Transformer-Based Capsule Network For Stock Movements Prediction

Jintao Liu[1*], **Xikai Liu**[1*], **Hongfei Lin**[1†], **Bo Xu**[1,2], **Yuqi Ren**[1], **Yufeng Diao**[1,3], **Liang Yang**[1]

[1]Dalian University of Technology, Dalian, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China
[3]Inner Mongolia University for Nationalities, Tongliao, China

liujintao@mail.dlut.edu.cn, ws_lxk@mail.dlut.edu.cn, hflin@dlut.edu.cn, xubo@dlut.edu.cn
ryq13@mail.dlut.edu.cn, diaoyufeng@mail.dlut.edu.cn, liang@dlut.edu.cn

Abstract

Stock movements prediction is a highly challenging study for research and industry. Using social media for stock movements prediction is an effective but difficult task. However, the existing prediction methods which are based on social media usually do not consider the rich semantics and relation for a certain stock. It leads to difficulty in effective encoding. To solve this problem, we propose a CapTE (Capsule network based on Transformer Encoder) model which uses the Transformer Encoder to extract the deep semantic features of the social media and then captures the structural relationship of the texts through a capsule network. In this paper, we evaluate our method with different benchmarks, and the results demonstrate that our method improves the performance of stock movements prediction.

1 Introduction

According to the Efficient Market Hypothesis (EMH) [Fama *et al.*, 1969], stock price movements are thought to be related to the news. In natural language processing (NLP), public news and social media are two primary content resources for stock movements prediction. Moreover, social media such as Twitter is better in timeliness than news, so the condition that text from social media like Twitter is used to predict the stock movements draws numerous attention recently. On the other hand, tweets are able to reflect the investor's mentality to some extent. It is useful for the prediction of stock movements.

Many studies focus on stock movements prediction based on social media. Xu and Cohen [2018] introduce recurrent and continuous latent variables for better treatment of stochasticity, use neural variational inference to address the intractable posterior inference, and also provide a hybrid objective with temporal auxiliary to flexibly capture predictive dependencies. Wu et al. [2018] propose a novel Cross-model attention based on Hybrid Recurrent Neural Network (CH-RNN), which is inspired by the recent proposed DA-RNN model [Qin *et al.*, 2017].

In order to cross the chasm of prediction ability between machine and human, a deeper level of semantic information need to be explored in the text. Obviously, the above methods do not consider the rich semantic information and structural information of the social media, which lead to the model being unable to mine the deep semantics of text. For example, "Seems Travis is Still Running Uber! Surprised?", this expression of the sentence is colloquial which is very difficult to get the real mentality of the author. Especially, for one stock, there are more than one tweets on the same day. Existing methods can not effectively extract semantic features from these complex texts which are vital for this task.

Therefore, insights on the solutions to stock movements prediction can be drawn from the novel structure for capturing a deeper level of semantic information. In this paper, we propose a CapTE (Capsule network based on Transformer Encoder) model which uses Transformer encoder to solve this problem due to its multi-head attention structure. The model is able to capture more important semantic information from different texts. Tang et al. [2018] prove that the Transformer encoder achieves better results in semantic feature extraction than other models based on CNN and RNN. From our results, we also prove that the Transformer encoder is better than other models in the task of stock movements prediction.

At the same time, there will be a lot of tweets for each stock on the same day. These tweets often contain different people's views on the same stock, and the views are often different or even opposite. For example, "4 Major Stocks That Analysts Want You to Buy Now $GE" and "$GE technical alerts: Non-ADX 1,2,3,4 Bearish, MACD Bullish Signal Line Cross, 1,2,3 Retracement Be". So how to capture valuable information from these different comments and ultimately get the right judgment is a very difficult problem. From the perspective of studying the relationship for different tweets contained on one day of one stock, we input the deep semantic information extracted by the Transformer encoder into a capsule network, which achieves the relationship between the semantic information for stock movements prediction. Ablation experiment proves that the capsule network effectively improves the accuracy of prediction. Finally, experimental results show that our integrated model is effective.

Our contributions are as follows:

[*]Both authors contributed equally to this paper
[†]Contact Author

- For the stock movements prediction, our model captures the deep effective semantic features of tweets more effectively through the Transformer encoder, compared with neural network models such as CNNs and RNNs.
- Up to date, no work introduces the Transformer to the task of stock movements prediction except us, and our model proves the Transformer improve the performance in the task of the stock movements prediction.
- The capsule network is also first introduced to solve the problem of stock movements prediction based on social media. The results show that the capsule network is effective for this task.

2 Related Work

Stock Market Prediction: There are a series of works predicting stock movements using text information [Lavrenko *et al.*, 2000; Schumaker and Chen, 2009; Xie *et al.*, 2013; Peng and Jiang, 2015; Li *et al.*, 2017]. Pioneering works extract different types of textual features from texts, such as bags-of-words, noun phrases, named entities, and structured events. Ding et al. [2014] showed structured events from open information extraction. Yates et al. and Fader et al. [2007; 2011] achieved better performance compared to conventional features, as they capture structured relations. However, one disadvantage of structured representations for events is that they lead to increased sparsity, which potentially limits the predictive power. Ding et al. [2015] proposed to address this issue by representing structured events and using event dense embeddings. Ding et al. [2016] leveraged ground truth from the knowledge graph to enhance event embeddings. Shah et al. [2018] retrieved, extracted, and analyzed the effects of news sentiments on the stock market. Liu et al. [2018] adopted a two-level attention mechanism to quantify the importance of the words and sentences in given news and designed a novel measurement for calculating the attention weights to avoid capturing redundant information in the news title and content. In this paper, we focus on capturing the deep semantic features of the social media appeared on the same day for prediction. The results show that our model is useful.

Transformer: Vaswani et al. [2017] presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention. Tang et al. [2018] evaluated RNNs, CNNs, and Transformer on two tasks: subject-verb agreement and word sense disambiguation. Their experimental results showed that: 1) Transformer and CNNs did not outperform RNNs in modeling subject-verb agreement over long distances; 2) Transformer performed distinctly better than RNNs and CNNs on word sense disambiguation. Radford et al. [Radford *et al.*, 2018] performed three different ablation studies and analyzed the effect of the Transformer by comparing it with a single layer 2048 unit LSTM using the same framework. They observed a 5.6 average score drop when using the LSTM instead of the Transformer. In our model, we obtain more semantic features through the Transformer. It is better than other baselines based on CNN and RNN.

Capsule Network: Hinton et al. [2011] firstly introduced the concept of "capsules" to address the representational limitations of CNNs and RNNs. Capsules with transformation matrices allow networks to automatically learn part-whole relationships. Consequently, Sabour et al. [2017] proposed capsule networks that replace the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. The capsule network has shown its potential by achieving a state-of-the-art result on MNIST data. Xi et al. [2017] further tested out the application of capsule networks on CIFAR data with higher dimensionality. Hinton et al. [2018] proposed a new iterative routing procedure between capsule layers based on the EM algorithm, which achieved significantly better accuracy on the small NORB dataset. Zhang et al. [2018] generalized existing routing methods within the framework of weighted kernel density estimation. Zhao et al. [2018] investigated the performance of capsule networks in NLP tasks. In comparison, our work combines the transformer encoder with a capsule network, further improves the results of the task of stock movements prediction.

3 CapTE Model

We predict the movements of stocks on the trading day td. And we use price data crawled from Yahoo Finance to label the tweets that where 1 denotes rise and 0 denotes fall,

$$y = 1(p^c_{td} > p^c_{td-1}) \tag{1}$$

where p^c_{td} denotes the adjusted closing price which is adjusted for actions affecting stock movements, e.g. dividends and splits. Before our work, the adjusted closing price has been used for predicting stock price movements [Xie *et al.*, 2013] [Xu and Cohen, 2018].

Generally, tweets of the same stock often contain more than one item in the same trading day. For learning more valuable information from multiple tweets, we adopt the transformer to encode the texts and then get the encoded representation as the input of the capsule network. By the capsule network, we capture the relationship between different tweets appeared on the same trading day that belonging to one stock. Finally, we obtain the probability of each category as the prediction results. Figure 1 shows the architecture of the proposed model, namely Capsule network based on Transformer Encoder (CapTE).

We merge all the tweets appeared on one day of the same stock as one sentence. For each sentence s_i, we utilize the pretrained word embeddings (word2vec) to project each word token onto the d_{model}-dimensional space as the input of the Transformer encoder.

3.1 Transformer Encoder

In order to obtain deep semantic features from complex texts, we introduce the Transformer encoder. The encoder maps an input sequence of symbol representations s_i= $(x_1,...,x_n)$ to a sequence of continuous representations $(z_1,...,z_n)$. And as the paper [Vaswani *et al.*, 2017] design, the encoder contains a stack of $N = 6$ identical layers. Each layer has two sub-layers. A multi-head self-attention mechanism is the

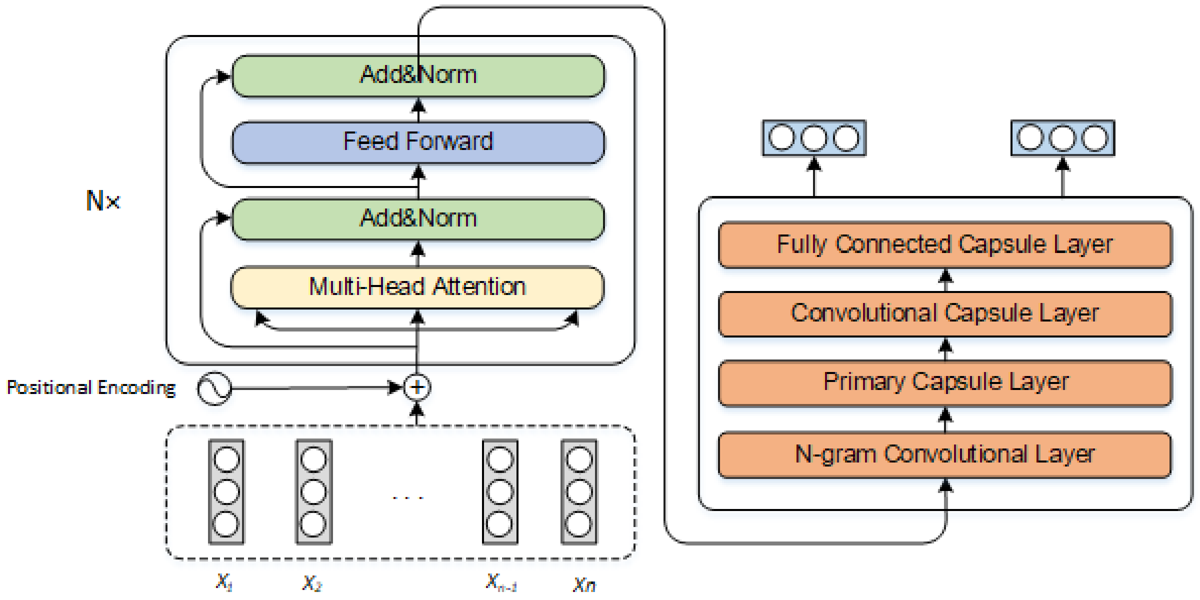


Figure 1: Transformer-Based Capsule Network

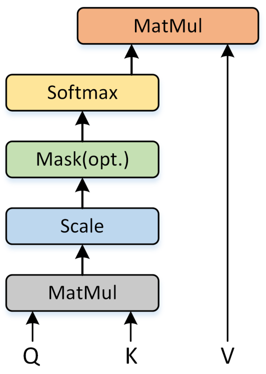


Figure 2: Scaled Dot-Product Attention.

first, while the second is a simple, position-wise fully connected feed-forward network. Around each of the two sub-layers exists a residual connection [He *et al.*, 2016], and a layer normalization follows after the two sub-layers. Hence, the output of each sub-layer is LayerNorm(x + Sublayer(x)). Sublayer(x) is the function implemented by the sub-layer itself.

Positional Encoding

The order of the words in the tweets is significant for the prediction. A reversal of the order of the words in a sentence often changes the original meaning. For example, "breakout and buying" and "buying and breakout". The former tweet means after breakout we can buy, but the later means we can buy immediately and wait for the raising of the stock price. So we adopt the "positional encodings" and add it to the input embeddings. In the end, we sum the two vectors as the final input at the bottom of the encoder. It is realized by the same dimension of the input embeddings and positional encodings. In our model, we employ sine and cosine functions of different frequencies as positional encodings:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \quad (2)$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \quad (3)$$

where pos is the position and i is the dimension. Each dimension of the positional encoding corresponds to a sinusoid.

Scaled Dot-Product Attention

After obtaining the representation summed by the word embeddings and the positional encoding, we compute the matrixes as the input of Scaled Dot-Product Attention, which consists of queries and keys of dimension d_k, and values of dimension d_v. And then, we get the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and utilize a softmax function to achieve the weights on the values. The matrix Q can be packed together into a matrix after computing the attention function on a set of queries simultaneously. The matrices K and V also apply the same method, which denotes the keys and values respectively. And the whole process is depicted in Figure 2. The matrix of outputs are as follows:

$$Attention(Q; K; V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

Multi-Head Attention

For the complex tweets, a single attention function is difficult to achieve enough information for improving the result of the prediction. So we linearly project the queries, keys, and values h times with different, which learns linear projections to d_k, d_k and d_v dimensions, respectively. On each of these queries, keys, and values, we perform the attention function in parallel and yielding d_v-dimensional output values. They are concatenated and projected, resulting in the final values.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. The function of Multi-Head Attention as follows:

$$MultiHead(Q;K;V) = Concat(head_1, ..., head_h)W^O \tag{5}$$

where W^O is the weight matrix used to multiply with the concatenated result of all the heads to produce the final output of the encoder. In our work we adopt h = 8 parallel heads. For each of the heads, we use d_k = d_v = d_{model}/h = 64.

Position-wise Feed-Forward Networks

Each of the layers in the encoder contains a fully connected feed-forward network. It is applied to each position separately and identically and consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0; xW_1 + b_1)W_2 + b_2 \tag{6}$$

while the linear transformations are the same across different positions, they use different parameters from layer to layer, just as two convolutions with kernel size 1. The dimensionality of input and output is $d_{model} = 512$.

3.2 Capsule Network

Different from the method of CNNs and RNNs, the capsule network increases the weights of similar information through its dynamic routing. The dynamic routing is proposed by Sabour et al. [2017], which displaces the pooling operation used in conventional convolution neural network. It maintains the position information for features, which is beneficial to the text representation. More importantly, this routing-by-agreement method has the ability to cluster the features into each class. People often focus on the important event, hence, the comments based on a hot event look alike. Since the price of a stock is usually decided by the significant event, we choose the capsule network to handle the information obtained from the Transformer encoder. After getting the output from the transformer encoder, we input the new representation into a capsule network to get the probability of each category. The capsule network consists of four layers: n-gram convolutional layer, primary capsule layer, convolutional capsule layer, and fully connected capsule layer.

N-gram Convolutional Layer

This layer is a standard convolutional layer which extracts n-gram features at different positions of a sentence according to various convolutional filters. In this part, the sentence is the new representation from the Transformer encoder.

Suppose $Z \in R^{L\times d_{model}}$ denotes the input representation, where L is the length of the emerged tweets of a certain day. And $z_i \in R^{d_{model}}$ is the d_{model}-dimensional vector corresponding to the ith word in the new representation. $W^a \in R^{K_1\times d_{model}}$ is the filter for the convolution operation, where K_1 is N-gram size. A filter W^a convolves with the word window $Z_{i:i+K_1-1}$ at each possible position (with stride of 1) to produce a column feature map $m^a \in R^{L-K_1+1}$, each element $m_i^a \in R$ of the feature map is produced by

$$m_i^a = f(Z_{i:i+K_1-1} \circ W^a + b_0) \tag{7}$$

where $\circ$ is element-wise multiplication, b_0 is a bias term, and f is a nonlinear activate function. That is the process by which one feature is extracted from one filter. For $a = 1, ..., B$, B filters with the same kernel size of the convolution operation. By assembling B feature maps together, we have a B-channel layer.

$$M = [m_1, m_2, ..., m_B] \in R^{(L-K_1+1)\times B} \tag{8}$$

Primary Capsule Layer

The feature maps generated from the n-gram convolutional layer are fed into this layer, piecing the instantiated parts together via another convolution. This is the first capsule layer in which the capsules replace the scalar-output feature detectors of CNNs with vector-output capsules to preserve the instantiated parameters of each feature.

By sliding over the feature map M, each filter W^b output a series of capsules $p_i \in R^d$, where d is the dimension of the capsule. These capsules comprise a channel p_i of the primary capsule layer.

$$p_i = g(W^b M_i + b_1) \tag{9}$$

where g is nonlinear squash function through the entire vector, b_1 is the capsule bias term. For all C filters, the generated capsule feature maps can be rearranged as

$$P = [p_1, p_2, ..., p_C] \in R^{(L-K_1+1)\times C\times d} \tag{10}$$

where totally $(L - K_1 + 1) \times C$ d-dimensional vectors are collected as capsules in P.

Child-Parent Relationships

Capsule network generates the capsules in the next layer using "routing-by-agreement". This process takes the place of pooling operation and usually discards the location information, which helps augment the robust of the network and cluster features for prediction. It allows the networks to automatically learn child-parent relationships. In the stock prediction task, different tweets with the same category are supposed to share a similar topic but with different viewpoints. For example, "4 Major Stocks That Analysts Want You to Buy Now $GE" and "$GE technical alerts: Non-ADX 1,2,3,4 Bearish, MACD Bullish Signal Line Cross, 1,2,3 Retracement Be". For the two comments, they talk about the same topic of technical analysis but get different views.

Between capsules i and j, a prediction vector $\hat{u_{j|i}} \in R^d$ is first calculated from the child capsule i, by multiplying a weight matrix $W^{t_1} \in R^{N\times d\times d}$, where N is the number of parent capsules in the layer above. Each corresponding vote is computed by:

$$\hat{u_{j|i}} = W_j^{t_1} u_i + \hat{b_{j|i}} \in R^d \tag{11}$$

where u_i is a child-capsule in the layer below and $\hat{b_{j|i}}$ is the capsule bias term.

The length of the capsule represents the probability that the input sample has the object capsule describes, and it is limited in range from 0 to 1 by a non-linear squashing function. The function pushes the short vectors to shrink to zero length and the long ones to one.

$$v_j = \frac{\|g_j\|^2}{1 + \|g_j\|^2} \frac{g_j}{\|g_j\|^2} \tag{12}$$

A capsule g_j is generated by the linear combination of all the prediction vectors with weights.

Dynamic Routing

The basic idea of dynamic routing is to construct a non-linear map in an iterative manner ensuring. It shows that the output of each capsule gets sent to an appropriate parent in the subsequent layer [Sabour *et al.*, 2017]:

$$\{\hat{u_{j|i}} \in R^d\}_{i=1,...H,j=1,...,N} \longmapsto \{v_j \in R^d\}_{j=1}^{N} \tag{13}$$

where v_j denotes each parent-capsule in the layer above. The parent capsules and their probabilities in the layer above are denoted as

$$v, a = Routing(\hat{u}) \tag{14}$$

where $\hat{u}$ denotes all of the child capsules in the layer below, v denotes all of the parent-capsules and their probabilities $a = |v|$.

Convolutional Capsule Layer

In this layer, each capsule is connected with a local region $K_2 \times C$ spatially in the layer below. Those capsules in the region multiple transformation matrices are to learn child-parent relationships followed by routing by agreement to produce parent capsules in the layer above. $K_2 \times C$ is the number of child capsules in a local region in the layer above. When the transformation matrices are shared across the child capsules, we get each potential parent capsule. And then, we use routing-by-agreement to produce parent capsules feature maps totally $(n - K_1 - K_2 + 2) \times D$ d-dimensional capsules in this layer. D is the number of parent capsules which the child capsules are sent to.

Fully Connected Capsule Layer

The capsules in the layer below are flattened into a list of capsules and fed into fully connected capsule layer in which capsules are multiplied by transformation matrix $W^{d_1} \in R^{G \times d \times d}$ or $W^{d_2} \in R^{H \times G \times d \times d}$ followed by routing-by-agreement to produce final capsule v_j and its probability a_j for each category. And H is the number of child capsules in the layer below, $G = 3$ is the number of categories plus an extra orphan category in this task. The orphan category helps us collect the less contributive capsules that contain too much background information. This method reduces the interference for normal categories.

Inspired by Zhao et al. [2018], we attempt to use the probability of the existence of parent capsules to iteratively amend the connection strength. The length of the vector v_j represents the probability of each relation. To increase the difference between the lengths of categories, we adopt a separate margin loss $Loss_k$ for each relation capsule k:

$$\begin{aligned} Loss_k = Y_k max(0, m^+ - \|v_k\|)^2 \\ +\lambda(1 - Y_k) max(0, \|v_k\| - m^-)^2 \end{aligned} \tag{15}$$

Data	Stocks	Days	Tweets	Words
Tweet	47	231	746,287	137,052

Table 1: Basic statistics of the dataset.

where v_k is the capsule for class k, m^+ and m^- is the top and bottom margins respectively. Y_k = 1 if the relation k is present. λ is the weight for the absent classes. In our model, m^+ = 0.9, m^- = 0.1 and λ = 0.5.

The Architectures of Capsule Network

The capsule network starts with a 1-gram ($K_1 = 1$) convolutional layer with 32 filters (B = 32) and a stride of 1 with ReLU non-linearity. All the other layers are capsule layers starting with a $B \times d$ primary capsule layer with 32 filters (C = 32), followed by a $1 \times C \times d \times d$ ($K_2 = 1$) convolutional capsule layer with 16 filters ($D = 16$) and a fully connected capsule layer in sequence.

Each capsule has 16-dimensional ($d = 16$) instantiated parameters and their length (norm) describe the probability of the existence of capsules. The capsule layers are connected by the transformation matrices, and each connection is also multiplied by a routing coefficient. It is dynamically computed by the routing of agreement mechanism. The final output with three classes ($G = 3$) in the fully connected capsule layer is obtained from the probability of each category. In this way, the capsule network learns more valuable information for the stock prediction.

4 Experiment

4.1 Datasets

We test our model on the open dataset [1]. It ranges from January 2017 to November 2017 and contains 47 stocks which have sufficient tweets from the Standard Poor's 500 list. The basic statistics of the dataset are shown in Table 1. The experimental dataset is still available until June 2019. Totally in our model and other baselines, we split the dataset with the ratio of approximately 5: 1: 1 in chronological order, which is the same as Wu et al. [2018].

4.2 Experimental Setups

In our experiment, the initial word embedding is obtained by word2vec. The dimension of word embedding is 512. We use the rise (1) and fall (0) of the stock price as the final output. The internal weights in our model are initialized by sampling from the uniform distribution and tuned in the training process. We adopt mini-batch in the training process, and the batch size is 128.

4.3 Evaluation Metrics

Following previous work for stock prediction [Xie *et al.*, 2013; Ding *et al.*, 2015; Xu and Cohen, 2018], we adopt the standard measure of accuracy and Matthews Correlation Coefficient (MCC) as evaluation metrics. With the confusion matrix which contains the number of samples classified as

[1]https://github.com/wuhuizhe/CHRNN

Model	Acc.	MCC
TSLDA	53.92	0.0561
HAN	57.14	0.0723
HCAN	58.72	0.0876
CH-RNN	59.15	0.0945
CapTE-nT	59.64	0.1073
CapTE-nC	60.12	0.1258
CapTE	**64.22**	**0.3481**

Table 2: Performance of baselines and CapTE variations in accuracy and MCC.

true positive, false positive, true negative and false negative, MCC is calculated as follows:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}} \quad (16)$$

4.4 Comparison Methods

We conduct extensive experiments to compare our model with several baselines:

- TSLDA: A generative topic model jointly learning topics and sentiments [Nguyen and Shirai, 2015].
- HAN: A state-of-the-art discriminative deep neural network with hierarchical attention [Hu *et al.*, 2018]. In our experiment, we adopt their open source code [2] to get the results.
- HCAN: A novel deep generative model jointly exploiting text and price signals [Xu and Cohen, 2018].
- CH-RNN: A novel Cross-model attention based Hybrid Recurrent Neural Network (CH-RNN) [Wu *et al.*, 2018]. And the dataset is the same as our model.
- CapTE-nT: In order to verify the effectiveness of the Transformer, we conduct experiments by utilizing our model without the Transformer Encoder.
- CapTE-nC: In order to verify the effectiveness of the Capsule network, we conduct experiments by utilizing our model without the Capsule Network.

4.5 Experimental Analysis

We test our model from two aspects. One is to predict the rise and fall based on the dataset. The results are shown in table 2. The other is a simulation test based on the transaction. The results are shown in table 3. In the end, we analyze the reasons for error based on the different model.

Comparison Methods

From table 2, we can see that on the 47 stocks CH-RNN gets the highest score in the baselines. However, on the same dataset that using the same way to split, our model obtains higher accuracy than CH-RNN, more than 5%, that shows our model capture deep semantic features more effectively.

[2]https://github.com/yumoxu/stocknet-code

Stock	CH-RNN	CapTE
AAPL	884$	901$
BAC	872$	996$
DIS	659$	869$
IBM	1092$	1768$
PFE	1025$	853$
WMT	1127$	1489$

Table 3: Profit comparison between CH-RNN and CapTE.

The CapTE-nC model gets a higher score by using the only Transformer than CH-RNN in both accuracy and MCC. It further illustrates that the Transformer captures deep semantic features compared to RNNs. At the same time, the performance of CapTE-nT is higher than the scores of CH-RNN in both the accuracy and the MCC. It further demonstrates that the capsule network obtains valuable relationship information. On the other hand, the score of CapTE-nC is higher than CapTE-nT which indicates that the capture of deep semantic features is more important for complex data such as tweets.

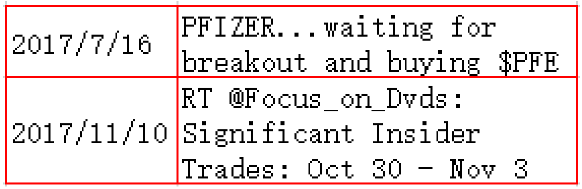
Especially, the results of CapTE-nT and CapTE-nC with the only partial model are quite different from those of the complete model. We believe that it is because the transformer encoder and capsule network complement each other in the extraction of deep semantic features. They constitute a complete system. And the function of the system performs more effectively than a single model.

Stock Trading Simulation

We simulate real stock trading by following the strategy proposed by Lavrenko et al. [2000]. If the model predicts that a stock price will raise the next day, we spend $10,000 to buy it at the opening price. And then, we hold the stock for one day. During this time, if the stock price increase 2% or more, we sell it immediately. If not, we sell the stock at the closing price. On the other hand, if the model predicts that a stock price will fall, when we can buy the stock at a price 1% lower than shorted, we buy the stock. Otherwise, we buy the stock at the closing price.

In table 3, we show the returns of six randomly selected stocks through CH-RNN and CapTE with $10,000 in 20 trading day. And the maximum return of IBM is over 17%. The results demonstrate consistently better performance, which indicates the robustness of our model.

Error Analysis



2017/7/16	PFIZER...waiting for breakout and buying $PFE
2017/11/10	RT @Focus_on_Dvds: Significant Insider Trades: Oct 30 - Nov 3

Figure 3: Examples for Error Analysis.

However, comparing with the PFE's return between the selected model, we find that CapTE is bad than CH-RNN. Hence, we compare the prediction results of CapTE with CH-RNN and analyze the cases which are wrongly predicted by CapTE but well predicted by CH-RNN.

Finally, we summarize two situations: a tweet is written to confuse the traders by the market makers. For example in Figure 3, “PFIZER...waiting for breakout and buying”, after this message, the stock price fell in the next few days. Instigating people to buy when they prepare to short and instigating people to sell when they want to bull is the main method to obtain profit by the market makers. Second, the event is just a fictive fact. For example, “Significant Insider Trades: Oct 30 - Nov 3”, and this message is apparently important to the stock trend. But it is well-known that the “Insider Trades” is unlikely to be made public. Most of the time, such news is just a rumor. For our model, it is hard to achieve the correct prediction without introducing the relevant knowledge in these conditions.

5 Conclusion

To capture the deep semantic information and structural relation for stock movements prediction task, we introduce the CapTE (Capsule network based on Transformer Encoder) model and demonstrate the reliability of our model. As shown in the results, we have no reason to doubt the importance of valuable information obtained through the Transformer. At the same time, with the aid of transformer encoder, the capsule network obtains the specific relationship between tweets that can improve the prediction accuracy of stock movements. Our model combines the advantages of the Transformer encoder and capsule network. In addition, because we introduce no financial data except texts in our model, our method has a generalization ability to the text classification tasks in the NLP field. However, our experimental dataset is only the day-level, the impact of tweets might be limited to the day when the event happens. Especially on the U.S. stock market, it allows people to trade many times on one trading day. For the task, this condition means the tweets have lost their impacts on the next day. Hence, how to predict the movements in a smaller period of time with information is the next topic we need to research.

Acknowledgments

This work is partially supported by a grant from the Natural Science Foundation of China (No.61632011, 61572102, 61702080, and 61602079), the Fundamental Research Funds for the Central Universities (No.DUT18ZD102, No.DUT19RC(4)016), the National Key Research Development Program of China (No.2018YFC0832101), the Postdoctoral Science Foundation of China (No.2018M631788, No.2018M641691), the Ministry of Education Humanities and Social Science Project (No.19YJCZH199), and the Foundation of State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China (COGOS-20190001,Intelligent Medical Question Answering based on User Profiling and Knowledge Graph).

References

[Ding *et al.*, 2014] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 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, 2014.

[Ding *et al.*, 2015] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[Ding *et al.*, 2016] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2133–2142, 2016.

[Fader *et al.*, 2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[Fama *et al.*, 1969] Eugene F Fama, Lawrence Fisher, Michael C Jensen, and Richard Roll. The adjustment of stock prices to new information. *International economic review*, 10(1):1–21, 1969.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hinton *et al.*, 2011] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

[Hinton *et al.*, 2018] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. 2018.

[Hu *et al.*, 2018] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 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, 2018.

[Lavrenko *et al.*, 2000] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, volume 2000, pages 37–44, 2000.

[Li *et al.*, 2017] Qing Li, Jun Wang, Feng Wang, Ping Li, Ling Liu, and Yuanzhu Chen. The role of social sentiment in stock markets: a view from joint effects of multiple information sources. *Multimedia Tools and Applications*, 76(10):12315–12345, 2017.

[Liu *et al.*, 2018] Qikai Liu, Xiang Cheng, Sen Su, and Shuguang Zhu. Hierarchical complementary attention network for predicting stock price movements with news. In *Proceedings of the 27th ACM International Conference* *on Information and Knowledge Management*, pages 1603–1606. ACM, 2018.

[Nguyen and Shirai, 2015] Thien Hai Nguyen and Kiyoaki Shirai. 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, 2015.

[Peng and Jiang, 2015] Yangtuo Peng and Hui Jiang. Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *arXiv preprint arXiv:1506.07220*, 2015.

[Qin *et al.*, 2017] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[Sabour *et al.*, 2017] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[Schumaker and Chen, 2009] Robert P Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12, 2009.

[Shah *et al.*, 2018] Dev Shah, Haruna Isah, and Farhana Zulkernine. Predicting the effects of news sentiments on the stock market. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4705–4708. IEEE, 2018.

[Tang *et al.*, 2018] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Wu *et al.*, 2018] Huizhe Wu, Wei Zhang, Weiwei Shen, and Jun Wang. Hybrid deep sequential modeling for social text-driven stock prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1627–1630. ACM, 2018.

[Xi *et al.*, 2017] Edgar Xi, Selina Bing, and Yang Jin. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

[Xie *et al.*, 2013] Boyi Xie, Rebecca Passonneau, Leon Wu, and Germán G Creamer. Semantic frames to predict stock price movement. In *Proceedings of the 51st annual meeting of the association for computational linguistics*, pages 873–883, 2013.

[Xu and Cohen, 2018] Yumo Xu and Shay B Cohen. 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)*, volume 1, pages 1970–1979, 2018.

[Yates *et al.*, 2007] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007.

[Zhang *et al.*, 2018] Suofei Zhang, Wei Zhao, Xiaofu Wu, and Quan Zhou. Fast dynamic routing based on weighted kernel density estimation. *arXiv preprint arXiv:1805.10807*, 2018.

[Zhao *et al.*, 2018] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*, 2018.