Deep learning solutions to some prediction problems in finance

Qinkai Chen

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAX025

Thèse de doctorat

Deep Learning Solutions to Some Prediction Problems in Finance

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'Ecole Polytechnique

École doctorale n°574 Ecole Doctorale de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques Appliquées

Thèse présentée et soutenue à Palaiseau, le 15 Mars 2023, par

QINKAI CHEN

Composition du Jury :

Jean-David Fermanian Professeur, ENSAE-CREST, ENSAE Paris	Président
Guillaume Coqueret Professeur Associé, QUANT, EM Lyon Business School	Rapporteur
Mihai Cucuringu Associate Professor, Department of Statistics, University of Oxford	Rapporteur
Romuald Elie Professor, LAMA, Université Gustave Eiffel	Examinateur
Sophie Laruelle Professeur Assistant, LAMA, Université Paris-Est Créteil	Examinateur
Charles-Albert Lehalle Visiting Professor, Department of Mathematics, Imperial College London	Examinateur
Christian-Yann Robert Professeur, LFA, ENSAE Paris	Directeur de thèse
Mathieu Rosenbaum Professeur, CMAP, Ecole Polytechnique	Directeur de thèse

Acknowledgements

Words cannot express my gratitude to my PhD supervisors Christian-Yann Robert and Mathieu Rosenbaum for their invaluable patience and feedback during the last three years. They were always ready to help me generously with their knowledge and expertise whenever I needed. They inspired me to become an independent researcher and demonstrated what brilliant and hard-working scientists can achieve. I could not have undertaken this journey without them.

My sincere thanks must also go to Professors Guillaume Coqueret and Mihai Cucuringu for accepting to be the referees of my thesis, as well as Professors Romuald Elie, Jean-David Fermanian, Sophie Laruelle and Charles-Albert Lehalle for being part of my thesis exam committee. Their comments and constructive suggestions broaden my view on the structure and the content of my thesis.

I am also grateful to my collaborators: Haoyang Cao, Mohamed El-Mennaoui, Antoine Fosset and Amine Rebei who supported me and allowed me to have different points of views during the research. My thank continues to Jianfei Zhang, with whom I spent a significant amount of time at ExodusPoint Paris office. His technical expertise and insightful comments on different research subjects helped me progress when I encounter difficulties.

There is no way to express how much it meant to have been a member of CMAP at Ecole Polytechnique and ExodusPoint Capital Management. These brilliant friends and colleagues helped and inspired me over the last three years: Quentin, Yuyang, Xiaofei, Alexandre, Thanh-Niem, Pierre-Jean, Bruno, François, Benoît, Johan, Mukunda, Jean-Sébastien, Mehdi, Nicolas, Johathan, Arthur and Thomas. Special thanks must go to Stéphane André who acted as the head of the team and always trusted me without reservation, and Marouane Anane who acted as my industrial supervisor and gave me abundant advice on different subjects.

Finally, I most deeply thank my parents, my family and my friends for their unconditional support.

Résumé

Cette thèse se compose de trois parties. Les deux premières parties examinent respectivement deux problèmes de prédiction en finance : la prédiction du rendement des actions et la prédiction de la volatilité réalisée à court terme. La troisième partie traite d'un problème connexe : la découverte de connexions d'artistes contemporains en utilisant les connaissances décrites dans les deux premières parties.

Dans la première partie, nous présentons une solution univariée et une solution multivariée pour le problème de la prédiction du rendement des actions avec les nouvelles financières. Nous introduisons d'abord une procédure de prédiction univariée qui prédit le rendement court terme d'une action après la publication d'une nouvelle qui l'associe (Chapitre 2). Dans cette procédure, nous appliquons d'abord une méthode d'apprentissage par transfert pour générer les embeddings contextualisés des mots dans le titre d'une nouvelle. Nous utilisons ensuite un réseau neurones récurrent pour faire la prédiction à partir des embeddings générés. Avec les expériences extensives qui comprennent les tests de précision et les simulations de trading, il est démontré que cette approche possède une meilleure performance par rapport aux autres modèles de référence. Nous étendons ensuite notre approche univariée à un modèle multivarié (Chapitre 3), dans lequel une nouvelle peut non seulement impacter la cour d'une action mais aussi toutes les autres actions associées par les relations venant de différentes sources. Nous modélisons cet effet de transmission à l'aide d'une structure innovative de réseau neurones convolutif en multi-graphe. Nous démontrons l'efficacité de ce modèle avec les expériences similaires à celles dans la première étude.

Dans la deuxième partie de cette thèse, nous nous intéressons à la prédiction de la volatilité réalisée à court terme à partir des carnets d'ordres à cours limité à l'aide d'un modèle multivarié (Chapitre 4). Pour atteindre cet objectif, nous concevons un réseau neuronal graphique contenant à la fois des relations temporelles et transversales. Des opérateurs de transformation graphique sont intégrés au modèle pour une meilleure précision et une meilleure efficacité de calcul sur ce grand graphe. Grâce à des expériences basées sur plus de cinq cents actions, nous démontrons qu'une approche multivariée basée sur un graphe a un meilleur pouvoir prédictif que les lignes de base univariées couramment utilisées. En particulier, notre gain de performance provient principalement des actions moins liquides avec moins d'informations dans leurs carnets d'ordres limités, ce qui correspond à l'intuition qu'un nœud peut bénéficier des informations de ses voisins dans un modèle de graphe lorsqu'il n'y a pas d'informations suffisantes sur lui-même.

Afin d'illustrer les champs étendus d'application des méthodes de traitement de langage naturel, nous nous intéressons à l'art contemporain. Dans la troisième partie, nous présentons leurs applications dans le domaine de l'art contemporain, où nous nous intéressons à l'identification des relations entre les artistes à travers leurs biographies (Chapitre 5). Pour cette tâche, nous

présentons un cadre générique de traitement de langage naturel, dans lequel nous continuons tout d'abord à pré-entraîner un modèle de langue anglaise générale existant avec une grande quantité de textes non étiquetés liés à l'art. Nous affinons ensuite ce nouveau modèle pré-entraîné avec des paires de biographies étiquetées. Grâce à des expériences approfondies, nous démontrons que notre approche atteint une précision de plus de 85% dans l'identification du lien entre deux artistes et qu'elle surpasse d'autres modèles de référence. Nous visualisons et analysons également quantitativement une œuvre d'artiste construite à partir des résultats de notre modèle.

Enfin, nous concluons la thèse par une vue d'ensemble des conclusions de toutes les recherches et fournissons plusieurs pistes possibles pour des recherches ultérieures, en particulier les applications du traitement du langage naturel et du réseau neuronal graphique dans la finance.

Mots clés: Prédiction de rendement des actions, traitement de langage naturel, apprentissage profond, trading algorithmique, réseau neurones en graphe, modèle multi-graphe, prédiction de la volatilité réalisée, carnet d'ordre, transformeur en graphe, art contemporain.

Abstract

This thesis consists of three parts. The first two parts examen respectively two prediction problems in finance: stock return prediction and short-term realized volatility prediction. The third part discusses a related issue: contemporary artist connection discovery using the knowledge described in the first two parts.

In the first part, we present a univariate and a multivariate deep learning solution to the problem of stock return prediction with financial news. We first introduce a univariate prediction procedure that predicts the short-term return of a stock after the publication of news associated with this stock (Chapter 2). In this procedure, we first use a transfer learning based method to generate contextualized embeddings of the words in a news' headline, a recurrent neural network is then used to make predictions from the generated embeddings. Through extensive experiments, including accuracy tests and trading simulations, we show that this approach outperforms other baseline models. We then extend our univariate approach to a multivariate model (Chapter 3), in which a single news can not only impact one stock but also all other related stocks. Through an innovative multi-graph convolutional network structure, we can model the information transmission process from one stock to others based on stock relationships built from different sources. We demonstrate the effectiveness of this approach with a similar experiment setup as the first study.

In the second part of this thesis, we are interested in predicting short-term realized volatility from limit order books with a multivariate model (Chapter 4). To achieve this goal, we design a graph neural network containing both temporal and cross-sectional relations. Graph transformer operators are integrated into the model for better accuracy and computing efficiency on this large graph. Through experiments based on more than 500 stocks, we demonstrate that a graph-based multivariate approach has better predictive power than commonly used univariate baselines. In particular, our performance gain mostly comes from less liquid stocks with less information in their limit order books, which corresponds to the intuition that a node can benefit from its neighbors' information in a graph model when there is no sufficient information on itself.

The Natural Language Processing (NLP) and transfer learning techniques that we studied in the first part can go beyond the finance. In the third part, we introduce their applications in contemporary art, in which we are interested in identifying the relations among the artists through their biographies (Chapter 5). For this task, we present a generic NLP framework, in which we first continue to pre-train an existing general English language model with a large amount of unlabeled art-related texts. We then fine-tune this new pre-trained model with labeled biography pairs. With extensive experiments, we demonstrate that our approach achieves more than 85% accuracy in identifying the connection between two artists and outperforms other baseline models. We also visualize and analyze quantitatively an artist work built from our

model outputs.

Keywords: Stock movement prediction, natural language processing, deep learning, transfer learning, algorithmic trading, graph neural network, multi-graph model, realized volatility prediction, limit order book, graph transformer, contemporary art.

List of papers being part of this thesis

- Q. Chen, *Stock Movement Prediction with Contextualized Embedding from BERT*, preprint, 2021
- Q. Chen and CY. Robert, *Graph-based learning for stock movement prediction with textual and relational data*, published in The Journal of Financial Data Science, 2022
- Q. Chen and CY. Robert, *Multivariate Realized Volatility Forecasting with Graph Neural Network*, published in The Proceedings of the Third ACM International Conference on AI in Finance, 2022
- Q. Chen, M. El-Mennaoui, A. Fosset, A. Rebei, H. Cao, P. Bouscasse, CE. O'Beirne, S. Shevchenko and M. Rosenbaum, *Towards mapping the contemporary art world with ArtLM: an art-specific NLP model*, submitted, 2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Financial markets are highly uncertain. However, the investors in financial markets seek to gain an advantage by predicting the market movement in advance, and this challenging topic has attracted much attention.

ExodusPoint Capital Management is an active asset manager seeking to gain extra profits for its clients by predicting the market in advance. This includes the directional stock return in the equity trading business and the non-directional volatility in the options trading business. Hence, to apply the rapidly developing deep learning methods to business needs, this thesis explores different deep learning methods to predict stock return and volatility from various data sources.

Fama [1965] shows that the movement of a stock at a given moment can be explained by all previously available information. However, it is difficult to collect and use this wide range of data. Hence, researchers usually use certain part of the information to capture a fraction of the market movement. Traditionally, the practitioners predict the market with easily accessible data such as historical prices and company fundamentals. With the development of machine learning, especially Natural Language Processing (NLP) and Graph Neural Networks (GNN), it is possible to handle more complex information such as news and stock relations.

This chapter introduces two prediction problems in finance examined in this thesis: stock return prediction with financial news and realized volatility prediction with limit order book. It also introduces a related artist connection classification problem, which involves some NLP methods applied in the first stock return prediction problem. This chapter is organized as follows:

In Section 1.1, we introduce the first topic: stock return prediction with financial news, and in Section 1.2, we introduce the second topic: realized volatility prediction with limit order book data. In each topic, we first give the context, the data we use and the problem definition. For the first topic, we introduce a univariate and multivariate methods in Sections 1.1.2 and 1.1.3, respectively. We then present a multivariate solution to the second topic in Section 1.2.2. In addition, we present a generic NLP framework to discover the connections among contemporary artists based on their biographies in Section 1.3, which uses a similar transfer learning scheme applied in the first topic. In all four studies, we give a more detailed problem definition in each context, the main challenges we face, our solutions to the challenges, the main innovations and a summarized result.

The details of all four studies are presented in Chapters 2, 3, 4 and 5.

We also add an overview of NLP and GNN in Appendix 2.B and 3.A for the readers who do not have prior knowledge in these fields.

1.1 Part I: Stock Return Prediction with Financial News

1.1.1 Context and Objective

Forecasting the stock movement has always been an interesting topic and this is particularly true for ExodusPoint Capital Management, an active asset manager. We seek to predict the stock movement in the future from various data sources to make a profit for investors.

There are various approaches based on different data in the literature. For example, econometricians use time series analysis to predict stock prices in the future based on historical prices and trading volumes [Mills and Mills, 1990; Chen et al., 2007; Mondal et al., 2014]. Financial analysts use company fundamentals [Chan et al., 1993; Wang and Xu, 2004; Ozlen, 2014] and macroeconomic data [Narayan et al., 2014; Hoseinzade and Haratizadeh, 2019] to achieve this goal. More recently, with the development of the natural language processing, we can handle more unstructured information such as financial news [Ding et al., 2014,0; Hu et al., 2018; Xu and Cohen, 2018].

We can also categorize this stock movement prediction task by univariate or multivariate methods. The univariate method is the most commonly used in this task, meaning we make predictions stock by stock while ignoring the connections among stocks. However, the multivariate approach further considers the stock relationships from various data sources to better model this signal transmission process. With the development of graph methods in deep learning, we can handle this relational information more efficiently. There are multiple multivariate methods applied in this task [Mehtab and Sen, 2020; Lof, 2012].

With the rapid development of the Internet, we observe significant growth in the number of texts available. Unlike the traditional price and volume information, the textual data remains largely unexplored. Hence, we want to focus on predicting the stock movement from the news data with the help of the most recent deep learning methods. At first, we aim to predict the short-term stock movement after a news (around 30 minutes for the news in trading hours and one day for the news out of trading hours) in a univariate approach. We then focus on predicting daily stock movement in a multivariate approach from the news and the relational data from other sources such as the activity sector and the supply chain.

The financial news we use is Bloomberg News, a news data source widely used in the financial industry. We show an example of this dataset in Table 1.1. The dataset includes the *Headline* of the news and the *TimeStamp* and *Ticker* associated with this news. In addition, Bloomberg also provides their sentiment analysis with *Score*, which is a value among -1, 0 and 1, denoting negative, neutral and positive sentiment, respectively. *Confidence* is a value between 0 and 100 which gives the confidence level of the *Score*. We use Bloomberg's analysis as one of our benchmarks.

We formulate this return prediction problem as follows.

Table 1.1: A small sample from the Bloomberg News dataset

Headline	TimeStamp	Ticker	Score	Confidence
1st Source Corp: 06/20/2015 - 1st Source announces the promotion of Kim Richardson in St. Joseph	2015-06-20 05:02:04.063	SRCE	-1	39
Siasat Daily: Microsoft continues rebranding of Nokia Priority stores in India opens one in Chennai	2015-06-20 05:14:01.096	MSFT	1	98
Rosneft, Eurochem to cooperate on monetization at east urengoy	2015-06-20 08:01:53.625	ROSN RM	0	98

Given a stock s, we define its market adjusted return r_s between t and $t + \Delta t$ as:

$$r_{s,t} = \frac{P_{s,t+\Delta t}}{P_{s,t}} - \frac{P_{m,t+\Delta t}}{P_{m,t}} \tag{1.1}$$

where $P_{s,t}$ denotes the price for stock s at time t, and $P_{m,t}$ denotes the market index value at time t. The market adjusted return removes the impact from the market index and better reflects the impact of the news. We do not use beta to adjust market return as proposed in the Capital Asset Pricing Model (CAPM) [Black et al., 1972], since Fama and French [2004] demonstrate that the beta can over-estimate the impact for high-beta stocks and under-estimate the impact for low-beta stocks.

We define the target of our stock movement prediction task for stock s between t and $t + \Delta t$ as:

$$Y_{s,t} = \begin{cases} 1, & r_{s,t} > 0 \\ 0, & r_{s,t} \leq 0 \end{cases}. \tag{1.2}$$

The return prediction task can be written as:

$$\widehat{Y}_{s,t} = f(E_t, \theta) \tag{1.3}$$

where f denotes the prediction model, E_t is all the information including the news available before t, θ represents all the trainable parameters and $\widehat{Y}_{s,t}$ is the prediction given by the model.

We first introduce a univariate solution in Section 1.1.2, which forecasts the short-term return for one stock after the news publication. We then further propose a multivariate solution in Section 1.1.3, which forecasts the daily returns for all stocks based on all the news in one day and the stock relationships built from different data sources.

1.1.2 Chapter 2: Stock Movement Prediction with Contextualized Embedding

This subsection introduces the first subject of this thesis. The detailed study is in Chapter 2.

Motivation and main challenges

In this subsection, our goal is to predict the stock movement from a news headline after publication. For the news recorded in the trading hours[1], the forecast horizon is 30 minutes. If the news is published outside trading hours, we forecast its return between the previous market close and the next market close. Our solution is univariate, meaning that we suppose the news can only impact the price of the associated the stock and this information does not transmit to other stocks. Equation 1.3 can therefore be refined as:

$$\widehat{Y}_{s,t} = f(E^0_{s,t}, \theta) \tag{1.4}$$

where $E^0_{s,t}$ denotes one piece of news about the stock s recorded at time t.

The first step in this task is to convert textual data into numerical data. In NLP, we can convert a word into a fixed-length dense vector called embedding (Appendix 2.B.2). There are three main challenges in this process:

1. We need a large number of corpus to extract the features in a language.
2. The embeddings trained with a large corpus do not have domain-specific knowledge. For example, some words may have specific meanings in finance compared with general English.
3. The embedding is static. It means that the vector representation of a word is the same in different sentences, even though their meanings can be significantly different depending on the context.

To solve the first challenge, researchers usually use a large amount of English corpus to train a general-purpose model for other downstream tasks. For example, Mikolov et al. [2013b] propose Word2Vec and Pennington et al. [2014] propose GloVe, both of which are trained on a vast corpus containing billions of tokens.

To solve the second challenge, Transfer Learning (Appendix 2.B.4) has recently become popular. The idea is to train a general English model with a large corpus, then continue the training process based on domain-specific corpus. Recently, several large-scale transfer learning models have been proposed, including Embedding from Language Model (ELMo) [Peters et al., 2018a], Bidirectional Encoder Representations from Transformers (BERT, Appendix 2.B.4) [Devlin et al., 2018] and XLNet [Yang et al., 2019]. The transfer learning approach has been proven to be robust and widely used in most NLP tasks.

To solve the last challenge, instead of using a pre-defined fixed word embedding from Word2Vec model, we input all the words together into an interconnected model, such as the aforementioned BERT model. In this case, a word embedding will be impacted by the embeddings of the words in the same sentence. We can then obtain a contextualized embedding from the hidden layers of this model.

Existing literature usually only partially solves these problems. For example, Ding et al. [2015] propose one of the first deep learning solutions to this problem while heavily relying on static embeddings generated from Word2Vec (points 2 and 3). Devlin et al. [2018] propose a classification method along with BERT but ignore the domain-specific knowledge (point 2).

[1] from 9:00 to 17:30 CEST for most of the European markets.

Vargas et al. [2017] train a Word2Vec model only with financial texts. However, this method cannot well train a model with the characteristics of general English, and the embeddings are still static (points 1 and 3).

A stock movement prediction procedure with contextualized embeddings

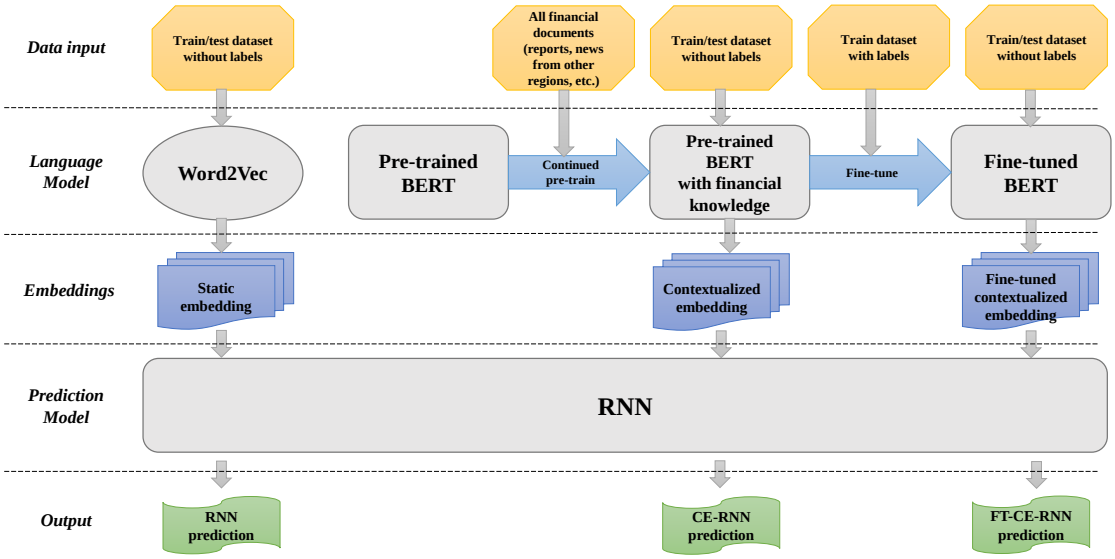


Figure 1.1: An illustration of the stock movement prediction procedure: FT-CE-RNN.

To consider all the factors above and close the gap in the research, we propose a prediction procedure called FT-CE-RNN[2] (Figure 1.1, details introduced in Section 2.5). We first continue the pre-training process of the BERT model with a broader range of financial texts to add financial knowledge to the model. We then fine-tune the model with labeled training data and extract the contextualized embeddings from the hidden layers of the fine-tuned model. We finally train a Recurrent Neural Network (RNN, Appendix 2.B.3) with the generated contextualized embeddings. In addition, to better incorporate the investors' needs, instead of evaluating the model's performance based on all the news, we propose a new evaluation metric based only on extreme news (Section 2.6.2).

With extensive experiments such as accuracy tests and trading simulations on about 600 stocks from the STOXX Europe 600 index, we prove that the FT-CE-RNN outperforms other baseline models in the literature. A summarized accuracy result is shown in Table 1.2, and the full results are shown in Section 2.6. We can see that the FT-CE-RNN outperforms the BERT model by 1.3% and the Bloomberg score by 9.5% in accuracy based on the 2% most extreme news. We also demonstrate that the contextualized embeddings from a fine-tuned model give a better prediction result than the static and non-fine-tuned embeddings. For example, using the fine-tuned contextualized embeddings records a 4.4% accuracy gain based on the 2% most extreme news compared with static embeddings and 4.0% accuracy gain compared with non-fine-tuned embeddings in the same situation.

[2]Fine-Tuned, Contextualized Embedding, Recurrent Neural Network

Table 1.2: Summarized accuracy result of FT-CE-RNN and other baseline models. The percentile denotes the percentage of the most extreme scores on which we perform the accuracy test.

	percentile	1%	2%	5%	10%
Baseline models	NBC [Maron, 1961]	59.8	56.1	54.3	53.4
	SSESTM [Ke et al., 2019]	56.3	55.4	54.4	53.2
	Bloomberg [Proprietary]	58.3	58.3	58	54.5
	BERT [Devlin et al., 2018]	73.6	66.5	59.3	56.1
	FinBERT [Yang et al., 2020b]	73.9	66.5	59.2	55.6
Proposed model and its variants	RNN	71.4	63.4	56.7	54.3
	CE-RNN	70.9	63.8	57.9	54.7
	FT-CE-RNN	**74.5**	**67.8**	**59.3**	**56.6**

We provide the details of this study in Chapter 2.

1.1.3 Chapter 3: Multivariate Stock Movement Prediction with Graph Convolutional Network

This subsection introduces the second subject of this thesis. The detailed study is in Chapter 3.

Motivation and main challenges

In the study introduced in Section 1.1.2, we suppose that a piece of news only impacts one associated stock. However, the stocks on the financial market can be highly correlated [Campbell et al., 1993]. For example, good news on one stock can positively impact the company's clients and suppliers but negatively impact its competitors. Our goal is to improve the prediction results by incorporating this relationship information into a multivariate prediction model. In addition, we observe that some news can have a longer impact in time than other news, we would like to predict the stock movement based on the news across multiple days in history instead of only one news.

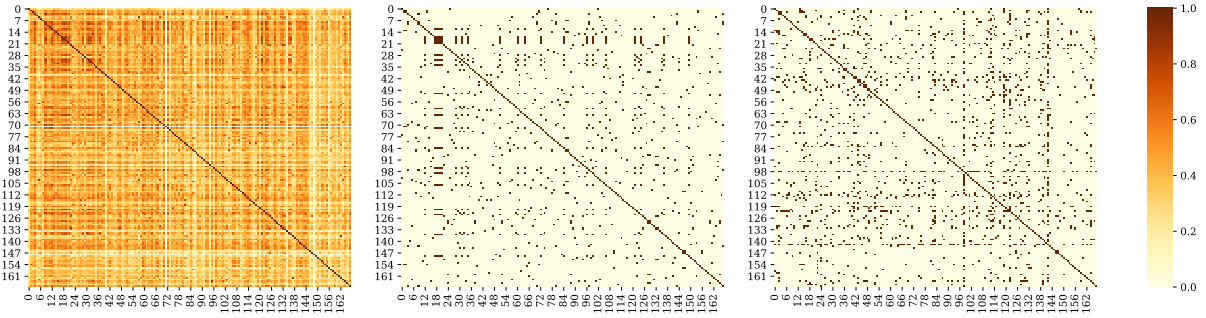
Hence, in addition to the news, we also use the data from different sources to build stock relationships. For example, we can get a stock return correlation from the stock price data or build a stock sector activity relationship with the activity sector data from GICS[3]. We can also create a supply chain relationship with the supply chain data from Factset[4]. We can visualize these relationships through their adjacency matrices in Figure 1.2.

Following the notations in the previous section, our goal can be rewritten as:

$$\widehat{Y}_{s,t} = f([E_{1,t}^{\Delta T}, ..., E_{n,t}^{\Delta T}], [G_1, ..., G_g], \theta) \tag{1.5}$$

where $E_{s,t}^{\Delta T}$ denotes all the news about stock s between t and $t - \Delta T$. In this equation, n is the total number of stocks, G_i is the stock relationship built from source i and g is the number of

[3]https://www.msci.com/our-solutions/indexes/gics
[4]https://go.factset.com/marketplace/catalog/product/factset-supply-chain-relationships



(a) Correlation Graph (b) Sector Graph (c) Supply Chain Graph

Figure 1.2: The adjacency matrices of the correlation graph, the sector graph and the supply chain graph. The adjacency matrix A is a $n \times n$ matrix where $A_{ij} \in [0, 1]$ denotes the relation between the stock i and stock j. $A_{ij} = 0$ means that there is no relationship between these two stocks while $A_{ij} = 1$ means that the relationship between two stocks is strong.

relations that we construct from different data sources. The main difference from Equation 1.4 is that we look at all news on all stocks when making predictions instead of only looking at one news on one stock. Another difference is that we look at all the news between t and $t - \Delta T$ when predicting instead of considering only one news at t.

To summarize, we want to build a model with the following characteristics:

1. The model predicts the stock movement from textual news and stock relations.
2. The model can ingest multiple relations built from different sources.
3. The model considers the temporal interactions of the news.

For point 1, we use Graphs (Appendix 3.A.1) to model the stock relations and extract the relational information. Recently, with the development of the machine learning, analyzing the graphs with neural networks has attracted more attention and multiple Graph Neural Networks (Appendix 3.A) have been proposed. Following Kipf and Welling [2016] and Chen et al. [2018], we use Graph Convolutional Network (Appendix 3.A.2) for this multivariate stock movement prediction task. In our case, each stock is a node in the graph and its node embedding (Appendix 3.A.2) is constructed as the aggregation of the news embedding for this stock in a day before t. The edge is constructed with relational data such as historical price correlation, activity sector and supply chain.

For point 2, we design a multi-graph structure in our network to extract information from different relational data, we then use an attention mechanism [Vaswani et al., 2017] to aggregate the outputs from all graphs in a non-linear way.

For point 3, we repeat the same procedure in point 1 and point 2 each day for ΔT days before t to get ΔT embeddings. These embeddings are then taken by a similar Recurrent Neural Network used in Section 1.1.2 to account for the temporal relations between news.

Previous research only partially solve this problem. Chen et al. [2015] use Long-Short Term Memory (LSTM, Section 2.B.3) to model the temporal interactions between the news while ignoring stocks' relationships (points 1 and 2). Sawhney et al. [2020] consider a single pre-built

stock relationship when predicting stock return from financial news (point 2). Li et al. [2021] propose a prediction model with both textual data and relation data as input while only accepting one stock graph and missing temporal considerations (points 2 and 3).

Multi-Graph Recurrent Network

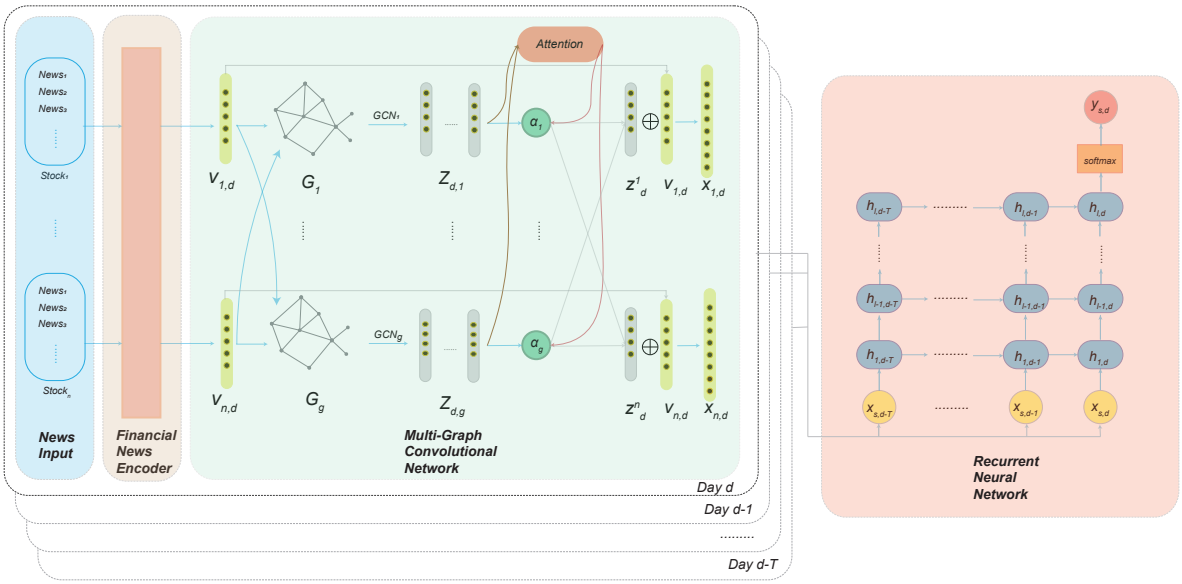


Figure 1.3: The structure of our Multi-Graph Recurrent Network.

Combining all the factors above, we design a Multi-Graph Recurrent Network (MGRN, Figure 1.3, details in Section 3.4). There are three subcomponents in the MGRN: (1) Financial News Encoder, which encodes textual news into a fixed-length vector for each stock and each day. (2) Multi-Graph Convolutional Network, which first performs graph convolution on the encoded daily news vectors and then combines all graph outputs through an attention mechanism. (3) Recurrent Neural Network, which takes the combined embeddings during a look-back window T as input and extracts temporal patterns among the news. Finally, through a fully-connected layer, we get the stock price movement probability for either direction.

Through an accuracy test and a trading simulation on around 600 stocks of the STOXX Europe 600 index, we demonstrate better performance of our model compared with other benchmarks. A summarized accuracy result is shown in Table 1.3, and the full results are shown in Section 3.5.4. We can see that the MGRN has a 13.2% accuracy gain compared with the Bloomberg score, a 6.9% accuracy gain compared with the MAN-SF model proposed by Sawhney et al. [2020] and 3.3% accuracy gain compared with the BERT model on the 10% most extreme scores. In addition, we demonstrate that the MGRN with all three relationships outperforms the vanilla MGRN without any relational data as input by 3.8%, which clearly shows the benefit of multivariate prediction. We also include a detailed analysis of the performance gained from each type of graph (Section 3.5.4), as well as a qualitative analysis of an example to show the intuitiveness behind the result (Section 3.5.5).

We provide the details of this study in Chapter 3.

Table 1.3: Summarized accuracy result of MGRN and other baseline models. The percentile denotes the percentage of the most extreme scores on which we perform the accuracy test.

	percentile	100%	20%	10%	2%
Baseline models	ARIMA [Ho and Xie, 1998]	0.491	0.494	0.487	0.510
	Bloomberg [Proprietary]	0.493	0.486	0.524	0.554
	Mean-BERT [Devlin et al., 2018]	0.512	0.586	0.623	0.694
	MAN-SF [Sawhney et al., 2020]	0.503	0.569	0.587	0.627
Proposed model and its variants	RNN	0.527	0.576	0.618	0.677
	MGRN-Corr	0.556	0.609	0.617	0.727
	MGRN-Sector	0.532	0.589	0.640	0.705
	MGRN-Supply	0.555	0.620	0.627	0.718
	MGRN	**0.561**	**0.629**	**0.656**	**0.728**

1.2 Part II: Realized Volatility Prediction with Limit Order Book

1.2.1 Context and Objective

Volatility is an important quantity in finance, it evaluates the price fluctuation and represents the risk level of an asset. It is one of the most important indicators used in risk management and equity derivatives pricing. Although the volatility is not observable, Andersen and Bollerslev [1998] show that realized volatility is a good estimator of the volatility. Forecasting realized volatility has therefore attracted the attention of various researchers. As an active asset manager, ExodusPoint seeks to predict short-term realized volatility to be more competitive in its options trading business.

Predicting realized volatility is not a new task, Brailsford and Faff [1996] propose using GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) model to predict realized volatility based on daily prices. Gatheral and Oomen [2010] introduce some simple volatility estimators based on Limit Order Book (LOB) data, showing that using LOB data can lead to better prediction results. Rahimikia and Poon [2020b] further propose using deep learning models based on LOB data to capture the non-linearity in the data. However, most previous researches only adopt univariate prediction models for this task while ignoring the connections among the stocks. For example, a sudden volatility jump on one stock can be transmitted significantly to other stocks in the same sector [Hassan and Malik, 2007]. Hence, combining the current situation in the academic research and our real-life needs, we are interested in predicting the short-term realized volatility (from 10 minutes to 30 minutes) in a multivariate approach after a given timestamp with the LOB data prior to this timestamp with a similar time window.

The Limit Order Book records outstanding limit orders maintained by the exchange. It records buy and sell orders (quotes) placed by all market participants and the trades executed on the market. It is the most detailed description of the market situation and is widely used to calculate the short-term realized volatility of an asset. We show examples of the quotes and trades in the LOB we use in our study in Table 1.4 and 1.5. For both quotes and trades, we have the records' *Date*, *Time* and *Symbol* (stock identifier). In addition, we have the best bid (buy) order and the best ask (sell) order with their price and volume for quotes and trade price and volume for

trades.

Table 1.4: An example of the quotes in the LOB of stock A from 9:30:01 to 9:30:02 on Jan. 4th, 2021. The best bid size and the best ask size are in the unit of lots, which equals to 1,000 shares of stocks.

Date	Time	Symbol	Best Bid Price	Best Bid Size	Best Ask Price	Best Ask Size
2021-01-04	9:30:01.002528512	A	118.53	1	118.94	2
2021-01-04	9:30:01.086944000	A	118.53	1	119	9
2021-01-04	9:30:01.086949376	A	118.53	1	118.87	2
2021-01-04	9:30:01.369885696	A	118.53	1	118.87	1
2021-01-04	9:30:02.513750784	A	118.54	1	118.87	1
2021-01-04	9:30:02.683774464	A	118.52	1	118.87	1

However, the raw LOB data shown in previous tables are not homogenous, meaning that the number of records in the same length of time is not equivalent. For example, there are 7 trades in the second of 9:30:01 while only 2 trades in the second of 9:30:02. For the sake of simplicity without losing too much information, we sample our LOB data every second. For quote data, we snapshot the best ask price (P_a^1), best bid price (P_b^1), best ask size (V_a^1) and best bid size (V_b^1) for each stock at the end of each second. For trade data, we aggregate all the trades for each stock during each second. We record the number of trades (N_t), the total number of shares traded (V_t) and the volume-weighted average price (P_t) of all trades, defined as:

$$P_t = \frac{\sum_i^{N_t} P_i V_i}{\sum_i^{N_t} V_i} \tag{1.6}$$

where P_i is the price for trade i, and V_i is the number of shares traded. N_t is the total number of trades recorded during second t. The sampled LOB of the example in Table 1.4 and 1.5 is shown respectively in Table 1.6 and 1.7. We note that the *second* signifies the number of seconds since the market start which is 9:30:00 in the US market.

We can formulate this realized volatility prediction task as follows.

We first define the log second return of the stock s at the t-th second as

$$r'_{s,t} = log(\frac{P_{s,t}}{P_{s,t-1}}) \tag{1.7}$$

Table 1.5: An example of the trades in the LOB of stock A from 9:30:01 to 9:30:02 on Jan. 4th, 2021.

Date	Time	Symbol	Trade Volume	Trade Price
2021-01-04	9:30:01.002522624	A	100	118.92
2021-01-04	9:30:01.002668032	A	3	118.91
2021-01-04	9:30:01.004687872	A	100	118.735
2021-01-04	9:30:01.051353600	A	13	118.94
2021-01-04	9:30:01.092121856	A	200	118.87
2021-01-04	9:30:01.369884928	A	1	118.87
2021-01-04	9:30:01.625717504	A	10	118.86
2021-01-04	9:30:02.081715968	A	102	118.7
2021-01-04	9:30:02.934643712	A	150	118.87

Table 1.6: The one second sampled quotes for the trades in LOB shown in Table 1.4. We note that V_b^1 and V_a^1 are in the unit of lots.

Date	Symbol	seconds	P_b^1	V_b^1	P_a^1	V_a^1
20210104	A	1	118.53	1	118.87	1
20210104	A	2	118.52	1	118.87	1

Table 1.7: The one second sampled trades for the trades in LOB shown in Table 1.5.

Date	Symbol	seconds	N_t	V_t	P_t
20210104	A	1	8	427	118.852
20210104	A	2	7	252	118.801

where $P_{s,t}$ is the last trade price of s at t.

Given a fixed window of ΔT seconds, our target $Y'_{s,t}$, the realized volatility for stock s between t and $t + \Delta t$ is defined as:

$$Y'_{s,t} = \sqrt{\sum_{i=t}^{t+\Delta T} r_{s,i}^2} \tag{1.8}$$

Using the same notation as Equation 1.3, this short-term realized volatility prediction task can therefore be written as

$$\widehat{Y'_{s,t}} = f'(E_t, \theta) \tag{1.9}$$

where f' is the prediction model. The main difference from the Equation 1.3 is that our goal is the realized volatility instead of the stock movement direction, which is a regression task instead of a classification task.

Unlike the existing univariate models, we introduce a multivariate solution to this realized volatility prediction task by combining LOB data and the relational data in Section 1.2.2.

1.2.2 Chapter 4: Multivariate Realized Volatility Prediction with Graph Neural Network

This subsection introduces the third subject of this thesis. The detailed study is in Chapter 4.

Motivation and main challenges

The simplest method of forecasting the short-term realized volatility for the next ΔT seconds from t is to predict only with a fragment of LOB data before t. For example, we can simply use the realized volatility calculated from $t - \Delta T$ as the prediction of our target: the realized volatility between t and $t + \Delta T$. Zhou [1996] further proposes a more mature but intuitive indicator for this task. This simple prediction process can be written as:

$$\widehat{Y'_{s,t}} = f'(D_{s,t}, \theta) \tag{1.10}$$

where $D_{s,t} = E_{s,t}^{\Delta T'}$ denotes the LOB data of the stock s between $t - \Delta T'$ and t. $\Delta T'$ is the look-back window and should be on the same scale as T.

More recently, researchers adopt time-series methods to improve the prediction while ignoring the connection among the stocks, such as Brailsford and Faff [1996], Gatheral and Oomen [2010] and Rahimikia and Poon [2020b]. The model is improved as:

$$\widehat{Y'_{s,t}} = f'([D_{s,t_1}, ..., D_{s,t_m}], \theta) \tag{1.11}$$

where $t_1 < ... < t_m = t$. The method assumes that the past volatility for the stock s has an impact on the future volatility of s, but the past volatility of other stocks has no impact on s.

However, as stated in Section 1.1.3, the stocks are not independent; we would like to consider this effect and predict the realized volatility in a multivariate approach. Given n stocks $s_1, ..., s_n$, we can rewrite our final model as:

$$\widehat{Y'_{s,t}} = f'\left(\begin{bmatrix} D_{s_1,t_1} & ... & D_{s_1,t_m} \\ ... & ... & ... \\ D_{s_n,t_1} & ... & D_{s_n,t_m} \end{bmatrix}, \mathcal{G}, \theta\right) \tag{1.12}$$

where $\mathcal{G}$ denotes the relations among all $D_{i,j}$. It includes both cross-sectional relationship $\mathcal{G}_s$ and temporal relationship $\mathcal{G}_t$.

In this study, we use the same cross-sectional relationships proposed in Section 1.1.3, including stock correlation, activity sector and supply chain.

There are three main challenges in this task:

1. How to encode the LOB data in a meaningful way.
2. How to build graphs based on both cross-sectional and temporal relationships.
3. How to compute efficiently on a large graph.

The first challenge states that the LOB data is in a tabular format. Therefore, we need to find a method to transform it into a standardized numerical format for computation. To solve this issue, we design a LOB data encoder (Section 4.4.1) that transforms both numerical (such as price and volume) and categorical (such as stock symbol) features in the LOB data into a fixed dimension vector. The numerical features are encoded with a set of intuitive indicators similar to Kercheval and Zhang [2015], and the categorical features are encoded with a trainable embedding layer (Appendix 2.B.2).

For the second challenge, we build a graph in which each node represents an observation $D_{s,t}$. For temporal relationships, we fix s and create edges among different t; for cross-sectional relationships, we build edges by fixing t and connecting different s. This method can represent both types of relationships in the same graph (Section 4.5.2).

Although the above-mentioned graph design is intuitive, the main fallback is that the graph becomes much larger than the graph containing only cross-sectional relations such as MGRN, since we have $m \times n$ nodes in this graph as opposed to n nodes in MGRN. To solve this third challenge, Gilmer et al. [2017] propose that the graph models can be seen as message passing

models (Appendix 3.A.3). It means that instead of applying the operators on the whole graph, we update each node embedding individually only based on its neighbors, the information is therefore passed from the neighbors to itself. This idea allows us to train a graph-based network in batches and avoid putting the graph in the memory as a whole. Based on this generalization, Shi et al. [2020] further propose a Transformer-like [Vaswani et al., 2017] operator and achieve the state-of-the-art performance. We use this Transformer-like operator to extract information efficiently from the large graph in our research.

To the best of our knowledge, there is no multivariate realized volatility prediction method using relationships built from external data in the literature. We also find no research which proposes the use of graph neural networks in this task.

Graph Transformer Network for Volatility Forecasting

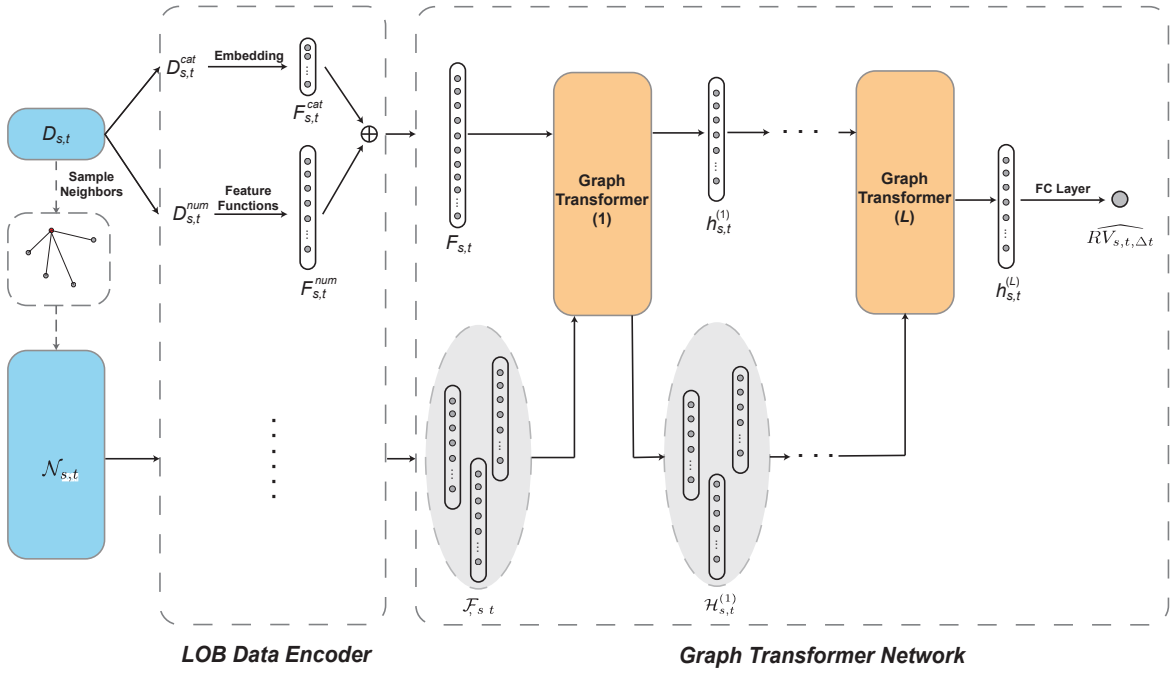


Figure 1.4: The structure of our Graph Transformer Network for Volatility Forecasting.

To further improve our realized volatility prediction result and close the gap in the research, combining all the three factors above, we design a Graph Transformer Network for Volatility Forecasting (GTN-VF, Figure 1.4, details in Section 4.4). There are two main components in GTN-VF: (1) LOB data encoder, which encodes the categorical and numerical from LOB into a fixed-length vector. (2) Graph Transformer Network, which makes prediction by aggregating the node embeddings of a node and its neighbors in an efficient way.

Through experiments based on about 500 stocks from S&P 500 index, we find better performance for our model than for other benchmarks on all prediction horizons ΔT. A summarized RMSPE[5] result based on test set is shown in Table 1.8, and the full results are shown in Section 4.5.4. For example, our GTN-VF with all three cross-sectional relationships and temporal relationship outperforms HAR-RV [Corsi, 2009] by 3.25% in RMSPE when ΔT equals 10 minutes, it also

[5]Root Mean Square Percentage Error

outperforms TabNet [Arık and Pfister, 2021] by 1.91% in the same condition. To prove the effectiveness of different relations, we include a detailed performance analysis for different types of temporal and cross-sectional relationships (Table 4.5). In addition, we add some ablation studies to show where the improvement in our model comes from (Section 4.6).

Table 1.8: Summarized RMSPE (lower is better) result of GTN-VF and baseline models.

	ΔT	600s	1200s	1800s
Baseline models	Naïve Guess	0.2834	0.2628	0.2364
	HAR-RV [Corsi, 2009]	0.2612	0.2061	0.1939
	lightGBM [Ke et al., 2017]	0.2492	0.2035	0.1963
	MLP [Rumelhart et al., 1985]	0.2514	0.2308	0.1999
	TabNet [Arık and Pfister, 2020]	0.2478	0.1996	0.2019
Proposed model ans its variants	Vanilla GTN-VF	0.2498	0.2251	0.2160
	GTN-VF Cross FC	0.2382	0.2196	0.2046
	GTN-VF Temp FC	0.2358	0.1921	0.1853
	GTN-VF Cross FC + Temp FC	0.2306	0.1917	0.1802
	GTN-VF Cross Sector	0.2422	0.2248	0.2091
	GTN-VF Cross Supply Chain	0.2411	0.2244	0.2000
	GTN-VF	**0.2287**	**0.1892**	**0.1798**

We provide the details of this study in Chapter 4.

1.3 Part III: Beyond Finance - NLP Applications in Contemporary Art

In this part, we introduce an application of NLP methodologies in contemporary art. Although not a finance related problem, this subject applies and expands the knowledge in Part I about NLP and transfer learning.

1.3.1 Context and Objective

The contemporary art market has been growing quickly since the beginning of this century [Kräussl et al., 2016]. Traditionally, collectors rely on social interactions, professional advice and media to find the artworks they wish to require. With an increasing number of artists and artworks, recommending suitable artworks to collectors with different tastes becomes a challenging problem. Recently, with the rapid development in artificial intelligence, researches start to modernize and automatize this art discovery process.

Most researchers focus on the visual aspect of the artworks to find the similarities among them. Elgammal et al. [2018] classify the styles of the paintings with some frequently used models in computer vision. Kim et al. [2018] further expand this method to group artworks into pre-defined art concepts and principles in multiple dimensions. However, these previous researches only focus on the visual aspect of the contemporary artworks since contemporary art can go beyond the aesthetics. For example, the artist practice and the influences used by the artist can both have a strong influence on the taste in art.

Hence, the easily available textual information such as the artists' biographies comes to our attention. In this work, we are interested in finding the relations among the artists through their biographies. More concretely, given the biographies of two artists, we classify if they are connected. We show an example of this task in Table 1.9.

Table 1.9: An example of our labeled artist biography pair data. *bio_a* is the biography of the *artist_a* and *bio_b* is the biography of the *artist_b*. The *label* is manually annotated by a team of professionals in the art industry signifying if two artists are connected (1 if connected and 0 if not connected). Our goal is to train a model which can tell if two artists are connected given their biographies.

artist_a	artist_b	bio_a	bio_b	label
Carl Edouard Keita	Kudzanai Violet Hwami	Born in 1992 in Abidjan, Carl-Edouard Keita now lives and works in New York. A 2021 graduate of the New York Academy of Art, Carl-Edouard Keita also won the prize for best draughtsman for his graduation work, some of which is presented in this group exhibition. Carl-Edouard Keita discovered the history of African art during his economics studies in Atlanta, through a course offered at his university. As he describes it himself, this discovery was a real aesthetic revelation for him.	Having fled her homeland due to the political unrest and turmoil when she was a child, Zimbabwe-born painter Kudzanai-Violet Hwami expresses her personal experiences of dislocation, displacement and fragmentation through her striking figurative paintings. The artist is interested in the collapsing of geography and time and space symptomatic of a globalised world and high-speed internet, through which both people and information can travel quickly.	1
Sun Xun	Cheng Xinhao	Sun Xun was born in 1980 in Fuxin, Liaoning province, China. Currently lives and works in Beijing. Recent and past histories, intransigent conflicts and tensions, sequential flashes of hand-created images of these are the irrevocable features of Sun Xun's artistic practice that fuses the line between art and animation. A graduate from the Printmaking Department of the China Academy of Arts in 2005, Sun Xun was a professor at the prestigious Academy before founding in 2006 his own Animation Studio. His work primarily involves making images using various materials such as colour powder, woodcuts and traditional ink, and collating these to produce a film, which is often presented in an immersive setting.	Cheng Xinhao (b.1985, Yunnan, China). After receiving his PhD on Chemistry from Peking University in 2013, Cheng continued his career as a photographer, investigating on the issues in the modernization, the construction of knowledge as well as the production of space in Chinese society.	0

We can formulate this artist connection discovery problem as a binary classification problem. Suppose that we have two artists A_i, A_j and their biographies are denoted by B_i, B_j. A team of art professionals can determine if they are connected based on different aspects, including background, themes, style, techniques, etc. We use $Y_{i,j}$ to denote this ground-truth, with

$$Y_{i,j} = \begin{cases} 1 & \text{if } A_i \text{ and } A_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}.$$

However, there are thousands of artists in our artist database, and it is impossible to determine all relationships through manual annotation. Hence, in this study, we are interested in building the connections among them automatically based on their biographies. We can describe this process as

$$p_{i,j} = f(B_i, B_j, \theta)$$

where $p_{i,j}$ denotes our predicted probability of a connection between A_i and A_j. f is our prediction model and θ denotes the trainable parameters.

Given f, our goal is to find θ which minimizes the cross-entropy loss [Good, 1992] defined as

$$-\sum_{i,j} Y_{i,j} \cdot \log p_{i,j} + (1 - Y_{i,j}) \cdot \log(1 - p_{i,j}).$$

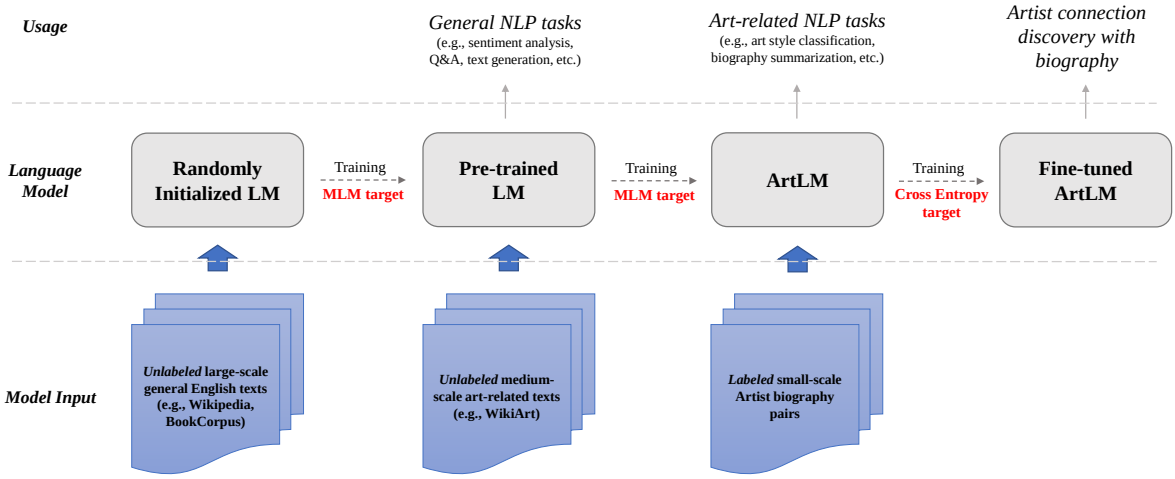


Figure 1.5: Overview of our artist connection discovery method, including two unsupervised pre-training phases and one supervised fine-tuning phase.

1.3.2 Chapter 5: Mapping the contemporary art world with ArtLM: an art-specific NLP model

This subsection introduces the fourth subject of this thesis. The detailed study is in Chapter 5.

Motivation and main challenges

We aim to solve two main challenges in this task:

1. How to efficiently classify the connection between two artists given their labeled biography pair.

2. How to improve the classification accuracy given some unlabeled art-related texts.

To find solutions to the first challenge, we can fine-tune some pre-trained Language Models (LMs) following the same idea introduced in Section 1.1.2. However, different from the previous research, our input is a pair of sentences instead of a single sentence. In this case, we concatenate the pair of sentences while splitting them with a special character to form a single sentence and continue the fine-tuning process (Section 5.4.3).

For the second challenge, we add another pre-training step before the fine-tuning. In this procedure, we continue to pre-train the original pre-trained models with the unlabeled art-related texts gathered from WikiArt[6] using the same training targets of the original models. We also apply this method on different publicly available pre-trained language models to prove the robustness of this approach.

[6]https://www.wikiart.org/

Although this pre-train/fine-tune scheme has been applied in the domain such as finance [Yang et al., 2020b], we see no application in arts in the literature to the best of our knowledge.

ArtLM for artist connection discovery

To consider all the factors above and close the gap in the research, we propose a generic NLP framework (called ArtLM, Section 5.4.2, Figure 1.5) to discover the connections among contemporary artists based on their biographies. We first continue to pre-train existing language models with unlabeled art-related texts to add artistic domain knowledge to the existing general models. We then fine-tune the model with the biography pairs labeled by the professionals in art. This procedure is robust and independent of the choice of base models, which is proved by the experiments based on multiple base models, such as BERT [Devlin et al., 2018], DistilBERT [Sanh et al., 2019] and RoBERTa [Liu et al., 2019].

Table 1.10: Summarized experiment results of ArtLM variants and baseline models

model	accuracy	F1	fine-tuning time (min. / fold)	pre-training time (min.)
Random Guess - 1	0.500	0.000	-	-
Random Guess - 0.5	0.500	0.500	-	-
word2vec embedding	0.539	0.196	-	-
BERT embedding	0.581	0.320	-	-
FT-Base-DistilBERT	0.838	0.826	6	-
FT-Base-BERT	0.844	0.832	12	-
FT-Base-RoBERTa	0.844	0.824	12	-
FT-Art-DistilBERT	0.846	0.826	6	10
FT-Art-BERT	**0.856**	**0.840**	12	18
FT-Art-RoBERTa	0.854	0.834	12	21

With extensive experiments, we demonstrate that our ArtLM achieves 85.6% accuracy and 84.0% F1-score and outperforms all other baseline models in a 5-fold cross-validation setup. A summarized accuracy and F1-score results is shown in Table 1.10, and the full detailed results are shown in Section 5.4. Comparing the results of the fine-tuned models (both `FT-Base` and `FT-Art`) and the baseline models, we find that our supervised fine-tuning approach largely outperforms the embedding based correlation methods. We also prove that adding domain-specific knowledge by continuing the pre-training process can help improve the model performance, if we focus on the accuracy discrepancies between the base models and the art models.

The generic ArtLM process can easily be generalized to other domains including finance. For example, we can continue to pre-train a LM with unlabeled financial documents, then make stock recommendations or sentiment analysis through manually labeled financial reports.

We provide the details of this study in Chapter 5.

Chapter 1

Introduction (en français)

Les marchés financiers sont très incertains. Cependant, les investisseurs sur les marchés financiers cherchent à obtenir un avantage en prévoyant l'évolution du marché à l'avance. Ce sujet difficile a attiré beaucoup d'attention.

ExodusPoint Capital Management est un gestionnaire d'actifs actif qui cherche à obtenir des bénéfices supplémentaires pour ses clients en prédisant le marché à l'avance. Cela inclut le rendement directionnel des actions dans le domaine d'actions et la volatilité non directionnelle dans le domaine des options. Ainsi, pour appliquer les méthodes d'apprentissage profond qui se développent rapidement aux besoins des entreprises, cette thèse explore différentes méthodes d'apprentissage profond pour prédire le rendement et la volatilité des actions à partir de diverses sources de données.

Fama [1965] montre que le mouvement d'une action à un moment donné peut être expliqué par toutes les informations précédemment disponibles. Cependant, il est difficile de collecter et d'utiliser ce large éventail de données. Par conséquent, les chercheurs utilisent généralement une partie de l'information pour capturer une fraction du mouvement du marché. Traditionnellement, les praticiens prédisent le marché à l'aide de données facilement accessibles, telles que les prix historiques et les fondamentaux des entreprises. Avec le développement de l'apprentissage automatique, en particulier le traitement du langage naturel (NLP) et les réseaux de neurones graphiques (GNN). Il est possible de traiter des informations plus complexes telles que les nouvelles et les relations entre les différentes action.

Ce chapitre présente deux problèmes de prédiction en finance examinés dans cette thèse : la prédiction du rendement des actions avec les nouvelles financières et la prédiction de la volatilité réalisée avec le carnet d'ordres à cours limité. Il présente également un problème connexe de classification de connexions d'artistes, qui fait appel à certaines méthodes NLP appliquées au premier problème de prédiction de rendement boursier. Ce chapitre est organisé comme suit :

Dans la section 1.1, nous présentons le premier sujet : la prédiction du rendement des actions à l'aide de nouvelles financières, et dans la section 1.2, nous présentons le deuxième sujet : la prédiction de la volatilité réalisée à l'aide de données du carnet d'ordres à cours limité. Dans chaque sujet, nous donnons d'abord le contexte, les données que nous utilisons et la définition du problème. Pour le premier sujet, nous introduisons des méthodes univariées et multivariées dans les sections 1.1.2 et 1.1.3, respectivement. Nous présentons ensuite une solution multivariée

pour le deuxième sujet dans la section 1.2.2. En outre, nous présentons un cadre NLP générique pour découvrir les connexions entre les artistes contemporains sur la base de leurs biographies dans la section 1.3, qui utilise un schéma d'apprentissage de transfert similaire à celui appliqué au premier sujet. Dans les quatre études, nous donnons une définition plus détaillée du problème dans chaque contexte, les principaux défis auxquels nous sommes confrontés, nos solutions à ces défis, les principales innovations et un résultat résumé.

Les détails de toutes les quatres études sont présentées dans les Chapitres 2, 3, 4 et 5, respectivement.

En outre, nous ajoutons un aperçu de NLP et GNN dans les Appendices 2.B et 3.A pour les lecteurs qui n'ont pas de connaissance dans ces deux domaines.

1.1 Part I: Prédiction du Rendement des Actions avec les Nouvelles Financières

1.1.1 Contexte et Objectif

La prévision de l'évolution des actions a toujours été un sujet intéressant et cela est particulièrement vrai pour ExodusPoint Capital Management, un gestionnaire d'actifs. Nous cherchons à prédire l'évolution des actions à l'avenir à partir de diverses sources de données afin de faire des bénéfices pour les investisseurs.

Il existe dans la littérature diverses approches basées sur des données différentes. Par exemple, les économétriciens utilisent l'analyse des séries chronologiques pour prédire les prix des actions dans le futur sur la base des prix historiques et des volumes de transactions [Mills and Mills, 1990; Chen et al., 2007; Mondal et al., 2014]. Les analystes financiers utilisent les fondamentaux des entreprises [Chan et al., 1993; Wang and Xu, 2004; Ozlen, 2014] et les données macroéconomiques [Narayan et al., 2014; Hoseinzade and Haratizadeh, 2019] pour atteindre cet objectif. Plus récemment, avec le développement du traitement du langage naturel, nous pouvons traiter davantage d'informations non structurées telles que les actualités financières [Ding et al., 2014,0; Hu et al., 2018; Xu and Cohen, 2018].

Nous pouvons également classer cette tâche de prédiction du mouvement des actions en méthodes univariées ou multivariées. La méthode univariée est la plus couramment utilisée dans cette tâche, ce qui signifie que nous faisons des prédictions stock par stock en ignorant les connexions entre les stocks. Cependant, l'approche multivariée prend en compte les relations entre les actions provenant de diverses sources de données afin de mieux modéliser ce processus de transmission de signaux. Avec le développement des méthodes de graphes dans l'apprentissage profond, nous pouvons traiter ces informations relationnelles plus efficacement. Plusieurs méthodes multivariées sont appliquées dans cette tâche [Mehtab and Sen, 2020; Lof, 2012].

Avec le développement rapide d'Internet, nous observons une croissance significative du nombre de textes disponibles. Contrairement aux informations traditionnelles de prix et de volume, les données textuelles restent largement inexplorées. Par conséquent, nous souhaitons nous concentrer sur la prédiction du mouvement des actions à partir des données d'actualité à l'aide des méthodes d'apprentissage profond les plus récentes. Dans un premier temps, nous cherchons

à prédire le mouvement des actions à court terme après une nouvelle (environ 30 minutes pour les nouvelles pendant les heures de négociation et un jour pour les nouvelles en dehors des heures de négociation) dans une approche univariée. Nous nous concentrons ensuite sur la prédiction du mouvement quotidien des actions dans une approche multivariée à partir des nouvelles et des données relationnelles provenant d'autres sources telles que le secteur d'activité et la chaîne d'approvisionnement.

Les nouvelles financières que nous utilisons sont celles de Bloomberg News, une source de données de nouvelles largement utilisée dans le secteur financier. Nous présentons un exemple de cet ensemble de données dans le tableau 1.1. L'ensemble de données comprend le *Headline* de l'actualité et les *TimeStamp* et *Ticker* associés à cette actualité. En outre, Bloomberg fournit également son analyse du sentiment avec le *Score*, qui est une valeur comprise entre -1, 0 et 1, indiquant respectivement un sentiment négatif, neutre et positif. *Confidence* est une valeur comprise entre 0 et 100 qui donne le niveau de confiance du *Score*. Nous utilisons l'analyse de Bloomberg comme l'un de nos points de référence.

Table 1.1: Un exemple de données de Bloomberg News

Headline	**TimeStamp**	**Ticker**	**Score**	**Confidence**
1st Source Corp: 06/20/2015 - 1st Source announces the promotion of Kim Richardson in St. Joseph	2015-06-20 05:02:04.063	SRCE	-1	39
Siasat Daily: Microsoft continues rebranding of Nokia Priority stores in India opens one in Chennai	2015-06-20 05:14:01.096	MSFT	1	98
Rosneft, Eurochem to cooperate on monetization at east urengoy	2015-06-20 08:01:53.625	ROSN RM	0	98

Nous formulons ce problème de prédiction du rendement comme suit.

Étant donné une action s, nous définissons son rendement ajusté au marché r_s entre t et $t + \Delta t$ comme :

$$r_{s,t} = \frac{P_{s,t+\Delta t}}{P_{s,t}} - \frac{P_{m,t+\Delta t}}{P_{m,t}} \tag{1.1}$$

où $P_{s,t}$ désigne le prix de l'action s au moment t, et $P_{m,t}$ désigne la valeur de l'indice de marché au moment t. Le rendement ajusté au marché élimine l'impact de l'indice de marché et reflète mieux l'impact de la nouvelle.

Nous n'utilisons pas le bêta pour ajuster le rendement du marché comme le propose le modèle d'évaluation des actifs financiers (MEDAF) [Black et al., 1972], puisque Fama and French [2004] démontrent que le bêta peut surestimer l'impact des actions à bêta élevé et sous-estimer l'impact des actions à bêta faible. sous-estimer l'impact des actions à faible bêta.

Nous définissons la cible de notre tâche de prédiction des mouvements de stock pour le stock s entre t et $t + \Delta t$ comme :

$$Y_{s,t} = \begin{cases} 1, & r_{s,t} > 0 \\ 0, & r_{s,t} \leq 0 \end{cases}. \tag{1.2}$$

La tâche de prédiction du retour peut être écrite comme suit :

$$\widehat{Y}_{s,t} = f(E_t, \theta) \tag{1.3}$$

où f désigne le modèle de prédiction, E_t est l'ensemble des informations, y compris les nouvelles, disponibles avant t, θ représente l'ensemble des paramètres entraînables et $\widehat{Y}_{s,t}$ est la prédiction donnée par le modèle.

Nous présentons d'abord une solution univariée dans la section 1.1.2, qui prévoit le rendement à court terme d'une action après la publication de la nouvelle. Nous proposons ensuite une solution multivariée dans la section 1.1.3, qui prévoit les rendements quotidiens de toutes les actions sur la base de toutes les nouvelles d'une journée et des relations entre les actions établies à partir de différentes sources de données.

1.1.2 Chapitre 2: Prévision des Mouvements des Actions avec l'embedding contextuelle

Cette sous-section présente le premier sujet de cette thèse. L'étude détaillée se trouve dans le chapitre 2.

Motivation et Principaux Défis

Dans cette sous-section, notre objectif est de prédire le mouvement des actions à partir d'un titre de nouvelle après sa publication. Pour les nouvelles enregistrées pendant les heures de négociation [1], l'horizon de prévision est de 30 minutes. Si la nouvelle est publiée en dehors des heures de cotation, nous prévoyons son rendement entre la clôture du marché précédent et la clôture du marché suivant. Notre solution est univariée, ce qui signifie que nous supposons que la nouvelle ne peut avoir un impact que sur le prix de l'action associée et que cette information ne se transmet pas aux autres actions. L'équation 1.3 peut donc être affinée comme suit :

$$\widehat{Y}_{s,t} = f(E^0_{s,t}, \theta) \tag{1.4}$$

où $E^0_{s,t}$ représente une nouvelle concernant l'action s enregistrée au moment t.

La première étape de cette tâche consiste à convertir les données textuelles en données numériques. En langage naturel, nous pouvons convertir un mot en un vecteur dense de longueur fixe appelé « embedding » (Appendice 2.B.2). Ce processus comporte trois défis principaux :

1. Nous avons besoin d'un grand nombre de corpus pour extraire les caractéristiques d'une langue.

[1] de 9h00 à 17h30 CEST pour la plupart des marchés européens

2. Les embeddings formés avec un grand corpus ne possèdent pas de connaissances spécifiques au domaine. Par exemple, certains mots peuvent avoir des significations spécifiques en finance par rapport à l'anglais général.

3. L'embedding est statique. Cela signifie que la représentation vectorielle d'un mot est la même dans différentes phrases, même si leur signification peut être très différente selon le contexte.

Pour résoudre le premier défi, les chercheurs utilisent généralement une grande quantité de corpus anglais pour entraîner un modèle polyvalent pour d'autres tâches en aval. Par exemple, Mikolov et al. [2013b] proposent Word2Vec et Pennington et al. [2014] proposent GloVe, qui sont tous entraînés sur un vaste corpus contenant des milliards de tokens.

Pour résoudre le deuxième défi, l'apprentissage par transfert (Appendice 2.B.4) est récemment devenu populaire. L'idée est de former un modèle anglais général à l'aide d'un grand corpus, puis de poursuivre le processus de formation sur la base d'un corpus spécifique au domaine. Récemment, plusieurs modèles d'apprentissage par transfert à grande échelle ont été proposés, notamment Embedding from Language Model (ELMo) [Peters et al., 2018a], Bidirectional Encoder Representations from Transformers (BERT, Appendix 2.B.4) [Devlin et al., 2018] et XLNet [Yang et al., 2019]. L'approche d'apprentissage par transfert s'est avérée robuste et largement utilisée dans la plupart des tâches de TAL.

Pour résoudre le dernier défi, au lieu d'utiliser un embedding de mots fixe prédéfini du modèle Word2Vec, nous introduisons tous les mots ensemble dans un modèle interconnecté, tel que le modèle BERT mentionné ci-dessus. Dans ce cas, l'embedding d'un mot sera influencée par l'embedding des mots de la même phrase. Nous pouvons alors obtenir un embedding contextualisé à partir des couches cachées de ce modèle.

La littérature existante ne résout généralement que partiellement ces problèmes. Par exemple, Ding et al. [2015] propose une des premières solutions d'apprentissage solutions d'apprentissage profond à ce problème tout en s'appuyant fortement sur des embeddings statiques générés à partir de Word2Vec (points 2 et 3). Devlin et al. [2018] proposent une méthode de classification avec BERT mais ignorent les connaissances spécifiques au domaine (point 2). Vargas et al. [2017] entraînent un modèle Word2Vec uniquement avec des textes financiers. Cependant, cette méthode ne permet pas d'entraîner un modèle avec les caractéristiques de l'anglais général, et les embeddings sont toujours statiques (points 1 et 3).

Une procédure de prédiction des mouvements boursiers avec des embeddings contextualisés

Pour prendre en compte tous les facteurs ci-dessus et combler les lacunes de la recherche, nous proposons une procédure de prédiction appelée FT-CE-RNN[2] (figure 1.1, détails présentés dans la section 2.5). Nous poursuivons d'abord le processus de pré-entraînement du modèle BERT avec un éventail plus large de textes financiers afin d'ajouter des connaissances financières au modèle. Nous affinons ensuite le modèle avec des données d'entraînement étiquetées et extrayons les embeddings contextualisées des couches cachées du modèle affiné. Enfin, nous formons un réseau neuronal récurrent (RNN, Appendice 2.B.3) avec les embeddings contextualisées générées.

[2]Fine-Tuned, Contextualized Embedding, Recurrent Neural Network

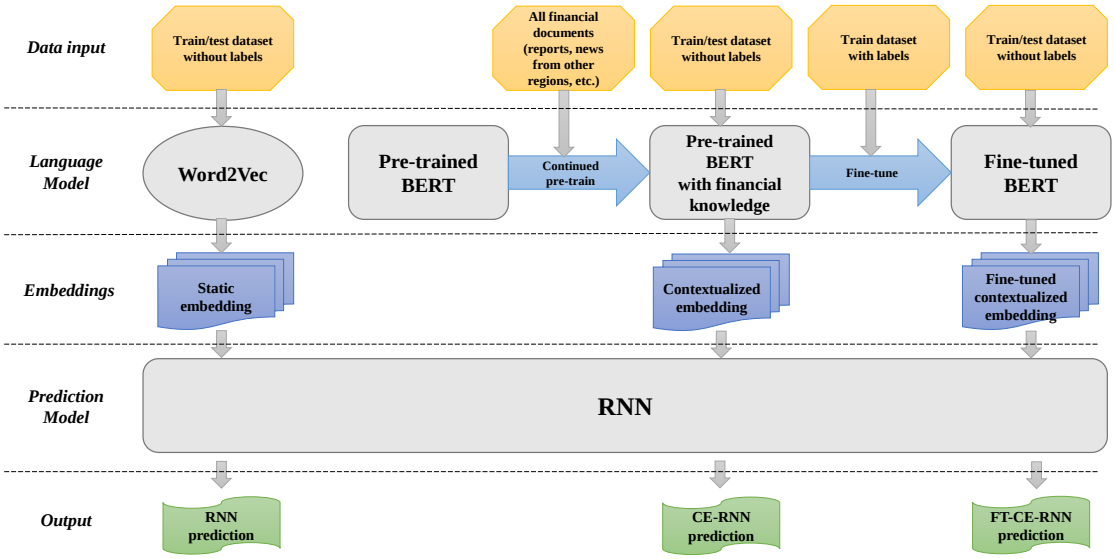


Figure 1.1: Une illustration de la procédure de prédiction des mouvements boursiers : FT-CE-RNN.

En outre, pour mieux intégrer les besoins des investisseurs, au lieu d'évaluer les performances du modèle sur la base de toutes les nouvelles, nous proposons une nouvelle métrique d'évaluation basée uniquement sur les nouvelles extrêmes (section 2.6.2).

Grâce à des expériences approfondies telles que des tests de précision et des simulations de transactions sur environ 600 actions de l'indice STOXX Europe 600, nous prouvons que le FT-CE-RNN surpasse les autres modèles de base de la littérature. Un résumé des résultats de précision est présenté dans le tableau 1.2, et les résultats complets sont présentés dans la section 2.6. Nous pouvons constater que le FT-CE-RNN surpasse le modèle BERT de 1,3% et le score Bloomberg de 9,5% en termes de précision sur la base des 2% de nouvelles les plus extrêmes. Nous démontrons également que les embeddings contextualisés d'un modèle affiné donnent un meilleur résultat de prédiction que les embeddings statiques et non réglés. Par exemple, l'utilisation des embeddings contextualisés affinés permet d'obtenir un gain de précision de 4,4% sur la base des 2% d'informations les plus extrêmes par rapport aux embeddings statiques et de 4,0% par rapport aux embeddings non affinés dans la même situation.

Nous fournissons les détails de cette étude dans le chapitre 2.

1.1.3 Chapitre 3 : Prévision Multivariée des Mouvements Boursiers avec un Réseau Convolutif Graphique

Cette sous-section présente le deuxième sujet de cette thèse. L'étude détaillée se trouve dans le chapitre 3.

Table 1.2: Résultat résumé de la précision de FT-CE-RNN et des autres modèles de base. Le percentile indique le pourcentage des scores les plus extrêmes sur lesquels nous effectuons le test de précision.

	percentile	1%	2%	5%	10%
Baseline models	NBC [Maron, 1961]	59.8	56.1	54.3	53.4
	SSESTM [Ke et al., 2019]	56.3	55.4	54.4	53.2
	Bloomberg [Proprietary]	58.3	58.3	58	54.5
	BERT [Devlin et al., 2018]	73.6	66.5	59.3	56.1
	FinBERT [Yang et al., 2020b]	73.9	66.5	59.2	55.6
Proposed model and its variants	RNN	71.4	63.4	56.7	54.3
	CE-RNN	70.9	63.8	57.9	54.7
	FT-CE-RNN	**74.5**	**67.8**	**59.3**	**56.6**

Motivation et Principaux Défis

Dans l'étude présentée dans la section 1.1.2, nous supposons qu'une nouvelle n'a d'impact que sur une seule action associée. Cependant, les actions sur le marché financier peuvent être fortement corrélées [Campbell et al., 1993]. Par exemple, une bonne nouvelle sur une action peut avoir un impact positif sur les clients et les fournisseurs de l'entreprise mais un impact négatif sur ses concurrents. Notre objectif est d'améliorer les résultats de la prédiction en incorporant cette information sur les relations dans un modèle de prédiction multivarié. En outre, nous observons que certaines nouvelles peuvent avoir un impact plus long dans le temps que d'autres, nous aimerions prédire le mouvement des actions sur la base des nouvelles sur plusieurs jours dans l'histoire au lieu d'une seule nouvelle.

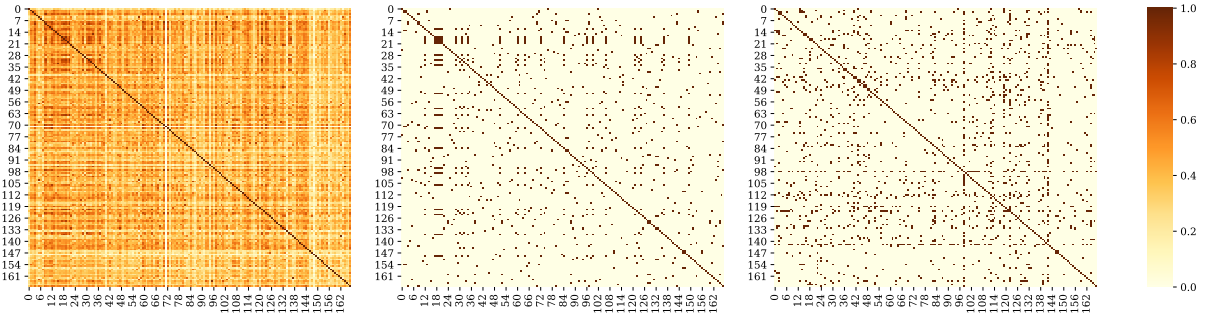
Ainsi, en plus des nouvelles, nous utilisons également les données de différentes sources pour établir des relations entre les actions. Par exemple, nous pouvons obtenir une corrélation entre les rendements des actions à partir des données sur les cours des actions ou établir une relation entre les secteurs d'activité des actions à partir des données sur les secteurs d'activité de GICS[3]. Nous pouvons également créer une relation de chaîne d'approvisionnement avec les données de chaîne d'approvisionnement de Factset[4]. Nous pouvons visualiser ces relations par le biais de leurs matrices d'adjacence dans la figure 1.2.

En suivant les notations de la section précédente, notre objectif peut être réécrit comme suit :

$$\widehat{Y}_{s,t} = f([E_{1,t}^{\Delta T}, ..., E_{n,t}^{\Delta T}], [G_1, ..., G_g], \theta) \tag{1.5}$$

où $E_{s,t}^{\Delta T}$ désigne toutes les nouvelles concernant le titre s entre t et $t - \Delta T$. Dans cette équation, n est le nombre total d'actions, G_i est la relation entre les actions construite à partir de la source i et g est le nombre de relations que nous construisons à partir de différentes sources de données. La principale différence par rapport à l'équation 1.4 est que nous prenons en compte toutes les nouvelles sur toutes les actions lorsque nous faisons des prédictions au lieu de ne prendre en compte qu'une seule nouvelle sur une action. Une autre différence est que nous considérons toutes les nouvelles entre t et $t - \Delta T$ lors de la prédiction au lieu de ne considérer qu'une seule nouvelle à t.

[3]https://www.msci.com/our-solutions/indexes/gics
[4]https://go.factset.com/marketplace/catalog/product/factset-supply-chain-relationships



(a) Correlation Graph (b) Sector Graph (c) Supply Chain Graph

Figure 1.2: Les matrices d'adjacence du graphe des corrélations, du graphe des secteurs et du graphe des chaînes d'approvisionnement. La matrice d'adjacence A est une matrice $n \times n$ où $A_{ij} \in [0, 1]$ désigne la relation entre le stock i et le stock j. $A_{ij} = 0$ signifie qu'il n'y a pas de relation entre ces deux stocks tandis que $A_{ij} = 1$ signifie que la relation entre deux stocks est forte.

Pour résumer, nous voulons construire un modèle avec les caractéristiques suivantes :

1. Le modèle prédit le mouvement des actions à partir des nouvelles textuelles et des relations entre les actions.
2. Le modèle peut ingérer de multiples relations construites à partir de sources différentes.
3. Le modèle prend en compte les interactions temporelles des nouvelles.

Pour le point 1, nous utilisons des graphes (Appendice 3.A.1) pour modéliser les relations entre les actions et extraire les informations relationnelles. Récemment, avec le développement de l'apprentissage automatique, l'analyse des graphes avec des réseaux neuronaux a attiré plus d'attention et de multiples réseaux neuronaux graphes (Appendice 3.A) ont été proposés. À la suite de Kipf and Welling [2016] et Chen et al. [2018], nous utilisons un réseau convolutif graphique (Appendice 3.A.2) pour cette tâche de prédiction multivariée du mouvement des actions. Dans notre cas, chaque action est un nœud du graphe et son embedding de nœud (Appendice 3.A.2) est construit comme l'agrégation de l'embedding des nouvelles pour cette action un jour avant t. L'arête est construite avec des données relationnelles telles que la corrélation historique des prix, le secteur d'activité et la chaîne d'approvisionnement.

Pour le point 2, nous concevons une structure multi-graphes dans notre réseau pour extraire des informations de différentes données relationnelles, nous utilisons ensuite un mécanisme d'attention [Vaswani et al., 2017] pour agréger les sorties de tous les graphes de manière non linéaire.

Pour le point 3, nous répétons la même procédure au point 1 et au point 2 chaque jour pour ΔT jours avant t pour obtenir ΔT embeddings. Ces incorporations sont ensuite prises en compte par un réseau neuronal récurrent similaire à celui utilisé dans la section 1.1.2 pour tenir compte des relations temporelles entre les nouvelles.

Les recherches précédentes ne résolvent que partiellement ce problème. Chen et al. [2015] utilisent la mémoire à long et court terme (LSTM, section 2.B.3) pour modéliser les interactions temporelles entre les nouvelles tout en ignorant les relations entre les actions (points 1 et

2). Sawhney et al. [2020] considèrent une seule relation préétablie entre les actions lors de la prédiction du rendement des actions à partir des nouvelles financières (point 2). Li et al. [2021] proposent un modèle de prédiction avec à la fois des données textuelles et des données de relation en entrée tout en n'acceptant qu'un seul graphique d'actions et en omettant les considérations temporelles (points 2 et 3).

Réseau Récurrent Multi-Graphes

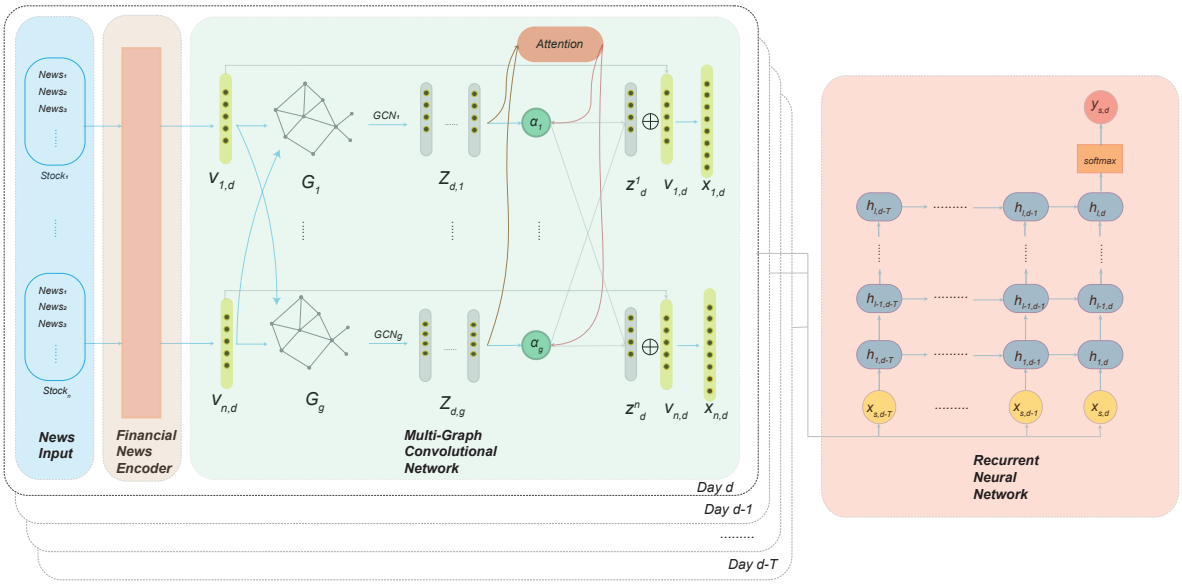


Figure 1.3: La structure de notre Réseau Récurrent Multi-Graphes.

En combinant tous les facteurs ci-dessus, nous concevons un réseau récurrent multi-graphes (MGRN, figure 1.3, détaillé dans la section 3.4). Le MGRN comprend trois sous-composants : (1) le codeur de nouvelles financières, qui code les nouvelles textuelles en un vecteur de longueur fixe pour chaque action et chaque jour. (2) Réseau convolutionnel multigraphique, qui effectue d'abord une convolution graphique sur les vecteurs d'informations quotidiennes codées, puis combine toutes les sorties graphiques par un mécanisme d'attention. (3) Réseau neuronal récurrent, qui prend en entrée les incorporations combinées pendant une fenêtre de retour en arrière T et extrait les modèles temporels parmi les nouvelles. Enfin, grâce à une couche entièrement connectée, nous obtenons la probabilité de mouvement du prix de l'action dans l'une ou l'autre direction.

Par le biais d'un test de précision et d'une simulation de transactions sur environ 600 actions de l'indice STOXX Europe 600, nous démontrons que notre modèle est plus performant que d'autres références. Un résumé des résultats de précision est présenté dans le tableau 1.3, et les résultats complets sont présentés dans la section 3.5.4. Nous pouvons constater que le MGRN présente un gain de précision de 13,2% par rapport au score Bloomberg, un gain de précision de 6,9% par rapport au modèle MAN-SF proposé par Sawhney et al. [2020] et un gain de précision de 3,3% par rapport au modèle BERT sur les 10% de scores les plus extrêmes. De plus, nous démontrons que le MGRN avec les trois relations surpasse le MGRN vanille sans aucune donnée relationnelle en entrée de 3,8%, ce qui montre clairement l'avantage de la prédiction multivariée. Nous incluons également une analyse détaillée des performances obtenues avec chaque type de graphe (section 3.5.4), ainsi qu'une analyse qualitative d'un exemple pour montrer l'intuitivité

derrière le résultat (section 3.5.5).

Table 1.3: Résumé des résultats de précision de MGRN et d'autres modèles de base. Le percentile indique le pourcentage des scores les plus extrêmes sur lesquels nous effectuons le test de précision.

	percentile	100%	20%	10%	2%
Baseline models	ARIMA [Ho and Xie, 1998]	0.491	0.494	0.487	0.510
	Bloomberg [Proprietary]	0.493	0.486	0.524	0.554
	Mean-BERT [Devlin et al., 2018]	0.512	0.586	0.623	0.694
	MAN-SF [Sawhney et al., 2020]	0.503	0.569	0.587	0.627
Proposed model and its variants	RNN	0.527	0.576	0.618	0.677
	MGRN-Corr	0.556	0.609	0.617	0.727
	MGRN-Sector	0.532	0.589	0.640	0.705
	MGRN-Supply	0.555	0.620	0.627	0.718
	MGRN	**0.561**	**0.629**	**0.656**	**0.728**

Nous fournissons les détails de cette étude dans le chapitre 3.

1.2 Part II: Prédiction de la Volatilité Réalisée avec un Carnet d'Ordres Limité

1.2.1 Contexte et Objectif

La volatilité est une quantité importante en finance, elle évalue la fluctuation des prix et représente le niveau de risque d'un actif. C'est l'un des indicateurs les plus importants utilisés dans la gestion du risque et la fixation du prix des dérivés d'actions. Bien que la volatilité ne soit pas observable, Andersen and Bollerslev [1998] montrent que la volatilité réalisée est un bon estimateur de la volatilité. La prévision de la volatilité réalisée a donc attiré l'attention de divers chercheurs. En tant que gestionnaire actif d'actifs, ExodusPoint cherche à prévoir la volatilité réalisée à court terme afin d'être plus compétitif dans son activité de négociation d'options.

La prédiction de la volatilité réalisée n'est pas une tâche nouvelle. Brailsford and Faff [1996] proposent d'utiliser le modèle GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) pour prévoir la volatilité réalisée sur la base des prix quotidiens. Gatheral and Oomen [2010] présentent quelques estimateurs de volatilité simples basés sur les données du carnet d'ordres à cours limité (LOB), montrant que l'utilisation des données LOB peut conduire à de meilleurs résultats de prédiction. Rahimikia and Poon [2020b] proposent en outre d'utiliser des modèles d'apprentissage profond basés sur les données LOB pour capturer la non-linéarité des données. Cependant, la plupart des recherches précédentes n'adoptent que des modèles de prédiction univariés pour cette tâche et ignorent les connexions entre les actions. Par exemple, une hausse soudaine de la volatilité d'une action peut se transmettre de manière significative à d'autres actions du même secteur [Hassan and Malik, 2007]. Par conséquent, en combinant la situation actuelle dans la recherche académique et nos besoins réels, nous sommes intéressés par la prédiction de la volatilité réalisée à court terme (de 10 minutes à 30 minutes) dans une approche multivariée après un timestamp donné avec les données LOB avant ce timestamp avec une fenêtre de temps similaire.

Le carnet d'ordres à cours limité enregistre les ordres à cours limité en cours tenus par la bourse. Il enregistre les ordres d'achat et de vente (quotes) passés par tous les participants au marché et les transactions exécutées sur le marché. Il s'agit de la description la plus détaillée de la situation du marché peut être utilisé pour calculer la volatilité réalisée à court terme d'un actif. Nous montrons des exemples de cotations et de transactions dans la LOB que nous utilisons dans notre étude dans le tableau 1.4 et 1.5. Pour les cotations et les transactions, nous disposons des enregistrements *Date*, *Heure* et *Symbole* (identifiant du titre). En outre, nous disposons du meilleur ordre d'achat (bid) et du meilleur ordre de vente (ask) avec leur prix et leur volume pour les cotations et le prix et le volume des transactions pour les transactions.

Table 1.4: Un exemple des cotations dans la LPP de l'action A de 9:30:01 à 9:30:02 le 4 janvier 2021. La meilleure taille d'offre et la meilleure taille de demande sont dans l'unité de lots, ce qui équivaut à 1 000 actions.

Date	Time	Symbol	Best Bid Price	Best Bid Size	Best Ask Price	Best Ask Size
2021-01-04	9:30:01.002528512	A	118.53	1	118.94	2
2021-01-04	9:30:01.086944000	A	118.53	1	119	9
2021-01-04	9:30:01.086949376	A	118.53	1	118.87	2
2021-01-04	9:30:01.369885696	A	118.53	1	118.87	1
2021-01-04	9:30:02.513750784	A	118.54	1	118.87	1
2021-01-04	9:30:02.683774464	A	118.52	1	118.87	1

Cependant, les données brutes LOB présentées dans les tableaux précédents ne sont pas homogènes, ce qui signifie que le nombre d'enregistrements dans une même durée n'est pas équivalent. Par exemple, il y a 7 transactions dans la seconde de 9:30:01 alors que seulement 2 transactions dans la seconde de 9:30:02. Par souci de simplicité, sans perdre trop d'informations, nous échantillonnons nos données LOB toutes les secondes. Pour les données de cotation, nous enregistrons le meilleur cours vendeur (P_a^1), le meilleur cours acheteur (P_b^1), la meilleure taille du cours vendeur (V_a^1) et la meilleure taille du cours acheteur (V_b^1) pour chaque action à la fin de chaque seconde. Pour les données sur les transactions, nous regroupons toutes les transactions pour chaque action pendant chaque seconde. Nous enregistrons le nombre de transactions (N_t), le nombre total d'actions négociées (V_t) et le prix moyen pondéré par le volume (P_t) de toutes les transactions, défini comme suit :

$$P_t = \frac{\sum_i^{N_t} P_i V_i}{\sum_i^{N_t} V_i} \tag{1.6}$$

Table 1.5: Un exemple des transactions dans la LPP de l'action A de 9:30:01 à 9:30:02 le 4 janvier 2021.

Date	Time	Symbol	Trade Volume	Trade Price
2021-01-04	9:30:01.002522624	A	100	118.92
2021-01-04	9:30:01.002668032	A	3	118.91
2021-01-04	9:30:01.004687872	A	100	118.735
2021-01-04	9:30:01.051353600	A	13	118.94
2021-01-04	9:30:01.092121856	A	200	118.87
2021-01-04	9:30:01.369884928	A	1	118.87
2021-01-04	9:30:01.625717504	A	10	118.86
2021-01-04	9:30:02.081715968	A	102	118.7
2021-01-04	9:30:02.934643712	A	150	118.87

où P_i est le prix de la transaction i, et V_i est le nombre d'actions négociées. N_t est le nombre total de transactions enregistrées pendant la seconde t. La LOB échantillonnée de l'exemple du tableau 1.4 et 1.5 est présentée respectivement dans le tableau 1.6 et 1.7. Nous notons que le *second* signifie le nombre de secondes depuis le début du marché qui est de 9:30:00 sur le marché américain.

Table 1.6: Les cotations échantillonnées à une seconde depuis la LOB sont montrées dans Table 1.4. On note que V_b^1 et V_a^1 sont dans l'unité des lots.

Date	Symbol	seconds	P_b^1	V_b^1	P_a^1	V_a^1
20210104	A	1	118.53	1	118.87	1
20210104	A	2	118.52	1	118.87	1

Table 1.7: Les transactions échantillonnées à une seconde depuis la LOB sont montrées dans Table 1.5.

Date	Symbol	seconds	N_t	V_t	P_t
20210104	A	1	8	427	118.852
20210104	A	2	7	252	118.801

Nous pouvons formuler cette tâche de prédiction de la volatilité réalisée comme suit.

Nous définissons d'abord le logarithme du rendement secondaire de l'action s à la t-ième seconde comme suit

$$r'_{s,t} = log(\frac{P_{s,t}}{P_{s,t-1}}) \tag{1.7}$$

où $P_{s,t}$ est le dernier prix de transaction de s à t.

Étant donné une fenêtre fixe de ΔT secondes, notre cible $Y'_{s,t}$, la volatilité réalisée pour le stock s entre t et $t + \Delta t$ est définie comme suit :

$$Y'_{s,t} = \sqrt{\sum_{i=t}^{t+\Delta T} r_{s,i}^2} \tag{1.8}$$

En utilisant la même notation que l'équation 1.3, cette tâche de prédiction de la volatilité réalisée à court terme peut donc être écrite comme suit

$$\widehat{Y'_{s,t}} = f'(E_t, \theta) \tag{1.9}$$

où f' est le modèle de prédiction. La principale différence avec l'équation 1.3 est que notre objectif est la volatilité réalisée au lieu de la direction du mouvement des actions, ce qui constitue une tâche de régression au lieu d'une tâche de classification.

Contrairement aux modèles univariés existants, nous introduisons une solution multivariée à cette tâche de prédiction de la volatilité réalisée en combinant les données LOB et les données relationnelles dans la section 1.2.2.

1.2.2 Chapter 4: Prédiction Multivariée de la Volatilité Réalisée avec un Réseau Neurones Graphiques

Cette sous-section présente le troisième sujet de cette thèse. L'étude détaillée se trouve dans le chapitre 4.

Motivation et Defis Principaux

La méthode la plus simple pour prévoir la volatilité réalisée à court terme pour les prochaines ΔT secondes à partir de t est de prévoir uniquement avec un fragment de données LOB avant t. Par exemple, nous pouvons simplement utiliser la volatilité réalisée calculée à partir de $t - \Delta T$ comme prédiction de notre objectif : la volatilité réalisée entre t et $t + \Delta T$. Zhou [1996] propose en outre un indicateur plus mature mais intuitif pour cette tâche. Ce processus de prédiction simple peut être écrit comme suit :

$$\widehat{Y'_{s,t}} = f'(D_{s,t}, \theta) \tag{1.10}$$

où $D_{s,t} = E_{s,t}^{\Delta T'}$ désigne les données LOB du titre s entre $t - \Delta T'$ et t. $\Delta T'$ est la fenêtre de retour en arrière et devrait être sur la même échelle que T.

Plus récemment, les chercheurs ont adopté des méthodes de séries chronologiques pour améliorer la prédiction tout en ignorant la connexion entre les actions, comme Brailsford and Faff [1996], Gatheral and Oomen [2010] et Rahimikia and Poon [2020b]. Le modèle est amélioré comme :

$$\widehat{Y'_{s,t}} = f'([D_{s,t_1}, ..., D_{s,t_m}], \theta) \tag{1.11}$$

où $t_1 < ... < t_m = t$. La méthode suppose que la volatilité passée de l'action s a un impact sur la volatilité future de s, mais que la volatilité passée des autres actions n'a aucun impact sur s.

Cependant, comme indiqué dans la section 1.1.3, les actions ne sont pas indépendantes. Nous souhaitons prendre en compte cet effet et prédire la volatilité réalisée avec une approche multivariée.

Étant donné n stocks $s_1, ..., s_n$, nous pouvons réécrire notre modèle final comme :

$$\widehat{Y'_{s,t}} = f'\left(\begin{bmatrix} D_{s_1,t_1} & ... & D_{s_1,t_m} \\ ... & ... & ... \\ D_{s_n,t_1} & ... & D_{s_n,t_m} \end{bmatrix}, \mathcal{G}, \theta\right) \tag{1.12}$$

où $\mathcal{G}$ désigne les relations entre tous les $D_{i,j}$. Elle comprend à la fois la relation cross-sectionnelle $\mathcal{G}_s$ et la relation temporelle $\mathcal{G}_t$.

Dans cette étude, nous utilisons les mêmes relations cross-sectionnelles proposées dans la section 1.1.3, incluant la corrélation entre les stocks, le secteur d'activité et la chaîne d'approvisionnement.

Cette tâche comporte trois grands défis :

1. Comment coder les données LOB de manière significative.

2. Comment construire des graphes basés sur des relations cross-sectionnelle et temporelles en même temps.

3. Comment calculer efficacement sur un grand graphe.

Le premier défi est que les données LOB sont dans un format tabulaire. Par conséquent, nous devons trouver une méthode pour les transformer en un format numérique standardisé pour le calcul. Pour résoudre ce problème, nous concevons un codeur de données LOB (Section 4.4.1) qui transforme les caractéristiques numériques (telles que le prix et le volume) et catégorielles (telles que le symbole d'action) des données LOB en un vecteur de dimension fixe. Les caractéristiques numériques sont codées à l'aide d'un ensemble d'indicateurs intuitifs similaires à ceux de Kercheval and Zhang [2015], et les caractéristiques catégorielles sont codées à l'aide d'une couche d'embedding entraînable (Appendice 2.B.2).

Pour le second défi, nous construisons un graphe dans lequel chaque nœud représente une observation $D_{s,t}$. Pour les relations temporelles, nous fixons s et créons des arêtes entre différents t ; pour les relations cross-sectionnelles, nous construisons des arêtes en fixant t et en reliant différents s. Cette méthode permet de représenter les deux types de relations dans le même graphe (Section 4.5.2).

Bien que la conception du graphe susmentionné soit intuitive, le principal inconvénient est que le graphe devient beaucoup plus grand que le graphe ne contenant que des relations cross-sectionnelle comme le MGRN, puisque nous avons $m \times n$ nœuds dans ce graphe par opposition à n nœuds dans le MGRN. Pour résoudre ce troisième défi, Gilmer et al. [2017] propose que les modèles de graphes soient vus comme des modèles de passage de messages (Appendice 3.A.3). Cela signifie qu'au lieu d'appliquer les opérateurs sur l'ensemble du graphe, nous mettons à jour l'embedding de chaque nœud individuellement, uniquement sur la base de ses voisins, l'information est donc transmise des voisins à lui-même. Cette idée nous permet de former un réseau basé sur un graphe par batch et d'éviter de mettre le graphe en mémoire dans son ensemble. Sur la base de cette généralisation, Shi et al. [2020] propose en outre un opérateur de type Transformer [Vaswani et al., 2017] et obtient des performances de pointe. Nous utilisons cet opérateur de type Transformer pour extraire efficacement des informations d'un grand graphe dans notre recherche.

À notre connaissance, il n'existe pas dans la littérature de méthode de prédiction de la volatilité réalisée multivariée utilisant des relations construites à partir de données externes. Nous ne trouvons également aucune recherche qui propose l'utilisation de réseaux de neurones à graphes dans cette tâche.

Réseau de Graphes à Transformer pour la Prédiction de la Volatilité

Pour améliorer encore notre résultat de prévision de la volatilité réalisée et combler les lacunes de la recherche, en combinant les trois facteurs ci-dessus, nous concevons un réseau de graphes à transformer pour la prediction de la volatilité (GTN-VF, Figure 1.4, détails dans la Section 4.4). Il y a deux composants principaux dans GTN-VF : (1) L'encodeur de données LOB, qui encode les données catégorielles et numériques de la LOB dans un vecteur de longueur fixe. (2) Graph Transformer Network, qui fait la prédiction en agrégeant les embeddings de nœuds d'un nœud et de ses voisins d'une manière efficace.

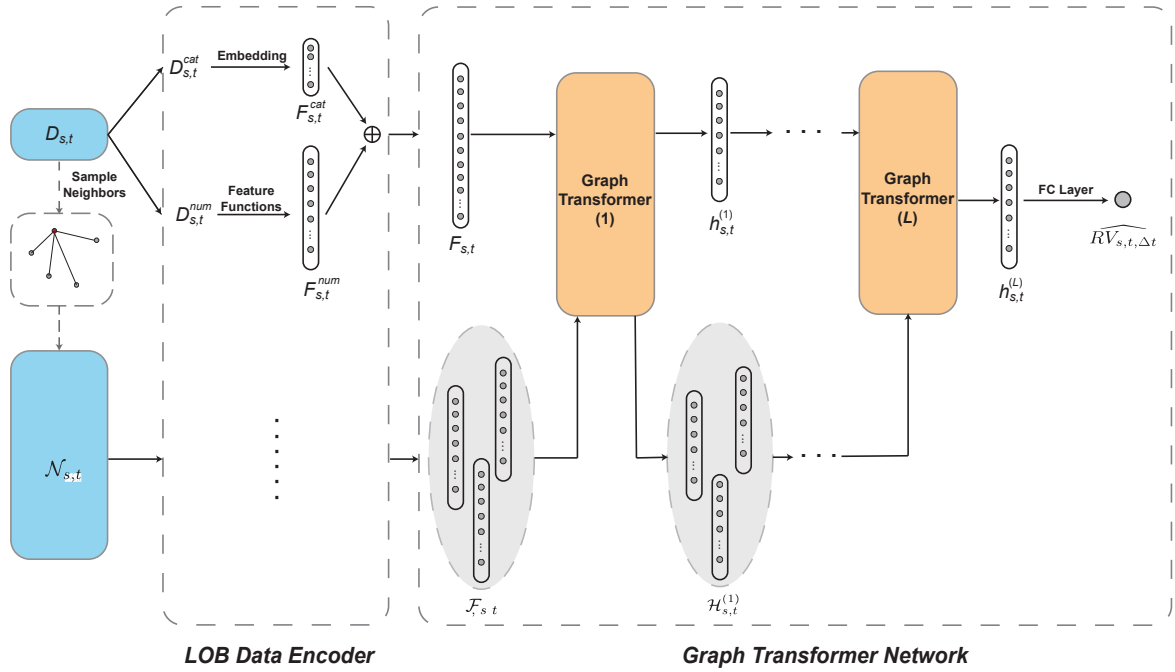


Figure 1.4: La structure de notre Réseau de Graphes à Transformer pour la Prédiction de la Volatilité (GTN-VF)

Par le biais d'expériences basées sur environ 500 actions de l'indice S&P 500, nous trouvons de meilleures performances pour notre modèle que pour les autres benchmarks sur tous les horizons de prédiction ΔT. Le tableau 1.8 présente un résumé du résultat RMSPE[5] basé sur l'ensemble de test, et les résultats complets sont présentés dans la section 4.5.4. Par exemple, notre GTN-VF avec les trois relations cross-sectionnelles et la relation temporelle surpasse HAR-RV [Corsi, 2009] de 3,25% en RMSPE lorsque ΔT est égal à 10 minutes, il surpasse également TabNet [Arık and Pfister, 2021] de 1,91% dans la même condition. Pour prouver l'efficacité de différentes relations, nous incluons une analyse détaillée des performances pour différents types de relations temporelles et cross-sectionnelles (Tableau 4.5). En outre, nous ajoutons quelques études d'ablation pour montrer d'où vient l'amélioration de notre modèle (Section 4.6).

Nous fournissons les détails de cette étude dans le chapitre 4.

1.3 Part III: Au-delà de la finance - Applications de la NLP dans l'art contemporain

Dans cette partie, nous présentons une application des méthodologies NLP dans l'art contemporain. Bien que n'étant pas un problème lié à la finance, ce sujet applique et développe les connaissances de la première partie sur la NLP et l'apprentissage par transfert.

[5]Root Mean Square Percentage Error

Table 1.8: Résumé du résultat RMSPE (le plus bas est le meilleur) de GTN-VF et des modèles de base.

	ΔT	600s	1200s	1800s
Baseline models	Naïve Guess	0.2834	0.2628	0.2364
	HAR-RV [Corsi, 2009]	0.2612	0.2061	0.1939
	lightGBM [Ke et al., 2017]	0.2492	0.2035	0.1963
	MLP [Rumelhart et al., 1985]	0.2514	0.2308	0.1999
	TabNet [Arık and Pfister, 2020]	0.2478	0.1996	0.2019
Proposed model ans its variants	Vanilla GTN-VF	0.2498	0.2251	0.2160
	GTN-VF Cross FC	0.2382	0.2196	0.2046
	GTN-VF Temp FC	0.2358	0.1921	0.1853
	GTN-VF Cross FC + Temp FC	0.2306	0.1917	0.1802
	GTN-VF Cross Sector	0.2422	0.2248	0.2091
	GTN-VF Cross Supply Chain	0.2411	0.2244	0.2000
	GTN-VF	**0.2287**	**0.1892**	**0.1798**

1.3.1 Contexte et Objectif

Le marché de l'art contemporain connaît une croissance rapide depuis le début de ce siècle [Kräussl et al., 2016]. Traditionnellement, les collectionneurs s'appuient sur les interactions sociales, les conseils de professionnels et les médias pour trouver les œuvres d'art qu'ils souhaitent acquérir. Avec un nombre croissant d'artistes et d'œuvres d'art, recommander des œuvres d'art appropriées à des collectionneurs ayant des goûts différents devient un problème difficile. Récemment, avec le développement rapide de l'intelligence artificielle, les recherches commencent à moderniser et à automatiser ce processus de découverte de l'art.

La plupart des chercheurs se concentrent sur l'aspect visuel des œuvres d'art pour trouver les similitudes entre elles. Elgammal et al. [2018] classent les styles des peintures à l'aide de certains modèles fréquemment utilisés en computer vision. Kim et al. [2018] étendent cette méthode pour regrouper les œuvres d'art selon des concepts et des principes artistiques prédéfinis en plusieurs dimensions. Cependant, ces recherches précédentes ne se concentrent que sur l'aspect visuel des œuvres d'art contemporaines, car l'art contemporain peut aller au-delà de l'esthétique. Par exemple, la pratique de l'artiste et les influences utilisées par l'artiste peuvent toutes deux avoir une forte influence sur le goût pour l'art.

Par conséquent, les informations textuelles facilement disponibles telles que les biographies d'artistes retiennent notre attention. Dans ce travail, nous sommes intéressés à trouver les relations entre les artistes à travers leurs biographies. Plus concrètement, étant donné les biographies de deux artistes, nous classons s'ils sont connectés. Nous montrons un exemple de cette tâche dans le tableau 1.9.

Nous pouvons formuler ce problème de découverte de connexions d'artistes comme un problème de classification binaire. Supposons que nous ayons deux artistes A_i, A_j et que leurs biographies soient désignées par B_i, B_j. Une équipe de professionnels de l'art peut déterminer s'ils sont liés en se basant sur différents aspects, notamment le contexte, les thèmes, le style, les techniques,

Table 1.9: Un exemple de nos données de paires de biographies d'artistes labélisées. *bio_a* est la biographie du *artiste_a* et *bio_b* est la biographie du *artiste_b*. Le *label* est annoté manuellement par une équipe de professionnels de l'industrie artistique et indique si deux artistes sont connectés (1 si connecté et 0 si non connecté). Notre objectif est d'entraîner un modèle capable de dire si deux artistes sont liés à partir de leurs biographies.

artist_a	artist_b	bio_a	bio_b	label
Carl Edouard Keita	Kudzanai Violet Hwami	Born in 1992 in Abidjan, Carl-Edouard Keita now lives and works in New York. A 2021 graduate of the New York Academy of Art, Carl-Edouard Keita also won the prize for best draughtsman for his graduation work, some of which is presented in this group exhibition. Carl-Edouard Keita discovered the history of African art during his economics studies in Atlanta, through a course offered at his university. As he describes it himself, this discovery was a real aesthetic revelation for him.	Having fled her homeland due to the political unrest and turmoil when she was a child, Zimbabwe-born painter Kudzanai-Violet Hwami expresses her personal experiences of dislocation, displacement and fragmentation through her striking figurative paintings. The artist is interested in the collapsing of geography and time and space symptomatic of a globalised world and high-speed internet, through which both people and information can travel quickly.	1
Sun Xun	Cheng Xinhao	Sun Xun was born in 1980 in Fuxin, Liaoning province, China. Currently lives and works in Beijing. Recent and past histories, intransigent conflicts and tensions, sequential flashes of hand-created images of these are the irrevocable features of Sun Xun's artistic practice that fuses the line between art and animation. A graduate from the Printmaking Department of the China Academy of Arts in 2005, Sun Xun was a professor at the prestigious Academy before founding in 2006 his own Animation Studio. His work primarily involves making images using various materials such as colour powder, woodcuts and traditional ink, and collating these to produce a film, which is often presented in an immersive setting.	Cheng Xinhao (b.1985, Yunnan, China). After receiving his PhD on Chemistry from Peking University in 2013, Cheng continued his career as a photographer, investigating on the issues in the modernization, the construction of knowledge as well as the production of space in Chinese society.	0

etc. Nous utilisons $Y_{i,j}$ pour désigner cette vérité de base, avec

$$Y_{i,j} = \begin{cases} 1 & \text{si } A_i \text{ et } A_j \text{ sont connectés} \\ 0 & \text{sinon} \end{cases}.$$

Cependant, notre base de données compte des milliers d'artistes et il est impossible de déterminer toutes les relations par annotation manuelle. Par conséquent, dans cette étude, nous nous intéressons à l'établissement automatique des relations entre eux à partir de leurs biographies. Nous pouvons décrire ce processus comme suit

$$p_{i,j} = f(B_i, B_j, \theta)$$

où $p_{i,j}$ désigne notre probabilité prédite d'une connexion entre A_i et A_j. f est notre modèle de prédiction et θ désigne les paramètres pouvant être entraînés.

Étant donné f, notre objectif est de trouver θ qui minimise la perte d'entropie croisée [Good, 1992] définie comme suit

$$-\sum_{i,j} Y_{i,j} \cdot \log p_{i,j} + (1 - Y_{i,j}) \cdot \log(1 - p_{i,j}).$$

1.3.2 Chapitre 5: Cartographier le monde de l'art contemporain avec ArtLM : un modèle NLP spécifique à l'art

Cette sous-section présente le quatrième sujet de cette thèse. L'étude détaillée se trouve dans le chapitre 5.

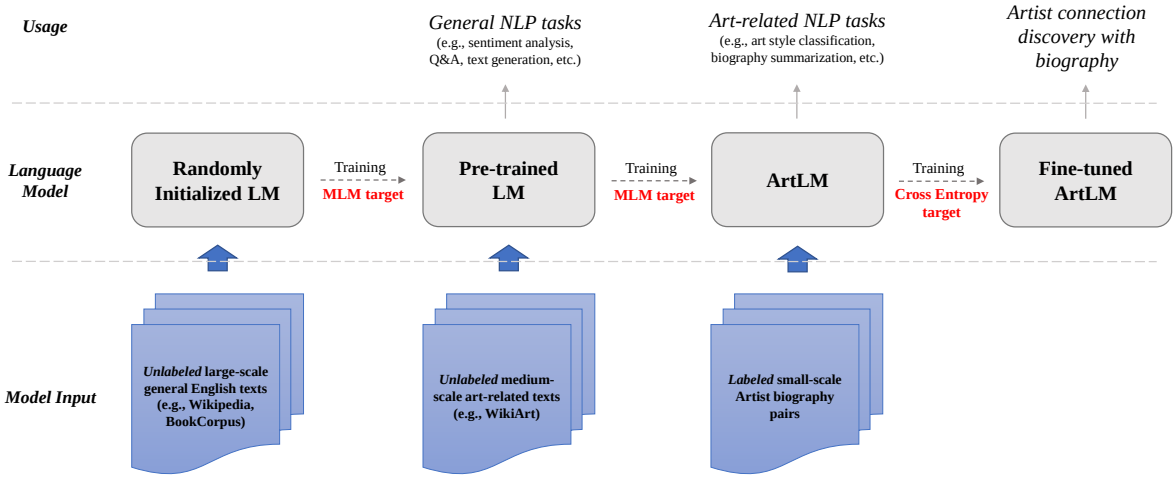


Figure 1.5: Aperçu de notre méthode de découverte des connexions entre artistes, comprenant deux phases de pré-formation non supervisées et une phase de fine-tuning supervisée.

Motivation et Defis Principaux

Nous cherchons à résoudre deux défis principaux dans cette tâche :

1. Comment classer efficacement le lien entre deux artistes à partir de leur paire de biographies étiquetées.

2. Comment améliorer la précision de la classification à partir de textes non labélisés relatifs à l'art.

Pour trouver des solutions au premier défi, nous pouvons affiner certains modèles linguistiques (LM) pré-entraînés en suivant la même idée que celle présentée dans la section 1.1.2. Cependant, à la différence des recherches précédentes, notre entrée est une paire de phrases au lieu d'une seule. Dans ce cas, nous concaténons la paire de phrases tout en les séparant avec un caractère spécial pour former une seule phrase et continuer le processus de fine-tuning (Section 5.4.3).

Pour le deuxième défi, nous ajoutons une autre étape de pré-entraînement avant le fine-tuning. Dans cette procédure, nous continuons à pré-entraîner les modèles pré-entraînés originaux avec les textes non labélisés relatifs à l'art recueillis sur WikiArt[6] en utilisant les mêmes cibles d'entraînement que les modèles originaux. Nous appliquons également cette méthode à différents modèles linguistiques pré-entraînés disponibles publiquement afin de prouver la robustesse de cette approche.

Bien que ce schéma de pré-entraînement et de fine-tuning ait été appliqué dans des domaines tels que la finance, nous ne voyons aucune application dans les arts à notre connaissance.

[6]https://www.wikiart.org/

ArtLM pour la découverte de connexions entre artistes

Pour prendre en compte tous les facteurs ci-dessus et combler les lacunes de la recherche, nous proposons un cadre NLP générique (appelé ArtLM, Section 5.4.2, Figure 1.5) pour découvrir les liens entre les artistes contemporains à partir de leurs biographies. Nous continuons d'abord à pré-entraîner les modèles de langage existants avec des textes non étiquetés liés à l'art afin d'ajouter des connaissances du domaine artistique aux modèles généraux existants. Nous affinons ensuite le modèle avec les paires de biographies labélisées par les professionnels de l'art. Cette procédure est robuste et indépendante du choix des modèles de base, comme le prouvent les expériences basées sur plusieurs modèles de base, tels que BERT [Devlin et al., 2018], DistilBERT [Sanh et al., 2019] et RoBERTa [Liu et al., 2019].

Table 1.10: Résultats d'expériences résumés des variantes d'ArtLM et des modèles de base

model	accuracy	F1	fine-tuning time (min. / fold)	pre-training time (min.)
Random Guess - 1	0.500	0.000	-	-
Random Guess - 0.5	0.500	0.500	-	-
word2vec embedding	0.539	0.196	-	-
BERT embedding	0.581	0.320	-	-
FT-Base-DistilBERT	0.838	0.826	6	-
FT-Base-BERT	0.844	0.832	12	-
FT-Base-RoBERTa	0.844	0.824	12	-
FT-Art-DistilBERT	0.846	0.826	6	10
FT-Art-BERT	**0.856**	**0.840**	12	18
FT-Art-RoBERTa	0.854	0.834	12	21

Grâce à des expériences approfondies, nous démontrons que notre ArtLM atteint une précision de 85,6% et un score F1 de 84,0%, et qu'il surpasse tous les autres modèles de base dans une configuration de validation croisée à 5 volets. Un résumé des résultats de précision et de score F1 est présenté dans la Table 1.10, et les résultats détaillés complets sont présentés dans la Section 5.4. En comparant les résultats des modèles fine-tuné (à la fois `FT-Base` et `FT-Art`) et les modèles de base, nous constatons que notre approche de fine-tuning supervisé surpasse largement les méthodes de corrélation basées sur l'embedding. Nous prouvons également que l'ajout de connaissances spécifiques au domaine en poursuivant le processus de pré-entraînement peut contribuer à améliorer les performances du modèle, si nous nous concentrons sur les écarts de précision entre les modèles de base et les modèles artistiques.

Le processus générique ArtLM peut facilement être généralisé à d'autres domaines, y compris la finance. Par exemple, nous pouvons continuer à pré-entraîner un LM avec des documents financiers non labélisés, puis faire des recommandations d'actions ou des analyses de sentiments avec des rapports financiers labélisés manuellement.

Nous fournissons les détails de cette étude dans le chapitre 5.

Part I

Stock Return Prediction with Financial News

Chapter 2

Stock Movement Prediction with Contextualized Embedding from BERT

Note:

- This chapter is authored by Qinkai Chen.
- This chapter is available as preprint *arXiv:2107.08721*.
- The method described in this chapter is applied in the trading system within ExodusPoint Capital Management.

Abstract

News events can greatly influence equity markets. In this paper, we are interested in predicting the short-term movement of stock prices after financial news events using only the headlines of the news. To achieve this goal, we introduce a new text mining method called Fine-Tuned Contextualized-Embedding Recurrent Neural Network (FT-CE-RNN). Compared with previous approaches which use static vector representations of the news (static embedding), our model uses contextualized vector representations of the headlines (contextualized embeddings) generated from Bidirectional Encoder Representations from Transformers (BERT). Our model obtains the state-of-the-art result on this stock movement prediction task. It shows significant improvement compared with other baseline models, in both accuracy and trading simulations. Through various trading simulations based on millions of headlines from Bloomberg News, we demonstrate the ability of this model in real scenarios.

Contents

2.1 Introduction

Stock movement prediction has attracted a considerable amount of attention since the beginning of the financial market, although the stock prices are highly volatile and non-stationary. Fama [1965] showed that the movement of stock prices can be explained jointly by all known information.

With the development of Internet, there is a rapid increase in the amount of financial news data (Figure 2.1), and more studies have been done to use computational methods to predict stock price changes based on financial news [Oliveira et al., 2013; Si et al., 2013; Xie et al., 2013; Nguyen and Shirai, 2015; Luss and d'Aspremont, 2015; Rekabsaz et al., 2017; Ke et al., 2019; Li et al., 2020a; Coqueret, 2020]. Following previous works, we explore an accurate method to transform textual information into stock movement prediction signal.

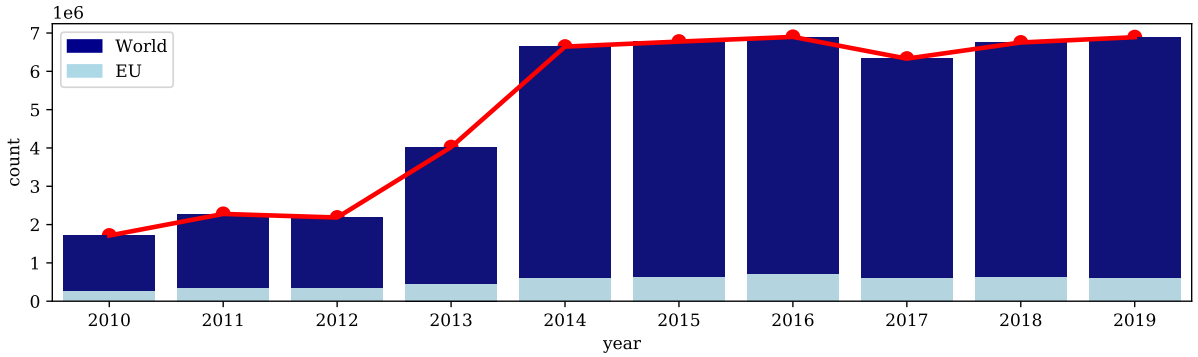


Figure 2.1: The number of news recorded by Bloomberg each year. For both world and European countries, there is a significant increase in the number of news from 2014.

Schumaker and Chen [2009] use a classical feature engineering method to predict the market behavior. More recently, deep learning methods are more frequently applied on this task. Ding et al. [2014,0] employ structured representations to normalize a news then apply a Convolutional Neural Network (CNN) on this formulation. Hu et al. [2018] apply an improved Transformer model [Vaswani et al., 2017] to handle all the words in the raw text simultaneously to predict the forward return. Luss and d'Aspremont [2015] propose a statistical learning method to combine text data and the historical returns. Xu and Cohen [2018] improve the idea from Luss and d'Aspremont [2015] by designing a deep neural network. Ke et al. [2019] adopts a simple but effective classification method combining both regression and Term-Frequency Inversed Document Frequency (TF-IDF) model [Jones, 1972]. Del Corro and Hoffart [2020] further introduce an unsupervised method to extract market moving events from text data, which overcomes the problem of lacking reliable labels in financial data. Wan et al. [2021] recently propose a sentiment propagation model to jointly predict the price movement of all the stocks.

Before applying computational models mentioned above, the first step usually involves converting words into fixed-length vectors (these fixed-length vectors are called embeddings in natural language processing, details are presented in Section 2.2.2). Mikolov et al. [2013b] propose Word2Vec model to embed words based on words co-occurrence prediction and Pennington et al. [2014] propose a similar GloVe model based on words co-occurrence frequencies. However, both methods can only generate static non-contextualized embeddings. It means that a word is converted to the same vector no matter its meaning or its context. This approach ignores

the fact that the meaning of a word can change significantly in different contexts, which impacts the performance of the model. As most of the previous researches rely heavily on static non-contextualized embedding such as Word2Vec or GloVe, there can be accuracy loss.

Cer et al. [2018] and Peters et al. [2018b] propose methods to generate contextualized embeddings by jointly considering all the words together. They published their models trained on large English corpus. Although effective, the model is fixed and does not contain domain-specific knowledge in finance.

Devlin et al. [2018] introduce a general-use language model called Bidirectional Encoder Representations from Transformers (BERT). It is one of the most promising models in natural language processing and it showed significant improvement on multiple benchmarks [Wang et al., 2018]. The BERT model is pre-trained on a very large scale of textual data to leverage all the features in natural languages, and it also provides the ability to fine-tune this pre-trained model with domain-specific data without needing to start from the scratch. As we have a large amount of financial texts, we can use them to add financial knowledge to the BERT model and generate contextualized embeddings with domain-specific knowledge in finance from BERT.

In addition, previous researches evaluate the performance of the models based on the accuracy calculated on all the news [Ding et al., 2015; Xu and Cohen, 2018; Hu et al., 2018]. However, this evaluation metric does not reflect the real capability of the model since investors only care about the news which can move the market significantly. The news identified as neutral have little impact on investors' decisions, as investors will simply ignore the news if they are classified as neutral.

In this paper, aiming to solve the problems mentioned above, we want our research to have the following characteristics:

- It adopts the contextualized embeddings instead of the static embeddings.
- The contextualized embeddings contain financial domain-specific knowledge.
- Our model has a better prediction on the news which can move the market significantly.

Hence, based on previous work (Sec. 2.2), we propose Fine-Tuned Contextualized-Embedding Recurrent Neural Network (FT-CE-RNN) to predict the stock price movement based on the headlines (Sec. 2.3). Using Bloomberg News dataset (Sec. 2.4), this model generates contextualized embeddings with domain-specific knowledge using all the hidden vectors from the BERT model fine-tuned on financial news. Then FT-CE-RNN uses a recurrent neural network (RNN) to make use of the generated embeddings. (Sec. 2.5) We also introduce a new evaluation metric which calculates the accuracy on various percentiles of the prediction scores on the test set instead of the whole test set to better incorporate investors' interests. Our experiments show that our FT-CE-RNN achieves a state-of-the-art performance compared with other baseline models. We also evaluate our model by running trading simulations with different trading strategies. (Sec. 2.6)

2.2 Related Work

2.2.1 Stock Movement Prediction

Stock movement prediction is a widely discussed topic in both finance and computer science communities. Researchers predict the stock market using all available information, including historical stock price, company fundamentals, third-party news sentiment score, financial news, social media texts and even satellite images.

The most classical method is to use the historical stock prices to predict the future prices. Kraft and Kraft [1977]; Sonsino and Shavit [2014]; Ariyo et al. [2014]; Kroujiline et al. [2016]; Jiang et al. [2018] use time series analysis techniques to extract the patterns of historical returns, and predict the future stock movement based on these patterns. More recently, researchers start to use neural networks to analyze this pattern [Kohara et al., 1997; Adebiyi et al., 2012; Tashiro et al., 2019; Chen and Ge, 2019; Mäkinen et al., 2019; Bai and Pukthuanthong, 2020].

Financial analysts usually use companies' fundamental indicators from their financial reports to predict the stocks' prices in the future [Zhang and Yan, 2018]. This includes the use of earnings per share (EPS) [Patell, 1976], debt-to-equity (D/E) ratio [Bhandari, 1988], cash flow [Liu et al., 2007], etc. Nonejad [2021] builds a conditional model to jointly consider historical prices and financial indicators.

With the rapid development of the natural language processing and deep learning, researchers start to focus on predicting stock movement based on textual data, such as financial news and social media texts, which were viewed as difficult to process systematically. Financial news data vendors such as Bloomberg, ThomsonReuters and RavenPack all include their proprietary sentiment analysis on the news. Coqueret [2020] thoroughly analyzes the sentiment classification given by Bloomberg and finds disappointing results on its predicting power. Ke et al. [2019] include the RavenPack's proprietary score as a benchmark and find it less performing than other models.

Hence, more researchers propose their own natural language processing models to improve the predictability based on financial news. Luss and d'Aspremont [2015] propose an improved Kernel learning method to extract the features in the texts. Ke et al. [2019] use statistical learning methods to determine the sentiment of the words in the news. More recently, computer scientists begin using state-of-the-art deep learning techniques to solve this problem. Ding et al. [2015]; Hu et al. [2018]; Xu and Cohen [2018]; Li et al. [2020a] propose different deep learning models to extract information from both financial news and social media texts.

There are also other researches which use uncommon data to predict future stock prices. The data includes key people compensation [Cooper et al., 2016], satellite images [Donaldson and Storeygard, 2016] and the pictures included in the news [Obaid and Pukthuanthong, 2021].

2.2.2 Contextualized Embedding

In the natural language processing, the first step usually involves transforming words or sentences into fixed-length vectors to allow numerical computations. These fixed-length vectors are known

as the embeddings of the words or the sentences.

Historically, researchers use one-hot embeddings [Stevens et al., 1946] to encode words. However, the dimension of the one-hot embedding is large since each unique word takes one dimension. Hence, researchers start to develop methods to make the word embeddings denser.

The most widely used methods for word embedding are Word2Vec [Mikolov et al., 2013b] and GloVe [Pennington et al., 2014], both of which are based on word co-occurrences. Such models take in a large number of texts and output a fixed vector for each word in the texts. The more frequently the two words co-occur, the more correlated two embeddings are. Once the model is trained, the embeddings of the words no long change, therefore we call these embeddings static embeddings. Such model generates the same embedding for one word no matter its context, although the meanings of the words can depend on the context in which this word occurs.

Researchers propose contextualized embeddings to solve this issue. Instead of taking only one words as input, the contextualized embedding model accepts the whole sentence as its input. The model then generates the embeddings for each word in the sentence by jointly considering the word and all the other words in the sentence. Cer et al. [2018] proposes Universal Sentence Encoder (USE) to encode the whole sentence contextually. However, USE only gives the embedding of the sentence as a vector without specifying the embedding of each word. Peters et al. [2018b] proposed Embeddings from Language Models (ELMo) to embed words based on their linguistic contexts, but ELMo is trained on general English language, making the generated embeddings lack of financial domain-specific knowledge. However, Yang et al. [2020b] showed domain-specific model outperforms general models in most of the tasks.

Recent researches on general-use language model such as BERT [Devlin et al., 2018] and XLNet [Yang et al., 2019] reported impressive result on all natural language processing tasks. More interestingly, these models propose a way to fine-tune its pre-trained model on general English with domain-specific data.

Hence, we propose FT-CE-RNN to complement existing researches. FT-CE-RNN generates contextualized embeddings with domain-specific knowledge from the BERT model, it then makes the stock movement prediction based on this more advanced embedding.

2.3 Problem Formulation

Suppose that we have a stock s with a headline $h_{s,t}$ recorded at time t, and the headline has N words, we denote them by $w_1, ..., w_N$. We first need to transform them into fixed-length embeddings. Suppose that the length of the embedding is l_e, this process can be written as:

$$Emb_i = f_s(w_i) \tag{2.1}$$

where $Emb_i \in \mathbb{R}^{l_e}$ is the embedding of the word w_i and f_s denotes the static embedding encoder. In this case, each word has a fixed embedding independent of its context.

A contextualized embedding encoder has the same function of converting a word into a vector, unless it considers all the words in a sentence together. We use f_c to denote this contextualized embedding encoder, it can be written as:

$$Emb_i = f_c(w_i|w_1, ..., w_N) \tag{2.2}$$

We concatenate the embeddings of all words to get the embedding of the headline $h_{s,t}$. We define the embedding of this headline as:

$$Emb_{h_{s,t}} = [Emb_1, ..., Emb_N] \tag{2.3}$$

where $Emb_{h_{s,t}} \in \mathbb{R}^{l_e \times N}$.

Following the work of Luss and d'Aspremont [2015]; Ding et al. [2015]; Xu and Cohen [2018]; Ke et al. [2019], we formulate the stock movement prediction as a binary classification task[1]. It means that we predict if a news has a **positive** impact or a **negative** impact on the related stock.

We define its market-adjusted return $r_{s,t}$ as

$$r_{s,t} = \frac{P_{s,t+\Delta t}}{P_{s,t}} - \frac{P_{m,t+\Delta t}}{P_{m,t}} \tag{2.4}$$

where $P_{s,t}$ denotes the price of stock s at time t and $P_{m,t}$ denotes the value of the equity index at time t.

We notice that it is necessary to use market-adjusted return instead of the simple return, as the information contained in the price change is partially due to the information related to this stock (such as news), and also partially due to the information related to other macroeconomic information (such as interest rate, fiscal policies, etc.). As the macroeconomic effect impacts all stocks, it can be explained by a weighted sum of all stocks, such as market index. We can simply remove this impact by subtracting the index return from the stock return, and this adjusted return can better explain the impact of the news.

Most researches in the stock movement prediction based on news simply suppose that all the news induce the market change in the same way [Luss and d'Aspremont, 2015; Xu and Cohen, 2018; Hu et al., 2018; Ke et al., 2019], and therefore use the same Δt to calculate the forward returns of all news. However, Fedyk [2018] suggests that the news during the trading hours and the news outside the trading hours have different market impact.

Hence, for different news, we choose different Δt. For example, for the news published during the trading hours, the price can change in several minutes after the arrival of the news. In this case, we can choose a smaller Δt of several minutes or several hours. However, for the news published out of the trading hours, as the market is already closed, we cannot observe the effect of the news until the next market open. Therefore, we need to choose a Δt of several days.

We define the stock price movement as:

$$Y_{s,t} = \begin{cases} 1, & r_{s,t} > 0 \\ 0, & r_{s,t} \leq 0 \end{cases} \tag{2.5}$$

The goal is to predict $Y_{s,t}$ from the embeddings of the headlines $Emb_{h_{s,t}}$. It can be written as:

$$\hat{Y}_{s,t} = g(Emb_{h_{s,t}}) \tag{2.6}$$

[1] We can also formulate this problem as a multi-class classification task [Pagolu et al., 2016]. It means that, instead of classifying a news into positive news and negative news, we can classify them into positive news, negative news or neutral news, making it a three-class classification task. Moreover, we can classify a news into different return intervals, making it a multi-class classification task. However, we find that the performance with multi-class classification setup is less impressive. We provide the details of this study in Section 2.6.7.

where g represents the prediction model.

2.4 Data

2.4.1 Data Description

The dataset that we use is Bloomberg News[2]. In this dataset, each entry contains a *timestamp* showing when this news is published, a *ticker* which tells the stock related to this news and the *headline* of this news. In addition to the necessary information above, there are two fields given by Bloomberg's proprietary classification algorithm. The *score* is among -1, 0 and +1, which indicates if the news is either negative, neutral or positive. *Confidence* is a value between 0 and 100 related to *score*. A higher *confidence* value means that Bloomberg's model is more sure about its *score*. Bloomberg's classification will serve as one of the benchmarks for our prediction model. We present a sample dataset in Table 2.1.

Table 2.1: A small sample from the Bloomberg News dataset

Headline	TimeStamp	Ticker	Score	Confidence
1st Source Corp: 06/20/2015 - 1st Source announces the promotion of Kim Richardson in St. Joseph	2015-06-20 05:02:04.063	SRCE	-1	39
Siasat Daily: Microsoft continues rebranding of Nokia Priority stores in India opens one in Chennai	2015-06-20 05:14:01.096	MSFT	1	98
Rosneft, Eurochem to cooperate on monetization at east urengoy	2015-06-20 08:01:53.625	ROSN RM	0	98

We need to address that in our dataset, we only have the headlines of the news instead of the whole article.

In our experiment, we use the news data on all the stocks from the STOXX Europe 600 index[3] which represents the 600 largest stocks of the European market. In order not to overfit, we select a short period (from 01/01/2016 to 30/06/2018) as our training set and another short period (from 01/07/2018 to 31/12/2018) as our development set. We tune the parameters of the models only based on this subset of the data, we then test on the whole period (from 01/01/2011 to 31/12/2019) on a 3-year rolling basis. It means that we generate the classification result of the year y using model trained between $y-3$ and $y-1$, without varying the parameters initially obtained. Detailed statistics of the dataset are in Table 2.2. We can also find the number of news in each year from Figure 2.1.

In addition to the Bloomberg news dataset, we also use the cooperate action adjusted share

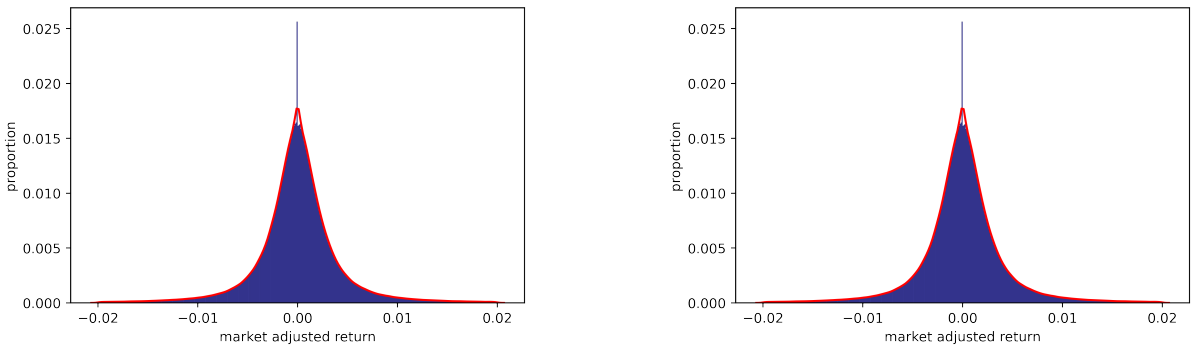
[2]https://www.bloomberg.com/professional/product/event-driven-feeds/

[3]https://www.stoxx.com/index-details?symbol=SXXP

[4]We note that we do not simply apply our model trained on the training set on the test set. The training set and the development set are only used to find the hyper-parameters for our models. We generate the scores on test set using different models trained on a 3-year rolling basis, as described in section 2.4.1

Table 2.2: Statistics of the Bloomberg News dataset

	Train	**Dev**	**Test**[4]
Total news	1,616,922	316,944	5,253,345
Word counts	17,650,629	3,554,324	55,410,309
From date	01/01/2016	01/07/2018	01/01/2011
End date	30/06/2018	31/12/2018	31/12/2019



(a) In trading hours

(b) Outside trading hours

Figure 2.2: The distribution of the $r_{s,t}$ for both the news in the trading hour and those outside the trading hour. The returns are calculated on all the news in the training set. For the news in the trading hour (Figure (a)), we use the forward return of 30 minutes ($\Delta t_i = 30$minutes), and for those outside the trading hour (Figure (b)), we use the forward return of 1 day($\Delta t_e = 1$day). We can see that the market adjusted returns are symmetrically distributed.

prices at market close and intraday minute bar share prices for all the stocks. The share prices are used to label our data and simulate our trading strategies.

2.4.2 Data Labelling

For this supervised learning task, we need to provide our model with the ground-truth as its target. However, our data is simply the headlines of the financial news, it does not tell us if a news is positive or negative. Hence, we need to give each news in the training set a label (positive or negative) before training our model.

The intuition behind our labelling method is simply that if a news is positive, the investors will start to overbuy the related stocks, the stock should therefore outperform the market and vice versa. We use Equation 2.4 to calculate the market adjusted return for each news. As discussed in Section 2.3, we consider the different effects of the news which occur during the trading hour[5] and those outside the trading hour. We use different Δt (Eq. 2.4) for the news during the trading hours and outside the trading hours. For the news inside trading hours, we choose Δt_i and for the news outside the trading hours, we use Δt_e.[6] We show an example of the distribution of the returns for all the news in Figure 2.2.

[5]9:00 to 17:30 CET for most European markets

[6]For Δt_i, we test 5,30,60,120 minutes; for Δt_e, we test 1,2,3,5 days.

We label the data based on the market-adjusted return mentioned above. As our task is to identify market-moving news which investors focus on, we need to remove news which do not have significant impact on the price. As we found that the return distribution is quite balanced for the training set (Figure 2.2), we simply label the 15% news with most positive return as 1 and the 15% news with most negative return as 0. This can be written as:

$$Y_{s,t,train} = \begin{cases} 1, & r_{s,t} \textit{ in top } 15\% \\ 0, & r_{s,t} \textit{ in bottom } 15\% \\ \textit{removed}, & \textit{otherwise} \end{cases} \tag{2.7}$$

where $Y_{s,t,train}$ is the label for the news $e_{s,t}$ for stock s recorded at time t in the training set.

However, for development and test sets, we label **all** news with positive return as 1 and all news with negative return as 0. This can be written as:

$$Y_{s,t,dev/test} = \begin{cases} 1, & r_{s,t} > 0 \\ 0, & r_{s,t} \leq 0 \end{cases} \tag{2.8}$$

This difference in labelling is simply to avoid the information leakage. In real-life scenario we cannot know the forward return of a news when it is published. Therefore, we cannot know if the news is in the top 15-percentile or the bottom 15-percentile. We are supposed to give each news a score when it is published regardless its forward return.

However, we can choose to exclude a news according to its score when calculating the metrics, as this information is available immediately after we receive the news. We use this idea to construct different test sets to evaluate of model. We present the details in Section 2.6.2.

2.5 Prediction Model

The general procedure of this stock movement prediction task is shown in Figure 2.3. In this procedure, we first input the data into different language models to get different news embeddings. We then use a prediction model such as RNN to generate predictions for all news. We mainly focus on the language model and the prediction model in this study.

There are two main components in our procedure: a contextualized embedding encoder from BERT model and a Recurrent Neural Network (RNN) which takes the contextualized embedding as input and outputs the classification probability for both classes.

2.5.1 Contextualized Embedding Encoder from BERT

The Bidirectional Encoder Representations from Transformers (BERT) proposed by Devlin et al. [2018] is a widely used language model in the natural language processing applications. It is first pre-trained on very large scale data (WikiBooks[7] and Wikipedia[8]) to learn the basic characteristics of a language. After this pre-training phase, we can obtain a pre-trained base

[7] https://en.wikibooks.org/
[8] https://en.wikipedia.org/

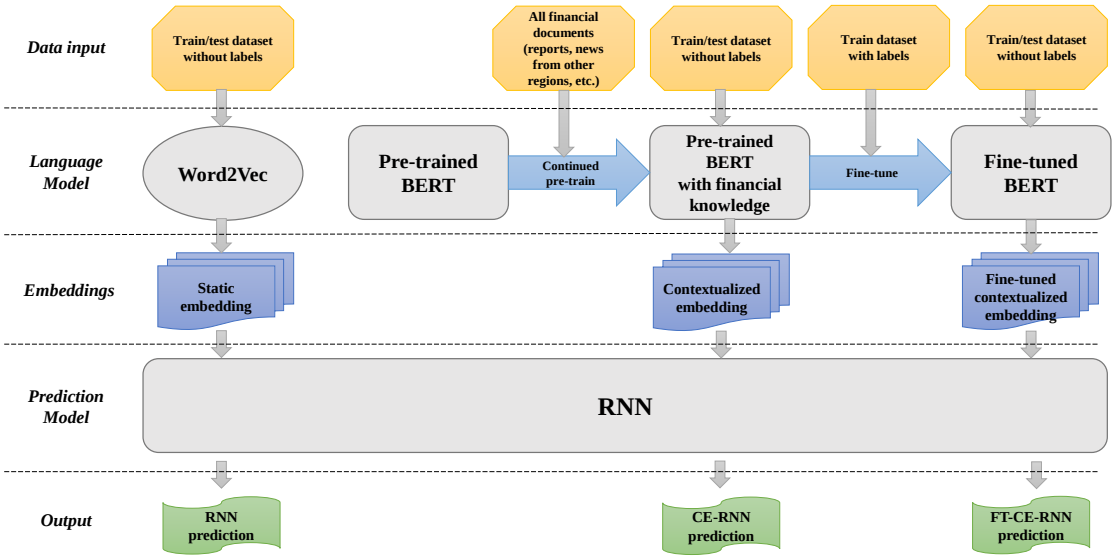


Figure 2.3: An illustration of the stock movement prediction procedure.

BERT model for all other downstream tasks (such as text classification). This pre-training process is computationally intensive, we use the pre-trained BERT model published by Google[9]. We continue the pre-training process using a large number of financial documents including financial reports and the news headlines from all the regions (only the news associated with the stocks from our European stock universe are used in the following fine-tuning process). We have 100 times more tokens in these unlabeled documents than the labeled headlines associated with European stocks. This process is to ensure that the model has a financial language context.

Figure 2.4 shows the structure of BERT model. It has L layers and each layer has N nodes, each node is a Transformer [Vaswani et al., 2017]. The first layer takes a tokenized headline as input and the BERT model generates $N \times L$ hidden vectors, denoted by $T_{i,j}$. The first token at the first layer is a special [CLS] token reserved for fine-tuning.

For a specific downstream task, we can fine-tune the base BERT model suitable for this specific task. It means that we do not initialize the parameters of BERT model randomly, we use the pre-trained BERT model as our initial state instead. We update the parameters in the base model with our domain-specific data. This approach adds domain knowledge to the large scale language model [Yang et al., 2020b], it can help the BERT model better understand the texts in specific situations. In our case, we can fine-tune the base BERT model with our labelled financial news data mentioned in Section 2.4.2 to make it specialize in financial texts.

The fine-tuning process is straightforward. We input the class label (0 or 1) together with the tokenized headline into the first layer of a pre-trained BERT model. We set the target to the class label and loss function to cross-entropy, defined as:

$$loss = \sum Y_i ln(P_i^+) + (1 - Y_i) ln(1 - P_i^+) \tag{2.9}$$

where Y_i is the label for the news i and P_i^+ denotes the probability that the news i is positive

[9] https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip

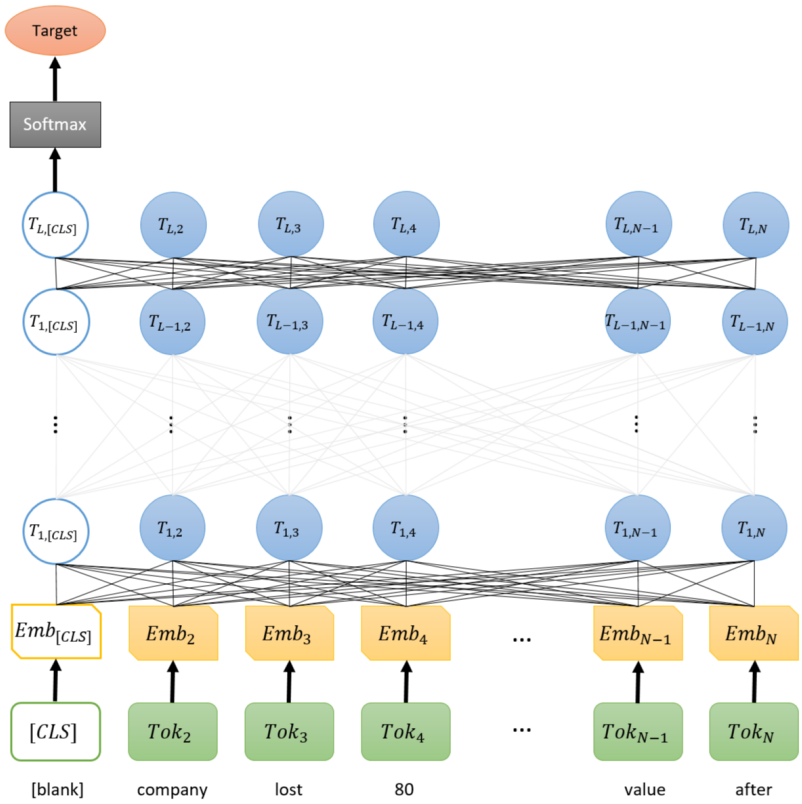


Figure 2.4: BERT model with input headline "*company lost about 80 percent of its market value after interim data*" and the maximum length of model is set to 10 ($N = 10$). The last two words *interim* and *data* are therefore trimmed. There are L layers in this BERT model. Tok_i denotes the *i-th* token of the tokenized news and N denotes the maximum length of a headline. $T_{i,j}$ represents the hidden vector for the j-th transformer at layer i. The first [CLS] label is simply a special string we choose to signify the class of this headline. This label is only used when fine-tuning the model, we leave it blank when we generate the embeddings. The original method to use BERT as classification model is to use directly [CLS] vector to represent the whole sentence, while we choose to use all hidden vectors at one layer to represent the sentence. The embedding generated with the model is shown in Equation 2.11.

given by the model. We use back-propagation [Hecht-Nielsen, 1992] to update the parameters in the model. We repeat such operation for several epochs until the loss converges.

In order to generate the contextualized embedding, we can either directly use the base BERT model or use the fine-tuned model. We first tokenize our headlines using SentencePiece tokenizer [Kudo and Richardson, 2018]. If the number of tokens is smaller than N, we simply pad it to N tokens by adding null tokens at the end. If there are more than N tokens, we remove the last tokens to make this sentence have exactly N tokens. We input these tokens into pre-trained BERT model as shown in Figure 2.4 with the first token which stands for [CLS] label left blank. Suppose that our BERT model has L layers, we can then generate L different embeddings, denoted by $Emb_{base,l}$. We have,

$$Emb_{base,l} = [T_{l,2}, T_{l,3}, ..., T_{l,N}] \tag{2.10}$$

where $Emb_{base,l}$ is a $size(T_{l,i}) \times (N-1)$ matrix representing this headline and l denotes the layer from which we generate the embedding. We have $1 \leq l \leq L$.

Similarly, we can generate another L embeddings from the fine-tuned BERT model, denoted by $Emb_{tuned,l}$. We have,

$$Emb_{tuned,l} = [T'_{l,2}, T'_{l,3}, ..., T'_{l,N}] \tag{2.11}$$

where $T'_{l,i}$ denotes the hidden vector for the i-th token at the l-th layer for the fine-tuned model.

2.5.2 RNN Prediction Model

The structure of our prediction model is simple and straightforward, it is shown in Figure 2.5.

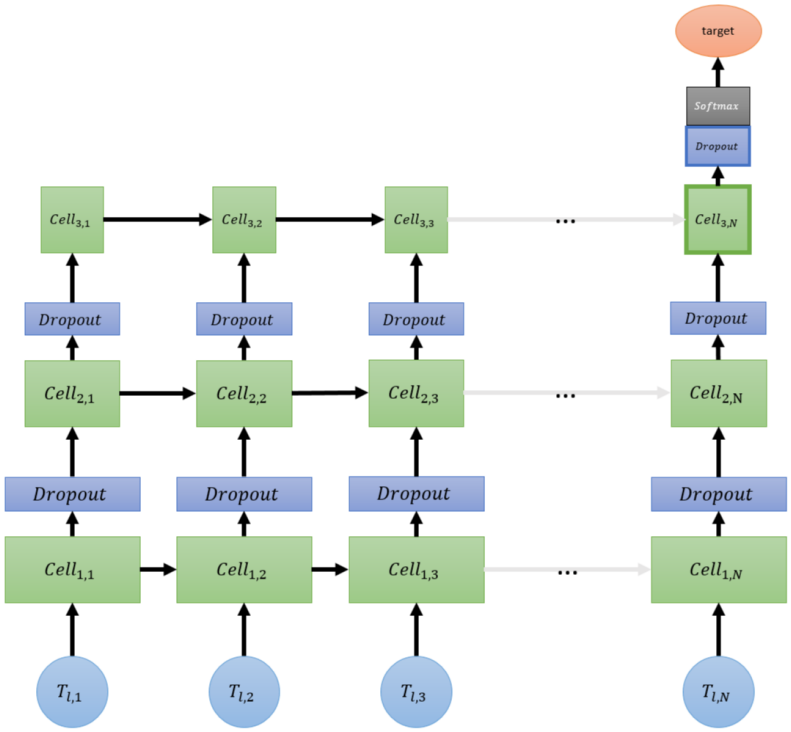


Figure 2.5: Structure of our RNN network. $Cell_{i,j}$ denotes the j-th cell on the i-th layer. The cell can be either Vanilla RNN, LSTM or GRU. In addition, the output size reduces when it approaches the last layer of the model.

There are several layers of cells which can either be Vanilla RNN [Cleeremans et al., 1989], Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] or Gated Recurrent Unit (GRU) [Cho et al., 2014]. The size of output at each layer shrinks in order to reduce the dimension of our features gradually and to make the remaining features more meaningful. Two neighbor layers are connected by a dropout to overcome the overfitting problem in the network. At the end of the last layer, a softmax is added to calculate the probability for each class based on the last vector on the last layer.

Suppose that we choose to use $Emb_{base,l}$ to represent a sentence. We first initialize the parameters in the RNN model randomly and input the embeddings of tokens ($T_{l,i}$) sequentially into the cells ($Cell_{1,i}$) in the first layer of the recurrent neural network. At the same time, we set the target to the corresponding label of the headline. We use the same cross-entropy loss

mentioned in Equation 2.10 as our loss function. We use the same back propagation procedure in Section 2.5.1 to update the parameters in the model until we have a stable loss.

2.6 Experiments

In this section, we introduce our experiment setup and results in detail. We also include the results of some baseline models to prove the effectiveness of our model. In addition to the final result, we add some ablation experiment results to show the effect of some factors in our model.

2.6.1 Training Setup

We use the pre-trained 12-layer, 768-dimension, 12-heads BERT model[10] to generate $E_{base,l}$, then we fine-tune this base model using our labelled dataset with the method mentioned in section 2.4.2.[11] We choose the best-performing fine-tuned model to generate $E_{tuned,l}$. Empirically, the layer of BERT used as embedding should not be too close to the first layer, otherwise the contextualized embedding will be too similar to the static embedding. Hence, we only test embeddings with $l = L, L-1$ *or* $L-2$. This choice will be discussed in detail in Section 2.6.6.

The maximum length of a sentence is set to 32 tokens, as there are at most 29 tokens for all headlines in our dataset. If there are fewer tokens, we pad it to 32 with null tokens.

For our RNN model, the cells are set to be LSTM. We use a four-layer single-directional RNN[12] with a dropout rate of 50%.

2.6.2 Evaluation Metrics

Because of the huge volume of news that we receive daily, it is not realistic for either human investors or systematic trading algorithms to react on all news. Otherwise, we lose a considerable amount of transaction fees on the news which do not significantly move the market. It is more logical that an investor first reads the news, then buys or sells the stock if he thinks that the news can have substantial impact on the stock price. If he thinks that the news is neutral, he will simply ignore the news. In this type of neutral-insensitive scenario, evaluating our model on all news is less meaningful. Instead, we evaluate our model only on certain "extreme" news chosen based on their sentiment classification results.

We define the score of a news, denoted by S_{news}:

$$S_{news} = (P_+(news) - 0.5) \times 2 \tag{2.12}$$

where $P_+(news)$ denotes the probability that this news belongs to the positive class given by the prediction model. S_{news} is therefore a value between -1 and 1.

[10] This is the base pre-trained model published by Google, it is available at https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip

[11] We choose batch size: 32, 64, 128 and learning rate: 2e-6, 5e-6, 1e-5

[12] The hidden size for each layer is set to 256, 128, 64, 32 respectively.

We use P_n to denote the n^{th} percentile of all scores on the **training** set. We can then choose the set on which we want to evaluate our model. We define this set E_{2n} by:

$$\begin{aligned} E_n^- &= \{news | S_{news} < P_n\} \\ E_n^+ &= \{news | S_{news} > P_{100-n}\} \\ E_{2n} &= E_n^- \cup E_n^+ \end{aligned} \tag{2.13}$$

We assume that the distributions of scores on the training set and the test set are the same. It should contain about $n\%$ highest-score news and $n\%$ lowest-score news from the test set. We evaluate our model on these subsets of news instead of all news in the test set.

Standard Metrics

Given a confusion matrix $\begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix}$ which contains the number of samples classified as true positive (tp), false positive (fp), true negative (tn) and false negative (fn). We use both the accuracy and the Matthews Correlation Coefficient (MCC) [Matthews, 1975] to evaluate our models. These two values are defined by:

Accuracy:

$$\frac{tp + tn}{tp + tn + fp + fn} \tag{2.14}$$

Matthews Correlation Coefficient (MCC):

$$\frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{2.15}$$

Trading Strategies

However, those two metrics introduced above do not perfectly reflect the reality, as the profits are quite different when the price goes up significantly or mildly, although they are both counted as true positive. Hence, it is necessary to simulate these trades on real markets. We use two simple trading strategies for simulations.

Strategy 1 (S1)

We simply follow the strategy used by Ke et al. [2019].

Before each market close, we search for all the news belonging to E_n with a maximum age of A days. We group the selected news by stock and we calculate the average score for each stock. We choose the 20 stocks with the highest scores, our target position for these stocks is \$T. We also choose the 20 stocks with the lowest scores, our target position for these stocks is -\$T.

Strategy 2 (S2)

The advantage of S1 is that it perfectly balances the long leg and the short leg.[13] As such, we have no exposure to the market, and it reduces the risk of the market movement. However, the fallback of S1 is that it only focuses on the highest scores instead of considering all the stocks. In this case, we will not be able to fully use our predictions. Hence, we design the following strategy to solve this problem:

1. We first calculate the average score for each stock using the same method as described in S1.

2. We choose all the stocks with positive scores, s_i^+ denotes the score for the stock i.

3. The target position for the stock i is $\$20T \times s_i^+ / \sum_j s_j^+$.[14]

4. We invest in the stocks with negative scores in the same way. For a negatively scored stock i, we invest $-\$20T \times s_i^- / \sum_j s_j^-$.

This strategy not only uses all the available classification results, but also has no exposure to the market as S1, since the long position and the short position are both $20T$.

To evaluate the performance of these strategies, we use the following two commonly used indicators in finance.

Annualized return: defined by

$$\frac{1}{N}\sum_{t=1}^{N} r_t \times D \tag{2.16}$$

where D is the number of trading days in one year[15], and r_t denotes the daily return of the portfolio for the day t, defined by the ratio of the profit on day t to the total position on day t.

Annualized Sharpe Ratio: defined by

$$\frac{\overline{\mathbf{r}}}{\sigma(\mathbf{r})} \times \sqrt{D} \tag{2.17}$$

$\overline{\mathbf{r}}$ denotes the mean of all the r_i and $\sigma(\mathbf{r})$ represents the standard deviation all the r_i.

2.6.3 Baselines and Proposed Models

We use the following models as baselines.

- **NBC** [Maron, 1961]: *Naive Bayes Classifier*

  One of the most traditional language classification models based on word frequency.

[13] The long leg means the total amount invested positively, the short leg means the total amount invested negatively

[14] The multiplier 20 is to guarantee the homogeneity with S1. We invest \$20T for each leg in both strategies.

[15] For the sake of simplicity, we choose 250 as the number of trading days for one year.

- **SSESTM** [Ke et al., 2019]: *Supervised Sentiment Extraction via Screening and Topic Modeling.*

  A regression model based on word frequency and stock returns.

- **Bloomberg** (Proprietary): *Bloomberg Sentiment Score*

  The sentiment score from Bloomberg's proprietary model, which comes along with the Bloomberg News dataset. An example of this sentiment score is shown in Table 2.1.

- **BERT** [Devlin et al., 2018]: *Bidirectional Encoder Representations from Transformers*

  A general and powerful language model for a wide range of NLP tasks. We directly use the [CLS] label as the final prediction, as proposed by the author.

- **FinBERT** [Yang et al., 2020b]: *Financial Sentiment Analysis with BERT*

  The same structure as the BERT model but pre-trained with financial domain-specific data.

To make a detailed analysis of the improvement brought by our proposed models, in addition to the final version of our model (FT-CE-RNN), we add two other intermediate variants of our RNN model.

- **RNN**: *Recurrent Neural Network*

  The recurrent neural network introduced in Section 2.5.2. Instead of using contextualized embeddings, we use the static Word2Vec embedding as its first layer.

- **CE-RNN**: *Contextualized Embedding - Recurrent Neural Network.*

  The network structure is the same as RNN, but we use contextualized embedding generated from base BERT model instead of Word2Vec.

- **FT-CE-RNN**: *Fine-Tuned - Contextualized Embedding - Recurrent Neural Network*

  The same RNN using contextualized embedding generated from fine-tuned BERT.

2.6.4 Results

The detailed results for standard metrics are shown in Table 2.3 and Figure 2.6. We also list the results from trading simulations in Table 2.4.

We find that in terms of accuracy, our FT-CE-RNN outperforms all the other baselines models. Especially, comparing the accuracy of RNN and FT-CE-RNN, there is an improvement of 4.1% when we test on the 1% most extreme news (E_1). This result shows the power of contextualized embedding against static embedding. We also notice that there is an improvement of 0.9% compared with BERT result. This result explains that using all hidden vectors instead of only one [CLS] vector helps improve the result. However, if we directly use the embedding from the base BERT (CE-RNN) instead of the fined-tuned BERT (FT-CE-RNN), there is a

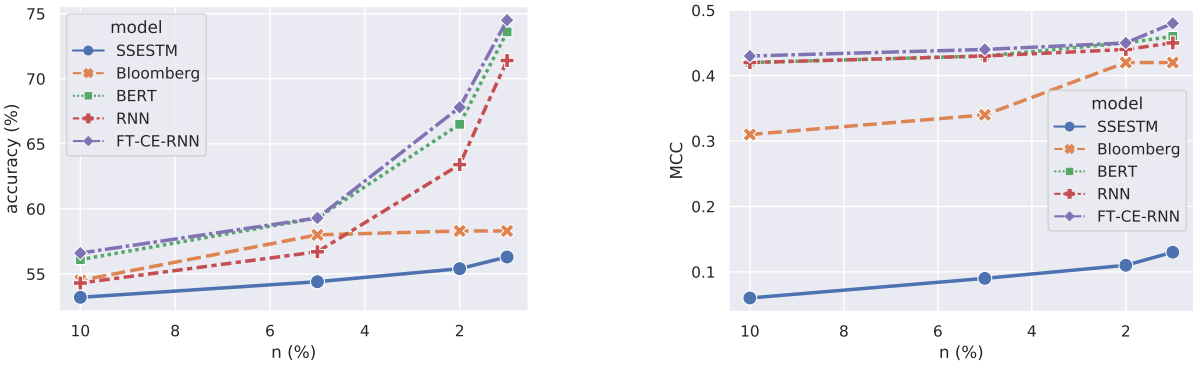
[16] 1% signifies that we test our result on E_1, which includes about 1% of all news on the test set.

[17] In Bloomberg dataset, we have about 4% of the news with a maximum level score, therefore we have the same result for E_1 and E_2.

[18] The annualized return, presented in percent.

Table 2.3: Accuracy and MCC of baseline models and our proposed RNN variants.

	1%[16]		2%		5%		10%	
	Acc	MCC	Acc	MCC	Acc	MCC	Acc	MCC
NBC	59.8	0.2	56.1	0.12	54.3	0.09	53.4	0.07
SSESTM	56.3	0.13	55.4	0.11	54.4	0.09	53.2	0.06
Bloomberg[17]	58.3	0.42	58.3	0.42	58.0	0.34	54.5	0.31
BERT	73.6	0.46	66.5	**0.45**	**59.3**	0.43	56.1	0.42
FinBERT	73.9	0.46	66.5	**0.45**	59.2	0.43	55.6	0.42
RNN	71.4	0.45	63.4	0.44	56.7	0.43	54.3	0.42
CE-RNN	70.9	0.42	63.8	0.28	57.9	0.16	54.7	0.09
FT-CE-RNN	**74.5**	**0.48**	**67.8**	**0.45**	**59.3**	**0.44**	**56.6**	**0.43**



(a) Accuracy

(b) MCC

Figure 2.6: Accuracy and MCC results of different models varied with E_n, the horizontal axis represents the value of n

clear disadvantage. This result shows the necessity of including the domain knowledge in the embeddings.

In addition, we observe that all our models have a significant margin compared with the Bloomberg and SSESTM sentiment score, which is completely independent of our data labelling and modelling process, this result can prove the efficiency of our models compared with other widely used models in the industry.

In our trading simulations, we use a look-back window of 5 days ($A = 5$ *days*). We find that the result of our FT-CE-RNN outperforms all other models in most of the cases. The most significant improvement is on Sharpe Ratio. When trading on the 1% most extreme news using strategy S2, there is an improvement of 0.69 (49%) in Sharpe ratio and an improvement of 6.62 (51%) in return if we compare BERT and FT-CE-RNN. It means that our model using contextualized embedding is not only more profitable but also more stable.

We can also find that S2 performs better than S1, since S2 uses all the signals while S1 only uses the signals on the top/bottom stocks. This proves that our classification is valid for most of the stocks, making it a robust method.

Table 2.4: Trading simulations of baselines and proposed models without transaction costs.

Strategy	Model	1%		2%		5%	
		Ret.[18]	Sharpe	Ret.	Sharpe	Ret.	Sharpe
S1	NBC	9.61	1.09	2.35	0.26	2.44	0.27
	SSESTM	2.39	0.41	1.57	0.24	2.74	0.35
	Bloomberg	8.83	1.19	8.83	1.19	8.03	1.10
	BERT	9.33	1.42	8.08	1.11	8.21	1.09
	FinBERT	8.83	1.22	8.29	1.10	7.86	1.13
	RNN	9.86	1.43	8.06	1.13	6.43	0.89
	CE-RNN	8.39	1.07	7.31	0.99	7.62	1.01
	FT-CE-RNN	10.75	1.50	12.31	1.70	11.32	1.50
S2	NBC	7.93	0.62	0.57	0.06	0.80	0.15
	SSESTM	4.75	0.47	3.02	0.35	2.76	0.38
	Bloomberg	10.76	1.61	10.76	1.61	9.82	1.56
	BERT	13.10	1.42	11.42	1.56	9.87	1.53
	FinBERT	11.35	1.24	9.85	1.21	12.85	1.97
	RNN	17.53	1.75	15.47	1.72	12.70	1.79
	CE-RNN	12.58	1.33	10.37	1.25	8.29	1.05
	FT-CE-RNN	**19.72**	**2.11**	**18.39**	**2.49**	**15.01**	**2.31**

An example of trading simulation is shown in Figure 2.7. It shows how our profit evolves with time. We observe that FT-CE-RNN is not only better on profitability and stability but also on the absolute profit in dollars.

2.6.5 Transaction costs

Our trading simulations ignore transaction costs thus far, since the primary goal of this research is to prove the effectiveness of our sentiment model with contextualized embedding. The transaction costs have no impact on the result because all the models are in the same no-cost environment.

That said, applying this model in real-life trading is another separate but interesting question. To understand the real gain of our FT-CE-RNN model for the asset management, we rerun our trading simulations with transaction costs.

In our simulations, we assume a transaction cost of 4bps[19] proportional to the daily turnover[20]. The simulation results with transaction costs are shown in Table 2.5.

Although the transaction costs cut our profits significantly, we can still have a profitable margin when using our FT-CE-RNN model.

[19] basis points, 10^{-4}

[20] defined as $|pos_i - pos_{i-1}|$ for day i

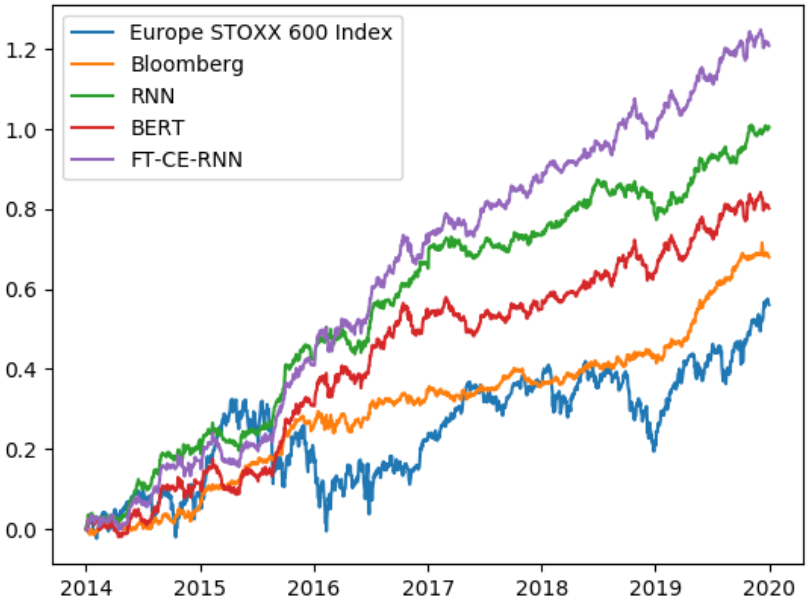


Figure 2.7: Trading simulation result in absolute profit. This trading simulation is run on 1% most extreme news (E_1) using strategy S2. We also include the market return, represented by Europe STOXX 600 Index. We can see our models largely exceed the market return and the Bloomberg sentiment score.

2.6.6 Effect of Embedding Layer

In this section, we discuss our choice of BERT hidden layer to be used as embedding.

Empirically, the final layer of BERT model should be used to generate our contextualized embedding as the final layer is more "mature" and contains more information compared with other layers which are closer to the first layer. However, our result listed in Table 2.6 shows the opposite.

We find that the best result for CE-RNN is acquired when we use layer L while the best result for FT-CE-RNN is obtained when using layer $L-1$. The reason for this phenomenon is that the last layer of the fine-tuned BERT is biased towards the classification result, since the goal for the fine-tuning process is to make the first token of the last layer close to the classification target. If we use the last layer of the fine-tuned BERT as the input for RNN, we are simply replicating the classification process of the BERT, instead of improving the result. Using one deeper layer ($L-1$) helps reduce this bias [Xiao, 2018]. However, the base BERT does not have this bias on the last layer since it has no previous knowledge on the training set. This explains why CE-RNN has a better performance when using the embedding from the last layer (L).

If we use an even lower layer to generate contextualized embedding, such as $L-2$, the performance declines as it is too close to the embedding layer and lacks contextualized characteristics.

[21] L denotes the last layer of the BERT model, $L-n$ represents the n-th last layer of the BERT model.

Table 2.5: Trading simulations with transaction costs. The result is obtained based on E_1 test set using the trading strategy S2.

Model	No cost		With cost	
	Ret.	Sharpe	Ret.	Sharpe
NBC	7.93	0.62	2.09	0.16
SSESTM	4.75	0.47	-1.80	-0.18
Bloomberg	10.76	1.61	4.49	0.67
BERT	13.10	1.42	5.98	0.65
FinBERT	11.35	1.24	3.88	0.42
RNN	17.53	1.75	11.57	1.15
CE-RNN	12.58	1.33	6.15	0.65
FT-CE-RNN	**19.72**	**2.11**	**12.60**	**1.35**

Table 2.6: Accuracy using different layers of BERT model as contextualized embedding. The result is acquired on the E_1 test set.

Embedding layer	L^{21}	$L-1$	$L-2$
CE-RNN	**70.9**	68.1	63.3
FT-CE-RNN	73.0	**74.5**	66.7

2.6.7 Effect of Classification Classes

During our initial researches, we also explored the possibility of using a 3-class classification instead of a 2-class classification. It means that we do not classify a news into a positive or a negative news, we classify if a news is either positive, negative or neutral. This is the method adopted in the Bloomberg's proprietary model.

Table 2.7: The accuracy for the 2-class classification and the 3-class classification model. The result is based on the test set E_1. For 3-class classification, we choose the $\frac{n}{2}\%$ largest scores for the positive class and the $\frac{n}{2}\%$ largest scores for the negative class as our E_n. The guarantees the same number of the news considered in both cases.

Acc	1%	2%	5%	10%
2-class	74.5	67.8	59.3	56.6
3-class	61.0	58.4	55.5	53.6

The result of using a 3-class classification model is shown in Table 2.7. We find a significant worse performance if we add another possibility to our model. This is because a 3-class model supposes a clear difference between the market-moving news and the neutral news, however, this is not always the case. It is not obvious to find a threshold, above which the news is positive and below which the news is neutral. In this scenario, we are not able to construct a clear training set for our model to learn the difference between a neutral news and a market-moving news.

Hence, in our final model, we decide to classify the news into two classes instead of three.

2.6.8 Qualitative Analysis of the Classification Result

We analyze the news in E_1 to see if there is any pattern, for example, some frequent words in them. We include the 50 most frequent words appeared in $E^+_{0.5}$ and 50 most frequent words in $E^-_{0.5}$ in Appendix 2.A, along with their frequencies. We exclude all the stopwords in English, such as *to*, *for*, *a*, etc.

We can find that for the news identified as the most positive, some common words include *buy*, *upgrade*, *raise*, etc. For the news identified as the most negative, *downgrade*, *cut* and *miss* are among common words. These are also logical keywords for the humans, making the result from the neural network intuitive.

We can also find in this collection that there are also some less natural words, such as *fly*, *say* *neutral*, etc. However, as these words appear in both categories, the effect of such words is neutralized if we empirically assume the effect of a word is its positive impact minus its negative impact.

This result is similar to the result we get from the word frequency-based method, such as NBC and SSESTM. However, we demonstrate that our FT-CE-RNN model is significantly more powerful than these two baseline models (Section 2.6.4). This phenomenon implies that our model is capable of capturing complex information in the news on top of the word frequency.

2.7 Conclusion

We build the whole pipeline for the stock movement prediction task with headlines from financial news, including labeling the news, generating contextualized embedding, training a neural network model, validating the model with various metrics and building trading strategies based on the model output.

We design a FT-CE-RNN model which uses fine-tuned contextualized embeddings from BERT instead of the traditional static embeddings. We also introduce our new evaluation metrics focusing on market-moving news, which are more suitable for asset manager's needs.

Through various experiments on the Bloomberg News dataset, we demonstrate the effectiveness of our FT-CE-RNN model. We find a better performance, in both accuracy and trading simulations, than other widely used baseline models. We also include other ablation studies to discuss the choice of some important parameters and to demonstrate the intuitiveness of the result.

In the future, we will continue our research on the stock movement prediction using natural language processing methods based on longer texts (such as earning call transcripts, financial reports, etc.) instead of the headlines. By using more information, we aim to build a model which helps achieve better stock movement prediction result.

Appendix

2.A Frequent Words in Market Moving News

Table 2.A.1: The most frequent words in the most positively scored news and the most negatively scored news

positive		**negative**	
word	*frequency*	*word*	*frequency*
buy	0.0344	downgraded	0.0252
upgraded	0.0187	cut	0.0240
deal	0.0172	fly	0.0173
said	0.0172	bank	0.0142
raised	0.0158	misses	0.0110
fly	0.0131	falls	0.0106
order	0.0107	neutral	0.0092
talks	0.0094	hold	0.0092
gets	0.0075	cuts	0.0088
neutral	0.0071	sell	0.0086
raises	0.0065	buy	0.0078
wins	0.0054	miss	0.0078
hold	0.0054	estimates	0.0073
billion	0.0053	outlook	0.0061
street	0.0052	sales	0.0058
stake	0.0048	profit	0.0054
unit	0.0047	earnings	0.0054
insider	0.0046	tradegate	0.0050
buyback	0.0041	says	0.0049
outlook	0.0038	sees	0.0043
offer	0.0037	downgrades	0.0036
agrees	0.0036	shares	0.0036
buys	0.0035	revenue	0.0035
says	0.0034	underperform	0.0029
group	0.0032	underweight	0.0029
berenberg	0.0030	credit	0.0028
near	0.0029	close	0.0028
sell	0.0029	lower	0.0027
bid	0.0026	results	0.0027
tradegate	0.0026	loss	0.0025
approval	0.0025	leave	0.0025
new	0.0025	forecast	0.0023
buyout	0.0025	guidance	0.0022
bank	0.0025	growth	0.0022
acquire	0.0025	overweight	0.0021
outperform	0.0024	negative	0.0020
rises	0.0023	downgrade	0.0020
worth	0.0023	outperform	0.0020
eu	0.0022	loses	0.0019
contract	0.0022	weight	0.0019
close	0.0021	seeking	0.0019
overweight	0.0020	drops	0.0018
credit	0.0020	equal	0.0018
set	0.0019	close	0.0017
fiat	0.0018	new	0.0017
gains	0.0018	perform	0.0017
sees	0.0017	indicated	0.0016
merger	0.0017	price	0.0016
takeover	0.0016	target	0.0016

2.B Overview of Natural Language Processing

In this section, we introduce the necessary knowledge in the Natural Language Processing (NLP) used in this chapter and the following chapters.

The goal of NLP is a computer capable of understanding the contents of documents. The NLP is seen as a challenging problem since Turing [2009] proposed using the automated interpretation and generation of natural language as criterions of intelligence.

In early ages, researchers adopt statistical models to achieve this goal. For example, Mosteller and Wallace [1963] propose using Naïve Bayes Classifier (NBC, Section 2.B.1) as a text classification model by counting the word appearance in different classes. Luhn [1957] and Jones [2004] further improve the NBC model by proposing Term Frequency - Inverse Document Frequency (TF-IDF) model which discriminates the most frequent words in the documents with Inverse Document Frequency.

Nowadays, with the development of the machine learning, researchers propose using neural networks to handle texts and achieve better performance compared with statistical models. Mikolov et al. [2010] prove the efficiency of the Recurrent Neural Network (RNN, Section 2.B.3) as a language model. It solves the problem that statistical models do not handle properly the context of a word, since the hidden state in the RNN can represent information about all the preceding words.

More recently, transfer learning (Section 2.B.4) approach has attracted more attention in the NLP community. The idea is to first train a model with very large corpus to extract the basic knowledge in general English and then fine-tune this model with domain specific data. This approach is similar to the learning procedure of humans and achieves state-of-the-art performance in all NLP tasks. One of the popular transfer learning models is the Bidirectional Encoder Representations from Transformers (BERT, Section subsubsec:bert) proposed by Devlin et al. [2018].

2.B.1 Statistical Models

In early ages, statisticians and linguisticians use statistical models to process natural languages. The method is usually based on the frequency of a word or a set of words. These methods are simple but effective during the time when the computation resources were not adequate. One of the most used methods is Naïve Bayes Classifier (NBC) and it is still frequently used as a benchmark model in modern researches.

Naïve Bayes Classifier

As its name states, NBC is used for classification tasks. Its goal can therefore be formulated as

$$\hat{c} = \arg\max_{c \in C} P(c|d) \tag{2.18}$$

where C denotes all possible classes and d denotes a document to classify.

Using Bayes' rule, the Equation 2.18 becomes

$$\hat{c} = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c) \tag{2.19}$$

since $P(d)$ is a constant for a given document d.

We then decompose a document d into words $w_1, w_2, ..., w_n$, the Equation 2.19 can be then rewritten as

$$\hat{c} = \arg\max_{c \in C} P(w_1, w_2, ..., w_n|c)P(c) \tag{2.20}$$

In NBC, we have two assumptions

- Bag of words assumption: the positions of words do not matter
- Naïve Bayes assumption: the words are independent from each other

The previous equation then becomes:

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{w \in \mathcal{V}} P(w|c) \tag{2.21}$$

where $\mathcal{V}$ is the vocabulary set.

To avoid overflow in the computation, we calculate the log probability instead, it then becomes

$$\hat{c} = \arg\max_{c \in C} \log P(c) \sum_{w \in \mathcal{V}} \log P(w|c) \tag{2.22}$$

The problem now becomes learning the probability $P(w|c)$. A simple method is to count the number of words appearing in each class, where we use the frequency as an estimator of the probability:

$$\hat{P}(w|c) = \frac{count(w,c) + 1}{\sum_{w \in \mathcal{V}}(count(w,c) + 1)} = \frac{count(w,c) + 1}{\sum_{w \in \mathcal{V}} count(w,c) + |\mathcal{V}|} \tag{2.23}$$

where $count(w, c)$ denotes the number of w shown in the documents in the class c. We note that we add one to all *count* in order to avoid zero when we calculate log.

We can therefore build a Naïve Bayes Classifier using Equation 2.23 for training and Equation 2.22 for inference.

2.B.2 Word Embedding

The first task in the modern NLP applications is to convert a word into a vector for further processing. We usually convert the word into a fixed-dimension vector called word embedding. There are several methods for this conversion, including sparse embedding, static embedding and contextualized embedding.

Sparse Embedding

The simplest method for this task is to represent a word with a vector of dimension $|\mathcal{V}|$, where $\mathcal{V}$ is the vocabulary of the corpus. The i-th word in the vocabulary is hence represented by

$$[\underbrace{0,...,0}_{1\ to\ i-1}, 1, \underbrace{0,...,0}_{i+1\ to\ |\mathcal{V}|}]$$

However, this method has a few drawbacks. Firstly, the vocabulary of English is large. The dimension of the vector can be huge if we have many different words in our corpus. Secondly, this method does not show the connection between words since the cosine similarity between any two words is 0. For example, word *good* and *great* has a similar meaning and we want them to have a larger cosine similarity compared with other pairs with no connection. Hence, we need a better word embedding model to achieve this goal.

Static Embedding - Word2Vec model

Mikolov et al. [2013b] propose a Word2Vec model to convert a word into a static dense vector. The embeddings generated with this model show better performance than sparse embeddings and it has been widely use across different NLP tasks.

The Word2Vec model is trained with a simple binary classification task: is a context word c appears near the target word w? It means that given a tuple (w, c), the classifier will return the probability that c is a context word around w, denoted as:

$$P(+|w, c) = \sigma(-\boldsymbol{c} \cdot \boldsymbol{w}) \tag{2.24}$$

where $\boldsymbol{c}$ and $\boldsymbol{w}$ are the embeddings of the word c and w respectively. σ denotes the sigmoid function[22].

The probability that c is not a context word around w is therefore:

$$P(-|w, c) = 1 - P(+|w, c) = \sigma(\boldsymbol{c} \cdot \boldsymbol{w}) \tag{2.25}$$

Suppose that we have a positive training example (w, c_{pos}), we sample k different negative training examples from the vovabulary denoted by (w, c_{neg_i}) where $1 \leq i \leq k$. The loss function to minimize for the word w in this context can be expressed using a cross-entropy [De Boer et al., 2005] function:

$$\begin{aligned} L_w &= -log\left[P(w|w, c_{pos}) \prod_{i=1}^{k} P(-|w, c_{neg})\right] \\ &= -\left[log\sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} log\sigma(-c_{neg_i} \cdot w)\right] \end{aligned} \tag{2.26}$$

We minimize this function using stochastic gradient descent [Bottou, 2010]. The hidden vector $\boldsymbol{w}$ from the trained model is the embedding for the word w.

[22] $\sigma(x) = \frac{1}{1+exp(-x)}$

We note that we do not need data labels during the training process, instead we are using the texts themselves as implicitly labeled data. We then run supervised classification task based on the same dataset. This idea is often called self-supervised learning and it is widely used in other NLP models such as BERT.

Contextualized Embedding

The embeddings we get from the Word2Vec model are static embeddings, meaning that the method learns one fixed embedding for each word. However, the meaning of a word can largely depend on the context in English. For example, the word **bank** has very different meanings in the sentence ***bank of america*** and *river* ***bank***. Hence, it is important to have contextualized embeddings, in which the vector for each word is different in different contexts.

There are different models to generate contextualized embeddings. The idea is to input all the words in the model at once, the model then calculates the new embedding for one word based on its embedding and the embeddings of other words. The BERT model can generate contextualized embeddings and we introduce its details in Section 2.B.4.

2.B.3 Neural Networks for Natural Language Processing

Language is a temporal phenomenon. We read and speak a sentence word by word and we can view this process as a continuos temporal process. It is therefore important to consider this characteristic when we build language models. A simple feed-forward neural network has shown promising results in NLP tasks but does not well model the temporality in a language.

Hence, the researchers propose two famous solutions to this challenge, including recurrent neural network and transformer network. The recurrent neural network adopts a recurrent structure to consider both the current word and the words in the past at the same time. More recently, the transformer networks are proposed to further consider the relations between a word and other words in the past or in the future over a long distance through the self-attention mechanism.

Recurrent Neural Network

The simple recurrent neural network (RNN) is first introduced by Elman [1990]. It is a neural network that contains a cycle within the connections, meaning that the value of a unit depends on earlier outputs. It is proven to be powerful in language modeling since it models well the temporality in the languages.

An unit of simple RNN is shown in Figure 2.B.1. Same as a simple feed-forward neural network, we have the input x_t at time t, a hidden layer h_t and the output y_t. The difference in RNN is that the model can see the information passed from $t-1$: h_{t-1}, which is shown with the dashed line.

The propagation rules in the RNN can be written as:

$$h_t = g(Uh_{t-1} + Wx_t) \tag{2.27}$$

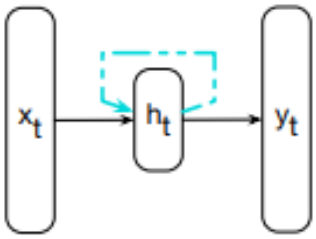


Figure 2.B.1: An unit of simple recurrent neural network. (Source: Jurafsky and Martin [2021])

$$y_t = f(Vh_t) \tag{2.28}$$

where $W \in \mathbb{R}^{d_h \times d_{in}}$, $U \in \mathbb{R}^{d_h \times d_h}$ and $V \in \mathbb{R}^{d_{out} \times d_h}$ are all trainable parameters. g and f are activation functions. In classification tasks, we usually use a softmax function as f to compute the probability distribution for each class. The cross-entropy loss function we use in the training process is therefore given as:

$$L = -\sum_{c \in \mathcal{C}} y_t[c] \log(\hat{y}_t[c]) \tag{2.29}$$

where c is the class and $y_t[c] \in 0, 1$ is an indicator function which denotes if the y_t is in the class c. $\hat{y}_t[c] \in [0, 1]$ is the probability that y_t belongs to c given by the prediction model.

Mikolov et al. [2010] first propose using RNN as a language model in NLP, the idea is simply that we treat x_t as the embedding of the t-th word in the sentence.

In practice, we can also stack this RNN structure to have a multi-layer RNN. This is shown in Figure 2.5 and used as a part of our model.

Long-Short Term Memory

In practice, it is difficult to train an effective language model with the aforementioned simple RNN since the current point of processing has limited access to distant information. Although h_t contains the information from all previous words in theory, this information is still quite local and more relevant to the most recent parts of the sentence. This is because the hidden layer performs two tasks simultaneously: recording the current information and carrying the information to the next state.

To solve this issue, researchers start using Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] in RNN instead of the simple RNN unit. The structure of a LSTM unit is illustrated in Figure 2.B.2. The main difference from the simple RNN unit is that we have two hidden vectors for each input: h_t and c_t. The goal of the newly added c_t is to represent the information in the longer context. In other words, h_t is the short memory and c_t is the long memory.

There are three steps in the computation of a LSTM unit. The first step is to forget some information that is no longer useful. It is written as:

$$f_t = \sigma(U_f h_{t-1} + W_f x_t) \tag{2.30}$$

$$k_t = c_{t-1} \odot f_t \tag{2.31}$$

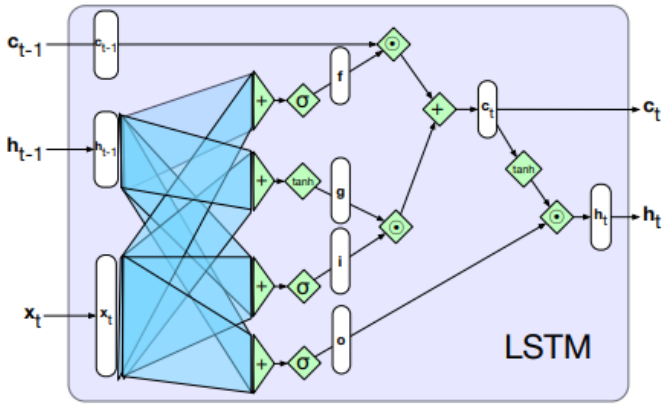


Figure 2.B.2: The structure of a LSTM. (Source: Jurafsky and Martin [2021])

where σ is the sigmoid function and $\odot$ denotes the point-wise multiplication. In this step, we first select the information to forget (f_t) with the previous state (h_{t-1}) and the current input (x_t). We then use a point-wise multiplication to force this forget process.

The second step is to select the information to remember and add this information to the context. We use the following equations to describe this step.

$$g_t = \tanh(U_g h_{t-1} + W_g x_t) \tag{2.32}$$

$$i_t = \sigma(U_i h_{t-1} + W_i x_t) \tag{2.33}$$

$$j_t = g_t \odot i_t \tag{2.34}$$

$$c_t = j_t + k_t \tag{2.35}$$

We first use the same feed forward structure as the simple RNN to get g_t. Then, similar to the first step, we use i_t to select the information to remember from g_t. We then add this information to the context to pass to the next step c_t.

The third and final step is to decide what information needed for the current hidden state (h_t). We use the similar idea by doing a point-wise multiplication:

$$o_t = \sigma(U_o h_{t-1} + W_o x_t) \tag{2.36}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.37}$$

The whole computation for a LSTM unit is illustrated in Figure 2.B.2.

Transformer Networks

Although using LSTM instead of simple RNN unit in recurrent neural networks allows it to deal with more distant information, there is still information loss in the propagation process. Hence, Vaswani et al. [2017] propose a transformer network, which uses full connections instead of recurrent connections. With this improvement, a word x_t can see all previous words , thus eliminating the need of a hidden vector which carries information from a long distance.

The key component in the transformer unit is self-attention mechanism used to combine x_t and the information which all previous words $\{x_i, i < t\}$. The self-attention is indeed a weighted sum, it can be written as:

$$y_t = \sum_{i \leq t} \alpha_{ti} x_i \tag{2.38}$$

. Hence, the main question is how we compute the weight α_{ti}.

There are three main components in this process: key (k_t), query (q_t) and value (v_t). They are calculated with:

$$q_t = W^Q x_t; k_t = W^K x_t; v_t = W^V x_t \tag{2.39}$$

where W^Q, W^K and W^V are all trainable parameters with the size of $d \times d$.

At each t, the query q_t is used to compute scores using the keys of all previous words:

$$\text{score}(x_t, x_i) = q_t \cdot k_i \tag{2.40}$$

.

We then normalize these scores with a softmax function and the normalized score is the *attention* (α_{ti}) of the word x_t on one previous word. This is written as:

$$\alpha_{ti} = \frac{\exp(\text{score}(x_t, x_i))}{\sum_{j=1}^{t} \exp(\text{score}(x_t, x_j))} \tag{2.41}$$

.

We finally sum the values of all words with their weights to get the final output y_t:

$$y_t = \sum_{i \leq t} \alpha_{ti} v_t \tag{2.42}$$

.

The computing process of self-attention is shown in Figure 2.B.3.

We can then use this self-attention mechanism to build a transformer unit with a structure shown in Figure 2.B.4,

The computation process can be expressed as:

$$z = \text{LayerNorm}(x + \text{SelfAttn}(x)) \tag{2.43}$$

$$y = \text{LayerNorm}(z + \text{FFNN}(z)) \tag{2.44}$$

where LayerNorm denotes the normalization [Ba et al., 2016].

The layer normalization is defined as:

$$\text{LayerNorm} = \gamma \hat{x} + \beta \tag{2.45}$$

where γ and β are trainable parameters and

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{2.46}$$

with μ denoting the mean of the vector x and σ denoting the standard deviation of x.

Similar to the RNN with LSTM units, we can the stack several layers of transform units to build a transformer network.

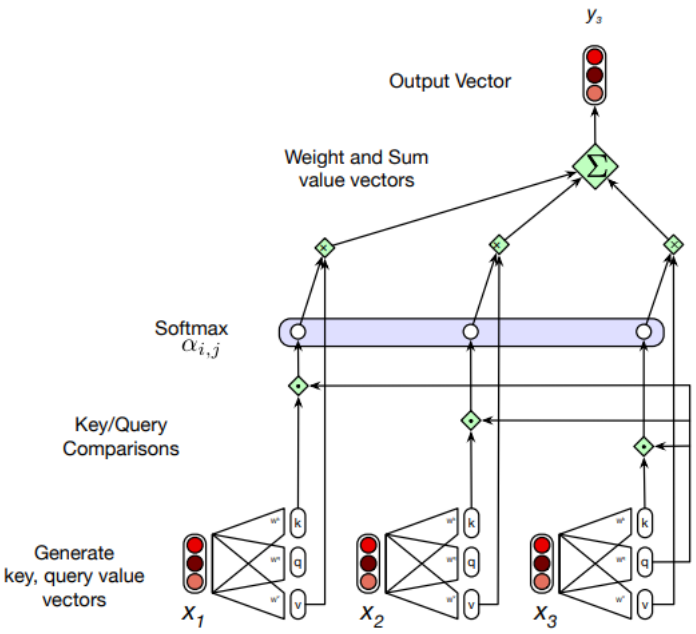


Figure 2.B.3: The structure of self-attention. (Source: Jurafsky and Martin [2021])

2.B.4 Transfer Learning Models

Transfer Learning

Transfer learning refers to storing knowledge gained while solving one problem and applying it to a different but related problem in machine learning. This approach is widely used in NLP and it is proven to be highly effective.

This transfer learning process is intuitive since it is similar to the learning process of humans. It is estimated that an adult native English speaker has a vocabulary size from 30,000 to 100,000, although only a small proportion of them are learnt from school classes. Hence, most of the knowledge in the language is acquired through daily life, such as talking and reading. This process leads us to believe that we can learn the meanings of the words through a large number of corpus without any groundings. This idea is usually referred as *pre-training*.

Another observation is that the knowledge one acquires from the pre-training process can be very useful during language processing long after its initial acquisition. For example, there are many words in the laws with specific meanings. For a student majoring in laws, the English knowledge is essential when learning these words, even though their initial meanings can be very different or he has never encountered these words. This process of continued learning based on previous knowledge is referred as *fine-tuning*.

Based on the two previous steps, researchers propose a transfer learning framework in NLP with two steps: pre-training and fine-tuning. In the pre-training process, we feed the model with a large number of documents without labels to learn the general structure of a language. We then use a smaller labeled dataset to fine-tune the model to make the model work on a specific task.

There are different transfer learning NLP models in the literatures with different structures,

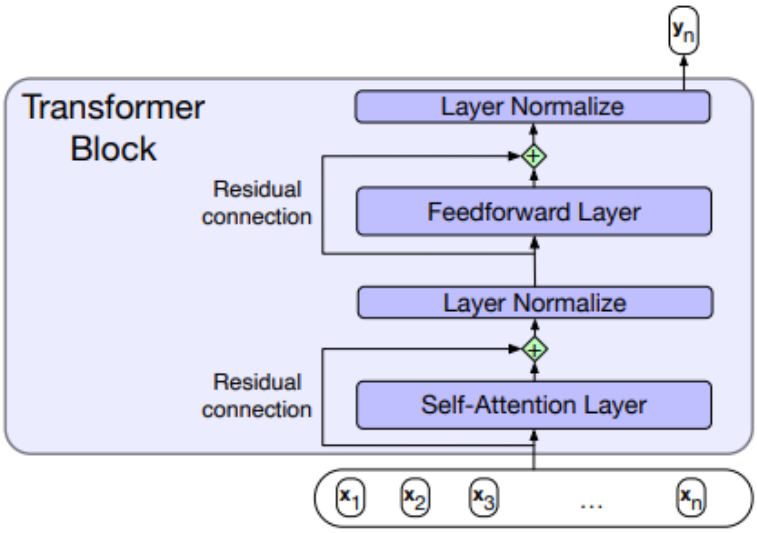


Figure 2.B.4: The structure of a transformer unit. (Source: Jurafsky and Martin [2021])

including BERT [Devlin et al., 2018], XLNet [Yang et al., 2019] etc. However, all of them follow this pre-training fine-tuning process in NLP tasks.

BERT

With the idea of transfer learning, Devlin et al. [2018] develop a Bidirectional-Encoder Representations from Transformers using the transformer units introduced in Appendix 2.B.3. The BERT model shows a significant improvement compared with RNN models on all NLP tasks included in the GLUE dataset [Wang et al., 2018].

The structure of the BERT model is already shown in Figure 2.4 and its fine-tuning process is introduced in Section 2.5.1. We introduce the pre-training of the BERT model in this section.

There are two training objectives used in the pre-training of the BERT model: masked language modeling (MLM) and next sentence prediction.

The MLM uses the same idea as training a Word2Vec model introduced in Section 2.B.2. However, instead of using a two class classification, the MLM lets the model predict among all the words in the vocabulary, meaning that it is a multi-class classification with the number of classes equaling the number of words in the vocabulary. We simply randomly select a number of words in a sentence and replace them with a special mask character. We also randomly select a proportion of words and replace them with incorrect words. The objective is to minimize the sum of cross-entropy for these replaced words. The MLM process is illustrated in Figure 2.B.5.

The goal of MLM is to predict the words from surrounding contexts and to produce effective word embeddings. However, an important number of NLP tasks involves determining the relationship between a pair of sentences. Hence, BERT proposes the next sentence prediction task in order to capture this relationship. In the training dataset, we have 50% of positive paris sampled from the corpus and the other 50% are negative pairs. During training, the two sentences are sent to the model with a specific separating character between them. We then use the same training

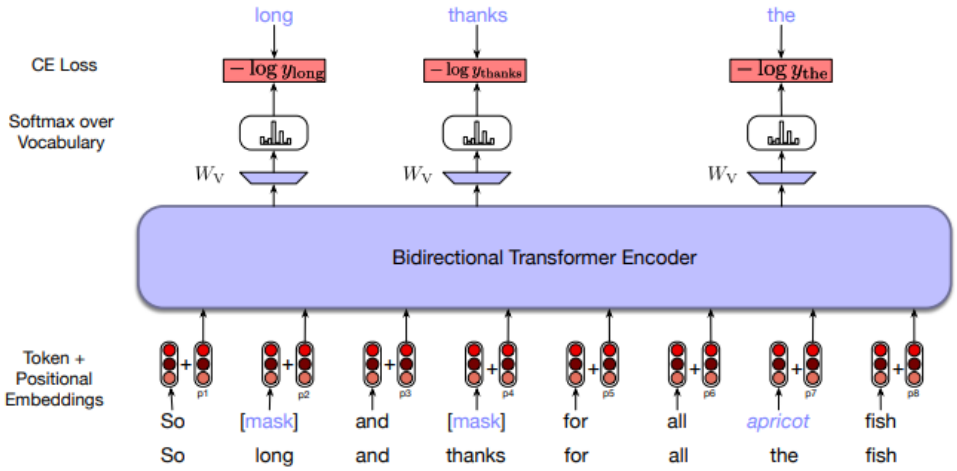


Figure 2.B.5: Masked language model. (Source: Jurafsky and Martin [2021])

method mentioned in Figure 2.4 for this binary classification task.

The pre-trained BERT model is trained on very large corpus, including 800 million words from BookCorpus [Zhu et al., 2015], 2.5 billion tokens from English Wikipedia. This large scale allows the BERT model to be effective in English language modeling.

Chapter 3

Graph-Based Learning for Stock Movement Prediction with Textual and Relational Data

Note:

- This chapter is co-authored by Qinkai Chen and Christian-Yann Robert.
- This chapter is published in Volume 4, Issue 4 of *The Journal of Financial Data Science*.
- The method described in this chapter is applied in the trading system within ExodusPoint Capital Management.

Abstract

Predicting stock prices from textual information is a challenging task due to the uncertainty of the market and the difficulty in understanding the natural language from a machine's perspective. Previous researches mostly focused on sentiment extraction based on single news. However, the stocks on the financial market can be highly correlated, one news regarding one stock can quickly impact the prices of other stocks. To take this effect into account, we propose a new stock movement prediction framework: Multi-Graph Recurrent Network for Stock Forecasting (MGRN). This architecture allows to combine the textual sentiment from financial news and multiple relational information extracted from other types of financial data. Through an accuracy test and a trading simulation on the stocks of the STOXX Europe 600 index, we demonstrate a better performance of our model compared with other benchmarks.

Contents

3.1 Introduction

Fama [1965] and Malkiel [1989] show that the movement of stock price can be explained jointly by all known information, although it is volatile and non-stationary [Adam et al., 2016]. The information can include all types of available information, such as historical prices [Kohara et al., 1997], macroeconomic indicators [Garcia and Liu, 1999], financial news [Ding et al., 2014], etc. Most of the research focuses on the time series analysis of the numerical indicators, i.e., using historical prices to predict future prices [Luo et al., 2017]. Although simple and efficient, this method does not consider the market sentiment and market moving events, based on which most rational human investors trade. With the development of the natural language processing, more recent research works start to use textual data for stock movement prediction [Ding et al., 2014,0; Hu et al., 2018]. However, these researches assume that all the stocks are independent and predict the price movement of each stock independently, although Hou [2007] shows that the movement of one stock can significantly impact other correlated stocks.

To take stock correlation into consideration, Guo et al. [2018] and Ye et al. [2021] integrate the relationship information into traditional time series analysis without using textual data. Cheng et al. [2020] and Sawhney et al. [2020] design neural networks to take both textual data and one pre-defined relationship graph into consideration. However, the stock relationships can come from multiple aspects, such as price correlation [Campbell et al., 1993], sector of economic activities [Vardharaj and Fabozzi, 2007] and supply chain [Pandit et al., 2011]. We will demonstrate that considering multiple relationships at the same time can benefit the prediction performance.

Hence, we want to design an improved model which has the following characteristics: (1) learn from both textual data and relational data, (2) incorporate a large number of relational graphs into the structure, (3) take temporal patterns of the news into account instead of learning from only one news at a time.

To address the above-mentioned challenges, we first present previous works (**Sec. 2**), we propose a new stock price movement prediction framework: the Multi-Graph Recurrent Network for Stock Forecasting (MGRN). MGRN combines textual information from a financial news provider and relationship data from different sources to predict the variation of stock prices (**Sec. 3**). MGRN jointly learns from texts and relationships through its graph-based structure, it can also learn from news' temporal patterns with its recurrent structure (**Sec. 4**). With various experiments, we show the performance of our MGRN model as well as other benchmark models (**Sec. 5**). We also perform trading simulations to show the profitability of our results in real-life scenario (**Sec. 6**).

3.2 Related Work

3.2.1 Stock Movement Prediction

There are various approaches to predict stock prices and the researches on this topic span on different domains. Econometricians use time-series analysis [Mills and Mills, 1990] to predict future prices based on historical prices and volumes data. Financial analysts rely on company fundamental data such as earnings and debt ratios [Ozlen, 2014; Wang and Xu, 2004], or

macroeconomic data such as GDP and CPI index [Hoseinzade and Haratizadeh, 2019] to predict the trend of stock prices from a economic point of view. Computer scientists tend to use machine learning techniques to interpret the stock price movement. With the development of the natural language processing, more researches focus on predicting stocks prices based on financial news or social media texts.

Schumaker and Chen [2009] use a classical feature engineering method to extract features from text data, Ke et al. [2019] use a TF-IDF [Crnic, 2011] like method to identify positive and negative words in financial texts. Nowadays, more researches adopt deep learning methods to analyze financial news. Ding et al. [2014,0] use structured representations and convolutional networks to analyze news sentiments. Hu et al. [2018] apply attention mechanism to directly handle the raw text without using widely used recurrent neural network. Xu and Cohen [2018] propose a model which considers jointly text and price information. All these methods assume that all news are independent to simplify the problem. Although useful, this is contrary to the the common sense and some findings [Hou, 2007; Klößner and Wagner, 2014] which explain the price interactions among stocks.

3.2.2 Graph Neural Network

With the popularity of graph learning, more researchers start to use graph-based structure to capture complex non-linear interactions among the nodes. Graph Convolutional Network (GCN) is one of the most used graph networks, and it has gained more popularity since it obtains outstanding results on node classification task [Kipf and Welling, 2016]. Some recent researches apply this technique on stock movement prediction tasks.

Chen et al. [2018] and Kim et al. [2019] combine historical price and corporation relationship knowledge graph through graph-based models. However, they only take historical price data as input without considering the information from news or social media texts. Sawhney et al. [2020] design a Multipronged Attention Network (MAN-SF) to consider both textual data and relationship data at the same time. However, the study only considers one pre-built graph from Wikidata[1]. In the real world, the relationships among companies come from multiple dimensions and it can change significantly over time.

To close the gap in the researches, we propose MGRN, which can ingest both textual data and a large number of relationship graphs built from different sources, as opposed to the previous researches. In addition, MGRN contains a recurrent structure to model the temporal interactions of the news, instead of assuming the independence of the news.

3.3 Problem Formulation

Following Ding et al. [2015] and Xu and Cohen [2018], we formulate the stock movement prediction as a binary classification task. Given a universe of stocks S, for a stock $s \in S$, we define its market adjusted return r_s between t and $t + \Delta t$ as:

[1] https://www.wikidata.org/

$$r_{s,t} = \frac{P_{s,t+\Delta t}}{P_{s,t}} - \frac{P_{m,t+\Delta t}}{P_{m,t}} \tag{3.1}$$

where $P_{s,t}$ denotes the price for stock s at time t, and $P_{m,t}$ denotes the market index value at time t.

We define the target of our stock movement prediction task for stock s between t and $t + \Delta t$ as:

$$Y_{s,t} = \begin{cases} 1, & r_{s,t} > 0 \\ 0, & r_{s,t} \leq 0 \end{cases}. \tag{3.2}$$

For a traditional single stock movement prediction task, the goal is to predict $Y_{s,t}$ from all the news related to the stock s in a look-back window T. It can be written as:

$$\hat{Y}_{s,t} = f(E^T_{s,t}, \theta) \tag{3.3}$$

where $E^T_{s,t}$ denotes all the news for stock s between $t - T$ and t and θ denotes the trainable parameters.

However, our goal is to consider both news and cross effects among stocks when predicting stock movement. Our prediction is hence written as:

$$\hat{Y}_{s,t} = f([E^T_{1,t}, ..., E^T_{n,t}], [G_1, ..., G_g], \theta) \tag{3.4}$$

where n is the number of stocks in our universe S, G_i is the graph constructed from data source i and g is the number of graphs we construct from different data sources.

3.4 Multi-Graph Recurrent Network for Stock Forecasting

The architecture of our MGRN model is shown in Figure 3.1. It has three sub-components: Financial News Encoder, Multi-Graph Convolutional Network and Recurrent Neural Network. We introduce the details of each component in the following subsections.

3.4.1 Financial News Encoder

Single news embedding

For each news e, we need to represent it with an embedding $v_e \in \mathbb{R}^d$. Following the work of Sawhney et al. [2020], we simply use Universal Sentence Encoder [Cer et al., 2018] to convert a sentence into a fixed-length embedding.

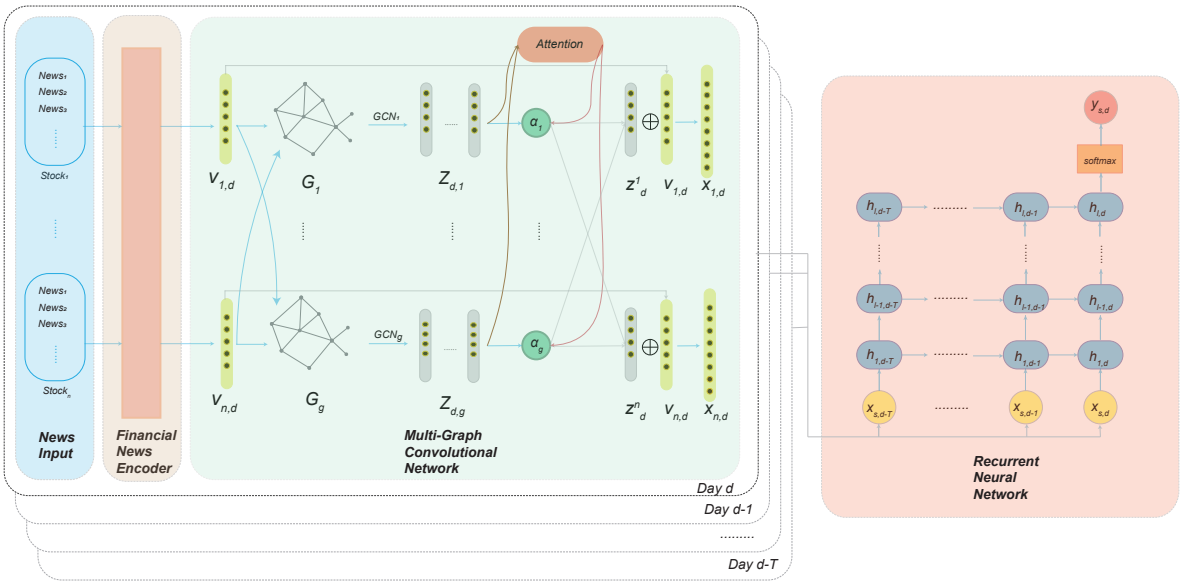


Figure 3.1: An overview of the architecture of the MGRN model. Our MGRN model includes three sub-components: (1) Financial News Encoder, which encodes textual news into a fixed length vector for each stock and each day ($v_{s,d}$). (2) Multi-Graph Convolutional Network, which takes the encoded daily news vectors and the graphs as input. Through this multi-graph structure, we get multiple node embeddings for each stock. We then combine these node embeddings into a single embedding ($\hat{x}_{s,d}$) through an attention mechanism. (3) Recurrent Neural Network, which takes the combined embeddings during a look-back window T as input and extracts temporal patterns among the news. $h_{i,j}$ denotes the j-th LSTM cell on the i-th layer. Finally, through a fully-connected layer, we predict whether the stock price increases or decreases ($\hat{y}_{s,d}$).

Aggregated news embedding

Unlike stock movement prediction based on single news, graph-based network structure requires a valid node embedding for each node when we train and predict. Hence, we need to choose a reasonable time window to make sure that for most of the stocks, there is at least one piece of news in this window. This is to avoid too many zero vectors as node embeddings. We simply choose a period of one day when we aggregate the news, following Kim et al. [2019] and Li et al. [2020b]. It means that for stock s and on day d, we select all the news concerning s between the market close time on day d and the market close time on day $d-1$ to get its aggregated embedding.

Iyyer et al. [2015] and Wieting et al. [2015] show that a simple average aggregation can have similar and even better performance than more complicated recurrent models such as LSTM. For the sake of simplicity without sacrificing the accuracy, we use an average over all news embeddings of a stock s as its aggregated news embedding on day d. We denote it by $v_{s,d}$. We have:

$$v_{s,d} = \frac{1}{|E^1_{s,d}|} \sum_{i=1}^{|E^1_{s,d}|} e^i_{s,t} \tag{3.5}$$

where $e^i_{s,t} \in E^1_{s,d}$ is the embedding of the i-th news about s occurring at time t between d and $d-1$.

3.4.2 Multi-GCN Attention Network

Graph Representation

We model a stock relationship with a graph G. We use the graph's adjacency matrix $A \in \mathbb{R}^{n \times n}$ to represent the relationships among the n stocks. The element $A_{i,j}$ denotes the intensity of relationship between stock i and stock j. We set $A_{i,i} = 1$.

There are two types of relationships: (1) boolean relationship represented by a simple graph and (2) continuous relationship represented by a weighted graph.

For a boolean relationship, we have $A_{i,j} \in \{0, 1\}$. If there is a connection between stocks i and j, $A_{i,j}$ is set to 1. Otherwise, it is set to 0. For example, GICS sector [2] relationship is a boolean relationship. If two stocks are in the same sector, we assert that they are connected. Supply chain relationship is also a boolean relationship. If one company is another company's supplier, we assert that they are connected.

However, for a continuous relationship, we have $A_{i,j} \in [0, 1]$. The more important the relation between two stocks, the larger this value. For example, the historical price relationship is a continuous relationship. The intensity of the relationship between two stocks is then calculated as the correlation coefficient of two stocks' daily return time series.

Following Duvenaud et al. [2015] and Kipf and Welling [2016], we normalize our adjacency matrix with a symmetric normalization:

$$\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{3.6}$$

where $D \in \mathbb{R}^{n \times n}$ is the generalized diagonal node degree matrix for both simple graphs and weighted graphs, defined as:

$$D_{i,j} = \begin{cases} \sum_k A_{i,k}, & i = j \\ 0, & i \neq j \end{cases} \tag{3.7}$$

Such normalization guarantees that the operations involving A do not change the scale of the result on both simple graphs and weighted graphs.

Single Graph Convolutional Network

We use the same GCN structure as proposed by Kipf and Welling [2016]. For day d, we construct our daily news matrix with $X_d = [v_{1,d}, ..., v_{n,d}]^T$. We also have one graph G and its adjacency matrix is A.

Our GCN with L layers can be written as the following function:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \tag{3.8}$$

[2] https://www.msci.com/gics

with $H^{(0)} = X_d$ and $H^{(L)} = Z_d$ as the final graph output. We have $H^{(l)} \in \mathbb{R}^{n \times f_l}$ where f_l denotes the number of output features for layer l. In Equation (3.8), σ denotes the activation function and $W^{(l)}$ represents the weight matrix for layer l.

With such an operation, we obtain a new node representation of dimension f_L for each stock from $H^{(L)}$.

Attention Aggregation Layer

Given g graphs $G_1, ..., G_g$ with their adjacency matrix $A_1, ..., A_g$, we attribute each graph an independent GCN. For day d, we have g graph outputs $Z_{d,1}, ..., Z_{d,g}$. We combine these graph outputs to get an aggregated graph output with an attention mechanism [Vaswani et al., 2017].

We define $W_a \in \mathbb{R}^{f_L \times w}$ and $q \in \mathbb{R}^{w \times 1}$, both of which are trainable parameters. We then calculate the attention coefficients $\alpha_i \in \mathbb{R}^{n \times 1}$ for graph i using the following formula:

$$\alpha_i = \frac{exp(Z_{d,i} W_a q)}{\sum_j exp(Z_{d,j} W_a q)}. \tag{3.9}$$

We then aggregate all the $Z_{d,i}$ using:

$$Z_d = \sum_i \alpha_i \otimes Z_{d,i} \tag{3.10}$$

where $\otimes$ denotes element-wise multiplication.

Finally, we concatenate the graph output Z_d with the original daily news embeddings. Our final output after the graph layer for day d becomes:

$$\hat{X}_d = X_d \oplus Z_d \tag{3.11}$$

where $\oplus$ denotes concatenation. This is to ensure that we can capture the information from both graphs and the orignal text embeddings.

3.4.3 Recurrent Neural Network

We then build a recurrent network to capture the temporal patterns in the news. The structure of the RNN is shown in Figure 3.2.

We first repeat the same process described in Section 3.4.2 from day d to day $d - T$. We have the outputs from the graph layer denoted by $\hat{X}_d, ..., \hat{X}_{d-T}$ as the input of our recurrent network.

We use a straightforward multi-layer recurrent neural network with LSTM cells [Hochreiter and Schmidhuber, 1997] shown on the right-hand side of Figure 3.1. At the final layer, we use a fully connected layer followed by a softmax to make the final prediction.

We input the concatenated outputs from the graph layer and financial news encoder layer sequentially into the first layer of the RNN model. For each stock at each day, we get the

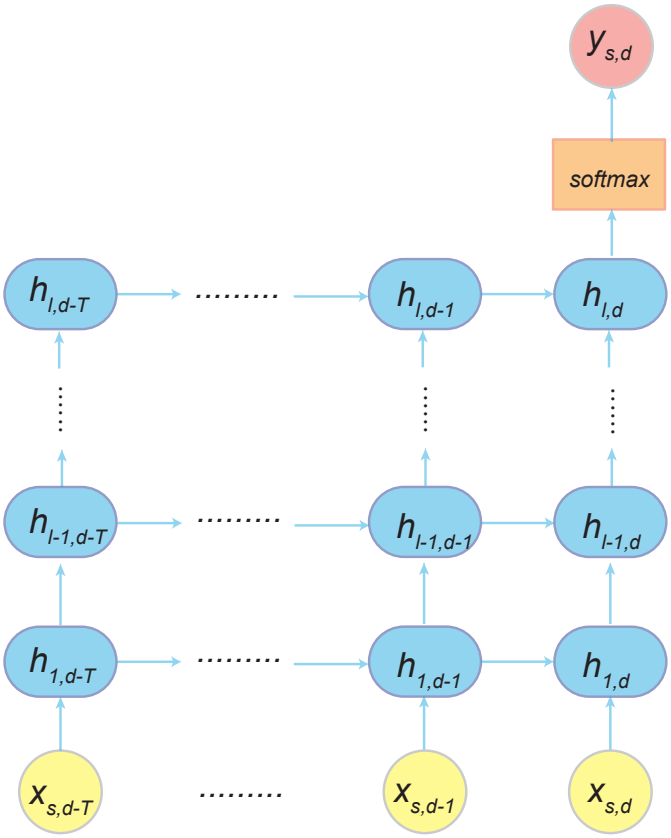


Figure 3.2: Structure of the Recurrent Neural Network in MGRN. The input $\hat{x}_{s,d}$ comes from the graph layer: the concatenated output for the stock s on day d. We use this RNN model to extract the temporal patterns in the news. $h_{i,j}$ denotes the j-th LSTM cell on the i-th layer. We use the last hidden state at the last layer ($h_{l,d}$) to make the final prediction $y_{s,d}$.

probability $P^+_{s,d}$ that the stock price will increase the next day and $P^-_{s,d} = 1 - P^+_{s,d}$ representing the price drop probability.

We train our MGRN network with an Adam optimizer [Kingma and Ba, 2014] by minimizing the binary cross entropy loss, given as:

$$l = \sum_s \sum_d Y_{s,d} ln(P^+_{s,d}) + (1 - Y_{s,d}) ln(1 - P^+_{s,d}) \tag{3.12}$$

where $Y_{s,d}$ is the true stock price movement defined in Equation (3.2).

3.5 Experiments

3.5.1 Datasets and Graph Building

Financial News Dataset

The dataset that we use is Bloomberg News[3]. In this dataset, each entry contains a *timestamp* denoting when this news happened, a *ticker* which signifies the stock related to this news and the *headline* of this news. In addition to the necessary information above, we can also find a *score* which is among -1, 0 and +1, and a *confidence* between 0 and 100 associated with the *score*. These two fields are given by Bloomberg's proprietary classification algorithm, it will serve as one of the benchmarks for our prediction model. We use the sample news dataset as the previous chapter, an example is shown in Table 2.1.

It is worth noting that we remove the stocks which do not have enough news. This is to ensure that we do not have too many zero vectors as our daily news vector (Equation (3.5)). We only select the stocks which have more than 2 news per day in average. With a such filter, we have 168 stocks in the stock universe, and we observe that there are only 15% (Table 3.1: Zero vector rate) zero vectors among all daily news vectors, meaning that given a stock and a date, there is a 85% chance there is at least one piece of news.

This is to ensure that we do not have too many zero vectors for the daily news vector of Equation (3.5) and still keep the simplicity of the method, since Rossi et al. [2021] show that GCN does not have a good training result when there are too many missing features. However, this removal does not mean that our model cannot handle missing features, we can see that there are still around 20% of all the observations which are missing (Table 3.1, zero vector rate). For these missing points, the model predicts from other companies that are connected. This can be explained by Equation (3.8), when a row in $H^{(l)}$ is zero, the same row for $H^{(l)}W^{(l)}$ is also zero, but $\hat{A}$ is a normalized adjacency matrix, the same row for $H^{(l+1)}$ is therefore the weighted mean of its neighbors.

Stock Price Dataset

We extract all the market close prices for all the stocks in the universe, we also extract the Europe STOXX 600 index value at the market close time[4] for our market adjusted return calculation. We use the stock prices for both labelling and building a correlation graph from stock returns.

For labelling, we follow the procedure described in Section 3.3. However, we observe that there are some delisted stocks which no longer have prices after a certain date, preventing us from correctly calculating their returns. Hence, we remove the stocks which are delisted during our training period. There are three such stocks, leaving us 165 stocks in total in our experiments.

We also use the stocks prices to build a weighted graph G_c. For all stocks, we first calculate its market adjusted returns with Equation (3.1). We obtain a vector $v_s = [r_{s,1}, ..., r_{s,T_c}]$ containing all the returns from the first day until the last day in our training dataset. We calculate the

[3]https://www.bloomberg.com/professional/product/event-driven-feeds/
[4]17:30 Central European Time

Pearson Correlation Coefficient [Freedman et al., 2007] between stock i and stock j, such that its adjacency matrix A_c is given by:

$$A_{c,i,j} = \frac{cov(v_i, v_j)}{std(v_i)std(v_j)} \tag{3.13}$$

where *cov* represents the covariance and *std* denotes the standard deviation.

Stock Sector Data

In finance, each company is classified into a specific sector with Global Industry Classification Standard (GICS). We use this data to construct a sector graph G_s. Its adjacency matrix A_s is defined as:

$$A_{s,i,j} = \begin{cases} 1, & sector(i) = sector(j) \\ 0, & otherwise \end{cases} \tag{3.14}$$

There are four granularities in GICS sector data: Sector, Industry Group, Industry, Sub-Industry. We can therefore construct four graphs with this dataset. In our experiments, we use the Industry granularity as it gives the best performance. The performances with different sector graphs are discussed in Section 3.5.4

Supply Chain Data

We use the supply chain data from Factset[5] to construct a supply chain graph. This dataset describes the supplier-customer relationship (SCR) among different companies. We construct a supply chain graph G_{sc} such that

$$A_{sc,i,j} = \begin{cases} 1, & i \text{ and } j \text{ have } SCR \\ 0, & otherwise \end{cases} \tag{3.15}$$

We show the heatmaps of three graphs in Figure 3.3.

Dataset Split

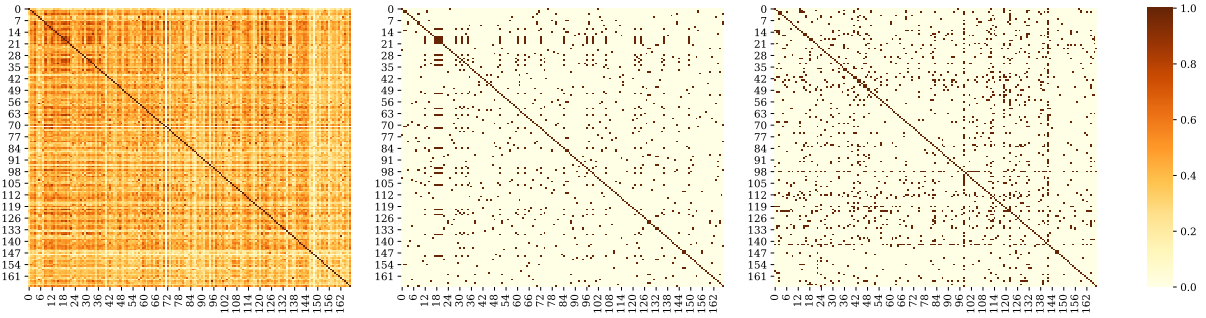
Following the standard in deep learning researches, we split our dataset into three sub-datasets: train, dev and test. The details are shown in Table 3.1.

Parameter Settings

We use a look-back window $T = 20$ days and we use a look-forward window $\Delta t = 1$ day to label our data.

[5]https://www.factset.com/marketplace/catalog/product/factset-supply-chain-relationships

[6]Number of stocks multiplied by number of trading days, this equals the total predictions we make in each dataset.



(a) Correlation Graph (b) Sector Graph (c) Supply Chain Graph

Figure 3.3: The heatmaps of our three graphs G_c, G_s and G_{sc}. We can see some common characteristics in these heatmaps, for example, the top-left corner of the correlation graph and the sector graph. However, the graphs are rather uncorrelated, we prove this with the experiment results in Section 3.5.4.

Table 3.1: Statistics of the news dataset used in our experiments. Zero vector rate means the ratio of zero vector among all embedded daily news vectors $v_{s,t}$. We only select the 165 stocks which have relatively more news to make this value as small as possible in order not to impact our GCN model.

	Train	**Dev**	**Test**
Total news	1,199,367	316,944	1,814,796
Start	01/2016	07/2018	01/2019
End	06/2018	12/2018	12/2021
Nb. Stocks	165	165	165
Trading days	652	118	757
Data points[6]	107,580	19,470	124,905
Zero vector rate	15%	17%	21%

The GCN model we use has two hidden layers, with 128 and 64 dimensions respectively. Our RNN model also has two layers, with 128 and 64 LSTM cells respectively. We train our model with an Adam optimizer for 10 epochs. We set the batch size to 32.

3.5.2 Evaluation Metrics

Standard Metrics

Following previous researches on the stock movement prediction task, we use accuracy [Ding et al., 2015; Hu et al., 2018] and Matthews Correlation Coefficient (MCC, Matthews [1975]) [Xu and Cohen, 2018; Sawhney et al., 2020] to evaluate the performance of our model.

Given a confusion matrix $\begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix}$ which contains the number of samples classified as true positive (tp), false positive (fp), true negative (tn) and false negative (fn), we define these two metrics as

$$\text{accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.16}$$

$$\text{MCC} = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}. \tag{3.17}$$

However, these simple metrics do not reflect the need of a real-life investor, since he does not need to make trades on all prediction results. The investor only trades when he is more confident about the prediction. In other words, the accuracy on the predictions with higher probability is more important than those with a mediocre probability. Hence, we also include "percentile accuracy" in our evaluation metrics.

We denote the score $S_{s,d} \in [-1, 1]$ for a stock s on day d as:

$$S_{s,d} = (P^{+}_{s,d} - 0.5) \times 2 \tag{3.18}$$

For **each day**, we choose the top $\frac{q}{2}$-percentile scores and the bottom $\frac{q}{2}$-percentile scores of that day, where q is a value between 0 and 100. We denote the the accuracy and the MCC calculated based on such selection as Acc_q and MCC_q, respectively.

Trading Simulation

We use a simple long/short trading strategy similar to Ke et al. [2019]. For each day, we attribute equally weighted (EW) long positions for the stocks whose scores are in the top $\frac{q}{2}$-percentile. For the stocks whose scores are in the bottom $\frac{q}{2}$-percentile, we give each stock the same short position. We also include a market capitalization weighted (MW) strategy to make sure that there is no bias on small-cap stocks. In both weighting methods, our long position equals to our short position[7], leaving no market exposure for our strategy.

We use annualized return and Sharpe ratio [Sharpe, 1994] to evaluate the performance of our strategies. The annualized Sharpe ratio is defined as the ratio of the expected return R to its standard deviation multiplied by square root of the number of trading days D_y in one year:

$$Sharpe = \frac{\mathbf{E}(R)}{\sigma(R)} \times \sqrt{D_y} \tag{3.19}$$

3.5.3 Baseline Models

We compare the performance of our MGRN model with other baseline models to demonstrate its performance.

We include the following baseline models:

[7]For MW strategy, we resize our long positions with a ratio to make sure that the sum of all long positions is the same as the sum of all short positions.

Table 3.2: The accuracy and MCC of baseline models and MGRN models with different q-percentiles. The results are based on test set.

q	100%		50%		20%		10%		2%	
metric	Acc.	MCC	Acc.	MCC	Acc.	MCC	Acc.	MCC	Acc.	MCC
ARIMA	0.491	0.007	0.492	0.010	0.494	0.015	0.487	-0.002	0.510	0.035
BBG	0.493	0.010	0.490	0.007	0.486	0.007	0.524	0.140	0.554	0.188
Mean-BERT	0.512	0.037	0.551	0.121	0.586	0.200	0.623	0.259	0.694	0.410
MAN-SF	0.503	0.021	0.518	0.044	0.569	0.130	0.587	0.220	0.627	0.275
RNN	0.527	0.055	0.544	0.089	0.576	0.155	0.618	0.247	0.677	0.393
MGRN-Corr	0.556	0.114	0.593	0.192	0.609	0.226	0.617	0.243	0.727	0.458
MGRN-Sector	0.532	0.064	0.552	0.106	0.589	0.184	0.640	0.296	0.725	0.453
MGRN-Supply	0.555	0.112	0.589	0.183	0.620	0.250	0.627	0.264	0.718	0.501
MGRN	**0.561**	**0.124**	**0.598**	**0.200**	**0.629**	**0.271**	**0.656**	**0.339**	**0.728**	**0.543**

- `ARIMA`: Auto-Regressive Integrated Moving Average model [Ho and Xie, 1998] based on historical prices.
- `BBG`: The prediction given by Bloomberg which comes along with Bloomberg News dataset (Table 2.1).
- `Mean-BERT`: We fine-tune the Bidirectional Encoder Representations from Transformers (BERT) model proposed by Devlin et al. [2018] as a classification model. We use the average score of all the news for stock s on day d as its $S_{s,d}$.
- `MAN-SF`[8]: A stock movement prediction framework proposed by Sawhney et al. [2020]. The model combines price data, news data and relational data to predict stock return.
- `RNN`: The model introduced in Sec. 3.4.3 without adding any graph. This is the same as a MGRN model with an identity matrix as graph adjacency matrix.

To make a detailed analysis of the improvement brought by different graphs, we train our MGRN model with different graphs:

- `MGRN-Corr`: MGRN model with the return correlation graph G_c (Eq. (3.13)).
- `MGRN-Sector`: MGRN model with the sector graph G_s (Eq. (3.14)).
- `MGRN-Supply`: MGRN model with the supply chain graph G_{sc} (Eq. (3.15)).
- `MGRN`: the full MGRN model using three graphs G_c, G_s and G_{sc} at the same time.

3.5.4 Experiment Results

Table 3.2 shows the accuracy and MCC of different models on the test set with different q-percentiles. We find that our MGRN model shows the best performance, outperforming other baseline models in both accuracy and MCC.

[8]MAN-SF only allows to have one relationship, we choose the Pearson correlation for this model.

Table 3.3: Trading simulations of all models with different q-percentiles.

	q	100		50		20		10		2	
	metric	Ret.	Sp.	Ret.	Sp.	Ret.	Sp.	Ret.	Sp.	Ret.	Sp.
EW	ARIMA	-0.5	-0.25	-0.81	-0.09	0.36	0.09	0.61	0.24	2.12	0.26
	BBG	1.51	0.28	3.09	0.56	6.05	0.62	8.67	0.68	10.61	0.83
	Mean-BERT	2.11	0.55	4.42	0.83	7.59	1.06	11.28	1.21	14.33	1.05
	MAN-SF	1.01	0.09	3.97	0.85	3.97	0.7	5.25	0.51	6.08	0.63
	RNN	**3.00**	0.39	3.79	0.86	4.56	0.72	3.71	0.42	7.81	0.85
	MGRN-Corr	2.00	0.64	4.39	0.91	5.70	0.83	9.59	0.56	14.33	0.99
	MGRN-Sector	1.64	0.45	3.47	0.47	5.00	0.59	11.82	1.16	16.62	1.11
	MGRN-Supply	2.94	0.62	3.86	0.60	6.49	0.86	10.50	1.04	14.17	0.96
	MGRN	2.39	**0.76**	**5.24**	**1.11**	**8.84**	1.12	**16.99**	1.38	16.15	1.19
MW	ARIMA	-2.82	-0.55	-1.17	-0.14	1.53	0.13	0.00	0.00	4.97	0.38
	BBG	1.36	0.29	3.61	0.12	6.12	0.83	5.02	0.37	13.63	0.92
	Mean-BERT	2.08	0.56	3.19	0.62	6.82	0.84	12.15	1.30	11.05	0.99
	MAN-SF	0.61	0.11	2.15	0.54	4.33	0.7	3.01	0.42	7.99	0.68
	RNN	2.79	0.66	3.66	0.86	4.59	0.76	3.85	0.46	7.40	0.84
	MGRN-Corr	1.81	0.61	4.21	0.92	5.34	0.81	9.05	0.59	9.43	0.90
	MGRN-Sector	2.42	0.40	4.17	0.42	4.76	0.61	12.23	0.91	10.32	1.09
	MGRN-Supply	2.75	0.60	4.77	0.61	5.90	0.85	11.07	1.11	14.69	1.04
	MGRN	2.15	0.72	4.90	1.10	8.20	**1.14**	15.74	**1.43**	**17.02**	**1.26**

We compare the single graph models (MGRN-Corr, MGRN-Sector and MGRN-Supply) and the vanilla model without graph (RNN). We find that all the graphs can help improve the performance, especially for the most extreme scores (a smaller q value). However, it is difficult to say which graph has the best performance, since each graph has different optimal performances on different percentiles. For example, the sector graph has the most added value on the most extreme scores (highest with $q = 10$), while the return correlation graph is more powerful on less extreme scores (highest with $q = 50$ and $q = 100$). This also signifies that the information in each graph is rather complementary, making it more reasonable to combine different graphs.

We validate our hypothesis that combining different graphs can help improve model performance by comparing the full model (MGRN) with the single graph models. We find that when using all three graphs together, we have a significant improvement in accuracy (4.8% with $q = 10$ and 5,3% with $q = 20$). It proves that our model can absorb necessary information from multiple complementary graphs at the same time, validating the effectiveness of combining relationship from different sources.

We also notice that adding a graph can lead to a worse result compared with the no-graph RNN in some scenario, for example, MGRN-Corr is worse than RNN when $q = 10$. However, when combining with other graphs, the result is better than using any graph individually. This is because the errors usually come from several particular stocks, especially when we only have one source of information. If the source is incorrect, it can lead to significant error. The benefit of using multiple graphs is to reduce the impact of these cases by making decisions based on more than one source of information.

Table 3.3 shows the trading simulation result using the strategy described in Sec. 3.5.2. We can also confirm that our MGRN model outperforms other models and that combining the graphs together is beneficial. We also find that the equally weighted strategy (EW) has a similar performance as market-cap weighted strategy (MW), showing that there is little bias from small-cap stocks.

Sector Graphs

As we mentioned in Section 3.5.1, there are four granularities in our GICS sector data. We compare the performances for the four granularities, and we find that the Industry level (the third granularity) shows the best performance, especially on more extreme scores. Hence, we choose to use Industry level to build G_s. The detailed result is shown in Table 3.4.

Table 3.4: The accuracy of MGRN-Sector model but with the sector graphs built from different GICS sector granularities.

level	name	100%	20%	10%
1	Sector	0.540	0.582	0.626
2	Industry Group	0.519	0.563	0.618
3	Industry	0.532	**0.589**	**0.640**
4	Sub-Industry	0.528	0.564	0.617

3.5.5 Qualitative Analysis: An Example

We give a detailed study on one specific case to show how our MGRN model helps improve stock movement prediction.

We focus on the stock **ORSTED DC**[9] on the Feb. 8, 2021. We notice a news in the evening of that day: *Big Oil Takes Over Next Generation of U.K. Offshore Wind*. This is a positive signal since Orsted was expanding its business. Based on this piece of news among others, our vanilla MGRN (RNN) without any relational input gives a slightly positive score for this stock at 0.022. However, we observe a return of -4.7% on the next trading day which is contrary to our prediction result.

The same prediction from MGRN-Sector model is -0.065, which is a correctly predicted negative value among the bottom 20-percentile. The only reason this new prediction is very different from that of vanilla MGRN is the impact from other related stocks. We find that **IBE SM**[10] has the most negative score from vanilla MGRN in the same sector. When we look at the news, we can find plenty of negative news about this company on the same day, such as *Edinburgh power cut: These are all the Capital postcodes to report electricity outages*. These negative news caused the price drop of **IBE SM** by 1.5%, which potentially caused the negative return (-1.3%) in the same sector since we do not observe many negative news for other companies.

We can also see the same phenomenon with MGRN-Corr since the correlation between two stocks is relatively high (0.39), but the prediction from MGRN-Supply is still false because there is no supplier-customer relationship between these two stocks. We show the detail of this analysis in Table 3.5.

Table 3.5: Detailed results of the case study for **ORSTED DC** on the Feb. 8, 2021. MGRN-Corr and MGRN-Sector both give correct results because the negative signal from **IBE SM** can reach **ORSTED DC** through the graphs, but MGRN-Supply still gives the wrong prediction since these two stocks do not have connection on this graph.

Model	$A_{i,j}$	Ticker	Score	Result
RNN	0	ORSTED DC	0.023	False
		IBE SM	-0.239	True
MGRN-Corr	0.39	ORSTED DC	-0.005	**True**
		IBE SM	-0.220	True
MGRN-Sector	1	ORSTED DC	-0.065	**True**
		IBE SM	-0.157	True
MGRN-Supply	0	ORSTED DC	0.008	False
		IBE SM	-0.262	True

This example shows how our MGRN model helps improve prediction results compared with a traditional recurrent model without relational modeling: the related stocks can transmit their information through the meaningful graph. The model can then make decision based on both its own information and the transmitted information.

[9]Orsted A/S is a Danish multinational power company based in Fredericia, Denmark.

[10]Iberdrola is a Spanish multinational electric utility company based in Bilbao, Spain. Scottish Power is a subsidiary of Spanish utility firm Iberdrola.

3.6 Conclusion

We predict the stock movement by jointly considering financial news, multiple graph-based features and temporal patterns of the news. We introduce Multi-Graph Recurrent Network (MGRN) for this task. Through extensive experiments and trading simulations, we demonstrate the effectiveness of the model structure. The result also proves that adding relationship information, especially different relationship information from multiple sources, can help better predict stock movement. We plan to incorporate more types of data (such as time series) in our model and apply feature filling techniques [Taguchi et al., 2021] to further improve the prediction accuracy.

Appendix

3.A Overview of Graph Neural Networks

In this section, we introduce the necessary knowledge in the Graph Neural Networks (GNN) used in this chapter and the following chapters.

3.A.1 Graph

Graph is a data structure used to describe the relationship among elements. It is widely used in many domains such as social science [Myers et al., 2014], physics [Sanchez-Gonzalez et al., 2020] and biology [Fout et al., 2017]. Nowadays, researchers use neural networks to extract information from graphs and propose various Graph Neural Networks (GNN). The GNN shows ground-breaking results compared with traditional approaches in recent papers.

A graph G consists of a finite set of *vertices* (or nodes, V) and *edges* (E). A note usually describes an element and an edge describes the relationship between two notes. The edges can be either directed or undirected. We show a directed graph in Figure 3.A.1.

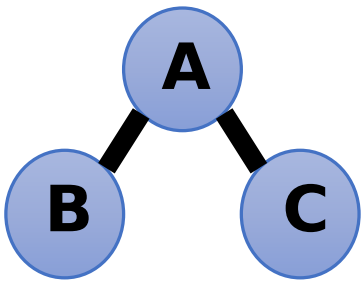


Figure 3.A.1: An example of undirected graph.

We introduce two important matrices linked to a graph: the adjacency matrix (A) and the degree matrix (D).

The adjacency matrix A is a $|V| \times |V|$ matrix such that its element A_{ij} is one if there is an edge from node i to node j. The adjacency matrix for the graph shown in Figure 3.A.1 is

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

We note that the adjacency matrix of an undirected graph is symmetric.

The degree matrix is a $|V| \times |V|$ diagonal matrix which contains information about the number of edges attached to each node. It is defined as:

$$D_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{3.20}$$

where $\deg(v_i)$ denotes the number of edges that terminate at that vertex. For example, the degree matrix of the graph shown in 3.A.1 is

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The node matrix can be used to normalize the adjacency matrix.

3.A.2 Graph Convolutional Network

The goal of Graph Convolutional Network (GNN) is to learn a function which takes both node features and the graph as input.

For each node i, we use a fixed number of features to represent the characteristics of this node. These features can be stored in a fixed length vector $x_i \in (R)^{d \times 1}$ where d denotes the number of features. This vector is also called the node embedding for the node i. Suppose that there are n nodes in the graph, we can therefore get a feature matrix $X \in \mathbb{R}^{n \times d}$. We use the adjacency matrix (A) to represent this graph. The goal can therefore be written as

$$Z = f_0(X, A) \tag{3.21}$$

where Z is the graph output.

The propagation process in a GCN can be composed of multiple layers, each layer can be written as a function

$$H^{(l+1)} = f(H^{(l)}, A) \tag{3.22}$$

where $H^{(l)} \in \mathbb{R}^{n \times f_l}$ with f_l being the number of features in the l-th layer. We also note that $H^{(0)} = X$ and $H^{(L)} = Z$ where L is the total number of layers in the GCN.

In GCN, the propagation rule f is given as

$$f(H^{(l)}, A) = \sigma(\hat{A}H^{(l)}W^{(l)}) \tag{3.23}$$

where $W^{(l)} \in \mathbb{R}^{f \times w_l}$ is a learnable weight matrix for the l-th layer and σ is an activation function. $\hat{A}$ is the normalized adjacency matrix calculated with

$$\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}. \tag{3.24}$$

where D is the degree matrix of the graph. This normalization guarantees that each row of the $\hat{A}$ sums to one, avoiding feature scale changes at each layer.

3.A.3 Message Passing Models

One of the fallbacks of the aforementioned GCN model is that it cannot handle a very large graph, since we need to put the A, a $|V| \times |V|$ matrix into the memory during the training process. When there is a large number of nodes, it is no longer possible to perform this task.

Hence, we can view this graph network as a spatial network instead of a spectral network [Zhou et al., 2020]. It means that we define the convolution operation based on the graph topology instead of the whole graph. This spatial view of the network is also known as a message passing.

Message passing works in three steps:

1. For each node in the graph, gather all neighbor node embeddings (messages).
2. Aggregate all messages via an aggregate function.
3. The pooled messages are passed through a neural network.

Suppose that the hidden vector for the node i in the l-th layer is $h_i^{(l)}$ and the neighbors of this node are in $\mathcal{N}(i)$. This message passing model can be written as

$$h_{\mathcal{N}(i)}^{(l)} = \mathrm{AGG}_l(\{h_j^{(l-1)}, \forall j \in \mathcal{N}(i)\}) \tag{3.25}$$

$$h_i^{(l)} = \sigma(W^{(l)} \cdot \mathrm{CONCAT}(h_i^{(l-1)}, h_{\mathcal{N}(i)}^{(l)})) \tag{3.26}$$

where AGG_l denotes the aggregation function for the l-th layer and $W^{(l)}$ are trainable parameters. Hamilton et al. [2017] propose three examples for the aggregation function: mean aggregator, LSTM aggregator and polling aggregator.

Hamilton et al. [2017] also prove that when using the mean aggregator, the spatial message passing model is equivalent to the spectral GCN model introduced in the previous section. We can therefore apply graph neural networks on large scale graphs with this generalized message passing models.

Part II

Realized Volatility Prediction with Limit Order Book

Chapter 4

Multivariate Realized Volatility Forecasting with Graph Neural Network

Note:

- This chapter is co-authored by Qinkai Chen and Christian-Yann Robert.
- This chapter is published in *Proceedings of the Third ACM International Conference on AI in Finance*.
- This chapter is presented in *the Third ACM International Conference on AI in Finance, New York NY, Nov 2022*
- This chapter is presented in *Statistics and Machine Learning in Finance seminar, Oxford UK, Nov 2022*

Abstract

The existing publications demonstrate that the limit order book data is useful in predicting short-term volatility in stock markets. Since stocks are not independent, changes on one stock can also impact other related stocks. In this paper, we are interested in forecasting short-term realized volatility in a multivariate approach based on limit order book data and relational data. To achieve this goal, we introduce Graph Transformer Network for Volatility Forecasting. The model allows to combine limit order book features and an unlimited number of temporal and cross-sectional relations from different sources. Through experiments based on about 500 stocks from S&P 500 index, we find a better performance for our model than for other benchmarks.

Contents

4.1 Introduction

Volatility is an important quantity in finance, it evaluates the price fluctuation and represents the risk level of an asset. It is one of the most important indicators used in risk management and equity derivatives pricing. Although the volatility is not observable, Andersen and Bollerslev [1998] show that realized volatility is a good estimator of volatility. Forecasting realized volatility has therefore attracted the attention of various researchers.

Brailsford and Faff [1996] propose using GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) models to forecast realized volatilities based on daily prices. Gatheral and Oomen [2010] introduce several simple volatility estimators based on Limit Order Book (LOB) data, showing that the use of LOB data can lead to better predicting results. More recently, Rahimikia and Poon [2020b] and Zhang and Rosenbaum [2020] use machine learning techniques such as Recurrent Neural Network (RNN) to improve such predictions.

The aforementioned literatures adopt a univariate approach in this task, which means that the model only considers one outcome for one stock at the same time, instead of jointly considering the situations of all stocks, although the asset returns on the financial markets can be highly correlated [Campbell et al., 1993]. Andersen et al. [2005] propose linear multivariate volatility forecasting methods based on daily price data to take into account this correlation, while Bucci [2020] further uses neural networks to forecast realized volatility covariance matrix non-linearly. Bollerslev et al. [2019] first propose a parametric multivariate model based on LOB data using covolatility and covariance matrices. More recently, a Kaggle multivariate realized volatility prediction competition[1] was sponsored by Optiver to challenge data scientists to propose new multivariate forecasting methods.

Compared with the univariate approach, multivariate models can capture the relations among observations. Most recently, Graph Neural Networks (GNN) [Bruna et al., 2013] are proposed to integrate such relationship into the commonly used non-linear neural networks. This approach achieves significant success in multiple applications, such as traffic flow prediction [Li et al., 2017], recommender systems [Berg et al., 2017] and stock movement prediction [Sawhney et al., 2020; Chen and Robert, 2021]. To the best of our knowledge, no graph-based structure for volatility forecasting has been proposed in the literatures.

Hence, to further improve the volatility forecasting performance, inspired by previous researches (**Sec. 2**), the Kaggle competition and our real-life use cases, we build a multivariate volatility forecasting model (**Sec. 3**) based on Graph Neural Network: Graph Transformer Network for Volatility Forecasting (GTN-VF). This model predicts the short-term volatilities from LOB data and both cross-sectional and temporal relationships from different sources (**Sec. 4**). With various experiments on about 500 stocks from the S&P 500 index, we demonstrate that GTN-VF outperforms other baseline models with a significant margin on different forecasting horizons (**Sec. 5**).

[1] https://www.kaggle.com/c/optiver-realized-volatility-prediction

4.2 Related Work

4.2.1 Volatility Forecasting

As introduced in Section 4.1, there are two types of volatility forecasting models: univariate models and multivariate models.

In univariate approaches, researchers can design intuitive estimators [Zhou, 1996; Zhang, 2006], without considering relations among observations. More commonly, researchers adopt time-series methods to model the time dependence while ignoring the cross-sectional relations and calibrating one set of parameters for each asset. For example, Brailsford and Faff [1996] propose GARCH models, Sirignano and Cont [2019] use RNN models with Long Short-Term Memory (LSTM) and Ramos-Pérez et al. [2021] adopt a Transformer structure [Vaswani et al., 2017].

Multivariate models add cross-sectional relationships and achieve better results. For example, Kwan et al. [2005] introduce multivariate threshold GARCH model and Bollerslev et al. [2019] propose a multivariate statistical estimator based on co-volatility matrices. It is worth noting that all models above only use asset covariance as the source to build the relationship while ignoring the intrinsic relations between the companies that issue the stocks.

4.2.2 Graph Neural Network

A graph is composed of nodes and edges, where a node represents an instance in the network and an edge denotes the relationship between two instances. It is an intuitive structure to describe the relational information. Recently, many researches focus on generalizing neural networks on graph structure to capture non-linear interactions among the nodes.

Bruna et al. [2013] first generalize the Convolutional Neural Network (CNN) on graph-based data, while Kipf and Welling [2016] propose Graph Convolutional Network (GCN) and Defferrard et al. [2016] introduce ChebNet, both of which have reduced network complexity and better predictive accuracy.

However, the aforementioned models are required to load all the graph data into the memory at the same time, making training larger relation networks impossible. Gilmer et al. [2017] state that Graph Neural Networks are essentially message passing algorithms. It means that the model makes decision not only based on one node's observation, but also the information passed from all other related nodes defined in the format of a graph. Based on this generalization, Hamilton et al. [2017] propose GraphSAGE which allows batch training on graph data.

Shi et al. [2020] further show that using a Transformer-like operator to aggregate node features and the neighbor nodes' features gives a better performance than a simple average such as GraphSAGE or an attention mechanism such as Graph Attention Network [Veličković et al., 2017].

To close the gap in the researches, we propose GTN-VF, which adopts the state-of-the-art Graph Neural Networks to model the relationships. In addition, GTN-VF allows to integrate

an unlimited number of relational information, including both widely used covariance and other external relations such as sector and supply chain, which were rarely used in previous researches.

4.3 Problem Formulation

We formulate this multivariate volatility forecasting problem as a regression task. The goal is to predict the realized volatility vector over the next ΔT seconds at a given time t with all previously available data.

We first define the return of stock s at time t as

$$r_{s,t} = log(\frac{P_{s,t}}{P_{s,t-1}}) \tag{4.1}$$

where $P_{s,t}$ is the last trade price of s at t.

We then use $RV_{s,t,\Delta T}$ to denote the realized volatility for stock s between t and $t + \Delta T$, it is defined as:

$$RV_{s,t,\Delta T} = \sqrt{\sum_{i=t}^{t+\Delta T} r_{s,i}^2} \tag{4.2}$$

Previous researches [Malec, 2016; Rahimikia and Poon, 2020a,0] usually calibrate one model for each stock. This prediction model f_s for stock s can be written as:

$$\widehat{RV_{s,t,\Delta T}} = f_s([D_{s,t_1}, ..., D_{s,t_m}], \theta) \tag{4.3}$$

where $D_{s,t}$ denotes the limit order book data related to stock s between t and $t - \Delta T'$, and $t_1 < ... < t_m < t$. $\Delta T'$ is a parameter denoting the backward window used to build features for t, while θ represents the model parameters.

However, as stated in Section 4.1, the realized volatilities of the stocks are related through their LOBs. We want to consider this effect in our model and we write our prediction model g_0 as:

$$\widehat{RV_{s,t,\Delta T}} = g_0(\begin{bmatrix} D_{s_1,t_1} & ... & D_{s_1,t_m} \\ ... & ... & ... \\ D_{s_n,t_1} & ... & D_{s_n,t_m} \end{bmatrix}, \mathcal{G}_s, \theta) \tag{4.4}$$

where $\mathcal{G}_s$ is the relationship of stock s with all other stocks $s_1, ..., s_n$. It means that our model jointly considers all the features from all other stocks when predicting realized volatility for stock s and its relationship with other stocks, instead of only taking its own features into account.

In additional to the relationship among stocks, we can also consider the relationship among the timestamps we predict. For example, at time t, we can check whether the behaviors of the stocks are similar to their behaviors at previous timestamps. We use $\mathcal{G}_t$ to denote this temporal relationship. Our prediction model g is finally written as:

$$\widehat{RV_{s,t,\Delta T}} = g(\begin{bmatrix} D_{s_1,t_1} & ... & D_{s_1,t_m} \\ ... & ... & ... \\ D_{s_n,t_1} & ... & D_{s_n,t_m} \end{bmatrix}, \mathcal{G}_s, \mathcal{G}_t, \theta) \tag{4.5}$$

4.4 Graph Transformer Network for Realized Volatility Forecasting

Our model consists of two main components: a LOB data encoder and a Graph Transformer Network. The LOB data encoder transforms numerical LOB data into multiple features. It also transforms categorical information, such as the stock ticker, into a fixed-dimension embedding. It finally concatenates the numerical features and the embedding for categorical features as the node feature.

The Graph Transformer Network then takes all the node features and the pre-defined relationship information as input. After training, it will give each node a new meaningful embedding which contains information from both LOB data and relational data. With a fully connected layer, we can get the final prediction of realized volatility for this node.

An illustration of the whole structure is shown in Figure 4.1.

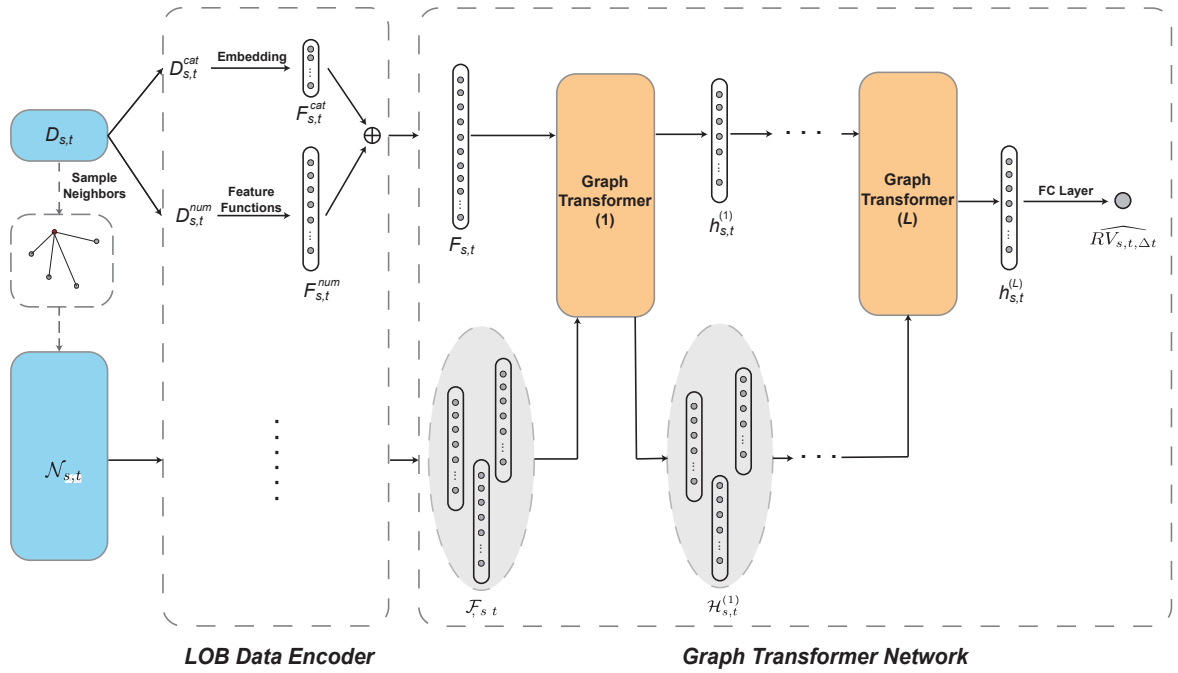


Figure 4.1: Structure of our Graph Transformer Network for Volatility Forecasting. This illustration shows the prediction process for one node $node_{s,t}$. The LOB data encoder first transforms its LOB data into a fixed-dimension node feature $F_{s,t}$. GTN then takes this node feature and all the features from all other nodes connected with this node ($\mathcal{F}_{s,t}$). After L Graph Transformer operations, we get the node embedding $h^{(L)}_{s,t}$ as output. We then use a fully connected layer to transform this node embedding into our final prediction $\widehat{RV_{s,t,\Delta T}}$.

4.4.1 LOB data encoder

We first divide our LOB data $D_{s,t}$ into two parts: numerical data $D^{num}_{s,t}$ and categorical data $D^{cat}_{s,t}$. For numerical data, we can define different functions to aggregate them into numerical values. Suppose that we have k_{num} such functions $f_1, ..., f_{k_{num}}$, we get k_{num} features and we

put them into a single vector $F_{s,t}^{num}$ where

$$F_{s,t}^{num} = [f_1(D_{s,t}^{num}), ..., f_{k_{num}}(D_{s,t}^{num})]^\top \tag{4.6}$$

$F_{s,t}^{k_{num}}$ is therefore a vector of size $k_{num} \times 1$.

Following Kercheval and Zhang [2015]; Bissoondoyal-Bheenick et al. [2019]; Mäkinen et al. [2019], we define a similar set of numerical features. We also add some other features which are suitable for our dataset. We show the detailed list of features we use in our experiments in Appendix 4.A.

For other categorical features, such as stock ticker, we simply adopt an embedding layer to transform them into a fixed-dimension vector $F_{s,t}^{cat}$. This vector is of size $k_{cat} \times 1$ where k_{cat} is the embedding dimension we can choose.

We then concatenate these two vectors into one vector $F_{s,t} \in \mathbb{R}^{k \times 1}$, where $k = k_{num} + k_{cat}$. This operation is written as

$$F_{s,t} = F_{s,t}^{num} \oplus F_{s,t}^{cat} \tag{4.7}$$

where $\oplus$ denotes the concatenation operation.

4.4.2 Graph Transformer Network

As stated in Section 4.2.2, Graph Transformer operator shows a better performance compared with other structures, we use it to build our network in this study.

We first build a graph with $m \times n$ nodes where m is the number of timestamps and n is the number of stocks. Each node $node_{s,t}$ represents the situation of stock s at time t. Its initial node feature is $F_{s,t}$ (Equation 4.7) encoded by LOB data encoder. From the relationship $\mathcal{G}_s$ and $\mathcal{G}_t$, we can find all other nodes connected with $node_{s,t}$. We use $\mathcal{N}_{s,t}$ to denote all the connected nodes. For each stock s at time t, the model takes both its own LOB features and the LOB features from other related pairs of (s,t) into account. It then forecasts the realized volatility based on both self node features and neighbor node features, instead of the traditional approach which considers only self node features.

The Graph Transformer operator for the l-th layer with C heads is written as:

$$\widehat{h}_{s,t,c}^{(l+1)} = W_{1,c} h_{s,t}^{(l)} + \sum_{node_{i,j} \in \mathcal{N}_{s,t}} \alpha_{i,j,c} W_{2,c} h_{i,j}^{(l)} \tag{4.8}$$

$$h_{s,t}^{(l+1)} = \sigma(\oplus_{c=1}^{C} \widehat{h}_{s,t,c}^{(l+1)}) \tag{4.9}$$

Equation 4.8 first calculates the output vector $\widehat{h}_{s,t,c}^{(l+1)}$ for one single head c, in which $h_{i,j}^{(l)} \in \mathbb{R}^{d_l \times 1}$ is the l-th layer hidden node embedding for $node_{i,j}$, $W_{1,c}, W_{2,c} \in (\mathbb{R}^{\widehat{d}_{l+1} \times d_l})^2$ are trainable parameters. $\alpha_{i,j,c}$ are attention coefficients associated with $node_{i,j}$ for head c. It is calculated via dot product attention [Bahdanau et al., 2014] by

$$\alpha_{i,j,c} = softmax(\frac{(W_{3,c} h_{s,t}^{(l)})^\top (W_{4,c} h_{i,j}^{(l)})}{\sqrt{d_l}}) \tag{4.10}$$

where $W_{3,c}$ and $W_{4,c}$ are both trainable parameters of size $\widehat{d}_{l+1} \times d_l$.

We then use Equation 4.9 to aggregate the output from all heads into a final output vector $h_{s,t}^{(l+1)}$ for the l-th layer. It is then used as the input for the $(l+1)$-th layer. In this equation, $\oplus$ denotes the concatenation operation and σ is an activation function such as ReLU [Glorot et al., 2011]. We show the structure of this operator in Figure 4.2.

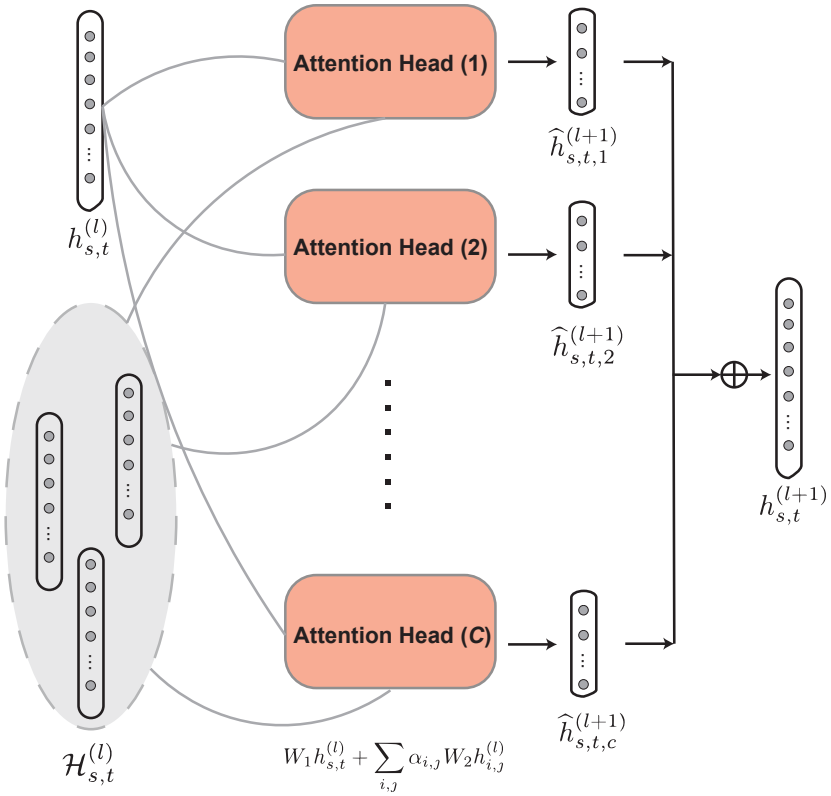


Figure 4.2: Illustration of Graph Transformer operator. $\mathcal{H}_{s,t}^{l}$ represents the l-th layer hidden vectors for all the nodes in $\mathcal{N}_{s,t}$. We can then accumulate multiple layers of this structure to build a Graph Transformer Network.

We can then accumulate multiple layers of this structure to better retreive information. Suppose that our Graph Transformer Network (GTN) has L layers in total, for each node, its initial node features $h_{s,t}^{(0)} = F_{s,t}$ will be transformed into a node embedding $h_{s,t}^{(L)} \in \mathbb{R}^{d_L \times 1}$. We then use a fully-connected layer to get the final predictions of realized volatility from the node embeddings. This operation is written as

$$\widehat{RV_{s,t,\Delta T}} = \sigma(W_0^\top h_{s,t}^{(L)}) \tag{4.11}$$

where $W_0 \in \mathbb{R}^{d_L \times 1}$ are trainable parameters in the fully connected layer.

We then use Root Mean Square Percentage Error (RMSPE) as our loss function to evaluate the model and propagate back into the model. It is define as

$$RMSPE = \sqrt{\frac{1}{N} \sum_{s,t} \left(\frac{\widehat{RV_{s,t,\Delta T}} - RV_{s,t,\Delta T}}{RV_{s,t,\Delta T} + \epsilon} \right)^2} \tag{4.12}$$

where N is the total number of nodes in the graph and ϵ is a small constant to avoid overflow.

We use RMSPE instead of the standard Mean Square Error (MSE) because the volatilities of different stocks are intrinsically different. For example, more liquid stocks are usually more volatile than less liquid stocks [Domowitz et al., 2001]. The RMSPE loss function helps normalize the difference among stocks to make sure that the model has a similar effect on all stocks.

4.5 Experiments

4.5.1 LOB data

We use NYSE daily TAQ data[2] as our limit order book data. The data contains all the quotes (only first limit, i.e., best bid and best ask) and trades happening in the US stock exchanges. We select the entries concerning all the stocks included in the S&P 500 index[3] as our universe. Compared with other researches (5 stocks in Mäkinen et al. [2019], 23 stocks in Rahimikia and Poon [2020b]), this large selection of around 500 stocks also covers some less liquid stocks. We will show that it is more difficult to have a good prediction on less liquid stocks in Section 4.6.1.

Data Sampling

Following Barndorff-Nielsen et al. [2009]; Rahimikia and Poon [2020a], we first sample the data with a fixed frequency T_f. Since we focus on short-term volatility forecasting in this paper, we use a *one second* sampling frequency instead of 5 minutes used by Rahimikia and Poon [2020a] to forecast daily volatility. Our sampling strategy is as follows:

- For quote data, we snapshot the best ask price (P_a^1), best bid price (P_b^1), best ask size (V_a^1) and best bid size (V_b^1) for each stock at the end of each second.
- For trade data, we aggregate all the trades for each stock during each second. We record the number of trades (N_t), the total number of shares traded (V_t) and the volume weighted average price (P_t)[4] of all trades.

An example of sampled quote and trade data for stock A on Jan. 3rd, 2017 is shown in Table 4.1 and 4.2. In addition to the previously defined LOB fields, we also have *Date*, *Symbol* and *seconds*. The *seconds* field signifies the number of seconds after the market open. We note that the *seconds* are not continuous. This is because there are occasions that there is no update in the LOB in that second. For quote data it implies that the quote is the same as the last second, for trade data it implies that there is no trade in that second.

[2] https://www.nyse.com/market-data/historical/daily-taq

[3] https://www.spglobal.com/spdji/en/indices/equity/sp-500/#overview

[4] $P_t = \frac{\sum_i^{N_t} P_i V_i}{\sum_i^{N_t} V_i}$ where P_i is the price for trade i and V_i is the number of shares traded. N_t is the total number of trades recorded during second t.

Date	Symbol	seconds	P_b^1	V_b^1	P_a^1	V_a^1
1/3/2017	A	0	45.92	1	46.09	1
1/3/2017	A	1	45.92	3	46	2
1/3/2017	A	2	45.92	2	46	2
1/3/2017	A	5	45.92	2	46	1
1/3/2017	A	6	45.94	1	46.05	3

Table 4.1: Sampled quote data

Date	Symbol	seconds	N_t	V_t	P_t
1/3/2017	A	0	8	1500	45.802
1/3/2017	A	1	7	24959	45.94963
1/3/2017	A	2	1	300	45.96
1/3/2017	A	3	1	100	45.9544
1/3/2017	A	5	7	916	45.97817

Table 4.2: Sampled trade data

Data Bucket

As introduced in Section 4.3, our goal is to forecast the realized volatility ΔT seconds after a given timestamp t based on the features built from a backward window of $\Delta T'$ seconds.

Hence, we need to build buckets which have the length of $\Delta T + \Delta T'$ seconds between $t - \Delta T'$ and $t + \Delta T$. In each bucket, we only use the returns between t and $t + \Delta T$ to calculate our target $RV_{s,t,\Delta T}$, we use both returns and other information from LOBs between $t - \Delta T'$ and t to build a set of features with LOB data encoder.

In our experiments on US stocks, we create 6 buckets for each stock each day. We select 10:00, 11:00, 12:00, 13:00, 14:00 and 15:00 EST as 6 different t. For the sake of simplicity, we use $\Delta T = \Delta T'$. We use three different ΔT of 600 seconds, 1200 seconds and 1800 seconds to show that our model is robust to this choice and is capable of forecasting volatility on different horizons. This choice also ensures that there is no overlap between buckets to avoid information leakage. The detailed experiment results are shown in Section 4.5.4.

Data Split

We split our LOB data into 3 parts: train, validation and test. We ensure that the validation set and the test set are no earlier than the training set to avoid backward looking. We also remove the buckets where there are no quotes or trades during ΔT. Detailed statistics of our dataset are shown in Table 4.3.

Table 4.3: The statistics of the LOB data with $\Delta T = 600$.

	train	val	test
start	Jan-17	Jan-20	Jan-21
end	Dec-19	Dec-20	Oct-21
# time	4,464	1,505	1,236
# stock	494	494	494
# bucket	2,141,108	743,068	607,770
proportion	61%	21%	18%

4.5.2 Graph Building

As stated in Section 4.3, we consider both temporal ($\mathcal{G}_t$) and cross-sectional ($\mathcal{G}_s$) relationships among buckets. In this subsection, we first introduce a method to construct relations without using other data than our LOB data. We also introduce other relations we constructed with external data, for example, stock sector and supply chain data.

Temporal Relationship

To construct the temporal relationship among the nodes, we only use LOB data. The intuition behind this temporal relationship is that at a new moment t, we check if there are moments in the history which are similar.

As introduced in Equation 4.6, for each $node_{s,t}$, we can calculate k_{num} features. Suppose that $f^i_{s,t}$ is the i-th feature for stock s at time t. For each time t, we let

$$Q^i_t = ([f^i_{1,t}, ..., f^i_{n,t}])^\top \tag{4.13}$$

where $Q^i_t \in \mathbb{R}^{n\times 1}$ represents the feature i of all stocks at time t.

Given a time t_0, we calculate the RMSPE (Equation 4.12) of Q^i between t_0 and all other t. We then choose the K-smallest RMSPE to form K pairs of times, represented by $\mathcal{G}_{t_0} = [(t_0, t_1), ..., (t_0, t_K)]$. Then for each such pair (t_i, t_j), we connect the two nodes where s is the same and the time is t_i and t_j respectively. This can be written as:

$$\begin{gathered}\forall (t_i, t_j) \in \mathcal{G}_{t_0}, \forall k \in [1, n], \\ \text{connect } node_{s_k,t_i} \text{ and } node_{s_k,t_j}\end{gathered} \tag{4.14}$$

After this operation, we get $K \times n$ single-directed edges in the graph for t_0. We then repeat the same process for all t, and get $K \times n \times m$ edges in total.

In our study, we use two features to get $2 \times K \times n \times m$ edges in the graph, namely, the average quote WAP[5] for the first 100 seconds and the average quote WAP for the last 100 seconds in the bucket.

[5]Weighted Average Price, defined as $\frac{P^1_b V^1_a + P^1_a V^1_b}{V^1_a + V^1_b}$

Cross-sectional Relationship

Unlike times, there are intrinsic relations among stocks since each stock represents a company in the real life. Hence, in addition to building relationship based on LOB features, we can also build the cross-sectional graph with external data.

Feature Correlation

Using the same idea introduced in 4.5.2, we first build

$$Q_s'^i = ([f^i_{s,1}, ..., f^i_{s,m}])^\top \tag{4.15}$$

$Q_s'^i \in \mathbb{R}^{m\times 1}$ represents the feature i of stock s at t.

We then build the edges for s_0 with

$$\begin{aligned} &\forall (s_i, s_j) \in \mathcal{G}_{s_0}, \forall k \in [1, m], \\ &\text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k} \end{aligned} \tag{4.16}$$

with $\mathcal{G}_{s_0} = [(s_0, s_1), ..., (s_0, s_{K'})]$ denoting the stock pairs which are among the K'-smallest feature RMSPE for stock s_0.

We repeat the same process for all stocks and we use the same features as in 4.5.2 to build another $2 \times K' \times n \times m$ edges.

Stock Sector

In finance, each company is classified into a specific sector with Global Industry Classification Standard[6] (GICS). It is shown that the performances of stocks in the same sector are often correlated [Vardharaj and Fabozzi, 2007]. Hence, we simply connect the times of a pair of stocks if they belong to the same sector. This is written as:

$$\begin{aligned} &\forall\ E_{sec}, \forall s_i \in E_{sec}, \forall s_j \in E_{sec} \text{ and } s_j \neq s_i, \\ &\forall k \in [1, m],\ \text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k} \end{aligned} \tag{4.17}$$

where E_{sec} is the ensemble of all the stocks in the sector *sec*.

There are four granularities in GICS sector data: Sector, Industry Group, Industry, Sub-Industry. We can therefore construct four different types of edges with this GICS sector data. In our experiments, we use the Industry granularity as it gives a good performance with a reasonable number of edges. The detail of this choice is discussed in Section 4.6.3.

Supply Chain

Supply chain describes the supplier-customer relation between companies and it is proved to be useful in multiple financial tasks such as risk management [Yang et al., 2020a] and performance prediction [Chen and Robert, 2021]. We use the supply chain data from Factset[7] to build this

[6] https://www.msci.com/our-solutions/indexes/gics
[7] https://www.factset.com/marketplace/catalog/product/factset-supply-chain-relationships

graph. We connect two companies if they have a supplier-customer relationship in the training period. This is described as:

$$\begin{gathered}\forall(s_i, s_j) \in \mathcal{G}_{supply}, \forall k \in [1, m] \\ \text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k}\end{gathered} \tag{4.18}$$

where $\mathcal{G}_{supply}$ is the ensemble of all the supplier-customer relations among the stocks.

We show a detailed statistics of each type of relationship we built in Table 4.4. In addition to using these relations separately, we can also join these relations by simply putting all the edges together in the same graph. We will show that combining the edges can help improve the result in Section 4.5.4. The pseudocode used to build all four relationships is included in Appendix 4.B.

Table 4.4: The number of edges in each relationship. The total number denotes the number of all the edges combined. It is not exactly the sum of all individual edge counts since there are duplicated edges. These numbers are based on the nodes in the training set for $\Delta T = 600$.

Type	Relation	**# edges**
Temporal	Feature Corr	8.36M
Cross-sectional	Feature Corr	8.21M
	Sector	47.86M
	Supply Chain	22.22M
Total		76.33M

4.5.3 Experiment Setup

In order to prove the effectiveness of our model structure, we also include the performance of some other widely used models as our benchmarks.

- `Naïve Guess`: We simply use the realized volatility between $t - \Delta T'$ and t to predict the target. It is written as $\widehat{RV_{s,t,\Delta T}} = RV_{s,t-\Delta T',\Delta T'}$.
- `HAR-RV`: Heterogeneous AutoRegressive model of Realized Volatility. A simple but effective realized volatility prediction model proposed by Corsi [2009].
- `LightGBM`: A gradient boosting decision tree model introduced by Ke et al. [2017]. It is proven to be highly effective on tabular data.
- `MLP`: Multi-Layer Perception network [Rumelhart et al., 1985]. We build a fully connected neural network with three layers, which have 128, 64, 32 hidden units respectively.
- `TabNet`: A neural network proposed by Arık and Pfister [2020] which specializes in dealing with tabular data.
- `Vanilla GTN-VF`: Our Graph Transformer Network for Volatility Forecasting trained without any relationship information.

In addition, we show the model performance with different relations individually to demonstrate how different types of relational data help improve the result compared with Vanilla GTN-VF and other benchmarks.

In addition, we compare the model performance with different relations individually to demonstrate how different types of relational data help improve the result compared with Vanilla GTN-VF and other benchmarks. These variants include Cross-sectional Feature Correlation (GTN-VF Cross FC), Temporal Feature Correlation (GTN-VF Temp FC), Cross-sectional Sector relationship (GTN-VF Sector) and Cross-sectional Supply Chain relationship (GTN-VF Cross Supply Chain). The full GTN-VF includes all four types of relations.

MLP, TabNet and all the variants of our GTN-VF model are implemented in PyTorch [Paszke et al., 2019] with Adam optimizer [Kingma and Ba, 2014]. For our GTN-VF, we use a 3-layer ($L = 3$) Graph Transformer Network with 8 heads ($C = 8$). All three layers have 128 channels ($d_1 = d_2 = d_3 = 128$). We embed our numerical features into a 73-dimension vector ($k_{num} = 73$, Section 4.4.1) and categorical features into a 32-dimension vector ($k_{cat} = 32$). Other baseline models are implemented without deep learning framework.

In order to provide a fair comparison and guard against hyperparameter hacking, we sweep over the same set of hyperparameters for all GTN-VF variants and choose the best setting for each variant according to the performance on the validation set. We then fix these parameters for all experiments on the test set. The appendix contains further implementation details on both baseline models and GTN-VF models.

4.5.4 Experiment Results

Table 4.5: RMSPE values of all the models on both validation set and test set. In addition to the baseline models, we also include the individual performance of the GTN-VF with four relations (Table 4.4), i.e., Cross-sectional Feature Correlation (GTN-VF Cross FC), Temporal Feature Correlation (GTN-VF Temp FC), Cross-sectional Sector relationship (GTN-VF Sector) and Cross-sectional Supply Chain relationship (GTN-VF Cross Supply Chain). The full GTN-VF includes all four types of relations.

	$\Delta T = 600$		$\Delta T = 1200$		$\Delta T = 1800$	
Model	val	test	val	test	val	test
Naïve Guess	0.2911	0.2834	0.2650	0.2628	0.2296	0.2364
HAR-RV	0.2684	0.2612	0.2149	0.2061	0.1968	0.1939
LightGBM	0.2583	0.2492	0.2414	0.2035	0.2349	0.1963
MLP	0.2431	0.2514	0.2200	0.2308	0.2270	0.1999
TabNet	0.2517	0.2478	0.2212	0.1996	0.2204	0.2019
Vanilla GTN-VF	0.2457	0.2498	0.2229	0.2251	0.2092	0.2160
GTN-VF Cross FC	0.2414	0.2382	0.2162	0.2196	0.2066	0.2046
GTN-VF Temp FC	0.2326	0.2358	0.1974	0.1921	0.1896	0.1853
GTN-VF Cross Sector	0.2406	0.2422	0.2067	0.2248	0.2071	0.2091
GTN-VF Cross Supply Chain	0.2430	0.2411	0.2105	0.2244	0.2057	0.2000
GTN-VF Cross FC + Temp FC	0.2326	0.2306	0.1936	0.1917	0.1848	0.1802
GTN-VF	**0.2314**	**0.2287**	**0.1916**	**0.1892**	**0.1809**	**0.1798**

In our short-term realized volatility forecasting task, we use a 3-layer ($L = 3$) Graph Transformer Network with 8 heads ($C = 8$). All three layers have 128 channels ($d_1 = d_2 = d_3 = 128$). We embed our categorical features into a 32-dimension vector ($k_{cat} = 32$).

The detailed results of our experiments are shown in Table 4.5. We can see that our full GTN-VF model with all four types of relational information outperforms all baseline models and each type of relational information individually on all prediction horizons. The improvement is significant. In average, we gain 6% in RMSPE compared with Naïve Guess and 2% compared with the best baseline model TabNet on test set. It proves that our GTN model structure and relation building methods are effective.

In terms of individual relationship, all relations are useful since they all show improvement compared with the vanilla model. The temporal feature correlation shows the most predicting power, contributing 1.2% RMSPE gain while the Sector relation only contributes 0.7% improvement when the window is set to 600 seconds although it has the largest number of edges. This can be explained by the 'noise' included in this type of information since we need to connect every two stocks in the same sector although not all of them have significant connection. However, feature correlation only selects the two most related stocks or times, making it more discriminational when building edges. This suggests that if we have the constraint on the number of edges in the graph, quality is more important than quantity. On the other hand, these relations are complementary to each other, adding more relations on top of existing relations can help improve the prediction if we have enough computing power.

It is also worth noting that when we forecast the realized volatility with a longer forward looking and backward looking window, the result is better. This is simple because the same volatility jump causes more volatility changes in shorter forecasting horizon [Ma et al., 2019].

4.6 Ablation Studies

4.6.1 Prediction accuracy and stock liquidity

In general, it is more difficult to have a good prediction on less liquid stocks because there are fewer market participants for them. One sudden change in quote or trade can cause significant volatility jump, which is difficult to foresee. We analyze the result to understand where the improvement comes from.

We first split the stocks into 50 buckets according to their average daily turnover, which represents the liquidity of a stock. We calculate the RMSPE in each bucket for both Naïve Guess and GTN-VF. The result is illustrated in Figure 4.3.

First, we notice that more liquid stocks have smaller RMSPE for both models, which is intuitive. The prediction for the most liquid stocks is 5% better than the least liquid stocks in terms of RMSPE, which is a large margin in realized volatility forecasting. We can also see that our graph based model has more improvement on the less liquid stocks (around 8% for more liquid stocks and 2% for less liquid stocks), although it is more effective than Naïve Guess on all scenarios.

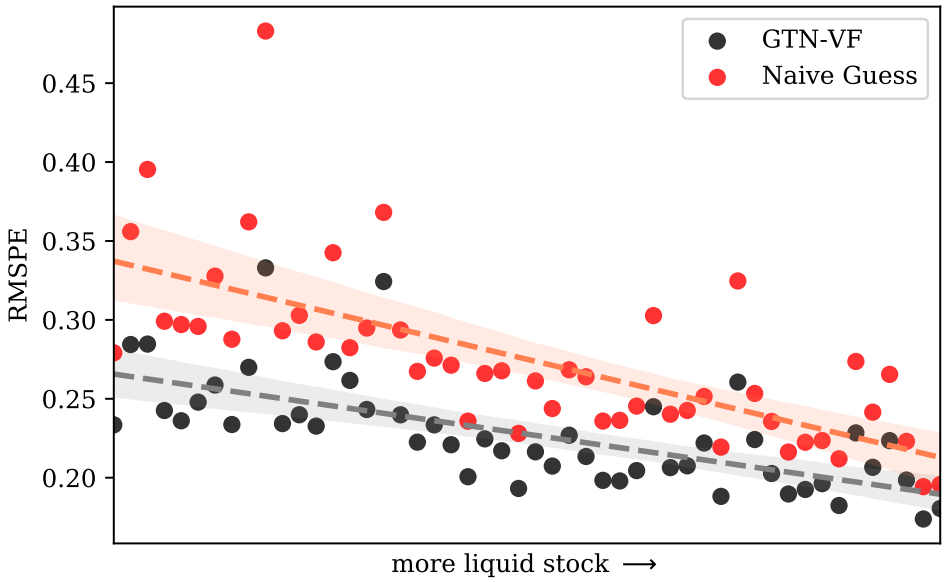


Figure 4.3: The relationship between stock liquidity and prediction RMSPE. The dots denote the RMSPE for the buckets and the dashed lines are the trend lines calculated with linear regression. This figure is based on test result and $\Delta T = 600$.

4.6.2 Prediction accuracy and node connection

We also investigate how our model performs on different nodes. We use the same approach in Section 4.6.1 by splitting nodes into buckets according to the number of edges connected to each node. We can see from Figure 4.4 that more connected nodes usually have better RMSPE result, with a 2% difference between the most connected and the least connected. This can be explained by the fact that mode connected nodes make decision based on more information from their neighbor, while the nodes with fewer or no connections can only rely on the information from themselves. This phenomenon proves again the effectiveness of our graph based method.

4.6.3 Sector Relationship

As introduced in Section 4.5.2, there are four granularities in the GICS sector data. We run different experiments to evaluate the performance of each type of sector relationship. The result is shown in Table 4.6.

Table 4.6: Test RMSPE with different granularities of GICS sector. The result for Sector is not available as the number of edges is too big to fit in the memory.

Granularity	# edges	test RMSPE
Sector	178.7M	N.A.
IndustryGroup	83.39M	0.2578
Industry	47.86M	0.2422
SubIndustry	19.88M	0.2441

We observe that Industry shows the best performance with a modest number of edges. If we

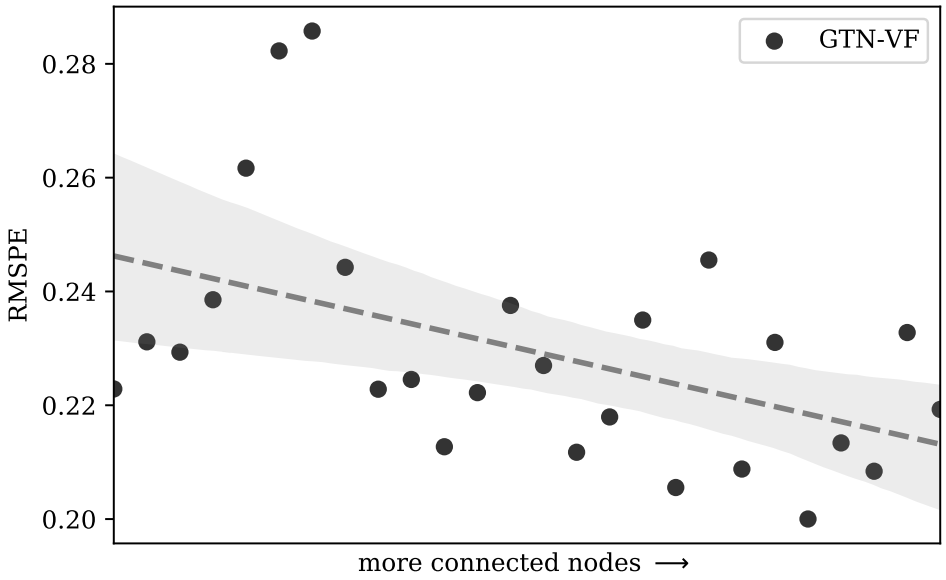


Figure 4.4: The relationship between node connection and RMSPE. This figure is based on test result and $\Delta T = 600$.

add more edges, such as IndustryGroup, the RMSPE decreases since the relations among stocks are less meaningful. For the Sector granularity, we are not even able to obtain a result since the number of edges exceeds the memory limit. Hence, in our final model combining multiple sources of relations, we choose Industry as our sector relationship.

4.7 Conclusion

We forecast the short-term realized volatility in a multivariate approach. We design a graph based neural network: Graph Transformer Network for Volatility Forecasting which incorporates both features from LOB data and relationship among stocks from different sources. Through extensive experiments on around 500 stocks, we prove the that our method outperforms other baseline models, both univariate and multivariate. In addition, the model structure allows to combine an unlimited number of relations, the study of the effectiveness of other relational data is open for future researches.

Appendix

4.A List of node features

We introduce the features we use in our experiments.

In each bucket, we have $\Delta T'$ lines as training data. We first calculate an indicator for each line, we then aggregate these indicators in the same bucket with an aggregation function (aggregator). In such way, we have one value per indicator per aggregator as one feature for the bucket. The full list of indicators and aggregators are listed in Table 4.B.1.

For some important indicators, we also calculate their progressive features. It means that instead of applying an aggregator on all the lines, we apply it on the lines between 0 and $\Delta T'/6$, $\Delta T'/3$, $\Delta T'/2$, $2\Delta T'/3$, $5\Delta T'/6$, $\Delta T'$. In such way, we have 6 features per indicator per aggregator for an progressive feature. This is shown in the column Progressive in Table 4.B.1.

We define our aggregation functions as follows. We use a_i to denote the i-th line in the bucket and N represents the total number of lines.

- gini coefficient [Gini, 1921]
$$\frac{\sum_i^N \sum_j^N |a_i - a_j|}{2N^2\overline{a}}$$
- percentage difference
$$\frac{\sum_i^N \mathbf{1}_{a_i \neq a_{i-1}}}{N}$$
- realized volatility (Equation 4.2)
$$\sqrt{\frac{1}{N}\sum_i^N a_i^2}$$
- percentage greater than mean
$$\frac{\sum_i^N \mathbf{1}_{a_i > \overline{a}}}{N}$$
- percentage greater than zero
$$\frac{\sum_i^N \mathbf{1}_{a_i > 0}}{N}$$

- median deviation

$$median(|a_i - \overline{a}|)$$

- energy

$$\frac{1}{N}\sum_{i}^{N} a_i^2$$

- InterQuartile Range (IQR)

$$Q_{75}(a) - Q_{25}(a)$$

where $Q_i(a)$ denotes the i-th percentile value for the series a.

4.B Graph Building Pseudocode

To help better understand our graph building process, we give the pseudocode we used in each type of graph. Temporal feature correlation relationship is built with Algorithm 1, cross-sectional feature correlation relationship is built with Algorithm 2, cross-sectional activity sector relationship is built with Algorithm 3 and cross-sectional supply chain relationship is built with Algorithm 4.

Algorithm 1 Graph building algorithm for temporal feature correlation relationship

Input:
Selected features: I
Selected features for all stocks and all times: $\{f_{s,t}^i \mid \forall s \in [1, ..., n], \forall t \in [1, ..., m], \forall i \in I\}$
Number of pairs selected for each t: K
Output:
Temporal feature correlation among the buckets: $\mathcal{R}_t$

for $i \in I$ **do**
  for $t = 1, ..., m$ **do**
    $Q_t^i \leftarrow ([f_{1,t}^i, ..., f_{n,t}^i])^\top$
  end for
end for
$\mathcal{R}_t \leftarrow \{\}$ ▷ Initialize an empty list for all relations
for $i \in I$ **do**
  for $t_0 = 1, ..., m$ **do**
    for $t = 1, ..., t_0$ **do**
      $L_{t_0,t} \leftarrow RMSPE(Q_{t_0}^i, Q_t^i)$
    end for
    $\mathcal{G}_{t_0} \leftarrow K$ pairs of (t_0, t) with the smallest $L_{t_0,t}$
    $\mathcal{R}_{t_0} \leftarrow \{\}$
    for $s = 1, ..., n$ **do**
      $\mathcal{R}_{t_0} \leftarrow \mathcal{R}_{t_0} \cup \{(node_{s,t_0}, node_{s,t})\}$
    end for
    $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup \mathcal{R}_{t_0}$
  end for
end for

Table 4.B.1: The list of features built from LOB data

Type	Notation	Description	Aggregators	Progressive	Count
Quote	$\frac{P_b^1 V_a^1 + P_a^1 V_b^1}{V_a^1 + V_b^1}$	WAP	mean, std, gini mean of first 100, mean of last 100	N	5
	P_a^1	Ask Price	% difference	N	1
	P_b^1	Bid Price	% difference	N	1
	$\frac{P_a^1 - P_b^1}{P_a^1 + P_b^1}$	Price Relative Spread	mean, std, gini	N	3
	$WAP - P_b^1$	WAP bid difference	mean, std, gini	N	3
	$log(\frac{WAP_i}{WAP_{i-1}})$	Return	realized volatility	Y	6
	$log(\frac{WAP_i}{WAP_{i-1}})^2$	Squared Return	std, gini	N	2
	$\frac{V_a^1 - V_b^1}{V_a^1 + V_b^1}$	Size Relative Spread	mean, std, gini	N	3
	V_a^1	Ask Size	% difference	N	1
	V_b^1	Bid Size	% difference	N	1
	$V_a^1 / \overline{V_a^1}$	Normalized Ask Size	mean, std, gini	N	3
	$V_a^1 + V_b^1$	Total Size	sum, max	N	2
	$\lvert V_a^1 - V_b^1 \rvert$	Size Imbalance	sum, max	N	2
Trade	P_t	Price	% greater than mean, % less than mean median diviation, energy, IQR	N	5
	$log(\frac{P_{t,i}}{P_{t,i-1}})$	Return	realized volatility	Y	6
			% greater than 0, % less than 0	N	2
	$log(\frac{P_{t,i}}{P_{t,i-1}})^2$	Squared Return	std, gini	N	2
	V_t	Size	sum	Y	6
			max, median diviation, energy, IQR	N	4
	t	Seconds	count	Y	6
	N_t	Order Count	sum	Y	6
			max	N	1
	$P_t \times V_t$	Amount	sum, max	N	2
	Total				73

Algorithm 2 Graph building algorithm for cross-sectional feature correlation relationship

Input:

Selected features: I

Selected features for all stocks and all times: $\{f_{s,t}^i \mid \forall s \in [1, ..., n], \forall t \in [1, ..., m], \forall i \in I\}$

Number of pairs selected for each s: K'

Output:

Cross-sectional feature correlation among the buckets: $\mathcal{R}_s$

for $i \in I$ **do**
  for $s = 1, ..., n$ **do**
    $Q_s'^i \leftarrow ([f_{s,1}^i, ..., f_{s,m}^i])^\top$
  end for
end for
$\mathcal{R}_s \leftarrow \{\}$ ▷ Initialize an empty list for all relations
for $i \in I$ **do**
  for $s_0 = 1, ..., n$ **do**
    for $s = 1, ..., s_0$ **do**
      $L_{s_0,s} \leftarrow RMSPE(Q_{s_0}^i, Q_s^i)$
    end for
    $\mathcal{G}_{s_0} \leftarrow K'$ pairs of (s_0, s) with the smallest $L_{s_0,s}$
    $\mathcal{R}_{s_0} \leftarrow \{\}$
    for $t = 1, ..., m$ **do**
      $\mathcal{R}_{s_0} \leftarrow \mathcal{R}_{s_0} \cup \{(node_{s_0,t}, node_{s,t})\}$
    end for
    $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \mathcal{R}_{s_0}$
  end for
end for

Algorithm 3 Graph building algorithm for cross-sectional activity sector relationship

Input:

Stock sectors: $\mathcal{S} = \{\mathcal{S}_1, ..., \mathcal{S}_{N_s}\}$

Output:

Sector relationship among the buckets: $\mathcal{R}_{sector}$

$\mathcal{R}_{sector} \leftarrow \{\}$ ▷ Initialize an empty list for all relations
for $i = 1, ..., N_s$ **do**
  for $s_j \in \mathcal{S}_i$ **do**
    for $s_k \in \mathcal{S}_i$ **do**
      if $s_j \neq s_k$ **then**
        for $t = 1, ..., m$ **do**
          $\mathcal{R}_{sector} \leftarrow \mathcal{R}_{sector} \cup \{(node_{s_j,t}, node_{s_k,t})\}$
        end for
      end if
    end for
  end for
end for

Algorithm 4 Graph building algorithm for cross-sectional supply chain relationship

Input:

Supply chain relations among stocks: $\mathcal{G}_{sc} = \{(s_i, s_j) \mid s_i \text{ and } s_j \text{ have supplier-customer relationship }\}$

Output:

Supply chain relationship among the buckets: $\mathcal{R}_{sc}$

$\mathcal{R}_{sc} \leftarrow \{\}$ ▷ Initialize an empty list for all relations
for $(s_i, s_j) \in \mathcal{R}_{sc}$ **do**
  for $t = 1, ..., m$ **do**
    $\mathcal{R}_{sc} \leftarrow \mathcal{R}_{sc} \cup \{(node_{s_i,t}, node_{s_j,t})\}$
  end for
end for

4.C Details on Experiment Setup

4.C.1 Model Details

In all baseline models, except for Naïve guess and HAR-RV that do not need features, the input features ($F_{s,t}$) are the same as GTN-VF, including 73 numerical features and 1 categorical feature. The categorical feature embedding is of size 32 whenever applicable. All loss functions are set to RMSPE. The hyperparameters are chosen based on validation set performance and the results presented are based on the best set of hyperparameters.

HAR-RV

The model is written as

$$\widehat{RV_{s,t+1d,\Delta T}} = c + \beta^{(d)} RV_{s,t}^{(d)} + \beta^{(w)} RV_{s,t}^{(w)} + \beta^{(m)} RV_{s,t}^{(m)} \tag{4.19}$$

where $RV_{s,t}^{(d)}$, $RV_{s,t}^{(w)}$ and $RV_{s,t}^{(m)}$ are respectively the daily, weekly and monthly average realized volatilities before t. $\beta^{(d)}$, $\beta^{(w)}$, $\beta^{(m)}$ and c are coefficients determined by linear regression. The weekly and monthly average realized volatilities are calculated with $RV_{s,t}^{(w)} = \frac{1}{5}(RV_{s,t} + ... + RV_{s,t-4d})$ and $RV_{s,t}^{(w)} = \frac{1}{21}(RV_{s,t} + ... + RV_{s,t-20d})$.

In our implementation, we calibrate one HAR-RV model for each sampling time since we find a better result than mixing all the sampling times and calibrate only one model. Therefore, we have 6 different HAR-RV models for 10:00, 11:00, 12:00, 13:00, 14:00 and 15:00.

LightGBM

We use the LightGBM package[8] to implement this baseline model. We use gradient boosting decision tree (GBDT) algorithm and set its learning to 0.1. The model is trained for a maximum

[8] https://github.com/microsoft/LightGBM

of 1,000 iterations and the training process is stopped earlier if there is no improvement on the validation set for 50 iterations.

MLP

There are three hidden layers in this fully-connected neural network. Their sizes are 128, 64 and 32 respectively. The learning rate of Adam optimizer is set to 0.01, and the batch size is set to 2048. The model is trained for a maximum of 200 iterations and the training process is stopped earlier if there is no improvement on the validation set for 20 iterations.

It is worth noting that we normalize our features to values between -1 and 1 before training. This is to avoid overflow in the computation process.

TabNet

We set the width of both prediction layer and attention embedding to 16. All other settings, including learning rate, batch size, training epochs, early stopping and feature normalization are the same as the MLP model.

GTN-VF and its variants

In our experiments, the GTN-VF and its variants share the same set of hyperparameter and model dimension. The model dimension is already introduced in Section 5.3.

During training, we set the batch size to 2048 and the initial learning rate to 0.001. The model is trained for a maximum of 100 iterations and the training process is stopped earlier if there is no improvement on the validation set for 20 iterations. We also reduce the learning rate by half if there is no improvement for 5 epochs.

In the neighbor sampling process, we sample all the connected neighbors without a maximum limit.

4.C.2 Hardware

Except for Naïve Guess, HAR-RV and LightGBM which can be quickly trained without GPU, we ran the experiments on a single machine with one NVIDIA Tesla V100 GPU (16GB RAM and 32GB/s bandwidth), 8 cores of Intel Xeon CPU (Broadwell E5-2686 v4) and 61GB of RAM. In average, a GTN-VF with all four relationships takes one hour to train. For comparison, the training time for MLP is 20 minutes and 3 hours for TabNet.

Part III

Beyond Finance - NLP Applications in Contemporary Arts

Chapter 5

Mapping the contemporary art world with ArtLM: an art-specific NLP model

Note:

- This chapter is co-authored by Qinkai Chen, Mohamed El-Mennaoui, Antoine Fosset, Amine Rebei, Haoyang Cao, Philine Bouscasse, Christy Eóin O'Beirne, Sasha Shevchenko and Mathieu Rosenbaum.
- This chapter is available as preprint *arXiv:2212.07127*.
- This chapter is submitted to *the 32nd International Joint Conference on Artificial Intelligence*.

Abstract

With an increasing amount of data in the art world, discovering artists and artworks suitable to collectors' tastes becomes a challenge. It is no longer enough to use visual information, as contextual information about the artist has become just as important in contemporary art. In this work, we present a generic Natural Language Processing framework (called ArtLM) to discover the connections among contemporary artists based on their biographies. In this approach, we first continue to pre-train the existing general English language models with a large amount of unlabeled art-related data. We then fine-tune this new pre-trained model with our biography pair dataset manually annotated by a team of professionals in the art industry. With extensive experiments, we demonstrate that our ArtLM achieves 85.6% accuracy and 84.0% F1 score and outperforms other baseline models. We also provide a visualization and a qualitative analysis of the artist network built from ArtLM's outputs.

Contents

5.1 Introduction

The contemporary art market has been growing at an unprecedented pace since the beginning of this century [Kräussl et al., 2016]. Traditionally, collectors rely on social interactions, professional advice and media to find the artworks they wish to acquire. However, with more artists and their artworks on the market, this process of recommending suitable artworks to collectors with different tastes became inefficient. Recently, with the wave of accelerated digitization in the art industry due to the COVID-19 pandemic [Noehrer et al., 2021] and the rapid development in artificial intelligence, researchers started to modernize this art discovery process with state-of-the-art technologies.

Most researchers focus on the visual aspect of the artwork. Elgammal et al. [2018] fine-tune some commonly used model structures in computer vision, such as AlexNet [Krizhevsky et al., 2017] and ResNet [He et al., 2016], to classify the styles of the paintings. Kim et al. [2018] further expand this method to group artworks into pre-defined art concepts and principles in multiple dimensions. However, the visual information has certain limits as contemporary art goes beyond the aesthetics. The surrounding narration is also essential: the artist practice, the message conveyed in the work, the influences used by the artist and many more contextual attributes also have a strong influence on the taste in art.

Hence, the easily available textual information, such as the artists' biographies, comes to our attention. Few existing research in the literature focus on this domain. Kim et al. [2022] transform style labels into a fixed-length embedding with the help of Natural Language Processing (NLP) models and combine it with visual elements. Fosset et al. [2022] tag the artworks by analyzing the authors' biographies through static (Word2Vec [Mikolov et al., 2013a]) or contextualized (BERT [Devlin et al., 2018; Chen, 2021]) embeddings without supervised fine-tuning or unsupervised pre-training.

To close the gap in the existing research introduced in Section 5.2, adopting the idea of transfer learning [Weiss et al., 2016], we introduce a generic NLP framework that we call ArtLM to solve this artist discovery problem formulated in Section 5.3. The details of ArtLM are introduced in Section 5.4. In ArtLM, we first reuse a general English language model pre-trained on a very large amount of English texts. We then continue the pre-training process with a large non-labeled art-related text dataset from WikiArt[1] in order to add art knowledge to the general English model. We finally fine-tune this pre-trained art model with a small set of biography pairs manually labeled by a team of professionals in the contemporary art industry (an example is shown in Table 5.1). This last step makes our model specialized in this particular artist pair classification task.

In section 5.5, with extensive experiments, we demonstrate that our ArtLM approach outperforms other baseline models in both accuracy and F1 score. We also prove that the continued pre-training and the fine-tuning are both essential to the state-of-the-art performance, and that these two steps are robust to the base model choices. In addition, we visualize the artist network built from our model output and compare it with the ground-truth.

[1] https://www.wikiart.org/

Table 5.1: An example of our labeled artist biography pair data. *bio_a* is the biography of the *artist_a* and *bio_b* is the biography of the *artist_b*. The *label* is manually annotated by a team of professionals in the art industry signifying if two artists are connected (1 if connected and 0 if not connected). Our goal is to train a model which can tell if two artists are connected given their biographies.

artist_a	artist_b	bio_a	bio_b	label
Carl Edouard Keita	Kudzanai Violet Hwami	Born in 1992 in Abidjan, Carl-Edouard Keita now lives and works in New York. A 2021 graduate of the New York Academy of Art, Carl-Edouard Keita also won the prize for best draughtsman for his graduation work, some of which is presented in this group exhibition. Carl-Edouard Keita discovered the history of African art during his economics studies in Atlanta, through a course offered at his university. As he describes it himself, this discovery was a real aesthetic revelation for him.	Having fled her homeland due to the political unrest and turmoil when she was a child, Zimbabwe-born painter Kudzanai-Violet Hwami expresses her personal experiences of dislocation, displacement and fragmentation through her striking figurative paintings. The artist is interested in the collapsing of geography and time and space symptomatic of a globalised world and high-speed internet, through which both people and information can travel quickly.	1
Sun Xun	Cheng Xinhao	Sun Xun was born in 1980 in Fuxin, Liaoning province, China. Currently lives and works in Beijing. Recent and past histories, intransigent conflicts and tensions, sequential flashes of hand-created images of these are the irrevocable features of Sun Xun's artistic practice that fuses the line between art and animation. A graduate from the Printmaking Department of the China Academy of Arts in 2005, Sun Xun was a professor at the prestigious Academy before founding in 2006 his own Animation Studio. His work primarily involves making images using various materials such as colour powder, woodcuts and traditional ink, and collating these to produce a film, which is often presented in an immersive setting.	Cheng Xinhao (b.1985, Yunnan, China). After receiving his PhD on Chemistry from Peking University in 2013, Cheng continued his career as a photographer, investigating on the issues in the modernization, the construction of knowledge as well as the production of space in Chinese society.	0

5.2 Related Work

5.2.1 Artwork Discovery and Recommendation

Recently, art discovery and application of recommendation systems to the contemporary art scene attracted attention from the research community. For example, Messina et al. [2018] study the impact of including metadata of artworks in addition to visual features on artwork recommendation. With images and transaction data from *UGallery*[2] online artwork store, they show that a hybrid approach improves the performance of the recommendation.

Fosset et al. [2022] propose a novel and innovative approach to build a recommendation engine suited for contemporary art complexities. They combine visual attributes of artworks and contextual information about artists to build similarity graphs, allowing them to make a global and informed recommendation close to users' artistic tastes. However, for the contextual aspect in this work, the authors rely mainly on static Word2Vec embeddings to tag text data with different aspects conveyed by the artists, such as themes, subjects, emotions, etc.

More recently, Wang et al. [2022] use Named Entity Recognition (NER) to extract the topics from the texts from Wikipedia and Graph Convolutional Networks (GCN) to build a artists graph. They further build a recommendation framework (SSAR-GNN) based on the relationship among the artists.

5.2.2 Language Model

In Natural Language Processing, a Language Model (LM) refers to a model that can represent the probability distribution over sequences of words in a language. LMs are generally trained on very large corpus with unsupervised learning tasks. They are usually used as general-propose models and can be adapted to various downstream tasks, including sentence pair classification, through transfer learning [Weiss et al., 2016].

There are two main types of LM in modern NLP: Masked Language Modeling (MLM) [Taylor, 1953; Devlin et al., 2018] and Causal Language Modeling (CLM) [Radford et al., 2018]. The main difference between these two types of models is the training target. MLM aims at predicting a randomly masked word in a sentence given the context before and after the word. This target is used in LMs such as BERT, RoBERTa [Liu et al., 2019], etc. In contrary, CLM's target is to predict the next word given its preceding words, such as GPT [Radford et al., 2019]. There are also models which use both training targets, for instance, XLNet [Yang et al., 2019] and XLM [Lample and Conneau, 2019].

Another commonly used training target is Next Sentence Prediction (NSP), which is a binary classification loss for predicting whether two segments follow each other in the original text. The NSP task is usually combined with other tasks such as MLM or CLM, although some researchers argue that the NSP task does not have a clear contribution in the language modeling process [Joshi et al., 2019; Lample and Conneau, 2019].

In this study, we use MLM as our training target since MLM shows better results when the goal

[2]https://www.ugallery.com/

is to learn a good representation of the input documents while CLM is more advantageous for text generation tasks such as machine translation and chatbots [Eo et al., 2021]. We also choose different base models with and without the NSP loss to demonstrate the usefulness of this task.

5.2.3 Natural Language Processing in Art

NLP has already been proven to be useful and is widely used in many domains, including machine translation [Stahlberg, 2020], finance [Chen and Robert, 2022], music [Oramas et al., 2018], etc. However, in the visual art industry, previous research mostly focuses on the visual aspect of the artworks and ignore the rich information in the texts associated with the artists and the artworks.

There are several early attempts in applying NLP technologies on artworks, such as Kim et al. [2022] and Fosset et al. [2022]. Nevertheless, both works only use the language models to generate embeddings from the texts and combine them with the visual elements, without training a task-specific model. To the best of our knowledge, there is no previous work which focuses on discovering artist relationships based on their biographies.

5.3 Problem Formulation

We formulate this artist connection discovery problem as a sentence pair binary classification problem. Our goal is to determine if two artists are connected given their respective biographies.

Suppose that we have two artists A_i, A_j and their biographies are denoted by B_i, B_j. A team of art professionals can determine if they are connected based on different aspects, including background, themes, style, techniques, etc. We use $Y_{i,j}$ to denote this ground-truth, with

$$Y_{i,j} = \begin{cases} 1 & \text{if } A_i \text{ and } A_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}.$$

However, there are thousands of artists in our artist database and it is impossible to determine all relationships through manual annotation. Hence, in this study, we are interested in building the connections among them automatically based on their biographies, which comprise the information of the artists in different dimensions. We can describe this process as

$$p_{i,j} = f(B_i, B_j, \theta)$$

where $p_{i,j}$ denotes our predicted probability of a connection between A_i and A_j. f is our prediction model and θ denotes the trainable parameters.

Given f, our goal is to find θ which minimizes the cross-entropy loss [Good, 1992] defined as

$$-\sum_{i,j} Y_{i,j} \cdot \log p_{i,j} + (1 - Y_{i,j}) \cdot \log(1 - p_{i,j}).$$

5.4 Identifying Artist Connection with ArtLM

There are three main components in our artist connection discovery method:

1. (unsupervised) pre-train a LM with generic English corpus
2. (unsupervised) continue to pre-train the LM with the same goal, but with art-related texts to get a LM with domain-specific knowledge in art (ArtLM)
3. (supervised) fine-tune the ArtLM with labeled artist biography pairs

The method is illustrated in Figure 5.1 and we introduce each component in detail in the following subsections.

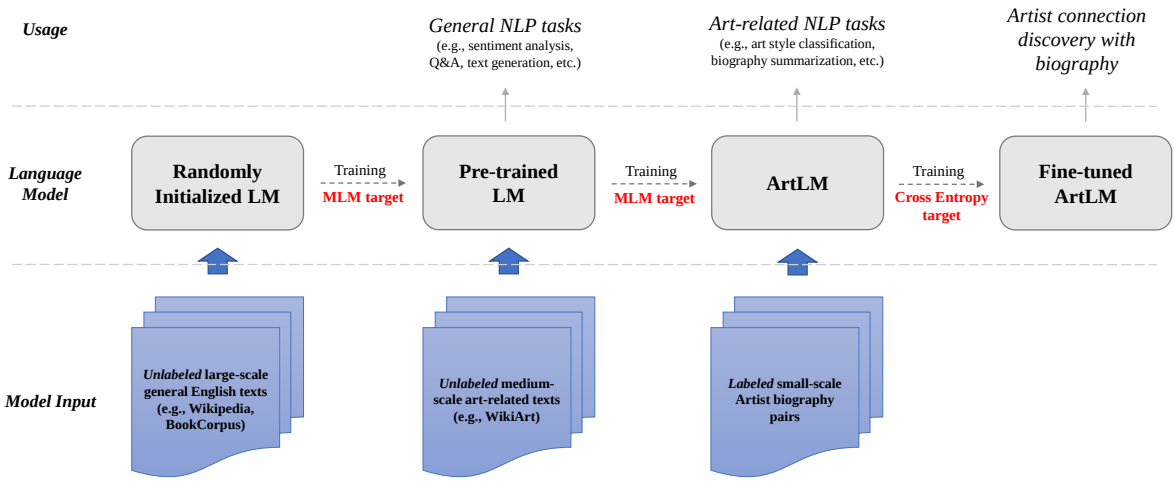


Figure 5.1: Overview of our artist connection discovery method, including two unsupervised pre-training phases and one supervised fine-tuning phase.

5.4.1 Pre-trained LM

At first, we want to learn the general characteristics of a language as our base model, which we refer as the pre-trained base language model.

In order to generalize, this model is usually trained on a very large corpus. For example, the BERT model is trained on BookCorpus [McEnery et al., 2006] and English Wikipedia which have more than 3.3 billion words in total. Hence, this process is computationally expensive and time consuming. In our experiments, we use publicly available pre-trained English LMs instead of training from scratch by ourselves.

To prove that our approach is generic and not model-dependant, we adopt three different pre-trained LMs of different sizes and training targets as our base models, including DistilBERT, BERT and RoBERTa. Their details are shown in Table 5.2.

Table 5.2: Details of the pre-trained models used in our experiments

model	size	# params. (millions)	training target	training dataset size
DistilBERT-based-uncased [Sanh et al., 2019]	small	66	MLM, NSP	16GB
BERT-base-uncased [Devlin et al., 2018]	large	110	MLM, NSP	16GB
RoBERTa-base [Liu et al., 2019]	large	125	MLM	160GB

5.4.2 ArtLM

Howard and Ruder [2018] show that further pre-training a LM with domain-specific data helps improve the performance. For example, Araci [2019] and Chen [2021] further pre-train BERT model with financial corpus, Wada et al. [2020] applies this methodology on medical texts, etc.

Following the same idea, we reuse the weights in the pre-trained base models and continue to train these models with MLM target (Section 5.2.2) with a large amount of art-related texts. Concretely, it means that we randomly replace around 15% of the words in the sentences with a special character `[MASK]` and we ask the model to predict this word. We minimize the sum of cross-entropy loss of all the masked characters using stochastic gradient descendant (SGD) [Robbins and Monro, 1951]. We use ArtLM (Art Language Model) to denote this derived model.

In our case, we use the texts from WikiArt for this propose. We introduce this dataset in detail in Section 5.5.1.

5.4.3 ArtLM Fine-tuning

Starting from a language model, we can continue to fine-tune it with labeled data to get a model specializing in our task: artist biography pair classification. As recommended by Devlin et al. [2018], we construct our labeled dataset in the format of `[CLS]`Trunc(B_i)`[SEP]`Trunc(B_j), where `[CLS]` denotes the special character for class labels (0 or 1 in our case) and `[SEP]` represents the special character separating two biographies. *Trunc* is the truncating function which removes the tailing words if the number of words in a biography exceeds the maximum limit. This is to ensure that two biographies are homogeneous with similar characteristics.

In this step, we add another fully-connected layer and a softmax function[3] at the end of the model. This layer takes the vector representing `[CLS]` as input and outputs $p_{i,j}$, the predicted probability of a connection between artists A_i and A_j. We train this fully-connected layer and all other layers in the model jointly with the target function shown in Equation 5.3. Our final prediction can therefore be written as

$$\widehat{Y_{i,j}} = \mathbb{1}_{p_{i,j}>0.5}$$

where $\mathbb{1}$ denotes the indicator function.

[3] $\text{softmax}(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

5.5 Experiments

5.5.1 Datasets

In this study, we use two different datasets to pre-train and fine-tune our ArtLM. We use a large dataset scraped from WikiArt to add art-related knowledge to the pre-trained base model and we use a smaller manually annotated artist biography pair dataset to fine-tune this model. We introduce the details of these two datasets in the following subsections.

WikiArt Data

WikiArt is an online art encyclopedia that contains art-related text data. We only use artists' biographies in this database, which is the most relevant to our task.

There are 3,110 articles scrapped from the WikiArt database, accounting for more than 1.3 million tokens in total.

Labeled Artist Biography Pair Data

To build the annotated biography pair data, we select about 850 contemporary artists spanning different artistic movements, techniques and expressing a variety of subjects and themes through their artworks. This leads to around 360,000 possible pairs in total, which is too large to annotate manually. To sample a smaller dataset for manual labeling, we first use the baseline model to get a set of potential connections. We then randomly sample a subset with around 1,500 pairs. We ask art curators from a renowned university to label these pairs as 0 if no relationship between the artists can be drawn based on the biographies, and label as 1 otherwise.

An example of the labeled data is already shown in Table 5.1. We provide some statistics of the dataset in Table 5.3.

The average number of words of all the biographies is 191, while the maximum number of words can reach 2,000. If there are more than 255 words in a sentence, we truncate the tailing words as mentioned in Section 5.4.3 to ensure that the sample fits in the model.

Table 5.3: Some statistics of the labeled artist biography pair data

label	meaning	count	proportion	word count
0	not connected	767	52.6%	315k
1	connected	691	47.4%	242k
	total	1,458	100.0%	557k

5.5.2 Baseline Models

In order to prove the effectiveness of our ArtLM approach, we compare its performance with other baseline models. We include the following baseline models.

Random Guess We randomly assign the label 0 or 1 to each biography pair, with probability of P_0 and P_1 respectively. We test two scenarios: $P_0 = 1$ (all pairs are predicted as 0) and $P_0 = 0.5$ (the proportion of the label 0 in the ground-truth). We use the expression `Random Guess -` P_0 to denote the random guess prediction results with different probabilities.

Static embedding from Word2Vec We use Word2Vec embeddings to create associations between artists biographies and themes based on keywords from a pre-defined list of themes. We consider that two artists are linked if they have at least one tag in common.

Contextualized embedding from BERT Instead of using static embeddings from Word2Vec, we use contextualized embeddings generated from the last layer of BERT.

Fine-tuned base models In addition to the above baseline models, we also report the results from fine-tuning base models without continuous pre-training on WikiArt data (we skip the step 2 mentioned in Section 5.4). This intermediate result helps demonstrate the performance gain from pre-training a ArtLM instead of directly using the pre-trained base models.

In the following sections, we use the prefix `FT-Base` to denote the results from a fine-tuned base model and the prefix `FT-Art` to denote the results from a fine-tuned ArtLM. For example, two BERT variants are respectively denoted by `FT-Base-BERT` and `FT-Art-BERT`. We report the performance from all three base models mentioned in Table 5.2.

5.5.3 Experiment Setup

Experiment Method

For the ArtLM pre-training process (Section 5.4.2), we randomly mask 15% of the words of all the sentences included in the WikiArt articles and continue to pre-train from the last checkpoint of one of the pre-trained base models. For all base models, we set the learning rate to 2e-5 and iterate on the dataset for 3 epochs.

In contrary, for the ArtLM fine-tuning process (Section 5.4.3), we split our data into 5 folds randomly for a cross-validation, since the size of our manually annotated data is relatively small. It means that we split the data equally into 5 parts, and we fine-tune 5 different models with one part being the test set and the remaining 4 parts being the training set and validation set. This helps avoid the bias caused by data split. In this fine-tuning process, we set the learning rate to 2e-5 and iterate on the dataset for 20 epochs. We select the best epoch according to the F1 score on the validation set and perform inference on the test set.

We use the mean and the standard deviation of the accuracy and F1 score of 5 folds to evaluate the model. Given a confusion matrix $\begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix}$[4], the accuracy is defined as

$$Acc = \frac{tp + tn}{tp + tn + fp + fn}$$

and the F1 score is defined as

$$F_1 = \frac{2tp}{2tp + fp + fn}.$$

Hardware

The experiments are conducted on a Nvidia Tesla V100 GPU with 16 GB memory and 900 GB/s bandwidth. The machine also has 8 cores of Intel Xeon E5-2686 CPU for other computationally inexpensive jobs. An ArtLM pre-training process usually takes 10-20 minutes to finish depending on the base model size, while an ArtLM fine-tuning process generally takes 30-60 minutes for all 5 folds.

5.5.4 Experiment Results

The detailed results of different ArtLM variants and the baseline models mentioned in Section 5.5.2 are shown in Table 5.4.

Table 5.4: Experiment results of the ArtLM variants and other baseline models. If applicable, the numbers of accuracy and F1 score in this table are shown in the format: average of 5 folds ± standard deviation of 5 folds. The fine-tuning time is calculated as the mean on 5 folds. We note that no training is involved in the embedding baseline methods, there is therefore no cross-validation and the accuracy and F1 score are calculated based on the whole dataset.

model	accuracy	F1	fine-tuning time (min. / fold)	pre-training time (min.)
Random Guess - 1	0.500	0.000	-	-
Random Guess - 0.5	0.500	0.500	-	-
Word2Vec embedding	0.539	0.196	-	-
BERT embedding	0.581	0.320	-	-
FT-Base-DistilBERT	0.838 ± 0.016	0.826 ± 0.019	6	-
FT-Base-BERT	0.844 ± 0.022	0.832 ± 0.020	12	-
FT-Base-RoBERTa	0.844 ± 0.021	0.824 ± 0.021	12	-
FT-Art-DistilBERT	0.846 ± 0.020	0.826 ± 0.019	6	10
FT-Art-BERT	**0.856 ± 0.023**	**0.840 ± 0.021**	12	18
FT-Art-RoBERTa	0.854 ± 0.024	0.834 ± 0.826	12	21

First, we note that all fine-tuned models outperform random guess and the baseline models based on embeddings. This phenomenon demonstrates that the task-specific information added during the fine-tuning process helps the model better predict the connections between the artists.

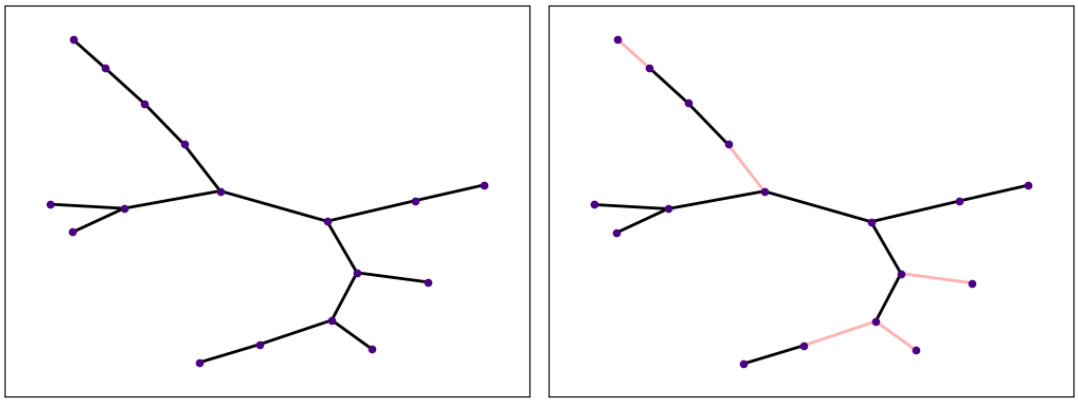
[4] tp, fn, fp and tn denote true positive, false negative, false positive and true negative respectively.

Secondly, comparing the results from the fine-tuned base models and the fine-tuned ArtLMs, the ArtLM outperforms the base model on all three cases. This proves the effectiveness of adding domain-specific information through continuing the pre-training process on a large amount of art-related texts. It also demonstrates that our ArtLM approach is effective on all base models and is robust to the model choices.

Finally, we observe that among the three ArtLM variants, `FT-Art-BERT` has the best performance, surpassing the both smaller `FT-Art-DistilBERT` and the larger `FT-Art-RoBERTa`, although RoBERTa reports a superior performance than BERT on most of the NLP tasks [Liu et al., 2019]. A possible cause is that the RoBERTa model removes the NSP task in the pre-training process, as mentioned in Section 5.2.2 and in Table 5.2. However, our artist pair classification usage is similar to the idea behind the NSP task, which leads to the performance degradation after its removal. Another possible explanation is that the RoBERTa is trained on a much larger corpus than BERT, it may require more art-related data in this second pre-training process in order to be significant.

5.5.5 Artist Network

With the prediction results in Section 5.5.4, we can build artist networks which can be used to visualize their relationships or to recommend related artists based on collector's interests in further studies. We visualize some artist networks in Figure 5.2. In each sub-figure, a node denotes an artist. An edge between two nodes signifies that the two authors are connected, either from the ground-truth or from the prediction. We first construct an artist network using the labeled ground-truth data. We then zoom into one of the sub-graphs (Figure 5.2a) for a qualitative analysis.



(a) a sub-network of the ground-truth network (b) `FT-Art-BERT` prediction of (a)

Figure 5.2: Artist network. Figure (a) is a sub-network of the artist network built from the ground-truth. We perform analysis on this sub-network. Figure (b) is the prediction of (a) made from our `FT-Art-BERT` model.

Figure 5.2b is the prediction of Figure 5.2a made from our `FT-Art-BERT` model. In this figure, black edge denote the correctly predicted edges, red edges denote the edges that are present in the ground-truth but identified as not connected by `FT-Art-BERT`. Blue edges represent the relations that are present in the predictions but not in the ground-truth. We can see that the ArtLM helps rebuild the framework of ground-truth network, although there are mismatches on

some biography pairs[5], especially in the tails of the graph.

A potential usage of this artist network is artwork recommendation. For example, we can recommend the frames of the connected artists if one collector likes an artist.

5.6 Conclusion

In this work, we classify if two contemporary artists are closely connected through their biographies. We introduce a transfer learning based approach: Art Language Model (ArtLM), which comprises an unsupervised pre-training step and a supervised fine-tuning step. Through extensive experiments, we demonstrate that ArtLM outperforms other benchmark models. We also prove that both pre-training and fine-tuning are essential to a better performance, and that this procedure is generic and robust to the choices of the base models.

For future study, we seek to integrate the ArtLM's outputs into a recommender system [Resnick and Varian, 1997] to further enhance the artwork discovery process. We can also combine the visual and textual aspects of the artworks in a more complex system.

We also plan to enrich our manually annotated biography pair dataset with the appreciations from different aspects. It means that instead of classifying a pair of artists in a binary manner, we label their relationships from different dimensions, for example, style, background, influence, etc. This problem will therefore become a multi-class classification problem and allow us to better understand the relationship between two artists.

[5]The Graph Edit Distance (GED) [Gao et al., 2010] between two graphs is 7.

Chapter 6

Conclusion and Outlook

This thesis aims to tackle some prediction problems in finance with deep learning methods.

In the first study, we built a pipeline for univariate stock movement prediction from financial news. It includes data labeling, contextualized embedding generation, model training and performance evaluation. The experiment results demonstrated the effectiveness of contextualized embeddings instead of static embeddings. We also compared the performance of our model with other models widely used in the industry to prove its power in real-life trading.

We then extended our study to a multivariate approach by considering news and stock relationships. In the second study, we introduced a graph-based model that considers jointly financial news, multiple stock relationships and temporal patterns of the news for stock movement prediction. The results of this study confirmed the necessity of stock relationship modeling in this stock movement prediction task. It was also proved that disjointed relationship information from multiple sources could further boost the prediction power.

In the third study, we forecasted the short-term realized volatility from limit order book in a multivariate approach by designing a graph transformer network. Unlike the previous study, we combined both cross-sectional and temporal relationships into the same graph, although both studies were based on graph neural networks. In addition, we adopted a spatial message passing model to compute on this large graph. Experiment results confirmed the effectiveness of this model design through performance gain compared with baseline models, especially on less liquid stocks where there was less available information.

In the fourth study, we applied some NLP methods studied in the first two parts on contemporary arts to discover the connections among the artists. For this task, we presented a generic NLP framework, in which we first continued to pre-train an existing general English language model with a large amount of unlabeled art-related texts. We then fine-tuned this new pre-trained model with labeled biography pairs. The results of this study demonstrated the effectiveness of this fine-tuning approach as well as the improvement brought by the extra pre-training step.

There are several research directions to pursue following these studies. For the stock return prediction task, our studies focused on using the headlines of the financial news. We can further extend to predicting from long texts such as financial reports or earnings call transcripts. We also notice that our second study is limited to the stocks with sufficient news to reduce the

number of missing features in the GCN, which can cause inefficient training. It is interesting to explore feature filling techniques [Taguchi et al., 2021] to apply this method to a larger universe of stocks.

For the realized volatility prediction task, we mainly focused on modeling the feature interactions among prediction buckets while using a fixed set of features constructed from limit order book. It is also interesting to understand the role each feature plays and select the most impactful features to further improve training efficiency.

I sincerely hope that the studies presented in this thesis will be helpful to both academic researchers wanting to know more about the application of deep learning methods in finance, and to the practitioners in the asset management aiming at improving their prediction algorithms.

Bibliography

[Adam et al., 2016] Adam, K., Marcet, A., and Nicolini, J. P. (2016). Stock market volatility and learning. *The Journal of Finance*, 71(1):33–82.

[Adebiyi et al., 2012] Adebiyi, A. A., Ayo, C. K., Adebiyi, M. O., and Otokiti, S. O. (2012). Stock price prediction using neural network with hybridized market indicators. *Journal of Emerging Trends in Computing and Information Sciences*, 3(1):1–9.

[Andersen and Bollerslev, 1998] Andersen, T. G. and Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International economic review*, pages 885–905.

[Andersen et al., 2005] Andersen, T. G., Bollerslev, T., Christoffersen, P., and Diebold, F. X. (2005). Volatility forecasting.

[Araci, 2019] Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

[Arık and Pfister, 2020] Arık, S. O. and Pfister, T. (2020). Tabnet: Attentive interpretable tabular learning. *arXiv*.

[Arık and Pfister, 2021] Arık, S. O. and Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, pages 6679–6687.

[Ariyo et al., 2014] Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. (2014). Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bai and Pukthuanthong, 2020] Bai, Y. and Pukthuanthong, K. (2020). Machine learning classification methods and portfolio allocation: An examination of market efficiency. *Available at SSRN 3665051*.

[Barndorff-Nielsen et al., 2009] Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., and Shephard, N. (2009). Realized kernels in practice: Trades and quotes.

[Berg et al., 2017] Berg, R. v. d., Kipf, T. N., and Welling, M. (2017). Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

[Bhandari, 1988] Bhandari, L. C. (1988). Debt/equity ratio and expected common stock returns: Empirical evidence. *The journal of finance*, 43(2):507–528.

[Bissoondoyal-Bheenick et al., 2019] Bissoondoyal-Bheenick, E., Brooks, R., and Do, H. X. (2019). Asymmetric relationship between order imbalance and realized volatility: Evidence from the australian market. *International Review of Economics & Finance*, 62:309–320.

[Black et al., 1972] Black, F., Jensen, M. C., Scholes, M., et al. (1972). The capital asset pricing model: Some empirical tests.

[Bollerslev et al., 2019] Bollerslev, T., Meddahi, N., and Nyawa, S. (2019). High-dimensional multivariate realized volatility estimation. *Journal of Econometrics*, 212(1):116–136.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[Brailsford and Faff, 1996] Brailsford, T. J. and Faff, R. W. (1996). An evaluation of volatility forecasting techniques. *Journal of Banking & Finance*, 20(3):419–438.

[Bruna et al., 2013] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[Bucci, 2020] Bucci, A. (2020). Cholesky–ann models for predicting multivariate realized volatility. *Journal of Forecasting*, 39(6):865–876.

[Campbell et al., 1993] Campbell, J. Y., Grossman, S. J., and Wang, J. (1993). Trading volume and serial correlation in stock returns. *The Quarterly Journal of Economics*, 108(4):905–939.

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[Chan et al., 1993] Chan, L. K., Hamao, Y., and Lakonishok, J. (1993). Can fundamentals predict japanese stock returns? *Financial Analysts Journal*, 49(4):63–69.

[Chen et al., 2015] Chen, K., Zhou, Y., and Dai, F. (2015). A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE.

[Chen, 2021] Chen, Q. (2021). Stock movement prediction with financial news using contextualized embedding from bert. *arXiv preprint arXiv:2107.08721*.

[Chen and Robert, 2021] Chen, Q. and Robert, C.-Y. (2021). Graph-based learning for stock movement prediction with textual and relational data. *arXiv preprint arXiv:2107.10941*.

[Chen and Robert, 2022] Chen, Q. and Robert, C.-Y. (2022). Graph-based learning for stock movement prediction with textual and relational data. *The Journal of Financial Data Science*.

[Chen and Ge, 2019] Chen, S. and Ge, L. (2019). Exploring the attention mechanism in lstm-based hong kong stock price movement prediction. *Quantitative Finance*, 19(9):1507–1515.

[Chen et al., 2007] Chen, T.-L., Cheng, C.-H., and Teoh, H. J. (2007). Fuzzy time-series based on fibonacci sequence for stock price forecasting. *Physica A: Statistical Mechanics and its Applications*, 380:377–390.

[Chen et al., 2018] Chen, Y., Wei, Z., and Huang, X. (2018). Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1655–1658.

[Cheng et al., 2020] Cheng, D., Yang, F., Wang, X., Zhang, Y., and Zhang, L. (2020). Knowledge graph-based event embedding framework for financial quantitative investments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2221–2230.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[Cleeremans et al., 1989] Cleeremans, A., Servan-Schreiber, D., and McClelland, J. L. (1989). Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381.

[Taguchi et al., 2021] Taguchi, c., Liu, c., and Murata, c. (2021). graph convolutional networks for graphs containing missing features. *Future Generation Computer Systems*, 117:155–168.

[Cooper et al., 2016] Cooper, M. J., Gulen, H., and Rau, P. R. (2016). Performance for pay? the relation between ceo incentive compensation and future stock price performance. *The Relation Between CEO Incentive Compensation and Future Stock Price Performance (November 1, 2016)*.

[Coqueret, 2020] Coqueret, G. (2020). Stock-specific sentiment and return predictability. *Quantitative Finance*, 20(9):1531–1551.

[Corsi, 2009] Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196.

[Crnic, 2011] Crnic, J. (2011). Introduction to modern information retrieval. *Library Management*.

[De Boer et al., 2005] De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67.

[Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

[Del Corro and Hoffart, 2020] Del Corro, L. and Hoffart, J. (2020). Unsupervised extraction of market moving events with neural attention. *arXiv preprint arXiv:2001.09466*.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Ding et al., 2014] Ding, X., Zhang, Y., Liu, T., and Duan, J. (2014). Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425.

[Ding et al., 2015] Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[Domowitz et al., 2001] Domowitz, I., Glen, J., and Madhavan, A. (2001). Liquidity, volatility and equity trading costs across countries and over time. *International Finance*, 4(2):221–255.

[Donaldson and Storeygard, 2016] Donaldson, D. and Storeygard, A. (2016). The view from above: Applications of satellite data in economics. *Journal of Economic Perspectives*, 30(4):171–98.

[Duvenaud et al., 2015] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.

[Elgammal et al., 2018] Elgammal, A., Liu, B., Kim, D., Elhoseiny, M., and Mazzone, M. (2018). The shape of art history in the eyes of the machine. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Eo et al., 2021] Eo, S., Park, C., Moon, H., Seo, J., and Lim, H. (2021). Comparative analysis of current approaches to quality estimation for neural machine translation. *Applied Sciences*, 11(14):6584.

[Fama, 1965] Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, 38(1):34–105.

[Fama and French, 2004] Fama, E. F. and French, K. R. (2004). The capital asset pricing model: Theory and evidence. *Journal of economic perspectives*, 18(3):25–46.

[Fedyk, 2018] Fedyk, A. (2018). Front page news: The effect of news positioning on financial markets. Technical report, working paper.

[Fosset et al., 2022] Fosset, A., El-Mennaoui, M., Rebei, A., Calligaro, P., Di Maria, E. F., Nguyen-Ban, H., Rea, F., Vallade, M.-C., Vitullo, E., Zhang, C., et al. (2022). Docent: A content-based recommendation system to discover contemporary art. *arXiv preprint arXiv:2207.05648*.

[Fout et al., 2017] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30.

[Freedman et al., 2007] Freedman, D., Pisani, R., and Purves, R. (2007). Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

[Gao et al., 2010] Gao, X., Xiao, B., Tao, D., and Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129.

[Garcia and Liu, 1999] Garcia, V. F. and Liu, L. (1999). Macroeconomic determinants of stock market development. *Journal of applied Economics*, 2(1):29–59.

[Gatheral and Oomen, 2010] Gatheral, J. and Oomen, R. C. (2010). Zero-intelligence realized variance estimation. *Finance and Stochastics*, 14(2):249–283.

[Gilmer et al., 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

[Gini, 1921] Gini, C. (1921). Measurement of inequality of incomes. *The economic journal*, 31(121):124–126.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

[Good, 1992] Good, I. J. (1992). Rational decisions. In *Breakthroughs in statistics*, pages 365–377. Springer.

[Guo et al., 2018] Guo, L., Peng, L., Tao, Y., and Tu, J. (2018). News co-occurrence, attention spillover, and return predictability. *Attention Spillover, and Return Predictability (November 18, 2018)*.

[Hamilton et al., 2017] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

[Hassan and Malik, 2007] Hassan, S. A. and Malik, F. (2007). Multivariate garch modeling of sector volatility transmission. *The quarterly review of economics and finance*, 47(3):470–480.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hecht-Nielsen, 1992] Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier.

[Ho and Xie, 1998] Ho, S. L. and Xie, M. (1998). The use of arima models for reliability forecasting and analysis. *Computers & industrial engineering*, 35(1-2):213–216.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hoseinzade and Haratizadeh, 2019] Hoseinzade, E. and Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285.

[Hou, 2007] Hou, K. (2007). Industry information diffusion and the lead-lag effect in stock returns. *The Review of Financial Studies*, 20(4):1113–1138.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

[Hu et al., 2018] Hu, Z., Liu, W., Bian, J., Liu, X., and Liu, T.-Y. (2018). Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 261–269. ACM.

[Iyyer et al., 2015] Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

[Jiang et al., 2018] Jiang, Z.-Q., Wang, G.-J., Canabarro, A., Podobnik, B., Xie, C., Stanley, H. E., and Zhou, W.-X. (2018). Short term prediction of extreme returns based on the recurrence interval analysis. *Quantitative Finance*, 18(3):353–370.

[Jones, 1972] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

[Jones, 2004] Jones, K. S. (2004). Idf term weighting and ir research lessons. *Journal of documentation*.

[Joshi et al., 2019] Joshi, M., Levy, O., Weld, D. S., and Zettlemoyer, L. (2019). Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

[Jurafsky and Martin, 2021] Jurafsky, D. and Martin, J. H. (2021). *Speech & language processing*. Pearson Education India.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

[Ke et al., 2019] Ke, Z. T., Kelly, B. T., and Xiu, D. (2019). Predicting returns with text data. Technical report, National Bureau of Economic Research.

[Kercheval and Zhang, 2015] Kercheval, A. N. and Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8):1315–1329.

[Kim et al., 2022] Kim, D., Elgammal, A., and Mazzone, M. (2022). Formal analysis of art: Proxy learning of visual concepts from style through language models. *arXiv preprint arXiv:2201.01819*.

[Kim et al., 2018] Kim, D. S., Liu, B., Elgammal, A., and Mazzone, M. (2018). Finding principal semantics of style in art. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 156–163. IEEE.

[Kim et al., 2019] 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*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

[Klößner and Wagner, 2014] Klößner, S. and Wagner, S. (2014). Exploring all var orderings for calculating spillovers? yes, we can!—a note on diebold and yilmaz (2009). *Journal of Applied Econometrics*, 29(1):172–179.

[Kohara et al., 1997] Kohara, K., Ishikawa, T., Fukuhara, Y., and Nakamura, Y. (1997). Stock price prediction using prior knowledge and neural networks. *Intelligent Systems in Accounting, Finance & Management*, 6(1):11–22.

[Kraft and Kraft, 1977] Kraft, J. and Kraft, A. (1977). Determinants of common stock prices: a time series analysis. *The journal of finance*, 32(2):417–425.

[Kräussl et al., 2016] Kräussl, R., Lehnert, T., and Martelin, N. (2016). Is there a bubble in the art market? *Journal of Empirical Finance*, 35:99–109.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kroujiline et al., 2016] Kroujiline, D., Gusev, M., Ushanov, D., Sharov, S. V., and Govorkov, B. (2016). Forecasting stock market returns over multiple time horizons. *Quantitative Finance*, 16(11):1695–1712.

[Kudo and Richardson, 2018] Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

[Kwan et al., 2005] Kwan, C., Li, W., and Ng, K. (2005). A multivariate threshold garch model with time-varying correlations. *Econometric reviews*, 24.

[Lample and Conneau, 2019] Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

[Li et al., 2020a] Li, J., Li, G., Zhu, X., and Yao, Y. (2020a). Identifying the influential factors of commodity futures prices through a new text mining approach. *Quantitative Finance*, 20(12):1967–1981.

[Li et al., 2020b] Li, W., Bao, R., Harimoto, K., Chen, D., Xu, J., and Su, Q. (2020b). Modeling the stock relation with graph network for overnight stock movement prediction. In *no. CONF*, pages 4541–4547.

[Li et al., 2021] Li, W., Bao, R., Harimoto, K., Chen, D., Xu, J., and Su, Q. (2021). Modeling the stock relation with graph network for overnight stock movement prediction. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4541–4547.

[Li et al., 2017] Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

[Liu et al., 2007] Liu, J., Nissim, D., and Thomas, J. (2007). Is cash flow king in valuations? *Financial Analysts Journal*, 63(2):56–68.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Lof, 2012] Lof, M. (2012). Heterogeneity in stock prices: A star model with multivariate transition function. *Journal of Economic Dynamics and Control*, 36(12):1845–1854.

[Luhn, 1957] Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

[Luo et al., 2017] Luo, L., You, S., Xu, Y., and Peng, H. (2017). Improving the integration of piece wise linear representation and weighted support vector machine for stock trading signal prediction. *Applied Soft Computing*, 56:199–216.

[Luss and d'Aspremont, 2015] Luss, R. and d'Aspremont, A. (2015). Predicting abnormal returns from news using text classification. *Quantitative Finance*, 15(6):999–1012.

[Ma et al., 2019] Ma, F., Liao, Y., Zhang, Y., and Cao, Y. (2019). Harnessing jump component for crude oil volatility forecasting in the presence of extreme shocks. *Journal of Empirical Finance*, 52:40–55.

[Mäkinen et al., 2019] Mäkinen, Y., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2019). Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data. *Quantitative Finance*, 19(12):2033–2050.

[Malec, 2016] Malec, P. (2016). A semiparametric intraday garch model.

[Malkiel, 1989] Malkiel, B. G. (1989). Efficient market hypothesis. In *Finance*, pages 127–134. Springer.

[Maron, 1961] Maron, M. E. (1961). Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417.

[Matthews, 1975] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

[McEnery et al., 2006] McEnery, T., Xiao, R., and Tono, Y. (2006). *Corpus-based language studies: An advanced resource book*. Taylor & Francis.

[Mehtab and Sen, 2020] Mehtab, S. and Sen, J. (2020). Stock price prediction using convolutional neural networks on a multivariate timeseries. *arXiv preprint arXiv:2001.09769*.

[Messina et al., 2018] Messina, P., Dominguez, V., Parra, D., Trattner, C., and Soto, A. (2018). Exploring content-based artwork recommendation with metadata and visual features. *UMUAI*.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mills and Mills, 1990] Mills, T. C. and Mills, T. C. (1990). *Time series techniques for economists*. Cambridge University Press.

[Mondal et al., 2014] Mondal, P., Shit, L., and Goswami, S. (2014). Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13.

[Mosteller and Wallace, 1963] Mosteller, F. and Wallace, D. L. (1963). Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed federalist papers. *Journal of the American Statistical Association*, 58(302):275–309.

[Myers et al., 2014] Myers, S. A., Sharma, A., Gupta, P., and Lin, J. (2014). Information network or social network? the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 493–498.

[Narayan et al., 2014] Narayan, P. K., Narayan, S., and Thuraisamy, K. S. (2014). Can institutions and macroeconomic factors predict stock returns in emerging markets? *Emerging Markets Review*, 19:77–95.

[Nguyen and Shirai, 2015] Nguyen, T. H. and Shirai, K. (2015). Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1354–1364.

[Noehrer et al., 2021] Noehrer, L., Gilmore, A., Jay, C., and Yehudi, Y. (2021). The impact of covid-19 on digital data practices in museums and art galleries in the uk and the us. *Humanities and Social Sciences Communications*, 8(1):1–10.

[Nonejad, 2021] Nonejad, N. (2021). Bayesian model averaging and the conditional volatility process: an application to predicting aggregate equity returns by conditioning on economic variables. *Quantitative Finance*, pages 1–25.

[Obaid and Pukthuanthong, 2021] Obaid, K. and Pukthuanthong, K. (2021). A picture is worth a thousand words: Measuring investor sentiment by combining machine learning and photos from news. *Journal of Financial Economics*.

[Oliveira et al., 2013] Oliveira, N., Cortez, P., and Areal, N. (2013). Some experiments on modeling stock market behavior using investor sentiment analysis and posting volume from twitter. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*, page 31. ACM.

[Oramas et al., 2018] Oramas, S., Espinosa-Anke, L., Gómez, F., and Serra, X. (2018). Natural language processing for music knowledge discovery. *Journal of New Music Research*, 47(4):365–382.

[Ozlen, 2014] Ozlen, S. (2014). The effect of company fundamentals on stock values. *European Researcher*, 71(3-2):595–602.

[Pagolu et al., 2016] Pagolu, V. S., Reddy, K. N., Panda, G., and Majhi, B. (2016). Sentiment analysis of twitter data for predicting stock market movements. In *2016 international conference on signal processing, communication, power and embedded system (SCOPES)*, pages 1345–1350. IEEE.

[Pandit et al., 2011] Pandit, S., Wasley, C. E., and Zach, T. (2011). Information externalities along the supply chain: The economic determinants of suppliers' stock price reaction to their customers' earnings announcements. *Contemporary Accounting Research*, 28(4):1304–1343.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Patell, 1976] Patell, J. M. (1976). Corporate forecasts of earnings per share and stock price behavior: Empirical test. *Journal of accounting research*, pages 246–276.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018a] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. In *NAACL*.

[Peters et al., 2018b] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Rahimikia and Poon, 2020a] Rahimikia, E. and Poon, S.-H. (2020a). Big data approach to realised volatility forecasting using har model augmented with limit order book and news. *Available at SSRN 3684040*.

[Rahimikia and Poon, 2020b] Rahimikia, E. and Poon, S.-H. (2020b). Machine learning for realised volatility forecasting. *Available at SSRN 3707796*.

[Ramos-Pérez et al., 2021] Ramos-Pérez, E., Alonso-González, P. J., and Núñez-Velázquez, J. J. (2021). Multi-transformer: A new neural network-based architecture for forecasting s&p volatility. *Mathematics*, 9(15):1794.

[Rekabsaz et al., 2017] Rekabsaz, N., Lupu, M., Baklanov, A., Hanbury, A., Dür, A., and Anderson, L. (2017). Volatility prediction using financial disclosures sentiments with word embedding-based ir models. *arXiv preprint arXiv:1702.01978*.

[Resnick and Varian, 1997] Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Rossi et al., 2021] Rossi, E., Kenlay, H., Gorinova, M. I., Chamberlain, B. P., Dong, X., and Bronstein, M. (2021). On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *arXiv preprint arXiv:2111.12128*.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[Sanchez-Gonzalez et al., 2020] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Sawhney et al., 2020] Sawhney, R., Agarwal, S., Wadhwa, A., and Shah, R. (2020). Deep attentive learning for stock movement prediction from social media text and company correlations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8415–8426.

[Schumaker and Chen, 2009] Schumaker, R. P. and Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

[Sharpe, 1994] Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, 21(1):49–58.

[Shi et al., 2020] Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2020). Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

[Si et al., 2013] Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., and Deng, X. (2013). Exploiting topic based twitter sentiment for stock prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 24–29.

[Sirignano and Cont, 2019] Sirignano, J. and Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459.

[Sonsino and Shavit, 2014] Sonsino, D. and Shavit, T. (2014). Return prediction and stock selection from unidentified historical data. *Quantitative Finance*, 14(4):641–655.

[Stahlberg, 2020] Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418.

[Stevens et al., 1946] Stevens, S. S. et al. (1946). On the theory of scales of measurement.

[Tashiro et al., 2019] Tashiro, D., Matsushima, H., Izumi, K., and Sakaji, H. (2019). Encoding of high-frequency order information and prediction of short-term stock price by deep learning. *Quantitative Finance*, 19(9):1499–1506.

[Taylor, 1953] Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

[Turing, 2009] Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer.

[Vardharaj and Fabozzi, 2007] Vardharaj, R. and Fabozzi, F. J. (2007). Sector, style, region: Explaining stock allocation performance. *Financial Analysts Journal*, 63(3):59–70.

[Vargas et al., 2017] Vargas, M. R., De Lima, B. S., and Evsukoff, A. G. (2017). Deep learning for stock market prediction from financial news articles. In *2017 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, pages 60–65. IEEE.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[Wada et al., 2020] Wada, S., Takeda, T., Manabe, S., Konishi, S., Kamohara, J., and Matsumura, Y. (2020). Pre-training technique to localize medical bert and enhance biomedical bert. *arXiv preprint arXiv:2005.07202*.

[Wan et al., 2021] Wan, X., Yang, J., Marinov, S., Calliess, J.-P., Zohren, S., and Dong, X. (2021). Sentiment correlation in financial news networks and associated market movements. *Scientific reports*, 11(1):1–12.

[Wang et al., 2018] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

[Wang and Xu, 2004] Wang, F. and Xu, Y. (2004). What determines chinese stock returns? *Financial Analysts Journal*, 60(6):65–77.

[Wang et al., 2022] Wang, M., Rao, X., Chen, L., Shang, S., and Zhang, B. (2022). Ssar-gnn: Self-supervised artist recommendation with graph neural networks.

[Weiss et al., 2016] Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1):1–40.

[Wieting et al., 2015] Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

[Xiao, 2018] Xiao, H. (2018). bert-as-service. https://github.com/hanxiao/bert-as-service.

[Xie et al., 2013] Xie, B., Passonneau, R., Wu, L., and Creamer, G. G. (2013). Semantic frames to predict stock price movement. In *Proceedings of the 51st annual meeting of the association for computational linguistics*, pages 873–883.

[Xu and Cohen, 2018] Xu, Y. and Cohen, S. B. (2018). Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979.

[Yang et al., 2020a] Yang, S., Zhang, Z., Zhou, J., Wang, Y., Sun, W., Zhong, X., Fang, Y., Yu, Q., and Qi, Y. (2020a). Financial risk analysis for smes with graph-based supply chain mining. In *IJCAI*, pages 4661–4667.

[Yang et al., 2020b] Yang, Y., UY, M. C. S., and Huang, A. (2020b). Finbert: A pretrained language model for financial communications.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

[Ye et al., 2021] Ye, J., Zhao, J., Ye, K., and Xu, C. (2021). Multi-graph convolutional network for relationship-driven stock movement prediction. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6702–6709. IEEE.

[Zhang and Yan, 2018] Zhang, H. and Yan, C. (2018). Modelling fundamental analysis in portfolio selection. *Quantitative Finance*, 18(8):1315–1326.

[Zhang and Rosenbaum, 2020] Zhang, J. and Rosenbaum, M. (2020). Universal volatility formation : perspective from rough volatility and deep learning. Technical report, Ecole Polytechnique.

[Zhang, 2006] Zhang, L. (2006). Efficient estimation of stochastic volatility using noisy observations: A multi-scale approach. *Bernoulli*, 12(6):1019–1043.

[Zhou, 1996] Zhou, B. (1996). High-frequency data and volatility in foreign-exchange rates. *Journal of Business & Economic Statistics*, 14(1):45–52.

[Zhou et al., 2020] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

[Zhu et al., 2015] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : Solutions d'Apprentissage Profond à Quelques Problèmes de Prédiction en Finance

Mots clés : Prédiction de cours de l'actions, volatilité réalisée, traitement de langage naturel, apprentissage profond, apprentissage par transfert, réseau neurone de graphe

Résumé : Cette thèse est composée de deux parties connectées qui examinent respectivement deux problèmes de prédictions en finance: la prédiction du rendement des actions et la prédiction de la volatilité réalisée court terme des actions. Une dernière partie additionnelle est consacré à résoudre un problème de découverte des relations parmi les artistes contemporains, avec quelques méthodes dérivées de deux premières parties.
Dans la première partie, nous présentons une solution univariée et une solution multivariée pour le problème de la prédiction du rendement des actions avec les nouvelles financières. Nous introduisons d'abord une procédure de prédiction univariée qui prédit le rendement court terme d'une action après la publication d'une nouvelle qui l'associe (Chapitre 2). Dans cette procédure, nous appliquons d'abord une méthode d'apprentissage par transfert pour générer les embeddings contextualisés des mots dans le titre d'une nouvelle. Nous utilisons ensuite un réseau neurones récurrent pour faire la prédiction à partir des embeddings générés. Avec les expériences extensives, il est démontré que cette approche possède une meilleure performance par rapport aux autres modèles de référence. Nous étendons ensuite notre approche univariée à un modèle multivarié (Chapitre 3), dans lequel une nouvelle peut non seulement impacter la cour d'une action mais aussi toutes les autres actions associées par les relations venant de différentes sources. Nous modélisons cet effet de transmission à l'aide d'une structure innovative de réseau neurones convolutif en multi-graphe. Nous démontrons l'efficacité de ce modèle avec les expériences similaires à celles dans la première étude.
Dans la seconde partie de cette thèse, nous nous intéressons à la prédiction multivariée de la volatilité réalisée court terme des actions avec les carnets d'ordre (Chapitre 4). Pour cette étude, nous désignons un réseau neurones en graphe qui prend en compte les relations temporelles et cross-sectionnelles en même temps. Nous intégrons les opérateurs de transformeur en graphe dans le modèle pour améliorer la précision et l'efficacité de calcul sur ce grand graphe. Avec les expériences sur plus de 500 actions, nous prouvons que cette méthode multivariée en graphe a une meilleure performance que les modèles de référence univariés dans la littérature.
Dans la troisième partie, nous introduisons une solution pour découvrir les relations parmi les artistes contemporains avec leurs biographies (Chapitre 5). Dans ce but, nous créons un modèle dans lequel nous continuons d'abord à entraîner un modèle de langage général avec les textes de l'art non-labélisés. Nous affinons ensuite le nouveau modèle avec les paires de biographes labélisées. Il est démontré que notre approches atteint plus de 85% en taux de précision en identification de connexion entre deux artistes, et qu'il surpasse les modèles de référence dans les expériences.

Title : Deep Learning Solutions to Some Prediction Problems in Finance

Keywords : stock movement prediction, realized volatility, natural language processing, deep learning, transfer learning, graph neural network

Abstract : This thesis consists of two connected parts that examen respectively two prediction problems in finance, stock return prediction and short-term volatility prediction. It also has another additional part which examens a related issue in contemporary artists connection discovery with some methods derived from the two first parts.
In the first part, we present a univariate and a multivariate deep learning solution to the problem of stock return prediction with financial news. We first introduce a univariate prediction procedure that predicts the short-term return of a stock after the publication of news associated with this stock (Chapter 2). In this procedure, we first use a transfer learning based method to generate contextualized embeddings of the words in a news' headline, a recurrent neural network is then used to make predictions from the generated embeddings. Through extensive experiments, we show that this approach outperforms other baseline models. We then extend our univariate approach to a multivariate model (Chapter 3), in which a single news can not only impact one stock but also all other related stocks. Through an innovative multi-graph convolutional network structure, we can model the information transmission process from one stock to others based on stock relationships built from different sources. We demonstrate the effectiveness of this approach with a similar experiment setup as the first study.
In the second part of this thesis, we are interested in predicting short-term realized volatility from limit order book with a multivariate model (Chapter 4). To achieve this goal, we design a graph neural network containing both temporal and cross-sectional relations. Graph transformer operators are integrated into the model for better accuracy and computing efficiency on this large graph. Through experiments based on more than 500 stocks, we demonstrate that a graph-based multivariate approach has better predictive power than commonly used univariate baselines.
In the third part, we introduce a solution to discovering the relations among the contemporary artists through their biographies (Chapter 5). For this purpose, we design a general NLP framework, in which we first continue to pre-train an existing general language model with unlabeled art-related texts. We then fine-tune this new pre-trained model with labeled biography pairs. We demonstrate that our approach achieves more than 85% accuracy in identifying the connection between two artists and outperforms other baseline models in the experiments.

Institut Polytechnique de Paris
91120 Palaiseau, France