AR-Stock: Deep Augmented Relational Stock Prediction

Tianxin Wei[1], Yuning You[2], Tianlong Chen[3]
[1]University of Science and Technology of China, [2]Texas A&M University, [3]University of Texas at Austin
ttxxwei@gmail.com, yuning.you@tamu.edu, tianlong.chen@utexas.edu

Abstract

Stock prediction aims to assess future price trends and assist investment decisions. With the recent success of graph convolutional networks (GCNs) in modeling relational data, they have shown promise for stock prediction too. However, vanilla GCNs lacking the ability to capture long-range dependencies in graphs and have not fully utilized the structured knowledge with data available. In this paper, we propose a novel framework of *Deep Augmented Relational Stock Prediction* (**AR-Stock**). We first detect the long-range links using pre-trained knowledge graph embeddings, leading to a new geometrically augmented edge type into the provided stock market graph. We then construct the GCN model on this augmented graph, that predicts each company's stock prices by leveraging its related corporations; specifically, to train the GCN better over this complex graph, we introduce two novel self-supervised regularizers (graph partition and graph completion) to inform the model with the global and local topology features. Unifying the above ingredients, AR-Stock has the unique strength in capturing long-term and hidden graph node dependencies better. Experiments on two popular stock market datasets, NASDAQ and NYSE, demonstrate the prediction superiority of AR-Stock. Particularly, in terms of the investment return ratio, AR-Stock improves **65.77%** in NASDAQ, and **30.48%** in NYSE, respectively over state-of-the-art models.

Introduction

Machine learning (ML) for financial market predictions, e.g., stock prices, has recently witnessed a surge of interest. Despite this cross-field enthusiasm, naïvely modeling a stock price as an isolated time series is apparently sub-optimal, as it ignored the rich inter-market and inter-company relations among stocks. For example, to predict a certain company's stock price, professional investors will thoroughly draw evidence ranging from its supplier/customer information (a.k.a, "short-range" or local dependency); to the entire industry, market, policy ad other macroeconomic factors (a.k.a, "long-range" or global dependency). Therefore, making better market predictions critically hinges on mapping this connectivity of companies and events and then utilizing this map. Those rich relations make the stock market a complex *knowledge graph*.

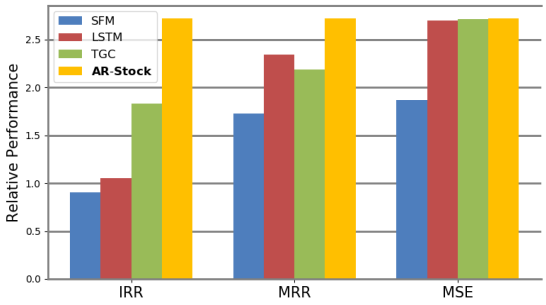

Figure 1: Results of three metrics (marked by three discrete coordinates on the y-axis) on the NASDAQ dataset with different methods. For each metric, we normalize all numbers using the top performer's. Then we normalize metrics with an exponential function.

Real data often come naturally in the form of graphs, e.g., social networks and gene expression networks. Graph convolutional networks (GCNs) (Kipf and Welling 2016; Gilmer et al. 2017) have gained increasing popularity in addressing those graph-based applications. A GCN stacks multiple graph convolution operators, and learn node representations by recursively aggregating information from their layer-wise neighbors. Typical GCN tasks include graph classification (Ying et al. 2018), node classification (Kipf and Welling 2016), link prediction (Zhang and Chen 2018), and collaborative filtering (Berg, Kipf, and Welling 2018). Lately, GCNs have achieved preliminary success in stock market prediction, by considering it as a node-level regression task (Feng et al. 2019; Chen, Wei, and Huang 2018).

Despite the demonstrated promise, GCNs suffer from several inherent limitations, constituting a major gap in accurately characterizing the huge, complex, and dynamic stock market. Specifically, GCNs notoriously lacks the ability to capture long-range dependencies over large graphs; even for local neighborhoods, the neighbor aggregation process of GCN may fail to well preserve the node's structural information well (Kondor et al. 2018). Such pose significant challenges to modeling the aforementioned "long-range" and "short-range" market factors. Moreover, the few-label problem is common in graph-structured data, which is difficult to be addressed with GCN solely. In view of that, we introduce two key innovations into GCN-based stock prediction:

- We firstly exploit a *geometric augmentation* approach to discover hidden long-range dependencies between stocks and to enrich the sparse market graph. Specifically, we utilize the knowledge graph to generate entity embeddings for stocks that encode various relation patterns, leading to new geometric edges being hallucinated to the original graph.
- We secondly leverage self-supervised learning to facilitate large GCN training, and to enforce global and local graph structure awareness respectively. Self-supervised learning was popular in convolutional neural networks (CNNs) (Chen et al. 2020), but not investigated in GCNs until lately (Sun, Lin, and Zhu 2020). We present two new dedicated self-supervision techniques for GCNs: graph partition and graph completion, and show them improving our GCN generalization.

We name our framework as *Deep Augmented Relational Stock Prediction* (**AR-Stock**). It is then applied to the tasks of stock price prediction (i.e., predictive regression of future stock prices) and stock returns ranking (e.g., identification of the highest-return stock). We perform extensive experiments on stocks listed in two datasets: NASDAQ and NYSE. In all metrics considered, AR-Stock substantially outperforms state-of-the-arts. For instance, in terms of the investment return ratio, AR-Stock improves **65.77%** in NASDAQ, and **30.48%** in NYSE, respectively over existing alternatives.

RELATED WORK

Stock Price Prediction

Stock price prediction is considered one of the most important problems in finance domain (Bao, Yue, and Rao 2017; Zhang, Aggarwal, and Qi 2017; Hu et al. 2018; Zhao et al. 2017; Nguyen and Shirai 2015). Traditionally, only the past information from the same source as the predicted target is taken into account. Bao, Yue, and Rao (2017) predict the next-day price of stocks based on the historical values by regarding the historical stock prices as a time serie and decompose it with Wavelet transform. Stacked Autoencoder (SAE) is designed to filter out the noises in the frequency domain, and the final prediction is made after a LSTM network. Zhang, Aggarwal, and Qi (2017) proposed State Frequency Memory (SFM) which combines LSTM network with Fourier transform to capture the multi-frequency trading patterns from past market data to make long and short term predictions over time. Instead of just using the past sequential information, Hu et al. (2018) taking financial news as input as well by devising Hybrid Attention Networks (HAN) to attentively aggregate both the stock related news information and stock sequential behaviors into a unified representation.

Recent trends demonstrate to incorporate different sources of information, exploiting the dependency relationship in the market to further assist prediction. Especially, the dependency information is naturally modeled as the adjacency matrix of a graph, facilitating the utilization of graph convolutional networks (GCNs) (Matsunaga, Suzumura, and Takahashi 2019; Feng et al. 2019). The feasibility of vanilla GCNs is verified in (Matsunaga, Suzumura, and Takahashi 2019) for stock prediction across different markets and longer time horizons using rolling window analysis. Feng et al. (Feng et al. 2019) propose temporal GCN to incorporate the relational information into stock modeling and use temporal attention to encode the relation-strength function to make GCN time-sensitive. They are also the first to formulate stock prediction as a ranking task, targeting at directly predicting which stock is more favored in terms of return ratio.

Graph Convolutional Networks

Graph convolutional networks (GCNs) have emerged in recent years for addressing applications of graph-structured data, such as social network processing and molecule representation learning (You, Ying, and Leskovec 2019; Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017; Ying et al. 2018; Veličković et al. 2017; Schlichtkrull et al. 2018; You et al. 2020b; Liu, Gao, and Ji 2020). Under the framework of message passing mechanism (Gilmer et al. 2017), GCN adopts mean pooling to aggregate the neighborhood information and updates representations layer-by-layer recursively (Kipf and Welling 2016), with several powerful variants proposed such as GraphSAGE (Hamilton, Ying, and Leskovec 2017) and GAT (Veličković et al. 2017). To improve GCN for capturing long-range dependencies in disassortative graphs, a geometric aggregation scheme is proposed in (Pei et al. 2020) to enhance the graph convolution, benefiting from a continuous space underlying the graph.

Self-Supervised Learning

Self-supervision is a promising technique for learning more transferable, generalized and robust representations from unlabeled data (Zhai et al. 2019; Kolesnikov, Zhai, and Beyer 2019; DeTone, Malisiewicz, and Rabinovich 2018; Goyal et al. 2019; Kocabas, Karagoz, and Akbas 2019; Liu et al. 2019) and is only investigated in (Sun, Lin, and Zhu 2020; You et al. 2020c,a; Jin et al. 2020; Zhu, Du, and Yan 2020) for GCN recently. It assists model training with the complementary pretext task through pre-training or multi-task learning, which is carefully designed in order to enforce learning down stream-related semantics features. A number of pretext tasks have been proposed for CNNs, including rotation (Gidaris, Singh, and Komodakis 2018), exemplar (Dosovitskiy et al. 2014), jigsaw (Noroozi and Favaro 2016) and relative patch location prediction (Doersch, Gupta, and Efros 2015). In GCN, using node clustering as the self-supervised task is proposed in (Sun, Lin, and Zhu 2020), where vertex attributes are utilized to group vertices with similar embeddings, achieving state-of-the-art performance in semi-supervised node classification. We notice that in graph-structured data, rich structure information is under-explored yet, leading to our innovations.

Deep Augmented Relational Stock Prediction

Problem Setting

Let the matrix $X^t = [x^{t-\mathrm{S}+1}, ..., x^t]^T \in \mathbb{R}^{\mathrm{S}\times\mathrm{D}}$ denote the stock sequential input feature, where S is the length of input sequence and D is the feature dimension, N is the

number of stocks and $X^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{S}\times\mathrm{D}}$ represents the sequential features of all the N stocks at time step t. The problem of stock prediction (i.e., price movement classification and price regression) is to learn a prediction function $y^{t+1} = f(X^t)$, mapping a stock's historical feature to the outcomes at time-step t. Specifically, we simultaneously predict the stock prices and rank their relative order, to select higher-ranked (more profitable) stocks, through incorporating each company's own input sequence with relationships among all N companies. Assuming there are K types of relationships between companies, we have a multi-hop binary vector $a_{ij} \in \mathbb{R}^{\mathrm{K}}$ where m-th dimension of a_{ij} is 1 if there is an edge of type m between stock i and stock j and 0 if there exists no edge between stock i and stock j. The pairwise relation between two stocks is represented by a multi-hop binary vector $a_{ij} \in \mathbb{R}^{\mathrm{K}}$ where the m-th entry of a_{ij} is 1 if there is an edge of type m between stock i and stock j, otherwise, the m-th entry of a_{ij} is 0. Therefore we formulate N stocks as vertices of a graph, and represent the complicated relationship among them as an adjacency matrix $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}\times\mathrm{K}}$ where the i-th row and j-th column is a_{ij}.

Given N stocks with their sequential input features $X^t \in \mathbb{R}^{\mathrm{S}\times\mathrm{D}}$ and their multi-hot binary adjacency matrix $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}\times\mathrm{K}}$, our proposed framework, Deep **A**ugmented **R**elational **Stock** Prediction (**AR-Stock**) aims to predict the close price of all stocks next day, and to rank the predicted return ratio and select the highest-revenue stocks.

Preliminary

Sequential Embedding Because stock markets are temporal volatile and dynamic, modeling the history of each stock is important for predicting future prices. Therefore, we use a sequential embedding layer to capture the sequential information in the historical stock values. Following (Feng et al. 2019), we choose LSTM because to capture the crucial long-term temporal relationships and factors of each stock. Therefore, we feed the historical sequential data of stock i at time-step t (X_i^t) to the LSTM network and take the last hidden state (e_i^t) as the sequential embedding of a stock at time t, i.e., we have,

$$E^t = \mathrm{LSTM}(X^t) \tag{1}$$

where $E^t = [e_1^t, ..., e_{\mathrm{N}}^t]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{U}}$denotes the initial temporary embeddings of all stocks at time t, and U is the dimension of embedding.

Temporal Graph Convolutional Network To model the relationships between different stocks, Feng et al. (Feng et al. 2019) propose temporal graph convolution, the variant of which with the best performance in (Feng et al. 2019) is introduced to our framework.

Given N stocks with their sequential embeddings $E^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{U}}$ (i.e., the output of sequential embedding layer) and their multi-hot binary relation encodings $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}\times\mathrm{K}}$, the aim of temporal graph convolution is to learn the embeddings $\overline{E^t} \in \mathbb{R}^{\mathrm{N}\times\mathrm{U}}$ that encode the relationship information between the stocks. To address the problem that the stock market is highly dynamic, they assume the strength of a relation are continuously evolving and propose to combine the temporal information and the relation information. The time-aware embedding propagation in the stock market is defined as follows:

$$\overline{e_i^t} = \sum_{\{j|\,\mathrm{sum}(a_{ij})>0\}} \frac{g\left(a_{ij}, e_i^t, e_j^t\right)}{d_j} e_j^t \tag{2}$$

where g is the weighting score and d is the normalization factor. The relation strength g is estimated by feeding the sequential embeddings and the relation multi-binary vector into a fully connected layer as:

$$g\left(a_{ji}, e_i^t, e_j^t\right) = \phi\left(w^{\mathrm{T}}\left[e_i^{t^{\mathrm{T}}}, e_j^{t^{\mathrm{T}}}, a_{ji}^{\mathrm{T}}\right]^{\mathrm{T}} + b\right) \tag{3}$$

where $w \in \mathbb{R}^{2\mathrm{U}+\mathrm{K}}$and $b \in \mathbb{R}^1$ are model parameters to be learned; ϕ is the activation function.

From Graph to Augmented Graph: Discovering Long-Range Dependencies

Graph convolution networks (GCNs) lose the structural information of nodes in neighborhoods and lacking the ability to capture long-range dependencies (Pei et al. 2020). To tackle this disadvantage, the first component of AR-Stock is *geometric augmentation*, which detects the structural neighborhood of each node in the stock knowledge graph and define a new relation type to represent the structural relationship. Then we propose an aggregation schema to unify the original knowledge graph relation and the proposed structural relationship.

Structural Neighborhood We first introduce how to define the structural neighborhood. Different from vanilla GCNs dependent on local dependency, we provide global information w.r.t the geometric structure preserved in the latent space of unsupervised graph embedding models. Here we adopt the state-of-the-art knowledge graph embedding method RotatE (Sun et al. 2019) to get the entity (stock) embedding which various relation patterns including symmetry/anti-symmetry, inversion, and composition are preserved. We use matrix $Z \in \mathbb{R}^{\mathrm{N}\times d}$ to represent the stock embedding obtained by the knowledge graph embedding model where d is the embedding dim and each entry of z resemble the embedding of each stock. One can employ various knowledge graph embedding methods to infer the latent space. Then the structural relationship in latent space is defined as:

$$N_s(i) = \{j | d(z_j, z_i) < \rho\} \tag{4}$$

where the number of original knowledge graph links is:

$$N_{kg} = \sum_i \sum_j a_{ij} \tag{5}$$

In our framework, ρ is selected by increasing the number of structural links from zero to more than N_{kg}, structural neighborhood can also be written as the adjacency matrix $A^s \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$.

We note that the structural edges are different from the original edges: while the latter were actually observed in the

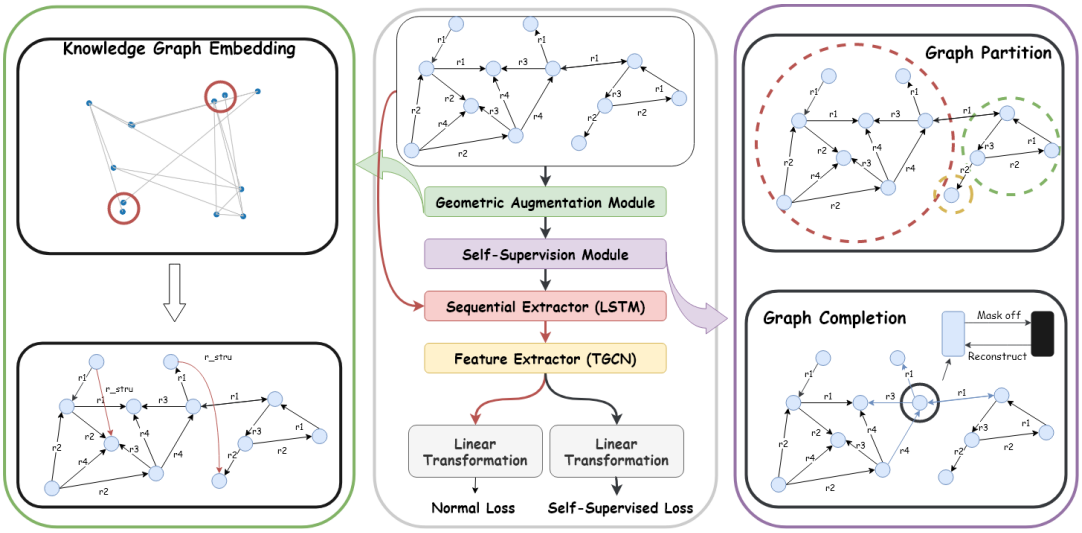

Figure 2: The middle figure illustrates the AR-Stock framework on the stock knowledge graph. The left figure stand for the Geometric Augmentation Module. The structural links are selected based on the pairwise distance of their knowledge graph embeddings. The right figure is our Self-Supervision Module. We adopt Graph Partition and Graph Completion to facilitate training. The normal loss and the self-supervised loss share the same sequential extractor and feature extractor with their individual linear transformation parameters.

market record, the former were "inferred" to denote "hidden relationships". Those "hidden relationships" may never actually happen, are not observable, or are never recorded; however, they represent the extra "latent correlations" between nodes as drawn from data, beyond those already on the record. Those inferred hidden relationships potentially abstract the deeper, even non-quantifiable economic market confounding factors, and greatly complement the observed graph.

Aggregation Scheme Next, we present the aggregation scheme to incorporate the structural adjacency matrix with GCN. The structural link can be regarded as a new kind of link. Therefore, we define a new adjacency matrix through concatenating the original knowledge graph adjacency matrix A with the structural adjacency matrix A^s as follows:

$$A^{\text{Total}} = \text{Concat}[A, A^s] \tag{6}$$

where $A^{\text{Total}} \in \mathbb{R}^{\text{N}\times\text{N}\times(\text{K}+1)}$. Then we compute the relation weight in graph convolution layer based on the new adjacency matrix:

$$g\left(a_{ji}^{\text{Total}}, e_i^t, e_j^t\right) = \phi\left(w^{\text{T}}\left[e_i^{t^{\text{T}}}, e_j^{t^{\text{T}}}, a_{ji}^{\text{Total}^{\text{T}}}\right]^{\text{T}} + b\right) \tag{7}$$

where $w \in \mathbb{R}^{2\text{U}+(\text{K}+1)}$. Other aggregation schemes are studied in Appendix D, and we find this one obtains the best performance.

Self-Supervision for Globel/Local Dependency Awareness

Furthermore, we introduce two kinds of self-supervised methods into AR-Stock via multi-task learning, which is shown to be a promising way in the visual domain to facilitate our model training. We incorporate self-supervised learning into stock prediction via multi-task learning in the following ways:

$$\ell = \ell_{\text{stock}} + \beta \times \ell_{\text{self}} \tag{8}$$

where ℓ_{stock} is the loss related to stock prediction, ℓ_{self} is the self-supervised learning loss and β is the trade-off hyper-parameter.

Graph Partition The first self-supervised task is graph partition, parting the graph into M parts where M is a hyper-parameter to be tuned, where the partition labels are used as self-supervised labels. Here METIS (Karypis and Kumar 1998) is used to perform graph partition.

However, the knowledge graph cannot be directly applied partitioned since there are multiple relation types existing such that there may exist multiple links between stocks. Moreover, the knowledge graph is a directed graph. To overcome so, we present approximation strategies below.

First, we treat every directed edge as an undirected link to eliminate the direction issue. Then, different types of edges are unified as the same type when partition, which is merged between stocks into an edge with a weight m where m is the number of links between the two stocks. Next, all m edges between two stocks are merged to form an edge with a weight m. With the approximation, we can use graph partition to get the subgraph to which each stock belongs to. These can be regarded as the node labels for which subgraph they belong to, and then fed into the regularization. Here we use the cross entropy loss as the regularization.

$$\ell_{\text{part}} = -\sum_i y_i^t \log(p^t) \tag{9}$$

where y is the stock partition label and p can be obtained as follows:

$$p^t = \text{softmax}(e_{\text{part}}^t) \tag{10}$$

$$e^t_{\text{part}} = \text{MLP}(\overline{e^t_i}) \tag{11}$$

where the embeddings after the graph convolution layer are feed into an neural network and softmax layer. The upper right of fig 2 depicts our Graph Partition method.

Graph Completion The second self-supervised method we propose is graph completion. Motivated by image inpainting a.k.a. completion (Yu et al. 2018) in computer vision (which aims to fill missing pixels of an image), we propose graph completion, a novel regression task, as a self-supervised task. As an analogy to image completion and illustrated in Figure1 1, our graph completion first masks target nodes by removing their features. It then aims at recovering/predicting masked node features by feeding to GCNs unmasked node features (currently restricted to second-order neighbors of each target node for 2-layer GCNs).

We design such a self-supervised task for the following reasons: 1) the completion labels are free to get, which is the node feature itself; and 2) we consider graph completion can aid the network for better feature representation, which teaches the network to extract feature from the context.

We next show how to implement our method. We first create an adjacency matrix called A_{self} same as the original adjacency matrix except that it doesn't contain self-loops. Then we input A_{self} to another GCN model to get the embeddings that are not affected by the self-loop. Then we get the regression loss as follows:

$$\ell_{\text{regre}} = ||e^t_{\text{regre}} - f^t||^2 \tag{12}$$

where f^t is the input feature at time t and e^t_{regre} is the embedding obtained through the graph convolution network with A_{self}. Fig 2 lower right depicts our Graph Completion.

Prediction Layer

Lastly, we feed the sequential embeddings and revised relational embeddings to a fully connected layer to predict the ranking score of each stock; the ranked list of stocks recommended to buy is then generated based on the prediction scores. To optimize the model, we propose an objective function that combines both pointwise regression loss and pairwise ranking-aware loss:

$$\begin{aligned}\ell_{\text{stock}}\left(\hat{r}^{t+1}, r^{t+1}\right) &= \left\|\hat{r}^{t+1} - r^{t+1}\right\|^2 \\ &+ \alpha \sum_{i=0}^{\text{N}} \sum_{j=0}^{\text{N}} \max\left(0, -\left(\hat{r}^{t+1}_i - \hat{r}^{t+1}_j\right)\left(r^{t+1}_i - r^{t+1}_j\right)\right)\end{aligned} \tag{13}$$

where $r^{t+1} = [r^{t+1}_1, ..., r^{t+1}_{\text{N}}]$ and $\hat{r}^{t+1} = \left[\hat{r}^{t+1}_1, \cdots, \hat{r}^{t+1}_{\text{N}}\right] \in \mathbb{R}^{\text{N}}$ are ground-truth and predicted ranking scores, respectively, and α is a hyper-parameter to balance the two loss terms. Since we focus on identifying the most profitable stock to trade, we use the 1-day return ratio of a stock as the ground-truth rather than the normalized price used in previous work. The first regression term punishes the difference between the scores of ground-truth and prediction. The second term is pair-wise max-margin loss (Zheng et al. 2007), which encourages the predicted ranking scores of a stock pair to have the same relative order as the ground-truth.

Table 1: Statistics of the sequetial and relation data.

Datasets	Stocks	Train\Valid\Test	Relation Types	Relation Ratio
NASDAQ	1026	756\252\237	42	0.21%
NYSE	1737	756\252\237	130	9.37%

For our method with self-supervised regularization, the loss function will be changed as follows:

$$\ell = \ell_{\text{stock}}\left(\hat{r}^{\text{t+1}}, r^{\text{t+1}}\right) + \beta \times \ell_{\text{self}} \tag{14}$$

where β is the trade-off hyper-parameter and ℓ_{self} is the two kinds of self-supervised learning loss.

EXPERIMENTS

To justify the effectiveness of AR-Stock, we conduct extensive experiments to answer the following research questions:

- **RQ1**: How does AR-Stock perform compared with state-of-the-art stock prediction methods?
- **RQ2**: How different components (*e.g.*, self-supervision, geometric augmentation, trade-off parameter) affect the results of AR-Stock?
- **RQ3**: What is the effect of our self-supervision methods on stocks with limited data?
- **RQ4**: Why does geometric augmentation work and what affects the performance of it?

Datasets

To evaluate the effectiveness of AR-Stock, we conducted experiments on two benchmark datasets from (Feng et al. 2019) that incorporate stock relations from NASDAQ and NYSE markets. The two datasets have transaction records between 01/02/2013 and 12/08/2017. Following (Feng et al. 2019), we chronologically separate the stock price data into three time periods for training (2013-2015), validation (2016), and evaluation (2017), respectively. The statistics about the sequential and relational data are summarized in Table 1. The detailed description of datasets can be found in Appendix A.

Experimental Setup

We adopt a daily buy-hold-sell trading strategy to evaluate the performance of stock prediction methods regarding the revenue. Since the target is to accurately predict the return ratio of stocks and appropriately rank the relative order of stocks, we employ three metrics, Mean Square Error (MSE), Mean Reciprocal Rank (MRR), and the cumulative investment return ratio (IRR), to report model performance. The detailed description can be seen in Appendix B.

To demonstrate the effectiveness of AR-Stock, we compare our proposed AR-Stock with three state-of-the-art methods: SFM (Zhang, Aggarwal, and Qi 2017), LSTM (Bao, Yue, and Rao 2017), TGCN (Feng et al. 2019). The detailed description of these baselines can be found in Appendix C. To explore various design options for AR-Stock, we design two variants of AR-Stock with two different self-supervised learning methods. The Part represents our AR-Stock with the self-supervised method of graph partitioning. The Comp represents our AR-Stock with the self-supervised method of graph completion.

Table 2: Experimental results of different methods over the NASDAQ and the NYSE datasets. Comp and Part in the table represent graph completion and graph partition methods of self-supervised learning, respectively.

	NASDAQ			NYSE		
	MSE	MRR	IRR	MSE	MRR	IRR
SFM	5.20e-4±5.77e-5	2.33e-2±1.07e-2	-0.25±0.52	3.81e-4±9.30e-5	3.82e-2±4.95e-3	0.49±0.47
LSTM	3.81e-4±2.20e-6	3.64e-2±1.04e-2	0.13±0.62	2.31e-4±1.43e-6	2.75e-2±1.09e-2	-0.90±0.73
TGCN	3.79e-4±3.58e-7	3.34e-2±3.19e-3	1.49±0.31	2.27e-4±1.24e-6	4.10e-2±2.65e-3	1.87±0.22
AR-Stock-Part	**3.78e-4±4.18e-7**	3.99e-2±4.32e-3	2.00±0.33	2.26e-4±5.14e-7	**5.13e-2±3.00e-3**	**2.44±0.16**
AR-Stock-Comp	3.78-4±4.50e-7	**4.27e-2±3.30e-3**	**2.47±0.35**	**2.25e-4±4.40e-7**	4.31e-2±4.98e-3	2.02±0.14

Table 3: Performance of geometric augmentation.

		TGCN	TGCN+Geom
NASDAQ	MSE	3.79e-4±3.58e-7	**3.78e-4±5.61e-7**
	MRR	3.34e-2±3.19e-3	**4.01e-2±3.16e-3**
	IRR	1.49±0.31	**1.77±0.27**
NYSE	MSE	2.27e-4±1.24e-6	**2.25e-4±2.34e-7**
	MRR	4.10e-2±2.65e-3	**4.47e-2±2.38e-3**
	IRR	1.87±0.22	**2.11±0.28**

Table 4: Performance of different self-supervision methods

		TGCN	TGCN+Part	TGCN+Comp
NASDAQ	MSE	3.79e-4±3.58e-7	**3.78e-4±5.00e-7**	3.78e-4±5.48e-7
	MRR	3.34e-2±3.19e-3	**3.74e-2±2.22e-3**	3.59e-2±3.13e-3
	IRR	1.49±0.31	1.60±0.14	**1.67±0.18**
NYSE	MSE	2.27e-4±1.24e-6	**2.26e-4±6.73e-7**	2.26e-4±1.29e-6
	MRR	4.10e-2±2.65e-3	4.07e-2±1.62e-3	**4.23e-2±2.70e-3**
	IRR	1.87±0.22	**2.01±0.23**	1.89±0.23

We implement our AR-Stock model in Tensorflow. We set the dimensions of embedding vectors of TGCN and AR-Stock models 64 by default. For TGCN and AR-Stock model, we utilize one layer of embedding propagation network to obtain the best results. For our method, we test the values of [1e-2, 1e-3, 1e-4, 1e-5, 1e-6] for the self-supervised trade-off parameter, finding a value of 1e-5 leads to the best result on both datasets. We determine ρ by increasing ρ from zero to more than N_{kg}. Moreover, the same with (Feng et al. 2019), we also pre-train the LSTM layer to obtain the sequential embedding. The number of training epochs is set to 50. The learning rate is fixed to 1e-3 and then Adam (Kingma and Ba 2014) is adopted as the optimizer. Besides, as the stock markets are rather sensitive, we repeat the testing procedure five times and report the average performance and corresponding standard deviation to eliminate the fluctuations.

Comparison with State-of-the-arts (RQ1)

We first compare the performance of AR-Stock with existing methods. Table 2 shows the two variants of AR-Stock and the performance comparison w.r.t. MSE, MRR, and IRR among the baseline methods on the NASDAQ, and NYSE datasets. We can find that AR-Stock with graph partition performs the best in general and AR-Stock with graph completion performs better than graph partition w.r.t some metrics of the two datasets. From the results in the table, we gain the following observations:

- **In general, our model yields the best performance on all the datasets in terms of all three metrics.** The best performing method is highlighted. In particular, our method improves over the strongest baselines w.r.t. investment return ratio by 65.77%, and 30.48% in NASDAQ, and NYSE dataset, respectively. By utilizing self-supervised learning and geometric augmentation, our method is capable of exploring the structural information explicitly, to benefit graph neural network effectively. We can find nearly each of our methods outperforms the original method. This verifies the significance of introducing self-supervised learning and geometric augmentation.
- **Our method with partition achieves the most promising result in general.** It makes sense since by graph partition method and geometric augmentation both enhance utilizing structural information. This indicates the limitations of graph neural network and shows a promising way to alleviate the problem.
- We find the TGCN method based on graph neural network significantly outperforms the traditional method such as LSTM and SFM as they do not consider the complex relationship between stocks. It again **verifies the importance to model the stock relations as a knowledge graph**.

Ablation Study (RQ2)

As the self-supervision and geometric augmentation play a pivotal role in AR-Stock, we investigate their impact on the performance. We first consider our method with only geometric augmentation, the ablation on the contribution of our method is in table 3. The settings of hyper-parameter are the same as before. We can have the following finding that the geometric augmentation method outperforms the basic graph neural network method. This verifies the usefulness of incorporating the geometric augmentation.

The ablation on the contribution of the two self-supervised learning methods in table 4. We can see the two self-supervised learning methods outperform the original GCN in most cases. Also we can see only using self-supervised learning decreases the performance a lot, this again verifies the importance of adding more links that can reflect the structural information.

Furthermore, we show how do these hyper-parameters introduced in our AR-Stock impact the performance and also shed light on how to set them. Due to space limitation, for the following experiments, we show the results on the NASDAQ dataset only, and the results on the NYSE dataset show the same trend. We mainly study the influence of the tradeoff

Table 5: Comparison of two self-supervised learning methods with different coefficients in terms of stock prediction on NASDAQ dataset.

	TGCN+Part			TGCN+Comp		
Coefficient	MSE	MRR	IRR	MSE	MRR	IRR
1e-4	3.79e-4±5.66e-7	3.45e-2±4.15e-3	1.50±0.29	3.84e-4±1.08e-6	3.02e-2±3.13e-3	1.66±0.44
1e-5	**3.78e-4±5.00e-7**	**3.74e-2±2.22e-3**	**1.60±0.14**	**3.78e-4±5.48e-7**	**3.59e-2±3.13e-3**	**1.67±0.18**
1e-6	3.79e-4±4.97e-7	3.71e-2±2.22e-3	1.55±0.11	3.80e-4±3.67e-7	3.57e-2±5.04e-3	1.51±0.36

Table 6: Experiments with different amount of training data on NASDAQ.

	TGCN	TGCN+Part	TGCN+Comp
1.00	3.79e-4±3.58e-7	**3.78e-4±5.00e-7**	3.78e-4±5.48e-7
0.10	3.89e-4±3.23e-6	**3.81e-4±7.89e-7**	3.82e-4±1.02e-6
0.05	4.86e-4±4.55e-5	4.12e-4±3.78e-5	**4.05e-4±1.89e-5**
0.01	3.15e-3±9.12e-5	2.24e-3±4.12e-5	**2.00e-3±2.43e-5**

Table 7: Experiments with different mask ratios on NASDAQ.

	10%	30%	50%	70%	90%
MRR	0.954	0.917	0.859	0.755	0.615
HR@1	0.925	0.870	0.791	0.673	0.536
HR@10	0.990	0.981	0.958	0.897	0.763

parameter used in the two self-supervised learning methods. The results are summarized in table 5. We can find that the best results are obtained when the coefficient is set to 1e-5. When the coefficient is set to 1e-6, our method decreases w.r.t the three metrics compared with the best results because it is insufficient to learn the self-supervision information, while when the coefficient is increased to 1e-6, our methods decreases greatly in all indexes, which indicates that too strong self-supervision regularization will negatively affect model normal training and is not encouraged. To conclude, this suggests that our approach is primarily complementary to the GCN. Also we find that graph partition performs better than graph completion. This again verifies our idea of using structure information. More ablation study about self-supervision and geometric augmentation can be seen in Appendix D.

Self-supervision with Limited Data (RQ3)

The data sparsity issue usually limits the expressiveness of machine learning systems, since few data of inactive stocks are insufficient to generate high-quality representations. We investigate whether exploiting self-supervision information helps to alleviate this issue.

Towards this end, we perform experiments over different amounts of training data. In particular, we keep a certain ratio of the latest data of each stock. In the experiments, the ratios per stock are 1.00, 0.10, 0.05, 0.01 respectively where 1.00 represents the original dataset. Table 6 illustrates the results w.r.t. MSE on different sizes of data in NASDAQ; we see a similar trend for performance w.r.t. MRR and IRR and omit the part due to the space limitation. We find that our proposed two self-supervision methods consistently outperform TGCN even with a very limited amount of data. It can also be found that the fewer data we have, the more significant improvements we can obtain. This verifies the effectiveness of our self-supervision strategies.

Table 8: Experiments of geometric augmentation with different number of geometric links in terms of stock prediction on NASDAQ dataset. The ratio is the proportion of geometric links to the links in the original dataset.

	TGCN+Geom		
Ratio of Links	MSE	MRR	IRR
30%	3.79e-4±3.87e-7	3.65e-2±4.01e-3	1.62±0.52
60%	3.79e-4±3.61e-7	3.96e-2±3.18e-3	1.64±0.46
90%	**3.78e-4±5.61e-7**	**4.01e-2±3.16e-3**	**1.77±0.27**
120%	3.79e-4±6.42e-7	3.90e-2±6.21e-3	1.54±0.51

Effect of Geometric Augmentation (RQ4)

We would like to investigate if the long-range dependencies are rich and stable enough to be captured from industry and supplier relationships. Towards this end, we perform experiments over knowledge graphs with different ratios of masked edges using our method in Section . In particular, we randomly mask 10%, 30%, 50%, 70%, 90% of edges in the original knowledge graph respectively, and discover whether our approach can well recover them. Table 7 illustrates the results w.r.t. MRR, HR@1, and HR@10 on different mask ratios of the knowledge graph in the NASDAQ dataset. We have the finding that known relationships are significantly more likely to be learned in the stock knowledge graph even with only 10% of remaining edges. Therefore, by performing the proposed method on the complete stock knowledge graph, potential long-range relationships can be well detected and GCNs training can be facilitated.

To further explore the role of geometric augmentation, we next show the influence of the different numbers of geometric links. We fix all other hyper-parameters and only adjust the number of long-range links added. We find that the best result is obtained when the geometric links close to the number of links in the original dataset were added. This can be explained as adding too many links will mix irrelevant information, and adding too few edges will lead to insufficient use of structural information. As shown by Table 8, we find adding 90% links yields the best result, and adding more or fewer links will decrease the performance.

Conclusions

In this paper, we propose AR-Stock via the incorporation of geometric augmentation and self-supervision to assist GCN training for stock trend prediction. Experiments on the NASDAQ and NYSE datasets showed AR-Stock significantly

outperforms state-of-the-art methods, which indicates the ability of our model to capture the long-range dependency relationships between corporations to make accurate predictions on the stock market. Our model is model-agnostic and has broader applications on knowledge graph-based learning problems, which will be explored in the future.

References

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

Berg, R. v. d.; Kipf, T. N.; and Welling, M. 2018. Graph convolutional matrix completion. *KDD* .

Chen, T.; Liu, S.; Chang, S.; Cheng, Y.; Amini, L.; and Wang, Z. 2020. Adversarial Robustness: From Self-Supervised Pre-Training to Fine-Tuning. In *CVPR*.

Chen, Y.; Wei, Z.; and Huang, X. 2018. Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction. In *CIKM*. ACM.

DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2018. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *ICCV*.

Dosovitskiy, A.; Springenberg, J. T.; Riedmiller, M.; and Brox, T. 2014. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *TOIS* .

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised representation learning by predicting image rotations. *ICLR* .

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry 1263–1272.

Goyal, P.; Mahajan, D.; Gupta, A.; and Misra, I. 2019. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Hu, Z.; Liu, W.; Bian, J.; Liu, X.; and Liu, T.-Y. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*.

Jin, W.; Derr, T.; Liu, H.; Wang, Y.; Wang, S.; Liu, Z.; and Tang, J. 2020. Self-supervised Learning on Graphs: Deep Insights and New Direction. *arXiv* .

Karypis, G.; and Kumar, V. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* .

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *ICLR* .

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *ICLR* .

Kocabas, M.; Karagoz, S.; and Akbas, E. 2019. Self-supervised learning of 3d human pose using multi-view geometry. In *CVPR*.

Kolesnikov, A.; Zhai, X.; and Beyer, L. 2019. Revisiting self-supervised visual representation learning. In *CVPR*.

Kondor, R.; Son, H. T.; Pan, H.; Anderson, B. M.; and Trivedi, S. 2018. Covariant Compositional Networks For Learning Graphs. In *ICLR Workshops*.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348.

Liu, P.; Lyu, M.; King, I.; and Xu, J. 2019. Selflow: Self-supervised learning of optical flow. In *CVPR*.

Matsunaga, D.; Suzumura, T.; and Takahashi, T. 2019. Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window Analysis. *arXiv* .

Nguyen, T. H.; and Shirai, K. 2015. Topic modeling based sentiment analysis on social media for stock market prediction. In *ACL*.

Noroozi, M.; and Favaro, P. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. *ICLR* .

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Sun, K.; Lin, Z.; and Zhu, Z. 2020. Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labeled Nodes. In *AAAI*.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *ICLR* .

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *ICLR* .

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. *ICML* .

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020a. Graph Contrastive Learning with Augmentations.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020b. L^2-GCN: Layer-Wise and Learned Efficient Training of Graph Convolutional Networks. In *CVPR*.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020c. When Does Self-Supervision Help Graph Convolutional Networks? *ICML* .

Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; and Huang, T. S. 2018. Generative image inpainting with contextual attention. In *CVPR*.

Zhai, X.; Oliver, A.; Kolesnikov, A.; and Beyer, L. 2019. S4l: Self-supervised semi-supervised learning. In *ICCV*.

Zhang, L.; Aggarwal, C.; and Qi, G.-J. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *NeurIPS*.

Zhao, S.; Wang, Q.; Massung, S.; Qin, B.; Liu, T.; Wang, B.; and Zhai, C. 2017. Constructing and embedding abstract event causality networks from text snippets. In *WSDM*.

Zheng, Z.; Chen, K.; Sun, G.; and Zha, H. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*.

Zhu, Q.; Du, B.; and Yan, P. 2020. Self-supervised Training of Graph Convolutional Networks. *arXiv* .