Economics and Finance: An Application to Firm Ownership*. Springer International Publishing, Cham, 331–355. https://doi.org/10.1007/978-3-030-66891-4_14

[7] Masashi Fujitsuka and Tsuyoshi Kudo. 2019. Latent Transaction Prediction Using Graph Embedding. In *The Japanese Society for Artificial Intelligence - 23th Workshop on Special Interest Group on Financial Informatics* (Tokyo, Japan) *(SIG-FIN '19)*.

[8] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA) *(KDD '16)*. Association for Computing Machinery, 855–864. https://doi.org/10.1145/2939672.2939754

[9] William L. Hamilton. 2020. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, 3 (2020), 1–159.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

[11] Anish Khazane, Jonathan Rider, Max Serpe, Antonia Gogoglou, Keegan Hines, C. Bayan Bruss, and Richard Serpe. 2019. DeepTrax: Embedding Graphs of Financial Transactions. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 126–133. https://doi.org/10.1109/ICMLA.2019.00028

[12] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *NIPS 2016 Workshop on Bayesian Deep Learning* (Barcelona, Spain) *(BDL '16)*. http://bayesiandeeplearning.org/2016/papers/BDL_16.pdf

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations* (Toulon, France) *(ICLR '17)*. https://openreview.net/forum?id=SJU4ayYgl

[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proceedings of the 7th International Conference on Learning Representations (ICLR '19)*. https://openreview.net/forum?id=H1gL-2A9Ym

[15] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, II–1188–II–1196.

[16] Xiaoxiao Li, Joao Saude, Prashant Reddy, Manuela Veloso, JP Morgan, and AI Research. 2020. Classifying and Understanding Financial Data Using Graph Neural Network. In *The AAAI-20 Workshop on Knowledge Discovery from Unstructured Data in Financial Services* (New York, NY, USA) *(AAAI-KDF '20)*.

[17] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A* 390, 6 (2011), 1150–1170.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations* (Scottsdale, Arizona, USA) *(ICLR '13)*. https://openreview.net/forum?id=idpCdOWtqXd60

[19] Kevin Patrick Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT press, Cambridge, MA, USA.

[20] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *Proceedings of the 8th International Conference on Learning Representations* (Addis Ababa, Ethiopia) *(ICLR '20)*. https://openreview.net/forum?id=S1ldO2EFPr

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[22] Valentina Shumovskaia, Kirill Fedyanin, Ivan Sukharev, Dmitry Berestnev, and Maxim Panov. 2021. Linking bank clients using graph neural networks powered by rich transactional data. *International Journal of Data Science and Analytics* 12, 2 (01 Aug 2021), 135–145. https://doi.org/10.1007/s41060-021-00247-3

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations* (Vancouver, Canada) *(ICLR '18)*. https://openreview.net/forum?id=rJXMpikCZ

[24] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. 2021. A Review on Graph Neural Network Methods in Financial Applications. (2021). http://arxiv.org/abs/2111.15367

[25] Boris Weisfeiler and A. A. Lehman. 1968. A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. *Nauchno-Technicheskaya Informatsia* Ser. 2, N9 (1968), 12–16.

[26] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 7th International Conference on Learning Representations* (New Orleans, Louisiana, USA) *(ICLR '19)*. 1–17. https://openreview.net/forum?id=ryGs6iA5Km

[28] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems 31* (Montréal, Canada) *(NeurIPS '18, Vol. 31)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 5165–5175. http://papers.nips.cc/paper/7763-link-prediction-based-on-graph-neural-networks.pdf

[29] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001