Learning Knowledge-Enriched Company Embeddings for Investment Management

Gary Ang
Singapore Management University
Singapore, Singapore
gary.ang.2019@phdcs.smu.edu.sg

Ee-Peng Lim
Singapore Management University
Singapore, Singapore
eplim@smu.edu.sg

ABSTRACT

Relationships between companies serve as key channels through which the effects of past stock price movements and news events propagate and influence future price movements. Such relationships can be implicitly found in knowledge bases or explicitly represented as knowledge graphs. In this paper, we propose Knowledge-Enriched Company Embedding (KECE), a novel multi-stage attention-based dynamic network embedding model combining multimodal information of companies with knowledge from Wikipedia and knowledge graph relationships from Wikidata to generate company entity embeddings that can be applied to a variety of downstream investment management tasks. Experiments on an extensive set of real-world stock prices and news datasets show that the proposed KECE model outperforms other state-of-the-art models on key investment management tasks.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Artificial intelligence**; **Knowledge representation and reasoning**.

KEYWORDS

Graph neural networks, transformers, attention mechanisms, time-series forecasting, networks, multimodality, embeddings, finance

ACM Reference Format:
Gary Ang and Ee-Peng Lim. 2021. Learning Knowledge-Enriched Company Embeddings for Investment Management. In *2nd ACM International Conference on AI in Finance (ICAIF'21), November 3–5, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3490354.3494390

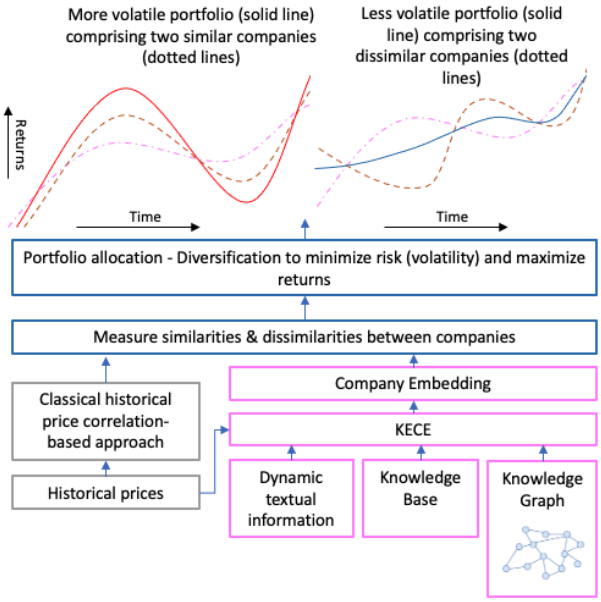

Figure 1: Overview of classical approach versus KECE

1 INTRODUCTION

Investments in stocks can generate substantial returns but also present significant risks. Forecasting of stock prices is hence an important task. However, forecasting individual stock prices is challenging as stock price time series are inherently noisy. Most methods for forecasting stock prices focus on individual stocks and do not capture underlying real-world relationships between companies that encapsulate valuable information on future stock price movements. For example, due to lead-lag effects, a drop in the stock price of one company could be a leading indicator of the future price movements of other related stocks. Relationships between companies serve as key channels through which the effects of past stock price movements and news events propagate and influence future price movements. E.g., a significant stock price movement of one supplier company could influence the stock prices of companies in the former's supply chain; an adverse news event affecting one company could also influence the stock prices of other companies in the same industry due to fears of similar events affecting these companies.

Stock price correlations (or co-variances) are usually used to measure such effects. Correlations provide an indication of the similarity and dissimilarity of companies based on stock prices, and are commonly used for financial tasks such as investment portfolio selection, allocation and risk measurement, where the focus is on managing an investment portfolio comprising a large number of stocks, with the objective of minimizing risk while maximizing returns. However, there are a number of limitations associated with correlations. Correlations may be spurious. They are unable to capture non-linear relationships. Correlations are also not designed to be applied directly to unstructured data such as textual news information.

ICAIF'21, November 3–5, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9148-1/21/11…$15.00
https://doi.org/10.1145/3490354.3494390

The availability of knowledge bases (KB) and graphs (KG) provides an alternative source of information on causal relationships between companies, e.g., parent-subsidiary, common product-type relationships, that can help address spurious correlations. Companies are present as entities within KBs such as Wikipedia, and relationships between companies can also be extracted from KGs such as Wikidata[43].

Instead of stock price correlations, we can instead use deep learning to generate latent representations or embeddings of companies to represent them in a high dimensional space that captures non-linear relationships among these companies and rich semantics of companies found in unstructured textual information(e.g., news).

The key idea in this paper is therefore to utilize knowledge from KBs and KGs together with textual information from financial news to enrich company entity embeddings for financial tasks. We focus on generating company embeddings that not only capture time varying multimodal data from different sources - numerical stock prices, textual news or blogs, but also incorporate knowledge from KBs and KGs. Such an approach, which injects stronger relational inductive biases within the model, will effectively guide model learning[12] and generate embeddings that can be used for many downstream tasks.

An overview of our proposed approach, compared against the classical historical price-correlation based approach, is depicted in Figure 1. Classical time series-based methods like the auto-regressive integrated moving average (ARIMA)[41] do not incorporate unstructured data nor company knowledge. Several recent works apply deep learning to financial time series forecasting but are designed for uni-variate and multi-variate numerical data only[13, 29, 36, 38]. [3, 4, 7, 16, 20, 47] apply NLP methods on unstructured textual input to perform financial forecasting, but they, together with the above-mentioned ones do not capture explicit causal relationships. Most of these works are also trained to make point predictions, and hence do not generate company embeddings for use in wide range of predictive tasks. They also fail to differentiate companies with different price dynamics and volatility, which is an important aspect to consider when measuring similarities and dissimilarities between companies. Hence, our proposed model KECE aims to offer several technical breakthroughs: i) jointly capture time-varying data with different modalities so that semantically rich dynamic embeddings can be learnt; ii) enable the propagation of information between companies based on underlying relationships to reflect real-world financial market dynamics; iii) enrich company embeddings with knowledge extracted from KBs and KGs to provide relational inductive bias to guide model learning; and iv) differentiate companies with different price dynamics and volatility. Our key contributions are as follows:

- To our knowledge, this is the first work to propose an approach that generates company embeddings that jointly captures time-varying structured and unstructured information from multiple modalities with knowledge from KBs and KGs.
- We propose a dynamic knowledge-based attention mechanism to dynamically weight the relevance of textual information (e.g. news, blogs) to companies across time.
- We propose a dual-stage attention-based model to encode and fuse time-varying multimodal features, before applying graph message passing to propagate company representations based on KG relationships.
- We propose a quantile regression training objective so that the model can differentiate between companies with different price dynamics and volatilities.
- Our experiments, which cover six datasets that are more extensive than most past works, show that KECE consistently out-performs state-of-the-art models on key investment management tasks - returns forecasting and investment portfolio allocation.

2 RELATED WORK

As this work involves financial time series forecasting, and the generation of network embeddings for company entities that are inter-connected in knowledge graphs, we review key related works in these areas.

Financial Time Series Forecasting ARIMA[41], which includes a variety of time series models such as autoregression (AR), moving average (MA), and autoregressive moving average models (ARMA), is a well studied classical method for time series forecasting, and has been commonly applied to financial time series. It is however not suitable for our multimodal and multi-variate setting. Classical multi-variate methods, such as VAR[30] which extends AR, cannot address our requirements as they are designed for numerical data but not unstructured data such as textual news articles. They are also unable to generate embeddings of company entities. Deep learning methods have been increasingly applied to time series forecasting. Deep feed-forward neural networks[36], convolutional neural networks[38] and recurrent neural networks[13, 29] have been applied to time series forecasting. A detailed review of these works can be found in [21, 28, 37, 40]. Many of these models are however designed for structured numerical inputs and uni-variate settings. Even when designed for multi-variate settings[8, 39], they do not capture causal real-world relationships between entities.

StockEmbed [4], a work that is closely related to ours, uses unstructured textual news articles to learn stock (or company) embeddings which are then used for investment portfolio allocation. Their approach is similar to other works that study the influence of news on stock prices[3, 7, 16, 20, 47] but differs from these works as StockEmbed focuses on capturing mutual effects between news articles and stock embeddings. It however does not capture causal relationships between stocks. Trained on a binary classification task (up/down price movements), StockEmbed does not distinguish between companies with different stock price dynamics and volatilities. Further, the work was only applied to a small sample of 140 companies using relatively old news datasets, whereas we conduct experiments that cover more than 2000 companies and use more recent news datasets.

Network Embeddings There are several related works on network embedding approaches which could be applied to a network of company entities. [23] applies a variational autoencoder (VAE) [22] framework to learn the node embeddings of networks. [33] uses two VAE channels to jointly encode and decode the node adjacency matrix and another node feature matrix. [34] extends [33] to co-embed both attributes and nodes of partially labelled

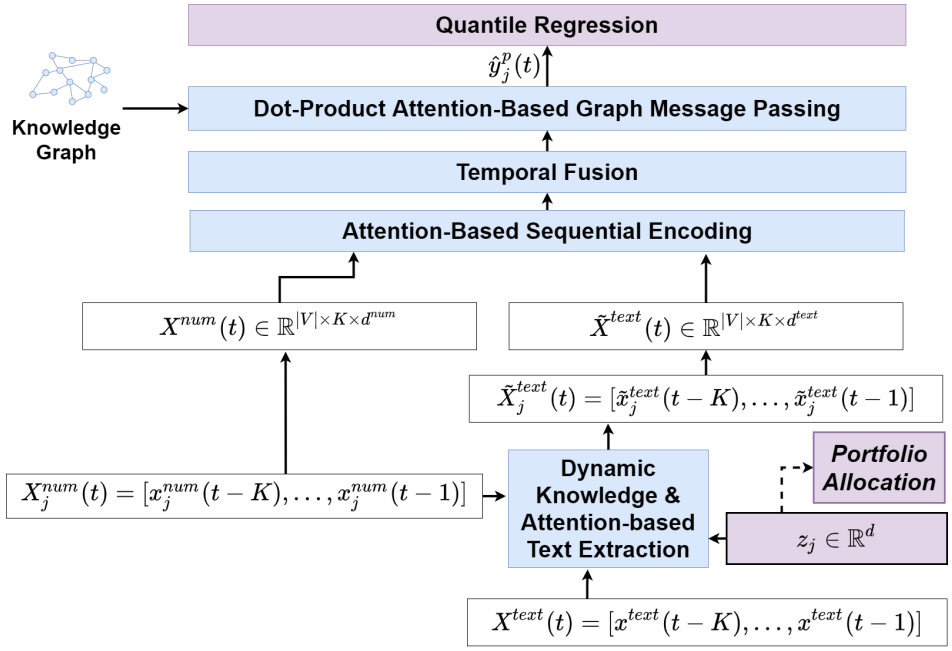

Figure 2: KECE architecture

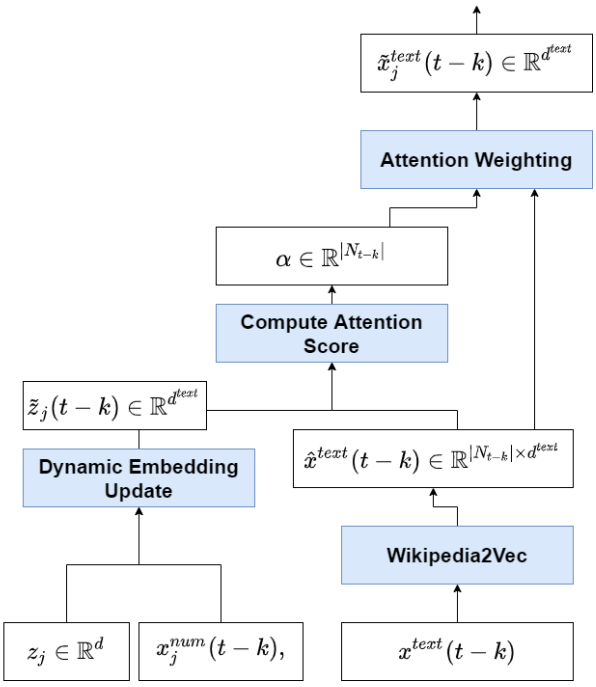

Figure 3: Dynamic Knowledge & Attention-based Text Extraction of step $(t-k)$ of a time window for company j.

networks. Multinomial VAE[26] is a VAE-based approach that generates embeddings of heterogeneous network using a multinomial distribution instead of the Bernoulli distribution used in [23]. Graph neural network (GNN) is another approach which composes messages based on network features, and propagates them to update the embeddings of nodes and/or edges over multiple neural network layers[12, 15]. Several GNN-based models have been developed. In particular, Graph Convolutional Network(GCN)[24] aggregates features of neighboring nodes and normalizes the aggregated representations by the node degrees. GraphSAGE[18] further considers mean, LSTM or pooling aggregation methods. Unlike GCN, GraphSAGE samples only a fixed number of neighbors for representation aggregation. Graph Attention Network(GAT)[42] assigns neighboring nodes with different importance weights during aggregation using additive attention. There are also GNN models designed for time varying networks or networks where the nodes have time varying attributes[1, 5, 9, 11, 17, 25, 45, 48]. However, these models are not designed for networks where the node attributes are multimodal financial time series. A few recent works[8, 14, 32] apply GNNs to prediction tasks on financial time series data, but these models were designed for attributes that are numerical only.

Among these works, Relational Stock Ranking(RSR)[14] is most closely related to ours as it captures causal relationships between companies using a graph-based model related to GCN. However, RSR is designed only for numerical information, and could not generate embeddings to capture unstructured textual information, or information from other modalities.

3 KNOWLEDGE-ENRICHED COMPANY EMBEDDING

Figure 2 depicts the architecture of our proposed Knowledge-Enriched Company Embedding (KECE) model. KECE represents companies in a network $G = (V, E, X)$, where V represents a set of company

nodes, E represents relationships between companies extracted from KGs, X represents sequences of multimodal attributes associated with companies. KGs such as Wikidata include real-world relationships E between companies V, e.g., company A *is-parent-company-of* company B, or company A *is-competitor-of* company B, which can be extracted to form the network G. More details on the KG extracted from Wikidata that we use for experiments in this paper are provided in Section 4.1. While we include only numerical stock prices and textual news data in $X = \{X^{num}, X^{text}\}$ in this paper, KECE can be further extended to incorporate data of other modalities in our future work.

Given a time step t, we define $X_j^{num}(t) = [x_j^{num}(t-K), ..., x_j^{num}(t-1)]$ to be the sequence of numerical price-related data associated with company v_j over a window of K time steps up to time step $t-1$. The matrix representing the numerical data of all company nodes at time step t with window size K is $X^{num}(t) \in \mathbb{R}^{|V|\times K\times d^{num}}$ where d^{num} is the embedding dimension size.

Similarly, we define $X^{text}(t) = [x^{text}(t-K), ..., x^{text}(t-1)]$ to be the sequence of textual news articles over the same window period. Initially, the textual news are not associated with any specific company, and the news articles are binned into each time step. KECE extracts from $X^{text}(t)$ the sequence of textual features at time step t of window size K relevant to company v_j as $\tilde{X}_j^{text}(t) = [\tilde{x}_j^{text}(t-K), ..., \tilde{x}_j^{text}(t-1)]$. We use $\tilde{X}^{text}(t) \in \mathbb{R}^{|V|\times K\times d^{text}}$ to encode the textual data for all company nodes where d^{text} is the embedding dimension size.

KECE then uses the structural KG relationship-based network information and multimodal information to learn company embeddings $Z \in \mathbb{R}^{|V|\times d}$, where d is the company embedding dimension size. We use quantile loss as the objective function to train KECE model. The goal is to learn Z that can perform prediction of stock returns at different percentiles, i.e., $\hat{y}(t)^p$ where $p = \{0.16, 0.5, 0.84\}$. Percentile 50% ($p = 0.5$) corresponds to median returns. Lower and upper percentiles of 16% ($p = 0.16$) and 84% ($p = 0.84$) are chosen based on the width (i.e. one standard deviation) of a normal distribution (68%), corresponding to the volatility of returns, a typical measure of risk in finance. More details on the tasks are provided in Section 4.2.

In the following, we elaborate on the processing steps in the KECE model.

Dynamic Knowledge & Attention-based Text Extraction. This step extracts textual features relevant to each company v_j for time step $t-k$ of window size K, $\tilde{X}_j^{text}(t) = [\tilde{x}_j(t-K), \cdots, \tilde{x}_j(t-1)]$. Each $\tilde{x}_j(t-k)$ is extracted following the steps shown in Figure 3.

For $1 \le k \le K$, the news articles at time step $t-k$ are combined. Each news article is first encoded using a pre-trained Wikipedia2Vec [10] model which represents both words and entities in Wikipedia in a common embedding space. We choose Wikipedia2Vec over other embedding models such as Word2Vec[35] or BERT[2] so that we are able to capture the rich knowledge present within the Wikipedia KB.

We initialize the company embedding at the beginning of training as z_j by the entity embeddings of the company v_j in Wikipedia2Vec (instead of random initialization). This strengthens the mutual effects between the textual news features and company embeddings so that more relevant textual features can be extracted.

As the embeddings of companies can drift across time, we perform the *dynamic embedding update step* to update z_j with numerical value $x_j^{num}(t-k)$, i.e.:

$$\tilde{z}_j(t-k) = LeakyReLU(Linear(z_j || x_j^{num}(t-k)) \tag{1}$$

where $Linear$ is a simple fully-connected linear layer.

We then use an attention mechanism to extract relevant textual features. We first encode all articles at time step $t-k$ as $\hat{x}^{text}(t-k) \in \mathbb{R}^{|N_{t-k}|\times d^{text}}$ where N_{t-k} denotes the set of news articles at time step $t-k$. Each news article is encoded by averaging the pre-trained Wikipedia2Vec word embeddings of the article's words. We then compute the inner product of the dynamically updated company embedding $\tilde{z_j}(t-k)$ with each encoded news article $n_i \in N_{t-k}$ to get the attention score: $score_{ij} = n_i \cdot \tilde{z_j}(t-k)$. The attention score is then normalized to obtain the attention weights: $\alpha_{ij} = \frac{exp(score_{ij})}{\sum_{n_i \in N_{t-k}} exp(score_{ij})}$, which are used to weight the textual features across the set of news articles for each time step $t-k$:

$$\tilde{x}_j^{text}(t-k) = \sum_{n_i \in N_{t-k}} \alpha_{ij} n_i \tag{2}$$

The output of this step across all company entities and the window period is hence $\tilde{X}^{text}(t) \in \mathbb{R}^{|V|\times K\times d^{text}}$.

Attention-Based Sequential Encoding. Next, we concatenate $\tilde{X}^{text}(t)$ and $X^{num}(t)$ to form multimodal time series sequence $X(t) \in \mathbb{R}^{|V|\times K\times (d^{num}+d^{text})}$. A self-attention mechanism based on the transformer [6] is chosen to encode the multimodal time series sequence as they have been shown to be more efficient and perform better than recurrent mechanisms[44]. We add fixed positional encodings $P \in \mathbb{R}^{K\times (d^{num}+d^{text})}$ as described in [6] and apply linear layers to generate queries, keys and values from the multimodal time series sequence - $Q_j(t) = Linear(X_j(t) + P)$, $K_j(t) = Linear(X_j(t) + P)$, $V_j(t) = Linear(X_j(t) + P)$. Learnt positional encodings can also be utilized, but we did not do so as, like [6], we did not see a significant improvement in task performance with learnt positional encodings. We apply scaled dot-product attention

$$X'_j(t) = softmax(\frac{Q_j(t)K_j(t)^T}{\sqrt{d^{num}+d^{text}}})V_j(t) \tag{3}$$

followed by a residual connection with layer normalization (LayerNorm), and finally a feed-forward network (FFN):

$$H_j(t) = FFN(LayerNorm(X'_j(t) + X_j(t)) \tag{4}$$

where $H_j(t) \in \mathbb{R}^{K\times (d^{num}+d^{text})}$.

Temporal Fusion. Representations at each time step are combined with temporal attention fusion, which weights the contributions of each time step based on its self-discovered importance. A non-linear transformation is applied to the representations to obtain a scalar $G_j(t-k)$ for each time step in the window where $k \in \{1, \cdots, K\}$: $G_j(t-k) = W^{(1)} tanh(W^{(0)} H_j(t-k) + b)$, where $W^{(0)}$ and $W^{(1)}$ are learnable weight matrices and b is the bias vector. We normalize each $G_j(t-k)$ to obtain the weights: $\beta_j(t-k) = \frac{exp(G_j(t-k))}{\sum_{k=1}^{K} exp(G_j(t-k))}$. We then fuse the representations of the time

series sequence:

$$H'_j(t) = \sum_{k=1}^{K} \beta_j(t-k) H_j(t-k) \quad (5)$$

where $H'_j(t) \in \mathbb{R}^{(d^{num}+d^{text})}$.

Dot-Product Attention-Based Graph Message Passing. For this step, we use a dot-product attention-based graph message passing module to propagate the fused company representations across Wikidata KG relations E. This module uses multiplicative attention and differs from GAT[42] which uses additive attention. We chose multiplicative attention as it is more efficient and demonstrated better performance. For each layer in the dot-product attention-based graph message passing module, we take the fused representation $H'_j(t)$ and apply a linear transformation $s_j^{(l)} = W^{(l)} H_j'^{(l-1)}(t)$ where $W^{(l)}$ is a learnable weight matrix of the l^{th} layer, and $H_j'^{(l-1)}(t)$ is the company representation from an earlier layer. We compute the pair-wise un-normalized attention score between company node v_j and each of its neighbors, say company node v_k with a dot-product operation: $e_{j,k}^{(l)} = s_j^{(l)} \cdot s_k^{(l)}$. Attention scores are computed to weight the hidden representations received by company node v_j from its neighboring nodes $N(v_j)$: $\alpha_{j,k}^{(l)} = \frac{exp(e_{j,k}^{(l)})}{\sum_{v_{k'} \in N(v_j)} exp(e_{j,k'}^{(l)})}$. The final step in each layer aggregates the hidden representations received by node j from its neighbors $N(v_j)$, weighted by the attention scores:

$$H_j'^{(l)}(t) = \sum_{v_{k'} \in N(v_j)} \alpha_{j,k'}^{(l)} s_{k'}^{(l)} \quad (6)$$

After two layers of message passing (representing the two-hop neighborhood), we represent the resultant hidden state for each company as $H''_j(t)$. More layers can be added to enable messages to be passed over more hops, but we find that this leads to over-smoothing[46] and negatively impacts performance. $H''_j(t)$ is then projected into one or more dimensions (corresponding to the number of quantiles q) with a single linear layer: $\hat{y}_j^p(t) = Linear(H''_j(t))$.

Quantile Regression KECE is trained by jointly minimizing the quantile loss:

$$\mathcal{L} = \sum_{p \in P} QuantileLoss(y_j^p(t), \hat{y}_j^p(t)) \quad (7)$$

where $p = \{0.16, 0.5, 0.84\}$; $QuantileLoss(y_j(t), \hat{y}_j(t), p) = p(y_j(t) - \hat{y}_j(t)) + (1-p)(\hat{y}_j(t) - y_j(t))$ per [27]; and $y_j(t)$ is the log return on day t, defined as $y_j(t) = \log price_j(t) - \log price_j(t-1)$. We use quantile loss for training so that the model is more robust to the variations in price dynamics and volatilities of different companies.

4 EXPERIMENTS

4.1 Datasets

To evaluate KECE and other models, we conduct experiments with three news datasets which differ significantly, either in the time span or the content as depicted in Table 1. Each dataset consists of both news articles and stock market price-information collected at daily time-steps over different time spans. The three news article sources are: i) older Reuters financial news article dataset (**RE1**) from [3][1] that has been used in a number of other works; ii) more recent Reuters financial news article dataset (**RE2**) that we extracted using the Newsfilter.io API; and iii) Investing.com news dataset (**INV**)[2], which contains news articles and commentaries from a much broader range of sources that include mainstream providers as well as blogs.

We also collected daily stock market price-related information - opening, closing, low & high prices, and trading volumes - of two separate stock markets - **NYSE** and **NASDAQ** - from the Center for Research in Security Prices. We include all companies in NYSE and NASDAQ with transactions in the respective time periods, and only filter out stocks that were not traded for an extended period, specifically more than 10% of trading days in the respective time periods.

For inter-company relationships, we use Wikidata, one of the largest and most active collaboratively constructed KGs. Companies such as Google, Apple and Microsoft are present within the Wikidata KG as entities, and relationships between them, e.g., Alphabet as a parent company of Google (first-order), both Apple and Microsoft are producing computer hardware (second-order), can be extracted from Wikidata. We extracted instances of 57 first and second-order relationship-types identified by [14] from the Wikidata dumps dated 15 Jan. 2018 and 7 Jan. 2019 and paired with the older RE1, and newer RE2/INV datasets respectively. The RE1 news dataset pre-dates the earliest Wikidata dumps[3]. We nevertheless adopt it following many previous works in financial forecasting, e.g., StockEmbed[4], one of the key baselines in our experiments. Notwithstanding this, the RE2-NYSE/NASDAQ and INV-NYSE/NASDAQ datasets are paired with the KG constructed from the Jan. 2019 Wikidata dump which is reasonably close to the end of the time spans. Furthermore, the coverage of the six datasets (RE1-NYSE, RE1-NASDAQ, RE2-NYSE, RE2-NASDAQ, INV-NYSE, INV-NASDAQ) used in this paper - over the time period 2006-2020, covering more than 2,000 companies and more than 300,000 articles in total - is more extensive than most existing works and provides strong assurance to our experiment findings.

4.2 Experiment Setup

Our experiments compare KECE with state-of-the-art baselines on two key investment management tasks:

- **Returns forecasting:** For this task, we predict returns for the percentiles $p = \{0.16, 0.5, 0.84\}$. The evaluation metric is quantile loss, as defined in Equation 7. We present results for the total quantile loss across all percentiles, as well as at individual percentiles. The datasets are divided into non-overlapping training/validation/test sets in the ratios 0.6/0.2/0.2[4].

[1]The original dataset also included news articles from Bloomberg, but these are no longer available for download.

[2]Subset extracted from https://www.kaggle.com/gennadiyr/us-equities-news-data

[3]The earliest Wikidata dumps were from 2014. However, we used Wikidata dumps from 15 Jan. 2018 and 7 Jan. 2019 as we found that knowledge graphs extracted from earlier Wikidata dumps were too sparse to be useful for our experiments, e.g. average knowledge graph node degrees for companies in RE1-NYSE is 0.3 for the Jan. 2016 Wikidata dump and 1.1 for the Jan. 2017 Wikidata dump, compared with 5.8 for the the Jan. 2018 Wikidata dump and 7.6 for the Jan. 2019 Wikidata dump.

[4]For the INV-NYSE and INV-NASDAQ datasets, we adjusted the starting date of the test sets to be after the 7 Jan 2019 date of the Wikidata dump used to construct the KG so datasets are divided in the ratios 0.6/0.22/0.18. No adjustment was made to

Table 1: Overview of datasets

	RE1-NYSE	RE1-NASDAQ	RE2-NYSE	RE2-NASDAQ	INV-NYSE	INV-NASDAQ
Time span	20 Oct. 2006 to 26 Nov. 2013		1 Jan. 2014 to 31 Dec. 2020		1 Jan. 2014 to 13 Feb. 2020	
No. of articles	87,064		66,779		198,718	
No. of companies (nodes)	1,235	723	1,385	854	1,433	893
No. of KG relationships (edges)	7,120	1,468	9,430	3,976	10,360	4,414
Average node degree	5.77	2.03	6.81	4.66	7.23	4.94

- **Investment portfolio allocation:** The aim of the investment portfolio allocation task is to optimize the proportion of capital invested in each stock in a portfolio, by finding an optimal set of weights $\mathbb{W}$ that minimizes portfolio risk for a given target return. The mean-variance risk minimization model by Markowitz [31] is formulated with the objective

$$min_{\mathbb{W}}\ risk = \mathbb{W}^T \Sigma \mathbb{W}$$

subject to $\mathbb{W}^T R^{hist} = E^{tgt}, \mathbb{W}^T \mathbf{1} = 1, 0 \leq \mathbb{W}^j \leq 1$, where R^{hist} is a vector of percentage stock returns over a selected historical period $\{1, \cdots, \tau\}$ and E^{tgt} is the targeted expected return of the portfolio. Σ is typically the historical co-variances of stock prices. We compute an alternative Σ for this task by taking the cosine distances between the company embeddings of companies v_j and v_k generated by each model, i.e.[5],

$$\Sigma'_{j,k} = cosine(z_j, z_k)$$

and use it to obtain the weights $\mathbb{W}$ via mean-variance risk minimization. This is a predictive task as we are using the resultant weights $\mathbb{W}$ to invest in stocks in year $\tau + 1$ in this paper; and then measuring the portfolio returns realized in this future period: $E^{real} = \mathbb{W}^T R^{real}$, where R^{real} is a vector of realized percentage stock returns over the selected future period. Given that the aim is to maximize portfolio returns while minimizing portfolio risk (volatility), we choose risk-adjusted realized portfolio returns over the selected future period as the evaluation metric, defined as: $\tilde{E} = \frac{E^{real}}{\sigma(E^{real})}$, where $\sigma(E^{real})$ is portfolio return volatility, defined as the one standard deviation of the portfolio returns over the same future period. Like [4], we conduct this evaluation with the walk-forward approach, i.e. we train the models using the first τ years of data and measure $\tilde{E}$ in year $\tau + 1$. We similarly perform the optimization over a range of targeted expected returns $E^{tgt} \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$, which cover the range of typical cases in real-world portfolio allocation (to cater to different investment preferences). The results shown are the averages for $\tau \in \{2, 3, 4, 5, 6\}$ for the RE1-NYSE/NASDAQ dataset; and $\tau \in \{2, 3, 4, 5\}$ for other datasets, decided based on the length of the time span covered by each of these datasets. We assume that there are 252 trading days in each year.

Baselines and settings. For both tasks, we compare KECE against LSTM and GRU models as baselines, as well as strong state-of-the-art baselines closest to our work as described in Section 2 - StockEmbed[4] and RSR[14]. StockEmbed[4] is chosen as a state-of-the-art baseline that captures textual information for stock forecasting. RSR[14] is chosen as a state-of-the-art baseline that captures causal relationships between stocks. For the investment portfolio allocation task, we additionally compare with the performance of the classical historical co-variance method. To train KECE, we chose the window period $K = 5$ days based on experiments with different window periods $K \in \{5, 10, 15, 20\}$. A longer window period did not lead to better performance, and differences in performance between KECE and baselines were generally consistent across all window periods. The window period $K = 5$ days corresponds to one trading week, and is also closest to the window period of 4 days used in StockEmbed[4]. A pre-trained Wikipedia2Vec[10] model which generates embeddings with a dimension of 100 is used. The dimensions of the company embeddings is also fixed at 100. An Adam optimizer with a learning rate of 0.001 with a cosine annealing scheduler is used. Models are implemented in Pytorch and trained for 100 epochs on a 3.60GHz AMD Ryzen 7 Windows desktop with NVIDIA RTX 3090 GPU and 64GB RAM.

4.3 Results

4.3.1 ***Returns forecasting.*** Tables 2-4 set out the results relating to returns forecasting. KECE outperforms all baselines on the totals and majority of the specific percentiles. We notice greater dispersion in performance on the percentiles $p = 0.16, 0.84$, as compared to $p = 0.50$, with KECE doing relatively better than the other baselines for these percentiles. This supports our point on the need to differentiate between companies with different price dynamics and volatilities, particularly when one is not just interested in making a point prediction, but capturing similarities and dissimilarities of companies. We also see slightly better performance for KECE on the RE1/RE2 news datasets relative to the baselines than on the INV news dataset, which could be due to the more diverse sources of textual news information in the INV news dataset. The diversity may have made it more challenging to capture the mutual effects between the textual information and company embeddings. Among the baselines, RSR generally does better, which indicates the importance of capturing relationships between companies.

4.3.2 ***Investment portfolio allocation.*** Figure 4 sets out the results relating to investment portfolio allocation. KECE consistently

RE1-NYSE and RE1-NASDAQ datasets as they pre-date the 15 Jan 2018 date of the Wikidata dump used to construct the corresponding KG. No adjustment was made to RE2-NYSE and RE2-NASDAQ datasets as the starting date of the test sets are already after the 7 Jan 2019 date of the Wikidata dump used to construct the KG.

[5] An important point to note is that Σ' needs to be positive semi-definite (PSD). One reason is that risk cannot be negative. Σ' computed from cosine distances between embeddings is not guaranteed to be PSD. Where the PSD condition is not met, we use Higham [19] to estimate the nearest PSD matrix.

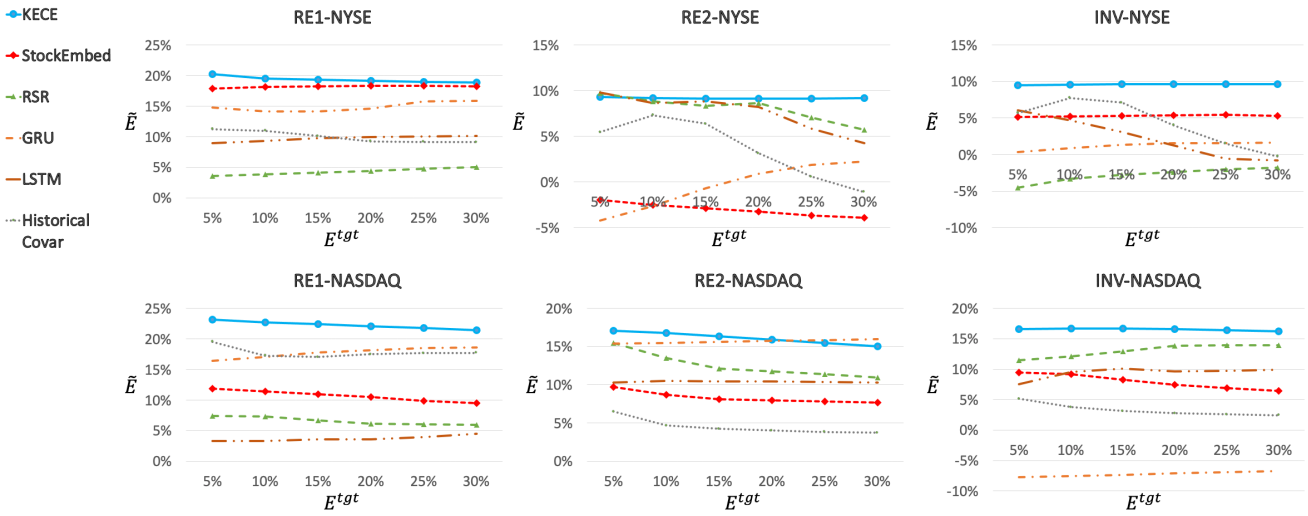

Figure 4: Investment portfolio allocation results. Vertical axis shows risk-adjusted realized portfolio returns $\tilde{E}$; horizontal axis shows targeted expected returns E^{tgt}. Higher $\tilde{E}$ is better.

Table 2: Returns forecasting results (RE1). Lower is better. Best performing model(s) in bold for totals; in italics for specific percentiles for this and subsequent tables.

	RE1-NYSE				RE1-NASDAQ			
Model	Total	p=0.16	p=0.50	p=0.84	Total	p=0.16	p=0.50	p=0.84
LSTM	0.0523	0.0187	*0.0189*	0.0147	0.0662	0.0233	*0.0236*	0.0193
GRU	0.0523	0.0187	*0.0189*	0.0147	0.0661	0.0233	*0.0236*	0.0192
RSR	0.0525	0.0187	*0.0189*	0.0149	0.0663	0.0233	*0.0236*	0.0194
StockEmbed	0.0535	0.0189	0.0191	0.0155	0.0664	0.0236	0.0237	*0.0191*
KECE	**0.0494**	*0.0160*	*0.0189*	*0.0145*	**0.0630**	*0.0203*	*0.0236*	*0.0191*

Table 3: Returns forecasting results (RE2)

	RE2-NYSE				RE2-NASDAQ			
Model	Total	p=0.16	p=0.50	p=0.84	Total	p=0.16	p=0.50	p=0.84
LSTM	0.1032	0.0359	0.0360	0.0313	0.1073	0.0371	0.0378	0.0324
GRU	0.1029	0.0359	*0.0359*	0.0311	0.1074	0.0371	0.0376	0.0327
RSR	0.1030	0.0359	0.0362	0.0309	0.1023	0.0371	0.0375	*0.0277*
StockEmbed	0.1034	0.0359	*0.0359*	0.0316	0.1065	0.0371	0.0373	0.0321
KECE	**0.0985**	*0.0318*	*0.0359*	*0.0308*	**0.1002**	*0.0320*	*0.0372*	0.0310

Table 4: Returns forecasting results (INV)

	INV-NYSE				INV-NASDAQ			
Model	Total	p=0.16	p=0.50	p=0.84	Total	p=0.16	p=0.50	p=0.84
LSTM	0.0644	0.0226	0.0231	0.0188	0.0743	0.0259	0.0266	0.0218
GRU	0.0640	0.0225	0.0230	*0.0184*	0.0737	0.0259	0.0265	0.0213
RSR	0.0630	0.0225	*0.0220*	0.0185	0.0726	0.0258	*0.0254*	0.0213
StockEmbed	0.0637	0.0225	0.0223	0.0188	0.0737	0.0260	0.0265	0.0213
KECE	**0.0603**	*0.0196*	0.0223	*0.0184*	**0.0689**	*0.0220*	0.0257	*0.0212*

outperforms all baselines across all datasets. The newer datasets - RE2-NYSE/NASDAQ and INV-NYSE/NASDAQ - appear to pose a greater challenge to some models. We see models such as StockEmbed (for RE2-NYSE) and RSR (for INV-NYSE) return negative risk-adjusted portfolio returns $\tilde{E}$ across all targeted expected returns E^{tgt}, which indicate that the embeddings generated by these models failed to capture similarities/dissimilarities between companies that hold in a future period, leading to sub-optimal weights $\mathbb{W}$. As such, performance on this task provides a clear indication of the quality of the company embeddings generated by KECE and the baseline models. The performance of StockEmbed and RSR were different for the RE2-NYSE and INV-NYSE datasets due to the market volatility subsequent to Feb. 2020 caused by Covid-19, which is captured in RE2-NYSE but not INV-NYSE. This and the general variability in performance of the baselines across the datasets indicates the sensitivity of these models to different data distributions, which KECE, because of the range of information it captures, is less affected by.

Table 5: Ablation Study (using RE1-NYSE). $\tilde{E}$ computed for $E^{tgt} = 0.05$ with testing dataset as selected future period.

	Returns Forecasting				Port. Allocation
Ablation	Total	p=0.16	p=0.50	p=0.84	$\tilde{E}$
w/o textual info.	0.0512	0.0170	0.0189	0.0153	18.4%
w/o network info.	0.0516	0.0187	0.0189	0.0140	18.9%
w ran. init. comp. embed.	0.0504	0.0170	0.0189	0.0145	22.3%
w/o dyn. update of comp embed.	0.0494	0.0160	0.0189	0.0145	22.9%
GAT for graph message-passing	0.0500	0.0159	0.0189	0.0152	22.4%
KECE	0.0494	0.0160	0.0189	0.0145	23.3%

4.4 Ablation Studies

Table 5 sets out the results of the ablation studies. We vary key aspects of the KECE model, namely: (a) excluding textual input and

related module (**w/o textual info.**); (b) excluding network input and related module (**w/o network info.**). We see that this leads to a significant drop in both the performance for returns forecasting as well as the investment portfolio returns, indicating the importance of effectively capturing these inputs. For KECE with random initialization of the company embeddings (**w rand. init. comp. embed.**) instead of using the entity embeddings in Wikipedia2Vec, we see that the drop in performance is less, but nonetheless material. Next, we evaluate KECE without the step of updating the initial company embeddings with numerical time series information before using it for the text extraction step (**w/o dyn. update of comp. embed.**). While there is no impact on returns forecasting performance, it does lead to a slightly lower risk-adjusted portfolio returns. We finally observe that KECE using the proposed dot-product attention-based graph message passing module performs better than the more commonly used GAT (**GAT for graph message-passing**).

4.5 Discussion

Based on the results of the experiments, we observe that being able to capture causal knowledge-based relationships significantly improves performance on investment returns forecasting and portfolio allocation tasks. KECE's performance across different percentiles p in returns forecasting, and across different targeted expected returns E^{tgt} for investment portfolio allocation is more consistent than the baseline models, which could be due to it being less affected by spurious correlations in stock prices since it captures causal knowledge-based relationships. We also see that textual news information, which contain valuable knowledge-based information, plays a key role in performance on these tasks. Greater variability in the sources of news information, e.g. RE1/RE2 vs. INV, can lead to differences in performance. This is likely due to differences in the style and nuances of textual information in mainstream news and blogs. Despite these differences, KECE is still able to perform well. Overall, we find evidence to support the usefulness of capturing multimodal information and enriching such information with knowledge from KBs and KGs to generate company embeddings for downstream investment management tasks.

5 CONCLUSION AND FUTURE WORK

The key insight demonstrated in this paper is that knowledge from KBs and KGs can be utilized to enrich multimodal information and generate embeddings of companies to improve performance on important downstream investment management tasks. We also showed that the proposed KECE outperforms recent state-of-the-art models consistently and by a significant margin in a number of instances. The datasets used in this paper are more extensive than most works and provide strong assurance on the validity of these results across different time periods, companies and types of textual information. Nonetheless, there is still potential scope for further work on the usefulness of this approach for other stock markets, other asset classes (e.g. interest rates and foreign exchange rates), other sources of textual information (e.g. from Reddit, Twitter or Facebook), and information from other modalities (e.g. visual news images).

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its Strategic Capabilities Research Centres Funding Initiative. Gary Ang is supported by a Monetary Authority of Singapore Postgraduate Scholarship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, nor the Monetary Authority of Singapore.

REFERENCES

[1] Jinyin Chen, Xuanheng Xu, Yangyang Wu, and Haibin Zheng. 2018. GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction. *CoRR* (2018).
[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
[3] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep Learning for Event-Driven Stock Prediction. In *IJCAI*.
[4] Xin Du and Kumiko Tanaka-Ishii. 2020. Stock Embeddings Acquired from News Articles and Price History, and an Application to Portfolio Optimization. In *ACL*.
[5] Aldo Pareja et al. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI Conference on Artificial Intelligence 2020*.
[6] Ashish Vaswani et al. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 2017*.
[7] Chen Deli et al. 2019. Incorporating Fine-grained Events in Stock Movement Prediction. In *Workshop on Economics and NLP*.
[8] Defu Cao et al. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. In *Advances in Neural Information Processing Systems*.
[9] Ehsan Hajiramezanali et al. 2019. Variational Graph Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 2019*.
[10] Ikuya Yamada et al. 2020. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. In *Conference on Empirical Methods in NLP (demo)*.
[11] Ling Zhao et al. 2020. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* 21, 9 (2020), 3848–3858.
[12] Peter W. Battaglia et al. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR* (2018).
[13] Yao Qin et al. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *International Joint Conference on Artificial Intelligence 2017*.
[14] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal Relational Ranking for Stock Prediction. *ACM Trans. Inf. Syst.* 37, 2 (2019), 27:1–27:30.
[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *CoRR* (2017).
[16] Paul Glasserman, Kriste Krstovski, Paul Laliberte, and Harry Mamaysky. 2020. Choosing News Topics to Explain Stock Market Returns. *ACM Int'l Conf. on AI in Finance* (2020).
[17] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR* (2018).
[18] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 2017*.
[19] Nicholas J. Higham. 1988. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra Appl.* 103 (1988), 103 – 118.
[20] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to Chaotic Whispers: A Deep Learning Framework for News-Oriented Stock Trend Prediction. In *ACM Int'l Conf. on Web Search and Data Mining*.
[21] Weiwei Jiang. 2020. Applications of deep learning in stock market prediction: recent progress. *CoRR* (2020).
[22] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations 2014*.
[23] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *NIPS Workshop on Bayesian Deep Learning*.
[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations 2017*.
[25] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations 2018*.
[26] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *World Wide Web Conference*.

[27] Bryan Lim, Sercan Ömer Arik, Nicolas Loeff, and Tomas Pfister. 2019. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *CoRR* (2019).

[28] Bryan Lim and Stefan Zohren. 2020. Time Series Forecasting With Deep Learning: A Survey. *CoRR* (2020).

[29] Yeqi Liu, Chuanyang Gong, Ling Yang, and Yingyi Chen. 2020. DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. *Expert Syst. Appl.* 143 (2020).

[30] Helmut Lütkepohl. 2011. *Vector Autoregressive Models.* Springer Berlin Heidelberg, Berlin, Heidelberg.

[31] Harry Markowitz. 1952. Portfolio Selection. *The Journal of Finance* 7, 1 (1952), 77–91.

[32] Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. 2019. Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window Analysis. *CoRR* (2019).

[33] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *ACM Int'l Conf. on Web Search and Data Mining*.

[34] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly Co-embedding Attributed Networks. In *Advances in neural information processing systems*.

[35] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Int'l Conf. on Learning Representations*.

[36] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Int'l Conf. on Learning Representations*.

[37] Serkan Özen, Volkan Atalay, and Adnan Yazici. 2019. Comparison of Predictive Models for Forecasting Time-series Data. In *Int'l Conf. on Big Data Research*.

[38] Leonardos Pantiskas, Kees Verstoep, and Henri E. Bal. 2020. Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks. In *IEEE Symposium Series on Computational Intelligence*.

[39] Youngjin Park, Deokjun Eom, Byoungki Seo, and Jaesik Choi. 2020. Improved Predictive Deep Temporal Neural Networks with Trend Filtering. *ACM Int'l Conf. on AI in Finance* (2020).

[40] José F. Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. 2021. Deep Learning for Time Series Forecasting: A Survey. *Big Data* 9, 1 (2021), 3–21.

[41] Granville Tunnicliffe Wilson. 2016. Time Series Analysis: Forecasting and Control. *Journal of Time Series Analysis* 37 (2016).

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *Int'l Conf. on Learning Representations* (2018).

[43] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* (2014).

[44] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. 2020. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *CoRR* (2020).

[45] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations 2020*.

[46] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *Int'l Conf. on Machine Learning*.

[47] L. Yang, Z. Zhang, S. Xiong, L. Wei, J. Ng, L. Xu, and R. Dong. 2018. Explainable Text-Driven Neural Network for Stock Prediction. In *IEEE Int'l Conf. on Cloud Computing and Intelligence Systems*.

[48] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Int'l Joint Conf. on Artificial Intelligence*.