Illuminating the Influence of Endogenous, Exogenous, and Learnable Linkages in Time Series Prediction Task

Anastasiya Nasykhova[*1], **Kseniia Miloradova**[*1], **Tatyana Kulikova**[*1], **Elizaveta Kovtun**[*2], **and Semen Budennyy** [2, 3]

[1] Higher School of Economics University, Moscow, Russia
[2] Sber AI Lab, Moscow, Russia
[3] Artificial Intelligence Research Institute (AIRI), Moscow, Russia

Abstract

Time series serves as a natural representation form to describe a spectrum of processes occurring with a specific entity. Often, various relationships exist between entities. Considering these interconnections can significantly enhance time series prediction accuracy. This is especially vital when dealing with stock price time series due to strong company interconnectedness. The majority of works leveraging company connections focus on one-sided links, neglecting consideration and comparison of alternative associations.

This study aims to bridge this gap by exploring different relationship types: endogenous, exogenous, and learnable, and comprehend the impact of varied links. Exogenous links originate from external sources like financial reports or corporate associations in WikiData. Endogenous dependencies arise from historical stock price behavior similarities. Learnable links emerge during model training and do not require any prior company knowledge or preliminary time series handling. Moreover, a unique architecture is proposed, enabling high-quality predictions and facilitating transparent relationship comparison simultaneously. The study's findings provide foundational insights into selecting the optimal connection type in time series prediction tasks.

Introduction

The stock market has become increasingly popular over the years as more individuals are looking for ways to grow their wealth and secure their financial future. With the advent of online trading platforms, investing in the stock market has become more accessible to the general public. Anyone with an internet connection can open a brokerage account and start buying and selling stocks. Market conditions can change rapidly, and unforeseen events can impact stock prices. Therefore, it is crucial for investors to use stock market trends forecasting as a tool for making informed decisions about buying or selling stocks. By analyzing market trends, economic indicators, and company performance, investors can identify potential opportunities and risks in the market. Accurate stock market forecasting can help build investor confidence. When investors have access to reliable forecasts and analysis, they are more likely to make conscious investment decisions and participate in the market.

[*]These authors contributed equally.

The purpose of this research is to study the influence of information about various types of company relationships on the stock movements forecasting. With the growing interconnectedness of global markets, understanding the relationships and dependencies between companies has become substantial for accurate stock market forecasts. Traditional stock market forecasting models often focus either on factors specific to individual companies, without explicitly taking into account the impact of linkages between companies, or use only one type of connection. However, different types of linkages have different benefits.

We selected companies from NASDAQ index that had the largest number of relationships on WikiData, based on this, we collected financial reports for 2017-2019 and uploaded time series to compare endogenous, exogenous linkages and associations from WikiData.

Exogenous. After gathering financial reports of NASDAQ index companies, we performed their clustering, utilizing the statistical approach, which was connected with the most considerable financial indicators' distributions.

Associations from WikiData. We selected the most relevant types of relationships between companies from WikiData open knowledge base and extracted data on the selected types of relationships for the companies in the NASDAQ index.

Endogenous. The endogenous approach involves the clustering of companies based on their historical time series data. Various clustering algorithms were employed in this process, utilizing the Dynamic Time Warping (DTW) distance metric as a measure of similarity.

Prediction Model. In this study, a novel neural network architecture has been developed to forecast stock price trends by considering the interrelationships among various companies. The proposed model comprised of three main stages. Firstly, the historical time-series data of each stock is decomposed into trend and seasonal components, which are then concatenated and passed through a linear layer to obtain sequential embeddings. Subsequently, the embeddings of the companies are recalculated by incorporating information from neighboring companies through a specific method that will be elaborated upon in the subsequent sections. This results in the generation of relational embeddings. Finally, the concatenation of sequential and relational embeddings is feed into fully connected layers for predicting the trend of the time series.

The key contributions of the article are summarized as follows.

- We explored the degree of influence of information about interconnections between companies in the market on the efficiency of predicting the share price trend.
- We collected data on different types of linkages between companies within the market and organised it into the linkage matrices.
- We picked out the most appropriate indicators for annual report based clustering.
- We created a baseline movement predicting model.
- We enhanced baseline by creating a transformation embedding attention layer. The layer implies adding data about the relationships between companies.

Literature review

Many scientists are studying the influence of information about the connections between objects on the quality of time series forecasting. For example, the authors of the article "Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction" (Xiang et al. 2022) propose an approach based on temporal and heterogeneous graph neural networks (THGNN) to study the relationships between price changes in financial time series taking into account dynamics. In particular, they generate a graph of company relationships for each trading day by their historical price. The experimental results demonstrate the proposed methods' effectiveness compared to the most modern fundamental indicators. The hypergraph describing the relationships is also used in the work entitled "Efficient Integration of Multi-Order Dynamics and Internal Dynamics in Stock Movement Prediction" (Huynh et al. 2022), the authors of which pay special attention to identifying individual patterns of behavior of stock prices of each company and analyzing their impact on each other. The authors of the article "HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction" (Kim et al. 2019) propose a hierarchical attention network for stock prediction, which selectively aggregates information about various types of relationships and adds this information to the representations of each company.

The article "A Comprehensive Survey on Graph Neural Networks" (Wu et al. 2021) provides a comprehensive review of graph neural networks (GNN) in the field of data mining and machine learning. The authors note that the complexity of graphical data creates significant problems for existing machine learning algorithms, so there has been much research recently on expanding deep learning approaches for graphical data. In particular, there are growing applications in which data is generated as graphs with complex relationships and interdependence between objects.

The authors of the work entitled "HGNN: Hierarchical graph neural network for predicting the classification of price-limit-hitting stocks" (Xu et al. 2022) are developing a new hierarchical architecture that takes into account the historical patterns of stock price series together with their relationships with each other in order to solve the classification problem, namely, to predict whether a stock that has reached its daily price limit will close at the same price level. Much research is connected with the task of forecasting the trend of stock prices. For example, this task is described in another article, the topic of which is "An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market" (Long et al. 2020). The authors of the article note that most of the works solving the same problem are focused on publicly available market data and do not use the behavior of traders due to the lack of data on actual transactions. In this paper, the researchers propose a deep neural network model that uses transaction records and publicly available market information to predict the trend of stock prices. Their method considers the correlation between time series of stock prices and builds a knowledge graph based on this information. Ultimately, an attention-based bidirectional network of long-term, short-term memory can predict stock price trends to aid financial decision-making. Experiments on forecasting the direction of price movement and trends have shown that this method provides the best efficiency compared to other basic forecasting methods.

Most existing solutions formulate stock forecasting as a classification problem (to predict the stock trend) or a regression problem (to predict the stock price). The paper entitled "Temporal Relational Ranking for Stock Prediction" (Feng et al. 2019) presents a method that adapts deep learning models to rank time series of stock prices and select the best stocks with the highest expected earnings. In addition, they fix the ratio of time series considering time. The key novelty of this work is the development of a new component in the modeling of neural networks, called convolution of time graphs, which models a network of time series relationships, taking into account changes in time. The study also considers a similar idea, which sounds like "Stock Selection via Spatiotemporal Hypergraph Attention Network: A Learning to Rank Approach" (Sawhney et al. 2021).

We also reviewed studies examining the effectiveness of various architectures of deep learning models. One such article is "Are Transformers Effective for Time Series Forecasting?" (Zeng et al. 2022), the authors dispute the information about the success of transformer–based architecture (Vaswani et al. 2023) and prove that linear models surpass existing transformer models in efficiency.

Another article – "Cluster Analysis with K-Mean versus K-Medoid in Financial Performance Evaluation" (Herman, Zsido, and Fenyves 2022), is related to the clustering of companies based on indicators from their financial reports. This paper examines the financial performance of Hungarian and Romanian food retail companies using two well-known cluster analysis methods (K-Mean and K-Medoid) based on the financial coefficients ROS (return on sales), ROA (return on assets), and ROE (return on equity). This study aims to highlight the differences between the results of these two clustering algorithms. The results demonstrate that the method chosen for grouping can influence the assessment of companies' financial indicators: the K-average method gives a greater variety of groups, and the range of results obtained after grouping is more significant. At the same time, the distribution by groups and the results obtained by the K-Medoid method are more balanced.

Methods

Exogenous linkages

Information from financial reports. Financial reports are written notes which reflect the results of companies' business activities and financial performance. Financial statements are usually inspected by stakeholders, such as investors, creditors, and management, to ensure the company's financial well-being and for tax, financing, or investing purposes. Investors and financial analysts make decisions based on financial data to analyze the state of a company, its financial health, and its profit potential for forecasting the future trend of its stock price. For getting the most reliable and relevant financial data, it is recommended to consider the annual report, which contains the firm's financial indicators.

The three major financial statement reports are the balance sheet, income statement, and cash flow statement.

- **Income Statement** - summarizes a company's revenues, expenses, and net income or loss over a specific period.
- **Balance Sheet** - provides an overview of a company's assets, liabilities, and shareholders' equity at a particular moment.
- **Cash Flow Statement** - presents the inflows and outflows of cash and cash equivalents from operating, investing, and financing activities during a given period. This report helps evaluate a company's ability to generate cash and manage its liquidity.

Most important financial variables. For defining companies' interconnections, we chose the most essential variables and performed clustering based on financial parameters. The indicators, which we have selected, are:

- **Total assets** - the sum of all current and non-current assets equal to the sum of total liabilities and stockholder's equity on the other side of the balance sheet.
- **Total shareholders' equity** - the remaining assets available to shareholders after all liabilities are paid. It is calculated as a firm's total assets, less its total liabilities
- **Operating income** - a company's profit after deducting operating expenses, which are the costs of running the day-to-day operations. Operating income, synonymous with operating profit, allows analysts and investors to drill down to see a company's performance by stripping out interest and taxes.
- **Net income** - is a company's profits or earnings. Net income is the bottom line since it sits at the bottom of the income statement and is the income remaining after factoring in all expenses, debts, additional income streams, and operating costs. Net income is calculated by netting items from operating income, including depreciation, interest, taxes, and other expenses. Sometimes, additional income streams add to earnings like interest on investments or proceeds from the sale of assets.

Since different companies are very diverse in terms of size and sales, for clustering, we utilized only relative indicators.

- $Return\ on\ Assets(ROA) = \frac{Net\ income}{Total\ Assets}$ - a measure of how efficiently a company uses the assets it owns to generate profits.
- $Return\ on\ equity(ROE) = \frac{Net\ income}{Total\ Equity}$ - considered a gauge of a corporation's profitability and how efficient it is in generating profits. The higher the ROE, the more efficient a company's management is at generating income and growth from its equity financing.
- $Net\ income\ fraction = \frac{Net\ income}{Operating\ income}$ - operating income is the income caused by conducting business operations, and net income is the profit left after considering all the expenditures. Operational efficiency should be increased by minimizing costs and wastage to increase operating income. So, the bigger the share of net income in operating income, the less the expenses that have been deducted, and the more efficient the company's performance.

Financial reports' download. In this study, we gathered financial statements of NASDAQ index companies, which have many WikiData linkages. The period is from 2017 to 2019; for our experiments, it is crucial to allocate the same periods for generating sets of connections of each type. Since the information about links on WikiData is hardly accessible, we focused on information from Wiki when selecting a period. We have chosen the Security and Exchange Commission filings database as the source of financial reports. The SEC filing is a financial report or other official document submitted to the U.S. Securities and Exchange Commission (SEC) database. Public companies are obliged to make regular SEC filings. Investors commonly rely on these filings for information about companies they evaluate for investment purposes. We used Finnhub - the real-time APIs for stock, forex, and cryptocurrency to upload data. This API provides an opportunity to get financials as reported. This data is available for bulk download on the Kaggle SEC Financials database. For easier interaction with Finnhub API, we applied its Python version, particularly the financials_reported() function, which allows the download of annual or quarterly reports by companies' tickers. The data format is an array of dictionaries for each year with parameters' names and their values. This research only examines annual reports for 2017, 2018, and 2019 years.

Financial reports-based clustering. We chose the statistical approach to divide companies into groups based on their relative financial variables. First, we made three different separations according to three indicators. Based on the ROE indicator, we clustered companies into three groups according to their ROA values. We got another three groups and divided companies into two groups, relating to the fraction of their net income in operating income. The statistical method consisted of quantile-based discretization. For this purpose, we applied the Pandas qcut function, which tries to split the underlying data into intervals of equal size. The function defines intervals using percentiles based on the distribution of data rather than the actual numerical boundaries of the intervals. Another approach we have considered is dividing into intervals of equal size, regardless of the data distribution in them. However, with this approach, clusters with companies are very uneven. However, we needed to get as much information as possible about the relationships between companies, which is why the first approach was chosen.

After splitting companies, according to their financial indicators, each company was matched with three numbers from 1 to 3 (1 corresponds to the lowest values, three corresponds to the highest values, 2 denotes the average values), where the first number was its cluster number according to the $Return\ on\ Assets$ indicator, second number - according to the $Return\ on\ Equity$ variable, third number - according to the $Net\ income\ fraction$ parameter. The next task was to divide the companies based on all indicators. First, the priority of each indicator was determined. In particular, $Return\ on\ Assets$ received the priority, $Return\ on\ Equity$ - the second priority, and we also gave the third priority to the $Net\ income\ fraction$ variable. Then, we split companies into three groups, according to their assigned numbers. As there are 18 variants of number sets matched to our companies (3 variants for the first number, three variants for the second number, and two variants for the third number), each group included companies, which were matched with 6 of 18 variants. The first group incorporates companies with the worst financial condition, which means that variables of first and second priority correspond to 1 or 2 categories). These are the following numbers: $111,\ 112,\ 121,\ 122,\ 211,\ 212$. The second is the set of companies with the average level of financial well-being. They correspond to the following numbers: $131,\ 132,\ 221,\ 222,\ 311,\ 312$. Moreover, the third group represents companies with the best performance; the numbers assigned to them are $332,\ 322,\ 232,\ 331,\ 321,\ 231$.

Connection matrix. The clustering process described above was held for 2017, 2018, and 2019. In the result, we obtained the 3D-matrix of shape $(number\_of\_companies, number\_of\_companies, 3)$, where the third dimension is the number of the year, and the 2D-matrix of size $(number\_of\_companies, number\_of\_companies)$ is constructed as follows: if two companies are in the same cluster in the current year, then one is placed at their intersection in the matrix, otherwise 0. Before submitting our matrix to the model, we decided to average its values in the third dimension by removing the time axis. Thus, we have obtained a 2D matrix in which, in each cell, there is a coefficient of the connection strength between pairs of companies. In particular, if two companies have been in the same cluster for all three years, they have the most vital connection. Conversely, if the companies have been in different clusters for all three years, their connection is 0.

Associations from WikiData

Analysts acknowledge that there is an influence of companies related in some characteristics on each other's stock prices. For example, in belonging to the same industry or in the same form of ownership. For illustrative instance, in February 2022, Meta Platforms shares fell by 23% and pulled the shares of other social media operators, namely Pinterest, Snap and Twitter. Snap shares fell more than 18% after the close of trading, Pinterest fell nearly 10% and Twitter fell about 8% [1].

To account for these types of links we have accessed the WikiData and we have extracted the most important relations.

Literature basis. There are studies by Feng et al.; Kim et al.; Matsunaga, Suzumura, and Takahashi; Sawhney et al. that confirm the improvement of forecasts when certain relationships between companies are fed into the model. The relations are correspond to the set of connections taken by Feng et al. (2019). We took the relationships highlighted in the paper for a set of 61 companies.

JSON dump, https://www.wikidata.org/ For intelligent prediction, a Wikidata dump is taken as of 01/05/2018 and the corresponding time period for historical stock prices.

Data structure. But the authors take 1026 NASDAQ. First of all, we excluded companies not represented in WikiData. Then, we removed companies that did not have any of the selected links. So we took out all the companies with no ties. Thus, the average number of links per company increased from 3 to 12.7. Also, now every company in the dataset has an addition of information when feeding a matrix with selected links from WikiData into the model.

There are two types of the interconnections. The first type is direct. Direct ones have got typical WikiData relationship structure of (subject; predicate; object). Indirect relationships ... We take the same relations as Feng et al.. The most popular of which (table)

For 61 companies, we ended up with 777 ties. This gives a high density of data, namely about 13 ties per company, with 3 ties per company for 1026 companies. We obtained a three-dimensional matrix of 61 by 61 companies for 43 types of relationships. Afterward, we collapsed the matrix to a two-dimensional. As a result, 705 ties of different strengths were formed as a symmetric matrix with integer values from 1 to 4. The matrix contains the number of links between companies within a couple.

A very detailed description of the each property could be found on the https://www.wikidata.org/wiki/Wikidata:List_of_properties

Endogenous linkages

The endogenous approach is based on clustering companies using their historical time series. To solve this problem, we need to determine how we will calculate the distance between them. There are many ways to calculate the distance between time series, but most of them have a key drawback: the metric does not take into account the time component and the possible shift between the series. This means that if in one time series there is a sequence from another series, similar in structure, but located in a different time interval,

[1]https://www.forbes.ru/investicii/454461-akcii-meta-upali-bolee-cem-na-23-i-potanuli-za-soboj-bumagi-drugih-socsetej

WikiData relation	Quantity
P361_P361	462
P452_P452	98
P127_P127	72
P463_P463	42
P1056_P1056	18
P1056_P452	6
P452_P1056	6
P452_P31	2
P121_P121	2
P31_P452	2
P127_P3320	1
P31_P1056	1
P1056_P31	1
P400_P1056	1
P3320_P127	1
P1056_P400	1

Table 1: Quantity of the linkages from WikiData

the metric will consider these series to be different. Since it is important to find a general pattern of the behavior of a series for the task of determining a synthetic index, such metrics as Euclidean, Manhattan distance, etc. cannot be used. That is why in this article the metric of similarity of time series is DTW (dynamic time warping).

DTW. Dynamic Time Warping (DTW) is a time series alignment algorithm that allows measure the similarity between two sequences, considering the possible offset (time shift) between them.

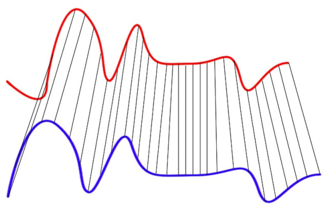

Figure 1: Dynamic time warping between two time series

The essence of the method is that two sequences are compared in pairs, and then aligned relative to each other. In the process of alignment, each point of one sequence correlates with a point of another sequence that is located at the closest distance in time, even if it differs significantly from their original position.

The DTW method works as follows:

1. First, the preliminary distance between two time series is calculated.
2. Then a distance matrix is constructed, where each element corresponds to the distance between a pair of points in the time series.
3. The optimal path with the minimum sufficient distance is calculated, points in two time series are compared.
4. The final distance is calculated as the sum of the distances between the points on the optimal path.

Relation lable	Name	Description
P1056	product or material produced or service provided	material or product produced by a government agency, business, industry, facility, or process
P121	item operated	equipment, installation or service operated by the subject
P127	owned by	owner of the subject
P31	instance of	that class of which this subject is a particular example and member
P3320	board member	member(s) of the board for the 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
P452	industry	specific industry of company or organization
P463	member of	organization, club or musical group to which the subject belongs

Table 2: WikiData linkages interpretation

The number of connections	0	1	2	3	4
Number of companies in the final matrix with the number of connections	3016	645	50	8	2

Table 3: Number of companies by number of WikiData connections

The general formula for calculating the DTW distance is as follows:

$DTW(x,y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i,y_j)^2}$,

where $\pi = [\pi_0, ..., \pi_K]$ - this is a path that satisfies a set of conditions:

- π - this is a list of paired indexes $\pi_k = (i_k, j_k)$, where $0 \leq i_k < n$ and $0 \leq j_k < m$, n is the length of the first time series, m is the length of the second;
- $\pi_0 = (0,0)$ and $\pi_K = (n-1, m-1)$ for every $k > 0$;
- $\pi_k = (i_k, j_k)$ and $\pi_{k-1} = (i_{k-1}, j_{k-1})$ satisfy the following inequalities: $i_{k-1} \leq i_k \leq i_{k-1} + 1$ and $j_{k-1} \leq j_k \leq j_{k-1} + 1$;

Time Series Clustering. Clustering is the process of grouping similar objects together to identify underlying patterns and structures. In general, time series clustering can be divided into 2 types:

• Feature based approach in which objects are clustered based on statistical data collected from a time series;

• Raw-data based approach in which clustering is applied to time series vectors without any transformations.

Both methods have proven themselves well when working with financial data, but the first approach is rarely used, because it does not take into account the hidden connections between time series.

Raw-data based approach

K-Means. K-Means is a clustering algorithm used to split a set of objects into a given number of clusters. It is based on minimizing the sum of squared distances between objects and cluster centers. To apply the algorithm to time series, you need to choose a metric by which the distances between the series will be estimated. Usually Euclidean distances or DTW are used.

After initializing the initial centroids of k clusters, the algorithm goes through the following steps:

1. Each time series refers to the nearest cluster by the distance to its centroid.
2. For each cluster, a new centroid is calculated as the average of all rows in this cluster.
3. Repeat steps 1 and 2 until the centroids stop changing or the maximum number of iterations is reached.

One of the main parameters of the algorithm is the choice of initial centroids. It is a good practice to use random points from a common dataset, as well as to conduct several runs with different initial centroids in order to more accurately determine the optimal number of clusters and avoid falling into the local optimum.

The limitations of K-Means include the need to specify a prefixed number of clusters, as well as sensitivity to outliers and heterogeneity of clusters in size. However, with the right choice of parameters and understanding of the features of the data, K-Means can be an effective tool for clustering time series.

Elbow method for determining the optimal number of clusters. The Elbow method algorithm is one of the main methods for determining the optimal number of clusters when clustering data. This method is based on estimating the variation of data within each cluster with a different number of clusters. The optimal number of clusters can be selected where the variation change begins to decrease significantly more slowly. The Elbow method algorithm consists of the following steps:

1. Determining the range of the number of clusters for which the analysis will be performed. This is usually a range from 1 to some maximum value.
2. Clustering data using a different number of clusters in a given range. For clustering, you can use any approach in which you can set the number of clusters.
3. Calculation of the intracluster total quadratic error (SSE) for each number of clusters. SSE is the sum of the squares of the distances between each object and the center of its cluster. The lower the SSE, the better the clustering.
4. Plotting a graph showing the number of clusters on the x axis and the corresponding SSE value on the y axis.
5. Visual analysis of the graph and determination of the ”elbow” on the graph, that is, the point after which the change in SSE becomes less noticeable. This place on the graph indicates the optimal number of clusters.

Determining the optimal number of clusters using the Elbow method requires some subjectivity, since the ”elbow” point may be unclear or ambiguous. In such cases, it may be necessary to use other methods to estimate the number of clusters or to make an additional assessment using external historical or subject information.

Hierarchical clustering. Hierarchical clustering is one of the most common time series clustering algorithms. This algorithm treats all time series as a set of points in n-dimensional space and builds a hierarchical cluster structure where each cluster consists of more similar data series.

The clustering algorithm can be divided into two stages:

1. Creating a distance matrix. The first stage is to create a matrix of distances between time series. The distance between time series can be determined in various ways.
2. Clustering. Each time series forms its own cluster. Then the algorithm finds the smallest distance between all possible pairs of clusters and combines them within a new cluster. This process continues until all time series are combined into a single cluster or until the number of clusters specified by the user is reached.

SOM. The basic idea of the SOM algorithm is that it creates a two-dimensional mapping of data vectors onto a grid of neurons, where each neuron is a multidimensional vector of weights. At the beginning of training, the weights of the neurons are randomly initialized.

Steps of the SOM algorithm:

1. The initial values of the weights of neurons are selected randomly or using other methods of initialization of the weights.
2. An input vector is randomly selected from the training dataset.
3. The distance between the input vector and the weights of neurons is calculated using a metric such as Euclidean distance or cosine distance.
4. The neuron with the weights closest to the input vector is selected as the winner.
5. The weights of the winning neuron and its neighbors are updated using a special update formula that requires taking into account the distance between the neurons and the winner.
6. Steps 2-5 are repeated for all input vectors in the training set.
7. Over time, the learning rate decreases, which allows the algorithm to eliminate the influence of random outliers and focus on the main data structures.
8. Training stops when a set number of epochs is reached or when a set stop condition is reached.

After training, SOM networks can be used for data visualization, cluster analysis, anomaly detection, and other tasks.

The SOM algorithm is a teacher-less method, which means that it does not require data markup and can detect hidden structures in the data. It also allows you to preserve the topological properties of the input data, which is one of its main advantages.

Feature based approach

VRAE. Variational Recurrent Autoencoder (VRAE) is a clustering algorithm that combines the ideas of recurrent neural networks (RNN) and variational autoencoders (VAE) to obtain a low—dimensional representation for each time series in an uncontrolled manner.

The VRAE model consists of two main components: an encoder and a decoder. The encoder takes an input time series and maps it to a hidden space where time dependencies and representations are fixed. The decoder then reconstructs the original time series from the representation of the hidden space. The VRAE encoder-decoder architecture is based on a long-term short-term memory (LSTM) network, which is a type of RNN designed to capture long-term dependencies.

In the process of learning, VRAE learns to match different time series with different areas of hidden space. This leads to the formation of cluster-like structures, where time series similar in their temporal behavior are located close to each other in a hidden space. This method is a subtype of the feature-based approach, because time series are not clustered directly, but their embeddings (representations in low-dimensional space) obtained using VRAE.

Visualization of clustering results To visualize the clustering results, 4 methods were used:

1. Rendering of all time series and their arithmetic mean on one graph;
2. Distribution of company sectors within the cluster in the form of a bar chart;
3. Distribution of companies by clusters in the form of a bar chart;
4. Distribution of companies by clusters on a two-dimensional graph using t-SNE.

T-SNE. The t-SNE (t-Distributed Stochastic Neighbor Embedding) algorithm can be applied to reduce the dimension of time series and their visualization. It is based on the idea of preserving similarity between objects and also applies a stochastic approach that avoids some problems, such as "sticking to local minima". The t-SNE algorithm consists of the following steps:

1. Calculation of pairwise similarities of objects in the original data space. In the case of time series, the DTW distance metric is used to measure the similarity between objects.
2. Initialization of a low-dimensional space randomly.
3. Calculation of conditional probabilities showing similarity between objects in the original data space and based on their representation in a low-dimensional space. To do this, a normal distribution is used, where similar objects have a high probability of being neighbors.
4. Determination of the probability distribution in a low-dimensional space and its comparison with the probability distribution in the original space. The distance between probability distributions is measured using the Kullback-Leibler divergence.
5. The use of gradient descent to optimize the positions of objects in low-dimensional space in such a way as to minimize the distance between probability distributions in the original and reduced spaces. Gradients are calculated based on the difference in conditional probabilities.
6. Repeat steps 3-5 until a stable structure is achieved.

Connection matrix In order to compare the effectiveness of different relationship matrices, it is necessary that all matrices have a single standard. These requirements were taken into account by me and my colleagues when creating the matrices:

- The matrix must be two—dimensional, the size of the matrix is N * N, where N = 61 is the number of companies.
- The matrix must be symmetrical.

In endogenous approach two ways of compiling the relationship matrix have implemented:

- Simple. if the companies fall into the same cluster, it means that there is a connection between the companies and there will be one in the corresponding cell of the table, in other cases — 0;
- Weighted. if the companies fall into the same cluster, it means that there is a connection between the companies and in the corresponding cell of the table there will be a DTW distance between the time series of these companies, in other cases — 0. The matrices obtained in this way will have a Weighted label in the final comparative table in the results section, distinguishing this method from a simple one.

Model

The study is devoted to the problem of binary classification of the trend of a time series. It is necessary to develop an effective classification algorithm with the help of a neural network that can determine the trend of a time series as increasing (class 1) or non-increasing (class 0). The developed model consists of three consecutive blocks:

- Sequential: getting embeds of companies;
- Relational: recalculation of embeddings taking into account the relationship matrix;
- Prediction: output of the predicted class.

Each block of the model was first developed as a separate neural network so that team members could work in parallel, and then everything was combined into one final model.

Data preprocessing To preprocess the data and create a dataset that meets the task, a CustomDataset class is created, the constructor of which receives a time series of historical stock prices of the companies we are considering (in the period from 2017 to 2019 inclusive, the series were downloaded from yahoo finance - a financial information provider owned by Yahoo!) and the value of the parameter (window_length + 1). The preprocessing of time series data consists of the following steps:

- From each time series, we take segments of the size equal to the parameter value passed to the dataset constructor. This is a moving window; that is, the values from 1 to window_length day will fall into the first segment, from 2 to window_length + 1 day will fall into the second segment, and so on until the end of the time series.
- Each last value in each window for each company is marked according to the following rule: if the difference between the last and penultimate values in the segment is positive, then the last value is assigned class 1; otherwise, class 0. We get for each company a set of segments of its time series of stock prices, in which the last value is the class number (target), and all other values are stock prices for which the model should predict the target, thereby solving the classification problem.
- In the matrix x, in order, we first put the first segments of the time series of each company, then the second segments, then the third, etc. In the matrix y, we put the corresponding target.
- Then we create an object of the CustomDataset class; the dataset length is equal to the number of time series segments of one company (the time series of all companies are equal in length), the dataset element is the matrices of the size number_of_companies * window_length and number_of_companies * 1.
- Then we divide our dataset into three parts – train (training sample), val (validation sample), and test (test sample), where train is the first 70% of the dataset length, val is the next 10% and test is the rest of the dataset, that is, when divided into parts, the chronological order of time series windows is preserved.
- The last step is wrapping each part of the dataset in an object of the DataLoader class, which divides our data into 64-bit chunks.
- The data is ready to be fed into the model

Sequential layer This part is formalized in a class called SequentialModel and represents the first part of the neural network that takes as input a matrix of sets of time series segments of all companies for a fixed interval of the time series - 20 days in the final version, i.e. a three-dimensional matrix with measurements: the number of sets of segments for a company, the number of companies, the length of the time series segment. The level consists of decomposition of time series into trend and seasonality. Where trend is considered to be a smooth price movement, and seasonality is considered to be fluctuations around the trend. Specially for this task the methods are implemented:

- MovingAvg - uses a sliding window to select a trend from the transmitted time series segments. The window size and shift are regulated through the parameters kernel_size (7 in the final version), stride (1 by default). To preserve the original length, the left and right rows are extended by half of kernel_size by elements equal to the first one for the beginning of the time series and the last one for the end of the time series.
- SeriesDecomp - receives time series segments as input, calls MovingAvg and extracts seasonality using the obtained trend.

Then for trend and seasonality separately recurrence layer, hyperbolic tangent as an activation function and normalization from torch.nn library are applied, then the trend and seasonality transformed in the described way are passed to the next, Relational part of the model, where transformation taking into account the interrelation matrices is applied, then it is the turn of Prediction part, which produces prediction using the transformed series.

Relational layer The model input is a matrix of relationships and embedding of the company's time series after passing through the SequentialModel model. RelationalModel model needs to update these embeddings so as to add information from the relationship matrix to them and keep their size.

Several neural network architecture designs have been invented for this layer.

Design 1: Naive approach. Let e_i be the embedding vector of i-the company for which we want to get (e_i) — the recalculated embedding vector, which takes into account the embeddings of related companies. The naive approach is to simply average such embeddings. To do this, you must first multiply all embeddings by their strength of connection with the current company, and then sum up and divide the resulting value by the number of non-zero connections of this company.

The final transformation of each embedding vector in this case looks like this:

$$\overline{e_i} = \sum_{j=0}^{N} \frac{a_{ij}}{d_i} e_j$$

where e_j is the embedding of the company that is associated with the i—th, d_i is the total number of companies that the i-th company is associated with (the strength of the connection is non—zero). The quality metrics of the prediction model obtained using this method will have the Naive label in the title in the final results table.

Design 2: Adding trainable weights (Explicit). The idea of this design is that the weights with which the embedding companies enter the final recalculated vector are selected by the neural network and recalculated throughout the training cycle. To do this, you need to apply a linear layer to the vector of relations of the i-th company. After that, as in the naive approach, we summarize and average the embedding values.

The general formula of transformation using a neural network in this case looks like this:

$$\overline{e_i} = \sum_{\{j \vee a_{ij} > 0\}} \frac{weight_j}{d_i} e_j$$

where $weight_j = \left[e_i * e_j * \varphi\left(w^T * a_i + b\right)\right]_j$ is the j-th element in the vector of weights and is the Softmax activation function.

The quality metrics of the prediction model obtained using this method will have the Explicit label in the title in the final results table.

Design 3: Adding complex trainable weights (Implicit). In this architecture of the embedding recalculation layer, a linear layer is created for the concatenated vector [e_i, e_j, a_{ij}]. It is assumed that this way the neural network itself additionally determines the connection between embeddings.

At the beginning, for the i-th embedding, we create a matrix of its concatenations with all other embeddings. Next, we apply a linear layer to this matrix to obtain a matrix of weights with which each values of the embedding vectors of related companies will enter (e_i). The resulting weights are then simply multiplied by the corresponding embedding and averaged, as described in the previous methods.

The general formula of transformation using a neural network in this case looks like this:

$$\overline{e_i} = \sum_{(\{j|a_{ij}>0\})} \frac{weight_j}{d_i} * e_j$$

where $weight_j = \left[\varphi\left(w^T * [e_{i,}e_j, a_{ij}] + b\right)\right]_j$ is the j-th row in the weight matrix, * is the element multiplication and is the Softmax activation function.

The advantage of this method is that it can take into account the time component of embedding due to the fact that it calculates its weight for each cell of the vector.

Prediction layer The layer responsible for the output of the model consists of two linear layers. The first layer receives an input matrix of the size batch_size * number_of_companies * hidden_size from the previous layer and returns a matrix of the size batch_size * number_of_companies * 50. The second layer receives a matrix from the first layer and returns a matrix of the size batch_size * number_of_companies * 1, in which the predicted probability of its belonging to class 1 (price increase) is stored for each company from each batch. In order for two linear layers not to be equivalent to one linear layer, it was necessary to add nonlinear transformations between them. Activation functions are designed for this purpose. The hyperbolic tangent was taken as the activation function in this layer. The activation function was selected experimentally depending on the Accuracy metric value of the test data. Also, before feeding to the second layer, a Dropout layer with the parameter p = 0.1 is applied to the input data, which turns off the neurons with probability 1 – p and, as a result, leaves them on with probability p. This layer is used to solve the problem of retraining, when the model explains well only examples from the training sample, losing the ability to generalize.

Final model At this stage, it was necessary to combine all the blocks in one model so that it would be possible to simultaneously adjust the optimizer to the parameters of all models, as well as simplify the type of training cycle that my colleague was developing.

The prototype for the final model was the device of modern autoencoders, which, like our model, consist of separate models, an encoder and a decoder.

The constructor creates objects of the Sequential Mode l, Relational Model, Prediction Model classes with parameters depending on the type of my layer: baseline or design number. In the redefined forward method, data is sequentially transferred from one block to another, eventually translating the original time series into the predicted class.

Training cycle The training cycle of the model consists of 30 iterations (epochs). Inside each iteration, there is an internal loop over all data butches (the size of the butch is 64). Binary cross-entropy was chosen as the loss function, a standard loss function for the binary classification problem. Adam is used as an optimization algorithm. Inside the batch loop, the optimization algorithm first zeroes the gradients. Then, the model is trained on the current batch. The model's conclusions pass through the sigmoid activation function, which returns a value from 0 to 1 – the probability of belonging to class 1 - and is most often used when solving the binary classification problem. If the model output is strictly greater than 0.5, then class 1 is assigned to this object; otherwise, class 0. After training the model on this batch, the loss function and metrics such as accuracy, precision, recall, and ROC-AUC are calculated using the model outputs and actual values. Then, the gradients of the loss function are calculated, taking into account the model's parameters (backward), and the optimization algorithm updates the model weights (step).

Results validation method To obtain statistically significant results, it is necessary to run the model several times with different initial randomization parameters. The eval_model method was developed for this purpose. It receives the following parameters as input:

- model_name (required parameter) — baseline or the name of the matrix according to the method of obtaining it;
- is_baseline (required parameter) — True if it is necessary to validate the baseline, False — otherwise;
- is_naive (default value False) — True if it is necessary to validate the naive method of recalculation of embeddings, False — otherwise;
- connection_matrix (default value None) — a matrix of relationships between companies corresponding to the type specified in model_name;
- relation_type (default value True) — True if it is necessary to validate Design 2, False — Design 3.

Inside the method there is a cycle of 5 iterations. At the beginning of the loop, the current randomization parameter for the torch and numpy libraries is set. Next, an object of the CommonModel class is created and the optimizer is updated. For each iteration of the loop , a file in the format is created .txt, which stores the entire learning process of the model. After training, the effectiveness of the model is checked on a test sample. The result is saved to the list. After the cycle, the metrics of all iterations are averaged and saved to a .csv file.

Results

Clustering based on financial statements results

Distribution of companies by clusters during clustering by the statistical method.

Year	Poor financial condition	Average financial condition	Good financial condition
2017	24	11	26
2018	22	16	23
2019	25	13	23

Table 4

The number of companies in the same cluster for all three years is 27 out of 61.

The composition of the matrix, based on the principle described in paragraph 6.3, is as follows:

- The number of pairs of companies that have never been in the same cluster is 1348
- The number of pairs of companies that were in the same cluster in one year out of three – 1214
- The number of pairs of companies that were in the same cluster in two out of three years is 726
- The number of pairs of companies that were in the same cluster in all years – 433

The results show that the relationship matrix with such clustering was informative.

Time series clustering result

In each clustering method, 7 clusters were obtained, the results are shown in the following pictures.

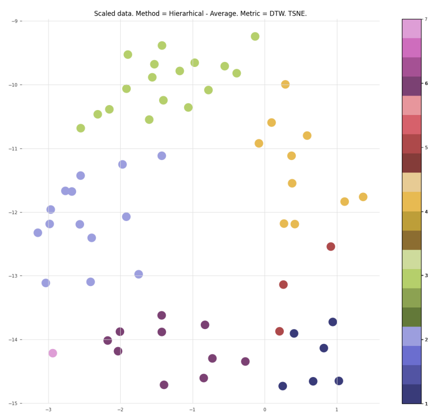

Figure 2: t-SNE for clustering using hierarchical clustering

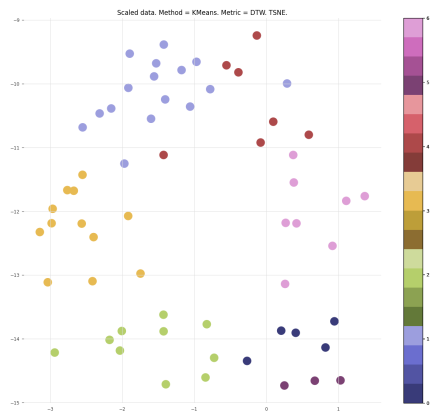

Figure 3: t-SNE for clustering using K-Means

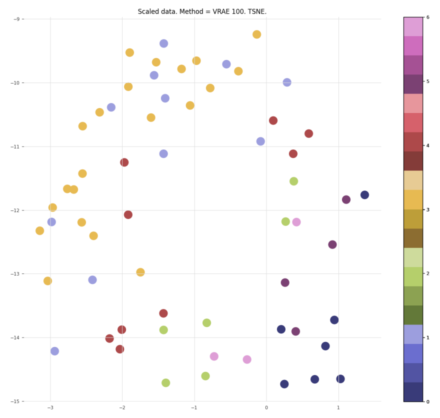

Figure 4: t-SNE for clustering using VRAE+K-Means

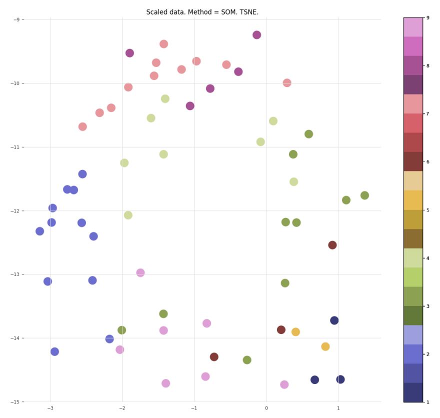

Figure 5: t-SNE for clustering using SOM

Model performance results

Table 5 contains metrics on the best approaches to using different types of relationships described in this report (financial reports, Wiki Data, similarity of time series). We see that according to Accuracy and F1-score metrics, the best option was to feed a matrix into the model containing information about the relationships between companies in accordance with their clustering by the KMeans method (modification of Weighted KMeans) with the recalculation of embeddings by the explicit method. In addition, according to these metrics, this approach is superior to Baseline. In accordance with the ROC-AUC metric, the Weighted VRAE Explicit approach (relationships based on time series clustering by the VRAE method) became the best, however, it also failed to surpass Baseline in this indicator.

Based on the results of the study, we concluded that in some cases, the quality of solving the classification problem for predicting time series can be slightly improved by taking into account information about the relationships between these time series or the objects whose characteristics they describe. However, since the quality of the model predictions turned out to be not much higher than the quality of the random generator predictions, it was suggested that perhaps the binary classification problem is not suitable for use in the field of time series forecasting and it is worth trying to consider other machine learning tasks.

References

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2): 1–30.

Herman, E.; Zsido, K.-E.; and Fenyves, V. 2022. Cluster Analysis with K-Mean versus K-Medoid in Financial Performance Evaluation. *Applied Sciences*, 12: 7985.

Huynh, T.; Nguyen, M.; Nguyen, T.; Nguyen, P. L.; Weidlich, M.; Nguyen, Q.; and Aberer, K. 2022. Efficient Integration of Multi-Order Dynamics and Internal Dynamics in Stock Movement Prediction.

Kim, R.; So, C. H.; Jeong, M.; Lee, S.; Kim, J.; and Kang, J. 2019. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999*.

Long, J.; Chen, Z.; He, W.; Wu, T.; and Ren, J. 2020. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. *Applied Soft Computing*, 91: 106205.

Matsunaga, D.; Suzumura, T.; and Takahashi, T. 2019. Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660*.

Sawhney, R.; Agarwal, S.; Wadhwa, A.; Derr, T.; and Shah, R. R. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 497–504.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2023. Attention Is All You Need. arXiv:1706.03762.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xiang, S.; Cheng, D.; Shang, C.; Zhang, Y.; and Liang, Y. 2022. Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction. In *Proceedings of the 31st ACM International Conference on Information Knowledge Management*. ACM.

Xu, C.; Huang, H.; Ying, X.; Gao, J.; Li, Z.; Zhang, P.; Xiao, J.; Zhang, J.; and Luo, J. 2022. HGNN: Hierarchical graph neural network for predicting the classification of price-limit-hitting stocks. *Information Sciences*, 607: 783–798.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2022. Are Transformers Effective for Time Series Forecasting?

Source of data	Loss	Accuracy	Recall	Precision	F1-score	ROC-AUC
Baseline	0.708396	51.742820	0.676162	0.544322	0.601381	0.503287
Reports (values from 0 to 3) naive	0.693108	53.616463	0.936069	0.541736	0.684722	0.499813
Reports (values from 0 to 3) explicit	0.712031	47.857582	0.220008	0.548373	0.309426	0.502057
Weighted KMeans Explicit	0.690742	54.111000	0.970330	0.542775	0.695877	0.502066
Weighted VRAE Explicit	0.701800	53.068653	0.775914	0.547534	0.635749	0.508668
Weighted Wiki naive	0.693278	53.009900	0.863075	0.543144	0.659260	0.500033
Weighted Wiki explicit	0.699389	52.210724	0.675385	0.547973	0.590339	0.508435

Table 5: Metrics when submitting different modifications of the relationship matrices of each type to the model