HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction

Raehyun Kim[a], Chan Ho So[b], Minbyul Jeong[a], Sanghoon Lee[a], Jinkyu Kim[a] and Jaewoo Kang[a,b,*]

[a]Department of Computer Science and Engineering, Korea University, 145, Anam-ro, Seongbuk-gu, Seoul, South Korea, 02841
[b]Interdisciplinary Graduate Program in Bioinformatics, Korea University, 145, Anam-ro, Seongbuk-gu, Seoul, South Korea, 02841

ARTICLE INFO

Keywords:
Stock market
Graph Neural Network
Corporate Relation
Financial data

ABSTRACT

Many researchers both in academia and industry have long been interested in the stock market. Numerous approaches were developed to accurately predict future trends in stock prices. Recently, there has been a growing interest in utilizing graph-structured data in computer science research communities. Methods that use relational data for stock market prediction have been recently proposed, but they are still in their infancy. First, the quality of collected information from different types of relations can vary considerably. No existing work has focused on the effect of using different types of relations on stock market prediction or finding an effective way to selectively aggregate information on different relation types. Furthermore, existing works have focused on only individual stock prediction which is similar to the node classification task.

To address this, we propose a hierarchical attention network for stock prediction (HATS) which uses relational data for stock market prediction. Our HATS method selectively aggregates information on different relation types and adds the information to the representations of each company. Specifically, node representations are initialized with features extracted from a feature extraction module. HATS is used as a relational modeling module with initialized node representations. Then, node representations with the added information are fed into a task-specific layer. Our method is used for predicting not only individual stock prices but also market index movements, which is similar to the graph classification task. The experimental results show that performance can change depending on the relational data used. HATS which can automatically select information outperformed all the existing methods.

1. Introduction

Stock markets are a symbol of market capitalism and billions of shares of stock are traded every day. In 2018, stocks worth more than 65 trillion U.S. dollars were traded worldwide and market capitalization of domestic companies listed in the U.S. exceeds the country's GDP [1]. Although stock movement prediction is a difficult problem, its solutions can be applied to industry. Many researchers in both industry and academia have long shown interest in predicting future trends in the stock market. Researchers focused on finding profitable patterns in historical data are known as quants in the financial industry and referred to as data scientists in general. Regardless of which term is used, such researchers are increasingly using more systematic trading algorithms to automatically make trading decisions.

Even though there is still room for debate [21], numerous studies have showed that the stock market is predictable to some extent [5], [23]. Existing methods are based on the ideas of fundamentalists or technicians, both of whom have different perspectives on the market.

Fundamentalists believe that the price of securities of a company corresponds to the intrinsic value of the company or entity [8]. If the current price of a company's stock is lower than its intrinsic value, investors should buy the stock as its price will go up and eventually be the same as its fundamental value. The fundamental analysis of a company involves an in-depth analysis of its performance and profitability. The intrinsic value of the company is based on its product sales, employees, infrastructure, and profitability of its investments [2].

Technicians, on the other hand, do not consider real world events when predicting future trends in the stock market. For technicians, stock prices are considered as only typical time series data with complex patterns. With appropriate preprocessing and modeling, patterns can be analyzed, from which profitable patterns may be extracted. The information used for technical analysis consists of mainly closing prices, returns, and volumes. The movement of stock prices is known to be stochastic and non-linear. Technical analysis studies focus on reducing stochasticity and capturing consistent patterns.

Some technical analysis works have focused on how to extract meaningful features from raw price data. In the finance industry, features extracted from such data are called technical indicators and include adaptive moving average, relative strength index, stochastics, momentum oscillator, and commodity channel index. Creating meaningful technical indicators is similar to manual feature engineering in general machine learning tasks. Like in any machine learning task, extracted features contain important information for models. Hence, some works have utilized indicators to more accurately predict the movement of stock prices [9], [22].

Technicians are also interested in finding meaningful pat-

*Corresponding author
E-mail address: kangj@korea.ac.kr (J. Kang), raehyun@korea.ac.kr (R. Kim), chanhoso@korea.ac.kr (C. So), minbyuljeong@korea.ac.kr(M. Jeong), a11525@korea.ac.kr(S. Lee), no100kill@korea.ac.kr(J. Kim)
ORCID(s):

[1]https://data.worldbank.org/

terns in raw price data. Numerous works have analyzed the effectiveness of different models. Although the majority of researchers agree that the stock market moves in a non-linear way, many empirical studies show that non-linear models do not outperform linear models [1], [2]. The results of these studies show that even though deep neural network based models have been successfully applied to many challenging domains, careful consideration should be given when trying to design profitable models for stock market prediction. Lately, more studies are finding that non-linear models outperform linear models [24]. Many studies have shown that recurrent neural network based models are effective in stock movement prediction [3].

As the amount of information on the web continues to rapidly increase, it becomes easier to obtain information on securities from different sources. Data scientists with a computer science background have begun to pay attention to unstructured data such as text from the news and Twitter [4], [10]. Models that use text data reflecting the real world events of companies can be categorized as fundamental analysis models. However, text-based stock prediction approaches try to capture investors' opinions about an event. Based on the assumption that the price of a company's stock can be based on the total aggregation of investors' opinions about the company, some works focused on reading investor's opinions about companies [20]. There also exist researches focusing on understanding the impact of events on stock price [11].

More recently, computer science research communities have been highly interested in utilizing graph-structured data [16], [28]. Stock market prediction methods using corporate relational data have also been proposed [7], [13]. Chen et al. created a network of companies based on financial investment information [7]. Using a constructed adjacency matrix, they trained a GCN model and compared its prediction performance with that of more conventional network embedding models'. Feng et al. developed a more general framework [13] that involves using many different types of relations in a publicly available knowledge database. They also proposed a GNN model that can capture temporal features of stocks. Although these models were the first to integrate relational data for stock market prediction, they can be still be improved. The quality of information varies considerably depending on the type of relation. However, no existing work has thoroughly investigated which types of relational data are more beneficial to stock movement prediction or focus on finding an effective way to selectively aggregate information on different relation types.

Furthermore, previous works have focused mainly on node classification. Node classification and graph classification are the two main tasks in graph-based learning. In a stock market network, individual nodes typically represent companies. Predicting future trends in individual stock prices is similar to the node classification task. We argue that previously proposed models can be used as a node representation updating function in the graph classification task which we propose in this work.

To address the limitations mentioned above, in this paper, we study how to effectively utilize graph-based learning methods and relational data in stock market prediction. We use different types of relations and investigated their effect on performance in stock price movement prediction of individual companies. In our experiments, we found that only relevant relations are useful for stock prediction. Information from some irrelevant relations even degraded prediction performance. We propose HATS which is a new hierarchical graph attention network method that uses relational data for stock market prediction. HATS selectively aggregates information from different relations and adds the information to the representations of companies. Specifically, node features are initialized with extracted features from the feature extraction module. HATS is used as relational modeling module to selectively gather information from neighboring nodes. The representations with the added information are then fed into a task-specific prediction layer.

We applied our method to the following two graph related tasks: predicting the movement of individual stock, which is the same node classification task performed in previous works, and predicting the movement of the market index, which is similar to the graph classification task. This is a new way of adapting graph-based learning in stock market prediction. Since market indices consist of individual stocks, we can predict the movement of a market index using a graph classification based approach. The experimental results on both tasks demonstrate the effectiveness of our proposed method.

The main contributions of this work can be summarized as follows.

- We thoroughly investigate the effect of using relations in stock market prediction and find characteristics of more meaningful relation types.
- We propose a new method HATS which can selectively aggregate information on different relation types and add the information to each representation.
- We propose graph classification based stock prediction methods. Considering the market index as an entire graph and constituent companies as individual nodes, we predict the movement of a market index using a graph pooling method.
- We perform extensive experiments on stocks listed in the S&P 500 Index. Our experimental results demonstrate that the performance of our method in terms of Sharpe ratio and F1 score was 19.8% and 3% higher than the existing baselines, respectively.

The remainder of this paper is organized as follows. In Section 2, we provide short preliminaries that can be helpful in understanding our work. Detailed descriptions of our proposed framework are provided in Section 3. In Section 4, we explain how we collected the data used in our experiments. We discuss our experimental results in Section Section 5 and we conclude our work in Section 6.

2. Preliminaries

Graph Theory A graph is a powerful data structure which can be used to deal with relational data. Various methods learn meaningful node representations in graphs. In this section, we provide a brief preliminary about a graph based method. Graph $\mathcal{G}$ consists of a set of vertices (nodes) V and edges E. If a node is denoted as $v_i \in V$ and $e_{ij} \in E$ is an edge connecting node i and j, the Adjacency matrix A is an $n \times n$ matrix with $A_{ij} = w_{ij} > 0$. The degree of a node is the number of edges connected to the node, and is denoted as D where $d_{ii} = \sum_j a_{ij}$. Each node can have node features (attributes) X, where $X \in R^{n \times f}$ is a feature matrix.

The features of nodes change over time in a *spatial-temporal graph* which can be defined using a feature matrix $X \in R^{t \times n \times f}$ where t is the length of time steps.

Graph Neural Networks With a growing interest in utilizing graph-structured data, a large amount of research has been conducted for learning meaningful representations in graphs. Most graph neural networks (GNNs) can be categorized as spectral or non-spectral.

Spectral graph theory based methods such as GCN [17] utilize convolutional neural networks (CNN [18]) to capture local patterns in graph-structured data. GCN applies a spectral convolution filter to extract information in the Fourier domain.

$$f_\theta(M, x) = U M U^T x, \tag{2.1}$$

Equation (2.1) describes a spectral convolution filter f_θ used for graph data $x \in \mathbb{R}^n$ (n companies) and a diagonal matrix M. U is the eigenvector matrix of a graph Laplacian matrix.

However, in large graph data, computing eigendecomposition of graph Laplacian is computationally too expensive. To address this problem, Kipf and Welling approximated spectral filters in terms of Chebyshev polynomials $T_k(x)$ up to k^{th} order based on Chebyshev coefficient θ_k, which can be defined as follows.

$$M \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\wedge}), \tag{2.2}$$

where $\tilde{\wedge} = \frac{2}{\lambda_{max}} \wedge -I$ with λ_{max} denotes the largest eigenvalue of graph Laplacian L.

Additionally, Chebyshev coefficients could be represented as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_1 x = x$ and $T_0 x = 0$. In [17], GCN is proven to be effective with the parameter setting of K=1. Also, they simply transformed Equation (2.2) as a fully connected layer with a built-in convolution filter.

On the other hand, non-spectral approaches directly define convolution operations directly on the graph, utilizing spatially close neighbors. For example, Hamilton et al. proposed a general framework for sampling and aggregating features from the local neighborhood of a node to generate embeddings. Specifically, features of neighboring nodes are aggregated iteratively using a learnable aggregation function, which is described as follows.

$$h^k_{\mathcal{N}(v)} \longleftarrow AGGREGATE(h^{k-1}_u, \forall u \in \mathcal{N}(v)) \tag{2.4}$$

$$h^k_v \longleftarrow \sigma\big(W^k \cdot CONCAT(h^{k-1}_u, h^k_{\mathcal{N}(v)})\big) \tag{2.5}$$

where h^k_u denotes the representation of node u at k-th iteration and $AGGREGATE$ is a learnable aggregation function. Many proposed methods can be considered as special types of aggregation functions. For example, Veličković et al. assigned different weights using attention mechanism to aggregate features of neighboring nodes [25].

Updated node representations can be used in both node classification and graph classification tasks. For a graph classification task, additional layers are needed to sum individual node representations and make graph representations. Graph pooling is a technique used in making graph representations. Numerous works which can effectively aggregate node features have been proposed [19], [28].

GNN methods have proven to achieve state-of-the-art performance in various tasks such as link prediction [29], social network community structure prediction [6], and recommendation [27].

3. Methodology

In this section, we will first explain our entire framework. Our framework is based on many different stock market prediction methodologies that use corporate relational data. Knowing how the general framework functions can help in understanding the importance of using relational data in stock prediction. The overall framework is shown in Fig. 1. After providing a general description of the framework, we will elaborate on the structure of our method HATS is a new type of relational modeling module.

3.1. General framework

Feature Extraction Module A stock market graph is a typical type of spatial-temporal graph. If we regard individual stocks (companies) as nodes, each node feature can represent the current state of each company with respect to price movement. Also, node features can evolve over time. As mentioned in Section 1, numerous types of data (e.g. historical price, text or fundamental analysis based sources) can be used as an indicator for the movement of a stock price. As data such as raw text or price data are not informative enough, we need a feature extraction module for obtaining meaningful representations of individual companies. In this study, we use only historical price data.

A feature extraction module is used to represent the current state of a company based on historical movement patterns. In this study, we use LSTM and GRU as our feature extraction modules. LSTM is the most widely used framework in time series modeling and [7] and [13] have also used LSTM as their feature extraction module. For a more detailed description of how node feature vectors are extracted from raw price data, we refer readers to [14]. We also use GRU as a feature extraction module as it is known to be more efficient than LSTM in time series tasks, and obtains similar performance with appropriate tuning. From our experiments, we found that LSTM performs slightly better than GRU on average. However, it was more difficult to train

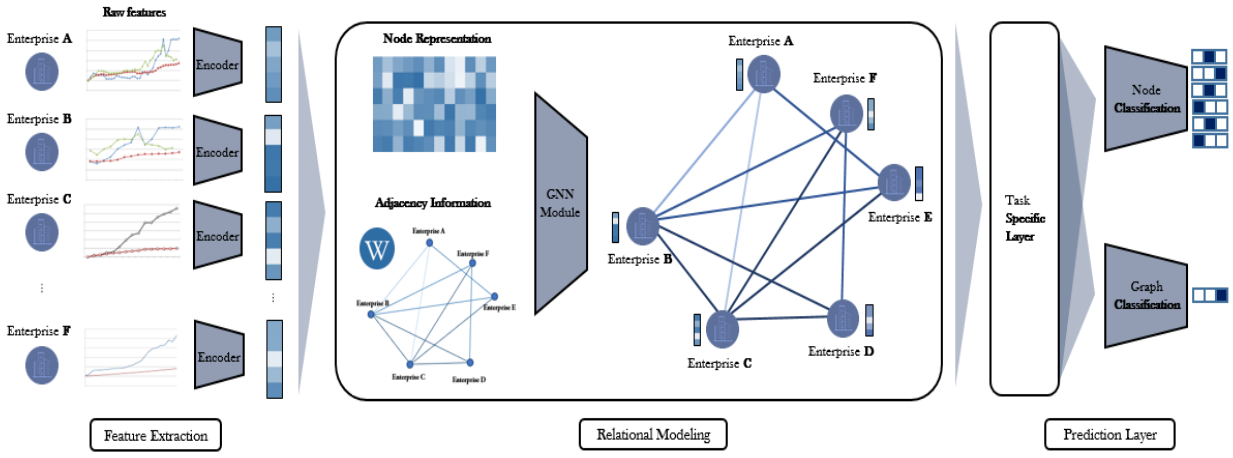

Figure 1: General framework of stock prediction using relational data.

LSTM especially when a model had more layers. For this reason, we use LSTM for the individual stock prediction task and GRU for the index movement prediction task where an additional graph pooling layer is needed.

Relational Modeling Module A relational modeling module is a node updating function. Gilmer et al. considered graph-based learning as information exchange between related nodes [15]. The main function of graph neural networks is information exchange between neighboring nodes. Information from neighboring nodes is aggregated and then added to each node representation. Information collected from different nodes and relation types needs to be effectively combined. To this end, we propose a new GNN based Hierarchical graph Attention Network for Stock market prediction (HATS) method. Each layer is designed to capture the importance of neighboring nodes and relation types. A detailed description of our proposed method HATS is provided in the below section Subsection.

Task-Specific Module After node representations are updated using relational modeling, the node representations are fed into the task-specific module. Since node representations can be used in various tasks with appropriate modeling, the layer is considered "task-specific." In this study, we performed experiments on the following two graph-based learning tasks: individual stock prediction and market index prediction. Individual stock prediction is similar to the node classification task which was performed in previous researches [7], [13]. As market indices consist of multiple related stocks, information on the current state of an individual company can be utilized to predict the movement of its index. As recently proposed graph pooling methods can be used to aggregate information of individual nodes to represent an entire graph, they can also be used for the index prediction task. The experimental results in Section 5 demonstrate that the graph pooling methods outperform all baseline methods.

In the next subsections, we describe HATS in more detail. We present our method HATS which aggregates information and adds it to node representations. Then, we explain how we use node representations with added information in two different tasks.

3.2. Hierarchical Attention Network

Let us denote a f-dimensional feature vector from a feature extraction module of a company i at time t as $e_i^t \in R^f$. In figure 2, we omit superscript t for simplicity, assuming that all the node representation vectors are calculated at time step t. We can define edges between different types of relations. For the graph neural network operation, we have to know the set of neighboring nodes for our target node i from each relation type. Let us denote the set of neighboring nodes of i for relation type m as $N_i^{r_m}$ and the embedding vector of relation type m as $e_{r_m} \in R^d$. Here, d is a dimension of a relation type embedding vector. Our goal is to selectively gather information on different relations from neighboring nodes. We want our models to filter information that is not useful for future trend prediction. This process is important because companies have many different types of relationships and some information is not related to movement prediction.

Attention mechanism is widely used to assign different weight values for information selection. With hierarchically designed attention mechanism, our Hierarchical Attention network for Stock prediction (HATS) selects only meaningful information at each level. Its hierarchical attention network is key in improving performance. The architecture of HATS is shown in Fig. 2.

At the first *state attention layer*, HATS selects important information on the same type of relation from a set of neighboring nodes. The attention mechanism is used to calculate different weights based on the current state (representation) of a neighborhood node. To calculate the state attention scores, we concatenate relation type embedding e_{r_m} and the

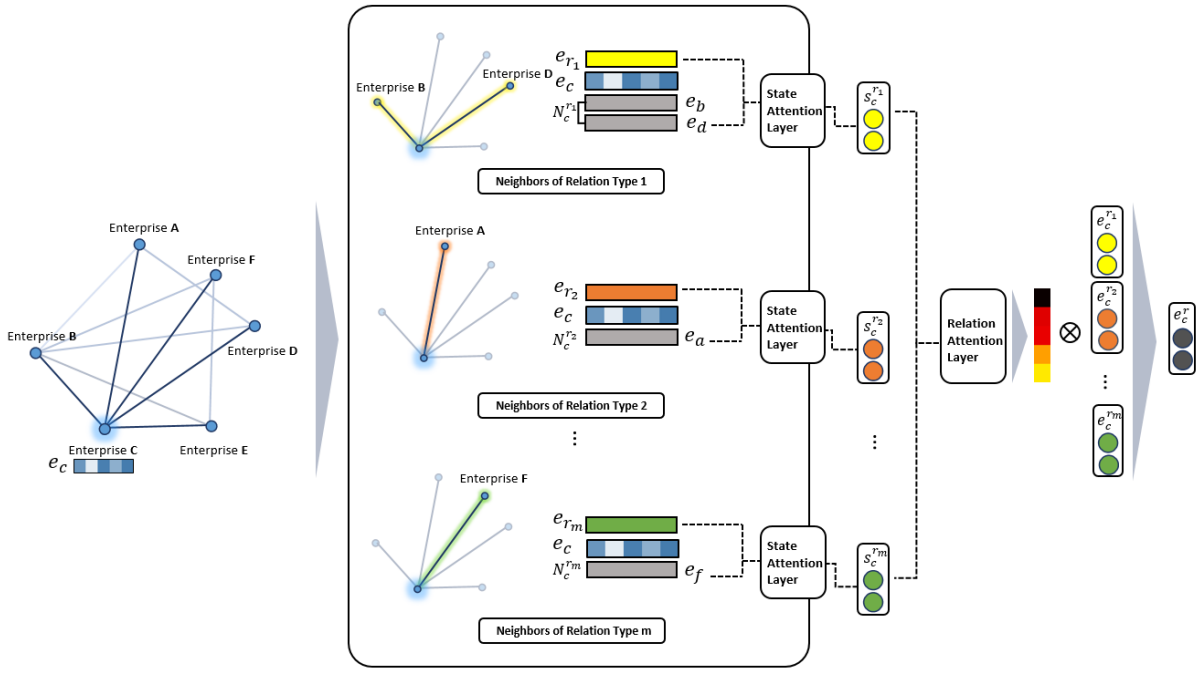

Figure 2: Hierarchical Attention Network for Stock Prediction

node representations of i and j into a vector where $j \in N_i^{r_m}$. If we denote the concatenated vector as $x_{ij}^{r_m} \in R^{(2f+d)}$, the state attention score is calculated as follows:

$$v_{ij} = x_{ij}^{r_m} W_s + b_s \tag{3.1}$$

$$\alpha_{ij}^{r_m} = \frac{\exp(v_{ij})}{\sum_k \exp(v_{ik})}, \quad k \in N_i^{r_m} \tag{3.2}$$

where $W_s \in R^{(2f+d)}$ and $b_s \in R$ are learnable parameters used to calculate the state attention scores. With attention weight calculated using Eq. (3.2), we combine all weighted node representations to calculate a vector representation of relation m for company i as Eq. (3.3).

$$s_i^{r_m} = \sum_{j \in N^{r_m}} \alpha_{ij}^{r_m} e_j \tag{3.3}$$

With above equation, all the representations of each type of relation are obtained. We selectively gathered information on specific relations from neighboring nodes. A representation can be considered as summarized information of a relation. Vector $s_i^{r_m}$ contains summarized information from relation m. For example, the representation of the industry relation summarizes the general state of the industry of our target company. Like human investors, our model should prioritize trading decisions based on summarized information of each relation. The second layer of HATS is designed to continuously assign importance weights to information using another attention mechanism.

We concatenate the summarized relation information vector $s_i^{r_m}$, representation of the current state of company e_i, and the relation type embedding vector e_{r_m} to use $\tilde{x}_i^{r_m} \in R^{(2f+d)}$ as input for the *relation attention layer*.

$$\tilde{v}_i^{r_m} = \tilde{x}_i^{r_m} W_r + b_r \tag{3.4}$$

$$\tilde{\alpha}_i^{r_m} = \frac{\exp(\tilde{v}_i^{r_m})}{\sum_k \exp(\tilde{v}_i^{r_k})}, \quad |N_i^{r_k}| \neq 0 \tag{3.5}$$

$W_r \in R^{(2d+f)}$ and $b_r \in R$ are learnable parameters, and weighted vectors of each relation type are added to form an aggregated relation representation as stated in Eq. (3.6) which is similar to Eq. (3.3).

$$e_i^r = \sum_k \tilde{\alpha}_i^{(r_k)} s_i^{r_k} \tag{3.6}$$

Finally, the representation of a node is added.

$$\bar{e}_i = e_i^r + e_i \tag{3.7}$$

In the next two subsections, we describe how updated node representations can be used in different tasks.

3.3. Individual Stock Prediction Layer

Like previous works such as [7] and [13], our model can be applied to the individual stock prediction task. We performed classification on the following three types of labels: *[up, neutral, down]*. A detailed description of the task setting is provided in Section 5. For the individual stock prediction task, we added only a simple linear transformation layer.

$$\hat{Y}_i = softmax(\bar{e}_i W_p^n + b_p^n) \tag{3.8}$$

where $W_p^n \in R^{d \times l}$, $b_p^n \in R^l$, and l is the number of movement classes. We trained models on all the corporate relational data using cross-entropy loss.

$$Loss_{node} = -\sum_{i \in Z_u} \sum_{c=1}^{l} Y_{ic} ln\hat{Y}_{ic} \tag{3.9}$$

where Y_{ic} is a ground truth movement class of company i and Z_u denotes all the companies in our dataset.

3.4. Graph Pooling for Index Prediction

A market index consists of multiple stocks chosen based on specific criteria. Let us denote a graph of a specific market index with n_k companies as $\mathcal{G}$, where a group of constituent companies of index k is V^k and its updated node representation is $\bar{X} \in R^{n_k \times f}$. To obtain the representation of the entire graph, the features of individual nodes need to be aggregated. Recently, numerous graph pooling methods for aggregation, such as [19] and [28], have been proposed. Stock market index data has its own historical price patterns which can be used as features. Therefore, we combine features obtained by graph pooling individual nodes and features directly extracted from historical price data.

We used mean pooling methods in our experiments to calculate graph representations as follows:

$$g_p^k = \frac{1}{n_k} \sum_{i \in V^k} \bar{e}_i \tag{3.9}$$

where $\bar{e}_i$ is the updated representation of company i. By denoting the target index's own feature vector extracted using the feature extraction module as g_e^k, the final representation of an entire graph can be obtained by combining the original representation of the graph and the representation obtained by graph pooling as follows.

$$g^k = g_p^k + g_e^k \tag{3.10}$$

We also concatenated the two representations; however, this did not have a significant impact on performance. As in the individual stock prediction task, we make predictions using simple linear transformation with $W_p^g \in R^{d \times l}$ and $b_p^g \in R^l$, and train models using cross-entropy loss as follows.

$$\hat{Y} = softmax(g^k W_p^g + b_p^g) \tag{3.11}$$

$$Loss_{graph} = -\sum_{c=1}^{l} Y_c ln\hat{Y}_c \tag{3.12}$$

Note that we use the most basic pooling method as this is the first work to apply graph pooling to the stock prediction task. There exists much room for improvement, which we we leave for future work.

4. Data

4.1. Price-related data

In this study, we focused on the U.S. stock market, one of the largest markets in the world based on market capitalization. We gathered corporate relational data from a public database which contains information on most of the S&P 500 companies. Among the S&P listed companies, there exist some companies without any type of relation with other companies in the database. After removing such companies, the remaining 431 companies were used as our target companies.

We sampled price data for our study from 2013/02/08 to 2019/06/17 (1174 trading days in total). Figure 3. shows the closing price of the S&P 500 index, which represents the overall market condition. As shown in Figure 3., although the index price has a tendency to go up, there are several crashes in our sample period. We splitted the entire dataset into several phases with varying degrees of volatility to evaluate performance. A more detailed description of the task settings is provided in Section 5.

As we described in Section 3, raw features of price-related data are fed to the feature extraction module. Many different types of raw features such as open price, close price, and volume can be used. In this study, following [14], we use historical price change rates as our input. Let P_i^t and P_i^{t-1} be the closing prices of a company i at time $tandt-1$, respectively. The price change rate at time t is calculated as $R_i^t = \frac{(P_i^t - P_i^{t-1})}{P_i^{t-1}}$. As our model can predict the movement of a stock price, the price change rate can also be predicted. Therefore, our model can predict the price change rate of the next day R_i^t, given the sequence of the historical price change rate of a company $[R_i^{t-l}, R_i^{t-l+1}, R_i^{t-l+2}, ..., R_i^{t-1}]$.

4.2. Corporate Relation data

The second type of data we used is corporate relational data. Following Feng et al., we collected corporate relational data from Wikidata [26]. Wikidata is a free collaborative knowledge base which contains relations between various types of entities (e.g. person, organization, country). Each entity has an index. If two entities have a relationship, it is considered as property. For example, the sentence "Steve jobs founded Apple" is expressed as a triplet *[Apple, Founded by, Steve jobs]*. In terms of graphs, each entity in Wikidata is a node and each property is an edge. Therefore, Wikidata can be understood as a heterogeneous graph with many different types of nodes and edges.

Here, companies are the only node type in which we are interested. However, there are a few types of edges between companies and their connections are very sparse. To address this problem, we utilize meta-path which is commonly used to deal with heterogeneous graphs [12]. If Steve Jobs was a board member of Google, we can make the triplet *[Steve jobs, Board Member, Google]*. Combined with the above mentioned relation *[Apple, Founded, Steve jobs]*, the two companies Apple and Google are now connected by the meta-path [Founded by, Board member] and share the node Steve Jobs. In this way, we found that there exist 75 types of relations including direct relations between companies and meta-paths. The entire lists of individual relations and meta-paths used in this study are provided in the appendix.

One of our main goals is to study the effect of using corporate relational data on stock market prediction performance. There are many ways to define a set of neighboring nodes. We used a meta-path with only 2 hops at maximum

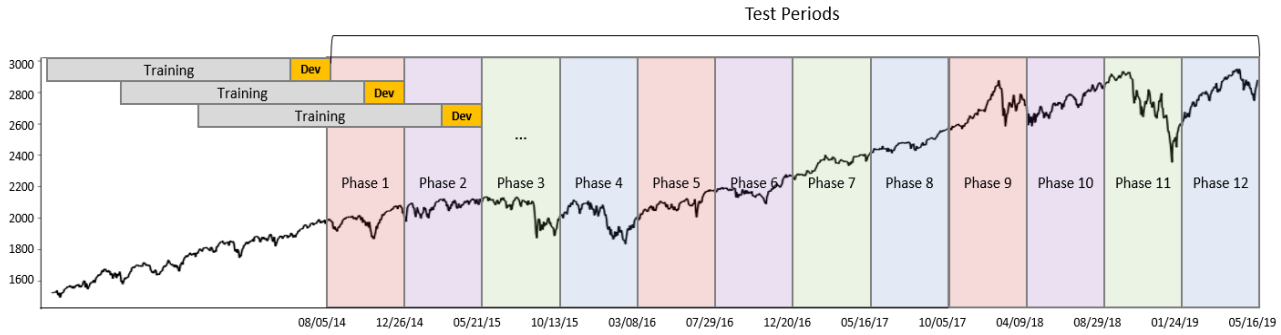

Figure 3: Dataset arrangement for the experiment. A 50-day period is used for evaluation right before the test period. A black line indicates an actual S&P 500 Index closing price.

to convert an originally heterogeneous graph into a homogeneous graph with only company nodes. Still, methods for building a corporate relational network from a large knowledge base can be much improved, which we leave for future work.

5. Experiments

5.1. Experiment design

General settings - As we mentioned in Section 3.3, we divided the training data into the following three classes based on the price change ratio: *[up, neutral, down]*. Specifically, two threshold values were used to divide the training data into the three classes and to assign labels to evaluation and test data. This labeling strategy labels small movements as neutral and uses only the significant movements as directional labels.

As shown in figure 3, although the price tends to go up eventually, there exist frequent stock market crashes. To ensure a strategy is profitable, it is important to keep your drawdown at a minimum level. Therefore, we should determine whether models are effective even in a highly volatile period. For this purpose, we divided our entire dataset into 8 smaller datasets that went through different phases, following [3]. Each phase consists of 250 days of training, 50 days of evaluation, and 100 days of testing.

For all the models in our experiments, we used a 50-day lookback period. As we used only the price change ratio as our input feature, the length of the input vector is 50. We used LSTM as a feature extraction module for individual stock prediction, and GRU for the index movement prediction task. We optimized all the models using the Adam optimizer, and tuned the hyper-parameters for each model within a certain range. Specifically, we used a learning rate between 1e-3 and 1e-5, weight decay between 1e-4 and 1e-5, and dropout between 0.1 and 0.9. Relu was used as our activation function. We measured the performance of the models on the evaluation set for each period. We performed early stopping based on F1 score. As the results of the stock prediction task tend to vary widely, all the experiments in this work were repeated five times. The results were averaged to obtain numbers in the table.

To measure the profitability of the models, we used a trading strategy based on movement prediction. Following Fischer and Krauss, we made a neutralized portfolio based on the prediction value obtained by models [14].Since there are three classes, the prediction vectors from all the models are three dimensional. Values of each dimension represent the predicted probability of each class. We selected 15 companies with the highest up class probability and the long position was taken. For the 15 companies with the highest down class probability, the short position was taken. This method is widely used when creating simple trading strategies for prediction models. We implemented our model in TensorFlow. Our source code and data are available at https://github.com/dmis-lab/hats.

Measurement We evaluated our models based on profitability and classification. In general, creating profitable trading strategies is the ultimate goal of stock movement prediction. Using the trading strategy mentioned above, we used two metrics to calculate profitability.

- **Return** We calculated the return of our portfolio as follows.

$$Return_i^t = \sum_{i \in F^{t-1}} \frac{(p_i^t - p_i^{t-1})}{p_i^{t-1}} * (-1)^{Action_i^{t-1}} \quad (5.1)$$

  where F^{t-1} denotes a set of companies included in the portfolio at time $t-1$, and p_i^t denotes the price of stock i at time t. $Action_i^t$ is a binary value between 0 and 1. $Action_i^t$ is 0 if the long position is taken at time t for stock i; otherwise, it is 1.

- **Sharpe Ratio** Sharpe ratio is used to measure the performance of an investment compared to its risk. The ratio calculates the earned return in excess of the risk-free rate per unit of volatility (risk) as follows:

$$Sharpe_a = \frac{E[R_a - R_f]}{std[R_a - R_f]} \quad (5.2)$$

where R_a denotes an asset return and R_f is the risk free rate. We used the 13-week Treasury bill to calculate the risk-free rates.

As price movement prediction is a special type of classification task, we used metrics widely used in classification tasks.

- **Accuracy and F1-Score** These two metrics are the most widely used for measuring classification performance.

  Each prediction can be labeled as True Positive(TP), True Negative(TN), False Positive(FP), or False Negative (FN). Accuracy and F-Score are calculated as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{5.4}$$

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{5.5}$$

After calculating the F1-score of each class, we averaged all the scores to obtain the macro F1-score.

Methods - We conducted experiments on the following baseline models. We describe the architecture of each model. We used different combinations of architectures and found that deeper structures generally suffer from overfitting.

Baselines without the relational modeling module.

- **MLP** Basic Multi Layer Perceptron model. We used an MLP consisting of 2 hidden layers with 16 and 8 hidden dimensions, respectively, and 1 prediction layer.
- **CNN** We used Convolutional Neural Network as it is known to be fast and as effective as RNN-based models in time series modeling. In our experiments, we used CNN with 4-layers and 2 convolutions and 2 pooling operations. The two convolutional layers with filter sizes of 32 and 8, respectively, and 5 kernels are used for each layer.
- **LSTM** Long Short-Term Memory is one of the most powerful deep learning models for time series forecasting. Many previous works have proven the effectiveness of LSTM. We used a LSTM network with 2 layers and a hidden size of 128. To train LSTM, we used the RMSProp optimizer which is known to be suitable for RNN-based models.

Baselines with the relational modeling module.

- **GCN**[7] Basic Graph Convolutional Neural network model. Following [7], we used a GCN model with two convolution layers and one prediction layer as stated in Eq. (5.6). All types of relations are used to create an adjacency matrix.
- **GCN-TOP20** We used the same GCN model but we used the edges from only the 20 best performed types of relations in the experiment, described in subsection 5.2, to create an adjacency matrix. In other words, we create adjacency matrix with manually selected relations for stock market prediction. By comparing GCN-Top20 with vanilla GCN, we analyzed the effect of using different relations on stock market prediction performance.
- **TGC**[13] Temporal Graph Convolution module for Relational modeling. Feng et al. proposed a general module for stock prediction. This module assigns values to the neighboring nodes of the target company based on the current state of the company and the relations between the nodes and the company. TGC aggregates all the information of a target company from its neighboring nodes while our HATS model summarizes information on different relation types.

Table 1
Results of using different relations from Phase 4.

Best 10	
Relation Type	F1
Industry-Legal form	0.3276
Industry-Product or material produced	0.3251
Parent organization-Owner of	0.325
Owned by-Subsidiary	0.3247
Parent organization	0.3247
Founded by-Founded by	0.3245
Follows	0.3244
Complies with-Complies with	0.3242
Owner of-Parent organization	0.3241
Subsidiary-Owner of	0.3241
...	
Worst 10	
Legal form-Instance of	0.311
Instance of-Legal form	0.3082
Location of formation-Country	0.307
Country-Location of formation	0.3053
Stock Exchange	0.2952
Country of origin-Country	0.2948
Country-Country of origin	0.2886
Country-Country of origin	0.2851
Instance of-Instance of	0.2748
Stock Exchange-Stock Exchange	0.2665

As mentioned in Section 3.1, for all models with a relational modeling module, LSTM is used as a feature extraction module in the individual stock prediction task. In the index movement prediction task, GRU is used as a feature extraction module. The simpler design of GRU makes it easier to train and helps obtain consistent results with deeper model architecture. Therefore, we used GRU as a feature extraction module for all models with the relational modeling module in the index movement prediction task.

5.2. Analysis of the effect of using relation data

We first conducted experiments to investigate the impact of using different types of relations for stock market prediction. The experiments were performed on the individual

Table 2
Classification accuracy scores on the individual stock prediction task

	MLP	CNN	LSTM	GCN	GCN20	TGC	HATS
F1							
Phase 1	0.2876	0.3111	0.3173	0.2874	0.3161	0.3110	**0.3314**
Phase 2	0.2862	0.3208	0.3228	0.3068	0.3339	0.3088	**0.3347**
Phase 3	0.2763	0.2938	0.3064	0.2692	**0.3113**	0.2237	0.3100
Phase 4	0.2810	0.3176	0.3030	0.2940	0.3240	0.2970	**0.3267**
Phase 5	0.2873	0.3354	0.3333	0.3116	0.3450	0.3329	**0.3496**
Phase 6	0.2855	0.3265	0.3229	0.2914	0.3140	0.2798	**0.3394**
Phase 7	0.2876	0.3111	0.3173	0.2874	0.3161	0.3110	**0.3314**
Phase 8	0.2862	0.3208	0.3228	0.3068	0.3339	0.3088	**0.3347**
Phase 9	0.2741	0.2390	0.2793	0.2980	0.3160	0.2851	**0.3219**
Phase 10	0.2529	0.2128	0.3134	0.3002	**0.3272**	0.2951	0.3243
Phase 11	0.2500	0.2270	0.2997	0.2714	0.3031	0.2577	**0.3091**
Phase 12	0.2678	0.2921	0.2968	0.2956	0.3299	0.3270	**0.3396**
Average	0.2769	0.2923	0.3113	0.2933	0.3225	0.2948	**0.3294**
Accuracy							
Phase 1	0.3455	0.3540	0.3597	0.3752	0.3700	**0.3811**	0.3725
Phase 2	0.3342	0.3626	0.3604	0.3735	0.3726	0.3701	**0.3752**
Phase 3	0.3547	0.3571	0.3771	**0.3860**	0.3834	0.3831	0.3859
Phase 4	0.3647	0.3855	0.3816	0.3992	0.3897	**0.4059**	0.3884
Phase 5	0.3208	0.3834	0.3684	0.4191	0.4164	**0.4239**	0.4176
Phase 6	0.3300	0.3803	0.3841	0.3627	0.3699	0.3716	**0.3869**
Phase 7	0.3553	0.4309	0.4252	**0.4510**	0.4488	0.4477	0.4502
Phase 8	0.3537	0.3901	0.3891	0.3993	0.4040	0.4022	**0.4049**
Phase 9	0.3711	0.3686	0.3511	**0.3955**	0.3836	0.3872	0.3923
Phase 10	0.3515	0.3484	0.3630	0.3552	0.3623	0.3667	**0.3674**
Phase 11	0.3813	0.3924	0.3608	0.3823	0.3794	0.3693	**0.3953**
Phase 12	0.3434	0.3615	0.3584	0.3530	0.3753	**0.3827**	0.3802
Average	0.3505	0.3762	0.3732	0.3877	0.3880	0.3910	**0.3931**

stock prediction task. To measure the effect of different relations, we used a basic GCN model that cannot distinguish the types of relations. Following [7], we used a GCN with two convolution layers and one prediction layer, which is defined as follows:

$$Y^{GCN} = softmax(\hat{A}ReLU(\hat{A}ReLU(\hat{A}X'W^{(0)})W^{(1)})W^{(2)}) \quad (5.6)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Here, $\tilde{A} = A + I$ is an adjacency matrix with added self-connections, and $\tilde{D}$ is a degree matrix of $\tilde{A}$. Therefore, changing the relation type changes the adjacency matrix that is fed into the GCN. We list the 10 best and 10 worst relations and their F1 scores on the test set of Phase 4 in Table 1 Table 1.

Our key findings are as follows.

- **Using relation data does not always yield good results in stock market prediction.** In our worst cases, using relation data significantly decreased performance. On the other hand, some relation information proved to be helpful in prediction. The best performance is 6% higher than the worst performance.
- **Densely connected networks usually have noise.** We confirmed this while analyzing the characteristics of the best and worst relations. Although the number of relations does not affect performance the most, less semantically meaningful relations such as country and stock exchange have very dense networks. Intuitively, densely connected networks carry a considerable amount of noise, which adds irrelevant information to the representations of target nodes.
- **Manually finding optimal relations is laborious.** Although semantically meaningful relations generally help improve performance, selecting such relations requires much work and expertise.

Based on the above findings, we can conclude that relational information should be selectively chosen when using it for stock market prediction. Furthermore, the framework should be designed to automatically select useful information to minimize the need for manual feature engineering. We conducted experiments on two different tasks to verify the effectiveness of different relational modeling approaches.

5.3. Individual Stock Prediction

Classification Accuracy The classification accuracy results of the experiments on individual stock market prediction are summarized in Table 2. Among the baselines without a relational modeling module, LSTM generally performs

Table 3
Profitability results on the individual stock prediction task

Average Daily Return (%)							
	MLP	CNN	LSTM	GCN	GCN20	TGC	HATS
Phase 1	0.0672	-0.0506	0.0904	-0.0264	-0.0103	-0.0517	**0.1231**
Phase 2	-0.0195	0.0929	0.1005	0.1057	0.2435	0.1247	**0.1759**
Phase 3	0.0029	-0.0623	0.0454	-0.0189	0.0246	-0.0100	**0.0703**
Phase 4	0.0945	-0.0578	0.1429	0.0028	0.0385	0.0113	**0.1779**
Phase 5	-0.0623	-0.0673	-0.0159	-0.0427	**0.0415**	-0.0202	0.0183
Phase 6	-0.0002	0.0140	0.0400	0.0748	**0.0828**	0.0581	0.0726
Phase 7	-0.0081	0.0272	0.0246	0.0201	-0.0389	-0.0143	**0.0860**
Phase 8	0.0319	-0.0122	0.0742	0.0837	**0.2356**	0.2175	0.0662
Phase 9	0.0143	-0.0311	0.0234	0.0375	-0.0437	-0.0211	**0.0394**
Phase 10	0.0040	-0.0239	-0.0209	0.0126	0.0164	-0.0523	**0.0612**
Phase 11	0.0395	-0.0106	-0.0085	0.0597	0.0685	0.1321	**0.1890**
Phase 12	0.0068	0.0051	0.0222	0.0334	-0.0241	0.0775	**0.0732**
Average	0.0142	-0.0147	0.0432	0.0285	0.0529	0.0376	**0.0961**
Sharpe Ratio (Annualized)							
Phase 1	2.4410	-1.4459	2.3553	-0.2802	-0.1013	-0.5029	**2.4796**
Phase 2	-1.0063	3.2835	4.0651	2.4700	**4.9007**	3.0525	4.3903
Phase 3	0.1070	-1.7872	1.0642	-0.2477	0.2994	-0.1796	**1.2503**
Phase 4	2.1602	-0.6064	2.2014	0.0289	0.3173	0.1085	**2.3961**
Phase 5	-1.6039	-1.7851	-0.4455	-0.7090	**0.6222**	-0.4131	0.4087
Phase 6	-0.0095	0.3565	1.1960	1.8324	2.0390	**2.9435**	1.6945
Phase 7	-0.4010	0.9306	0.8354	0.4078	-0.9107	-0.6618	**2.0334**
Phase 8	1.0398	-0.3917	2.1975	1.3746	3.1870	**4.1305**	1.4830
Phase 9	0.4915	-1.9624	0.4905	0.7758	-0.6896	-0.3619	**0.8060**
Phase 10	0.6667	-1.8774	-0.5671	0.3263	1.3576	-1.2023	**1.4382**
Phase 11	0.8059	-0.7052	-0.1983	2.5786	1.3053	3.3379	**3.6146**
Phase 12	0.1684	0.2055	0.6334	0.8066	-0.8201	1.7799	**1.9014**
Average	0.4050	-0.4821	1.1523	0.7803	0.9589	1.0026	**1.9914**

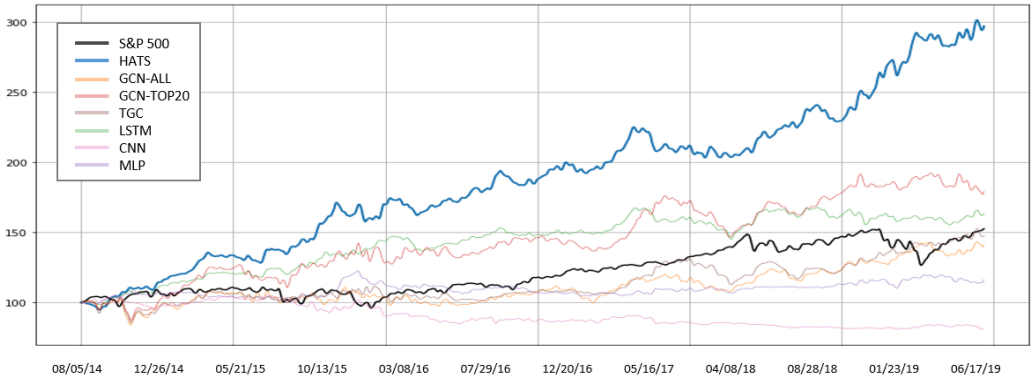

Figure 4: Comparison of different prediction models and their changes in asset value. The asset value is assumed to start at 100.

the best. Therefore, we compare the results of the models with a relational modeling module and the result of LSTM. In terms of accuracy, all models with a relational modeling module performed better than LSTM. However, not all relational models outperformed LSTM in terms of F1 score. As shown in Table 2, only GCN-Top20 and HATS achieved higher F1 scores. What we observed during our experiment is that GCN and TGC tend to make biased predictions on a specific class. By making biased predictions, the GCN and TGC models obtained higher accuracy but lower F1 scores.

Table 4
Classification accuracy scores on the market index prediction task

	S5CONS	S5FINL	S5INFT	S5ENRS	S5UTIL	Average
			F1			
MLP	0.2986	0.3002	0.2867	0.2785	0.2928	0.2913
CNN	0.3013	**0.3157**	0.3036	0.3011	0.3025	0.3049
LSTM	0.3405	0.2859	0.3454	0.3109	0.2942	0.3154
GCN	0.3410	0.3040	0.3423	0.2848	0.3111	0.3166
TGC	0.3322	0.3051	0.3391	0.2736	0.2911	0.3082
HATS	**0.3758**	0.3148	**0.3518**	**0.3267**	**0.3256**	**0.3389**
			Accuracy			
MLP	0.3290	0.3463	0.3318	0.3210	0.3282	0.3313
CNN	0.3429	0.3392	0.3434	0.4126	0.3235	0.3537
LSTM	0.3625	0.3506	0.3808	0.4550	0.3100	0.3718
GCN	0.3722	0.3637	0.3591	0.4531	0.3373	0.3771
TGC	0.4021	**0.3699**	0.3754	0.4468	0.3250	0.3819
HATS	**0.4095**	0.3662	**0.3834**	**0.4620**	**0.3531**	**0.3948**

Selectively aggregating information from different relations can help improve F1 scores. Although TGC performed better than vanilla GCN, TGC was outperformed by GCN-Top20 which was trained on manually selected relational data. In contrast, our proposed model HATS generally outperformed all the baselines in terms of both F1 score and accuracy. These results are consistent with the profitability test results which are provided in the following subsection.

Profitability test The individual stock prediction results on the profitability test are summarized in Table 3. We calculated the daily returns of the neutralized portfolio made using the strategy discussed in Section 5.1, and averaged them for each period. On average, GCN-Top20 and HATS obtained the highest average daily return. As mentioned above, GCN-Top20 and HATS outperformed GCN and TGC in terms of F1 score. TGC performed better than vanilla GCN but worse than LSTM. Surprisingly, the Sharpe ratio of GCN-Top20 was lower than that of LSTM and TGC. Without even calculating the Sharpe ratio, we can see in Table 3 that the expected return results of GCN-Top20 have large variance, which may be attributed to GCN-Top20 using relational data statically. Although relations used for GCN-Top20 are manually selected and expected to improve stock prediction, fixed relations may be useful only in a specific market condition. As GCN cannot assign importance to neighboring nodes based on the market condition and current state of a given node, its results vary widely. By selecting useful information based on the market situation, our HATS model obtains good performance in terms of expected return and Sharpe ratio.

5.4. Market Index Prediction

As mentioned in section 4, we gathered price and relational data for 431 companies listed in the S&P 500. There exist 9 different market indices each representing an industrial sector. We removed four indices with less than 20 constituent companies and have five remaining market indices. The five market indices are as follows: S5CONS (S&P 500 Consumer Staples Index), S5FINL (S&P 500 Financials Index), S5INFT (S&P 500 Information Technology Index), S5ENRS (S&P 500 Energy Index), S5UTIL (S&P 500 Utilities Index). As the graph of constituent companies is already sparse, we do not use GCN-Top20 as a baseline. The results are summarized in table 5.

Due to the space constraints, we provide only the averaged results for each index in table 5. Furthermore, we did not measure the profitability performance of a neutralized portfolio on the market index prediction task. It is not reasonable to make neutralized portfolio With only five assets as our portfolio selection universe.

On average, models with a relational modeling module outperformed LSTM on the market index prediction task. However, HATS is the only model that achieved significantly better performance than LSTM in terms of F1 score and accuracy. GCN performed slightly better than LSTM and TGC performed worse than LSTM in terms of F1 score. As we used the same pooling operation for all the models, the differences in performance can be mainly attributed to their relational modeling module. This again proves that HATS is effective in learning node representations for a given task. On the market index prediction task, HATS outperforms all the baselines in terms of F1 score and accuracy on average.

Unexpectedly, the other baselines with the relational modeling module did not perform significantly better than LSTM. The baselines cannot easily select information from different relation types and they use a naive structure to obtain graph representations. Many graph pooling methods such as [19] and [28] have already been proposed for learning graph representations, and proven to be more effective in many different tasks. We expect that more advanced pooling methods will further improve performance on the market index prediction task.

5.5. Case Study

Relation attention scores In this section, we conduct two case studies to further analyze the decision-making mecha-

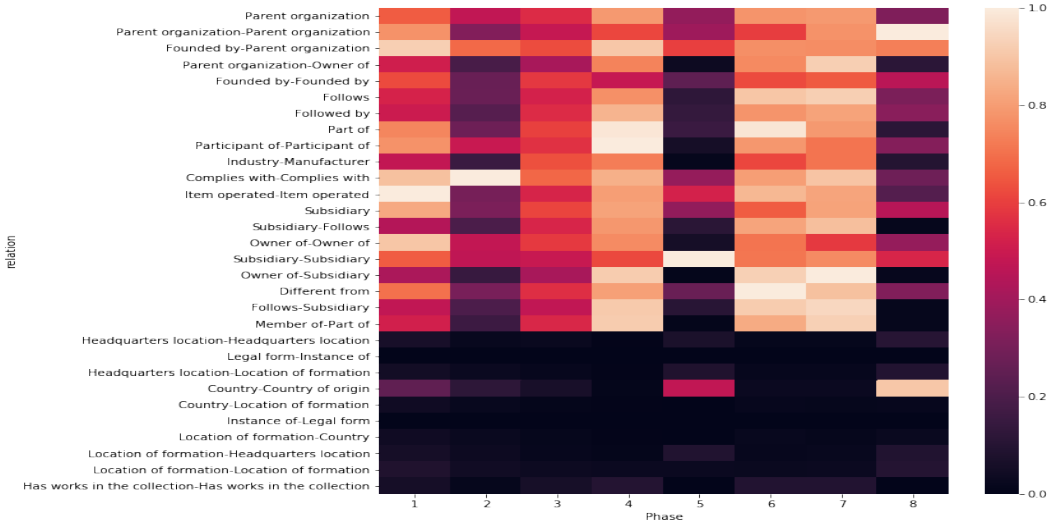

Figure 5: Visualization of attention scores of different relations. 20 relations with highest attention scores on average and 10 relations with the lowest scores on average.

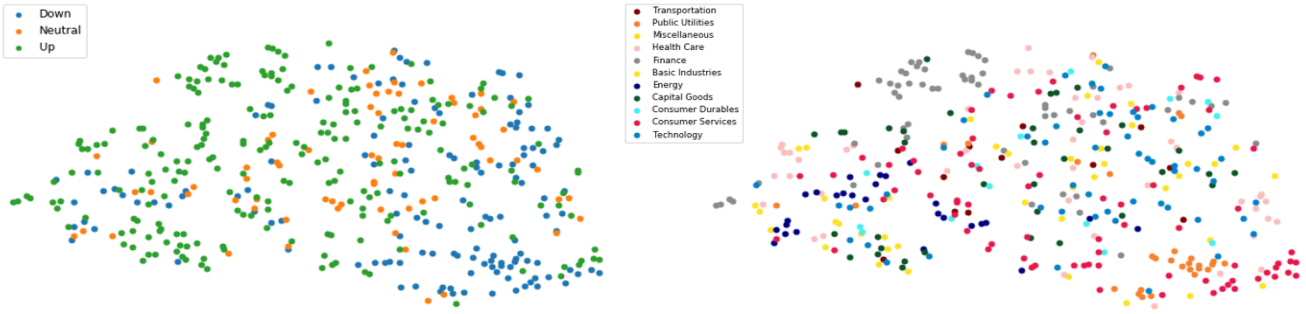

Figure 6: Visualization of node representations using T-SNE.

nism of HATS. As previously mentioned, HATS is designed to gather information from only useful relations. For our first case study, we calculated the attention score of each relation. By analyzing the relation types with the highest and lowest attention scores, we can understand what types of relations are considered to be important. Fig. 5 shows a visualization of the attention scores of all the relations. We calculated the average attention scores on the test sets from all the phases and selected 20 relations with the highest attention scores and 10 relations with the lowest attention scores. The visualization shown in Fig. 5 is based on the average scores calculated in each test phase. As shown in Fig. 5, the relations with the highest attention scores are mostly dominant-subordinate relationships such as parent organization-subsidiary relationships. Some relations with the highest scores represent industrial dependencies. On the other hand, most of the relations with the lowest attention scores are geographical features.

Node representation In studies on graph neural network methods, researchers are interested in representations obtained by GNN. We present the visualization node representation obtained by HATS in Fig. 6. We obtain the representations of all companies on a specific day and use the T-SNE algorithm to map each representation to a two-dimensional space. In Figure 6(a), the movement of a stock on a given day is denoted by any one of the three colors which represent the up/neutral/down labels we used in our experiment. In Figure 6(b), industries of companies are denoted by different colors. We can find a rough line that separates companies with up labels from companies with down labels in Figure 6(a). It is also interesting that representations of the neutral movement are widely spread. In Figure 6(b), there exists a group of clusters in the same industry. We can find these clusters in any time phase. Although the prices of two stocks in the same industry do not always move in the same direction, the clusters in Figure 6(b) show that HATS learned meaningful representations.

6. Conclusion

In this work, we proposed our model HATS which uses relational data in stock market prediction. HATS is designed to selectively aggregate information on different relation types to learn useful node representations. HATS performed the graph related tasks of predicting individual stock prices and predicting market index movement. The experimental results prove the importance of using proper relational data and show that prediction performance can change dramatically depending on the relation type. The results also show that HATS which automatically selects information to use outperformed all the existing models.

There exist many possibilities for future research. First, finding a more effective way to construct a corporate network is an important research objective that could be the focus of future studies. In this study, we define the neighborhood of a company as a cluster of companies connected by direct edges or meta-paths with at most 2 hops. However, the way in which we define it could be improved. Furthermore, we used a single database (WikiData) to create a company network. In future work, we could use another source of data and we could even create knowledge graphs from unstructured text of various sources. Applying more advanced pooling methods to obtain graph representations could improve the overall performance of GNN methods on the market index prediction task.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF-2017R1A2A1A17069645,
NRF-2017M3C4A7065887)

References

[1] Adebiyi, A.A., Adewumi, A.O., Ayo, C.K., 2014. Comparison of arima and artificial neural networks models for stock price prediction. Journal of Applied Mathematics 2014.

[2] Agrawal, J., Chourasia, V., Mittra, A., 2013. State-of-the-art in stock prediction techniques. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 2, 1360–1366.

[3] Bao, W., Yue, J., Rao, Y., 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one 12, e0180944.

[4] Bollen, J., Mao, H., Zeng, X., 2011. Twitter mood predicts the stock market. Journal of computational science 2, 1–8.

[5] Bollerslev, T., Marrone, J., Xu, L., Zhou, H., 2014. Stock return predictability and variance risk premia: statistical inference and international evidence. Journal of Financial and Quantitative Analysis 49, 633–661.

[6] Chen, J., Ma, T., Xiao, C., 2018a. FastGCN: Fast learning with graph convolutional networks via importance sampling, in: International Conference on Learning Representations.

[7] Chen, Y., Wei, Z., Huang, X., 2018b. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM. pp. 1655–1658.

[8] Dechow, P.M., Hutton, A.P., Meulbroek, L., Sloan, R.G., 2001. Short-sellers, fundamental analysis, and stock returns. Journal of Financial Economics 61, 77–106.

[9] Dempster, M.A., Payne, T.W., Romahi, Y., Thompson, G.W., 2001. Computational learning techniques for intraday fx trading using popular technical indicators. IEEE Transactions on neural networks 12, 744–754.

[10] Ding, X., Zhang, Y., Liu, T., Duan, J., 2015. Deep learning for event-driven stock prediction, in: Twenty-Fourth International Joint Conference on Artificial Intelligence.

[11] Ding, X., Zhang, Y., Liu, T., Duan, J., 2016. Knowledge-driven event embedding for stock prediction, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 2133–2142.

[12] Dong, Y., Chawla, N.V., Swami, A., 2017. metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, ACM. pp. 135–144.

[13] Feng, F., He, X., Wang, X., Luo, C., Liu, Y., Chua, T.S., 2019. Temporal relational ranking for stock prediction. ACM Transactions on Information Systems (TOIS) 37, 27.

[14] Fischer, T., Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research 270, 654–669.

[15] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org. pp. 1263–1272.

[16] Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, pp. 1024–1034.

[17] Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 .

[18] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105.

[19] Lee, J., Lee, I., Kang, J., 2019. Self-attention graph pooling. arXiv preprint arXiv:1904.08082 .

[20] Li, X., Xie, H., Chen, L., Wang, J., Deng, X., 2014. News impact on stock price return via sentiment analysis. Knowledge-Based Systems 69, 14–23.

[21] Malkiel, B.G., 2003. The efficient market hypothesis and its critics. Journal of economic perspectives 17, 59–82.

[22] Patel, J., Shah, S., Thakkar, P., Kotecha, K., 2015. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Systems with Applications 42, 259–268.

[23] Phan, D.H.B., Sharma, S.S., Narayan, P.K., 2015. Stock return forecasting: some new evidence. International Review of Financial Analysis 40, 38–51.

[24] Rather, A.M., Agarwal, A., Sastry, V., 2015. Recurrent neural network and a hybrid model for prediction of stock returns. Expert Systems with Applications 42, 3234–3241.

[25] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 .

[26] Vrandečić, D., Krötzsch, M., 2014. Wikidata: a free collaborative knowledge base .

[27] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J., 2018a. 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.

[28] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J., 2018b. Hierarchical graph representation learning with differentiable pooling, in: Advances in Neural Information Processing Systems, pp. 4800–4810.

[29] Zhang, M., Chen, Y., 2018. Link prediction based on graph neural networks, in: Advances in Neural Information Processing Systems, pp. 5165–5175.

Raehyun Kim received a B.S. degree in business from Korea University, Seoul, Republic of Korea, in 2018, where he is currently pursuing an M.S. degree in computer science. His current research interests include financial market prediction and recommendation systems.

Chan Ho So received a B.S. degree in computer science from Korea University, Seoul, Republic of Korea, in 2018, where he is currently pursuing an M.S. degree in computer science. His current research interests include bioinformatics and named entity recognition.

Minbyul Jeong received a B.S. degree in computer science from Korea University, Seoul, Republic of Korea, in 2019, where he is currently pursuing an M.S. degree in computer science. His current research interests include natural language processing and bioinformatics.

Sanghoon Lee received a B.S. degree in computer science from Korea University, Seoul, Republic of Korea, in 2019, where he is currently pursuing an M.S. degree in computer science. His current research interests include bioinformatics.

Jinkyu Kim received a B.S. degree in business from Korea University, Seoul, Republic of Korea, in 2019. His current research interests include financial market prediction and bioinformatics.

Jaewoo Kang received a B.S. degree in computer science from Korea University, Seoul, Republic of Korea, in 1994, an M.S. degree in computer science from the University of Colorado Boulder, CO, USA, in 1996, and a Ph.D. degree in computer science from the University of Wisconsin-Madison, WI, USA, in 2003.

From 1996 to 1997, he was a Technical Staff Member at AT&T Labs Research, Florham Park, NJ, USA. From 1997 to 1998, he was a Technical Staff Member at Savera Systems Inc., Murray Hill, NJ, USA. From 2000 to 2001, he was the CTO and Founder of WISEngine Inc., Santa Clara, CA, USA, and Seoul, Republic of Korea. From 2003 to 2006, he was an Assistant Professor in the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. Since 2006, he has been a Professor in the Department of Computer Science, Korea University, Seoul, Republic of Korea. He also serves as the Department Head for the Interdisciplinary Graduate Program in Bioinformatics at Korea University, Seoul, Republic of Korea. He is jointly appointed with the Department of Medicine, School of Medicine, Korea University, Seoul, Republic of Korea.

Table 5
List of relation types and their definitions used to make meta-paths. Combinations of relations below are used in our study.

Code	Relation Name	Description
P17	Country	sovereign state of this item; don't use on humans
P31	Instance of	that class of which this subject is a particular example and member (subject typically an individual member with a proper name label)
P112	Founded by	founder or co-founder of this organization, religion or place
P121	Item operated	equipment, installation or service operated by the subject
P127	Owned by	owner of the subject
P131	Located in the administrative territorial entity	the item is located on the territory of the following administrative entity.
P138	Named after	entity or event that inspired the subject's name, or namesake (in at least one language)
P155	Follows	immediately prior item in a series of which the subject is a part [if the subject has replaced the preceding item, e.g. political offices, use "replaces"]
P156	followed by	the immediately following item in some series of which the subject is part. Use P1366 if the item is replaced e.g. political offices, states
P159	Headquarters location	specific location where an organization's headquarters is or has been situated.
P166	Award received	award or recognition received by a person, organisation or creative work
P169	Chief executive officer	highest-ranking corporate officer appointed as the CEO within an organization
P176	Manufacturer	manufacturer or producer of this product
P355	Subsidiary	subsidiary of a company or organization, opposite of parent organization
P361	Part of	object of which the subject is a part
P400	Platform	platform for which a work was developed or released, or the specific platform version of a software product
P414	Stock Exchange	exchange on which this company is traded
P452	Industry	industry of company or organization
P463	Member of	organization or club to which the subject belongs
P488	Chairperson	presiding member of an organization, group or body
P495	Country of origin	country of origin of this item (creative work, food, phrase, product, etc.)
P625	Coordinate location	geocoordinates of the subject.
P740	Location of formation	location where a group or organization was formed
P749	Parent organization	parent organization of an organisation, opposite of subsidiaries (P355)
P793	significant event	significant or notable events associated with the subject
P1056	Product or material produced	material or product produced by a government agency, business, industry, facility, or process
P1343	Described by source	dictionary, encyclopaedia, etc. where this item is described
P1344	Participant of	event a person or an organization was/is a participant in,
P1454	Legal form	legal form of an organization
P1552	Has quality	the entity has an inherent or distinguishing non-material characteristic
P1830	Owner of	entities owned by the subject
P1889	Different from	item that is different from another item, with which it is often confused
P3320	Board member	member(s) of the board for the organization
P5009	Complies with	the product or work complies with a certain norm or passes a test
P6379	Has works in the collection	collection that have works of this artist

A. Appendix

Table 6
List of the relations used in our study. Both direct edges and meta-paths are included in the list.

Relation Index	Relation Combination (Code)	Relation Index	Relation Combination (Code)
1	P1454-P1454	37	P452-P176
2	P159-P159	38	P6379-P6379
3	P1454-P31	39	P155-P155
4	P159-P740	40	P495-P17
5	P17-P495	41	P749-P127
6	P17-P740	42	P749-P749
7	P414-P361	43	P4950-P495
8	P414	44	P495-P740
9	P452-P452	45	P355
10	P361-P361	46	P355-P155
11	P127-P749	47	P1830-P1830
12	P1344-P1344	48	P112-P749
13	P127-P127	49	P1056-P452
14	P740-P159	50	P1454-P452
15	P740-P740	51	P355-P127
16	P112-P112	52	P176-P452
17	P155	53	P159-P131
18	P1056-P1056	54	P1830
19	P1056-P31	55	P1830-P127
20	P127-P355	56	P1830-P749
21	P127-P1830	57	P355-P355
22	P361-P414	58	P131-P159
23	P127	59	P5009-P5009
24	P156	60	P31-P1056
25	P31-P1454	61	P1830-P355
26	P452-P31	62	P740-P17
27	P452-P1056	63	P740-P495
28	P463-P463	64	P1889
29	P625-P625	65	P749-P112
30	P361	66	P361-P463
31	P159-P17	67	P155-P355
32	P17-P159	68	P463-P361
33	P793-P793	69	P355-P1830
34	P166-P166	70	P121-P121
35	P31-P452	71	P749-P1830
36	P452-P1454	72	P749