INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 13(5), 881-894, October 2018.

A Financial Embedded Vector Model and Its Applications to Time Series Forecasting

Y.F. Sun, M.L. Zhang, S. Chen, X.H. Shi

Yanfeng Sun^{1,2}, **Minglei Zhang**^{1,2}, **Si Chen**^{1,2}, **Xiaohu Shi**^{1,2,3}*
1. Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education
Jilin University, China
Changchun, 130012, China
2.College of Computer Science and Technology
Jilin University, China
Changchun, 130012, China
3.Zhuhai Laboratory of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education
Zhuhai College of Jilin University, China
Zhuhai, 519041, China
sunyf@jlu.edu.cn, 632987245@qq.com, 1075083128@qq.com
*Corresponding author: shixh@jlu.edu.cn

Abstract: Inspired by the embedding representation in Natural Language Processing (NLP), we develop a financial embedded vector representation model to abstract the temporal characteristics of financial time series. Original financial features are discretized firstly, and then each set of discretized features is considered as a "word" of NLP, while the whole financial time series corresponds to the "sentence" or "paragraph". Therefore the embedded vector models in NLP could be applied to the financial time series. To test the proposed model, we use Radial Basis Functions (RBF) neural networks as regression model to predict financial series by comparing the financial embedding vectors as input with the original features. Numerical results show that the prediction accuracy of the test data is improved for about 4-6 orders of magnitude, meaning that the financial embedded vector has a strong generalization ability.

Keywords: embedded vector, financial daily vector, financial weekly vector, RBF neural networks.

1 Introduction

Financial time series is nonlinear, non-stationary, and often with high noise [7]. Moreover, the financial market is a dynamic complex system which is vulnerable to various external factors [23]. Therefore, it is a challenging job to analyze the financial time series, especially for its forecasting problem. Unlike the traditional predictive methods such as the Autoregressive Integrated Moving Average (ARIMA) model, the computational intelligence-based approaches are data-driven models that do not require any specific assumptions about the distribution of the data. They are widely used in the financial market, the exchange rate market and corporate financial forecasting and other fields [6].The intelligent computing methods, such as Multilayer Perceptron(MLP) [12, 29], Radial Basis Function (RBF) neural network [1, 24, 32], Adaptive Neural-Fuzzy Inference System [8], Support Vector Regression (SVR) [5, 31], Deep Belief Network [14, 25], simulation model [9], lifecycle model [22], and Particle Swarm Optimization (PSO) [21] are often used for forecasting jobs.

An important work in the field of natural language processing (NLP) is how to represent words or documents. With the applications of the deep learning algorithms [2, 16] and the machine learning algorithms [13, 27] in the field of NLP, a number of word representation methods have been proposed. One-hot Representation [30] is simple and one of the most widely used methods, but the weaknesses are involved in high dimension of the word vector, easily leading to dimension disaster and easily resulting in the phenomenon of lexical division.

The weaknesses have been solved to a great extent of the advent of the distributed representation methods, or word embedding, which are vectors whose relative similarities correlate with semantic similarity. One of the most influential early models is Latent Semantic Analysis (LSA), developed in the context of information retrieval [10]. Latent Dirichlet Allocation (LDA) is the refinement of LSA, which is the most well-known topic model [4]. Both LSA and LDA use documents as contexts, which become computationally very expensive on large data sets. Another type of distributed representation method is neural networks based. Bengio et. al. proposed a neural network statistical language model by learning a distributed representation of words which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences [3]. While until Mikolov et. al. proposed the word2vec model, and provided open source toolkit [17, 18], word embedded vector had been widely used in different kinds of NLP applications. Word2vec models are two-layer neural networks that are trained to reconstruct linguistic contexts of words. Unlike the earlier language model, word2vec represents all the words in the corpus with a real dense vector, namely the word embedding. During the learning process, the embedding vectors are trained together with language model parameters. The most advantage of word2vec is that it could be better represented by latent semantics. Two topology frameworks, namely Continuous Bag of Words (CBOW) and Skip-gram, could be adopted in word2vec model, as well as two training techniques, Hierarchical Softmax and Negative Sampling, could be selected, respectively. Compared with other neural network models, the hidden layer is removed from word2vec, and the mapping layer no longer emphasizes the order of words. Paragraph vector is an extension of word2vec, which adds the ID of a paragraph (might be a phrase, a sentence or a document) into the inputs and outputs paragraph representation vectors together with word embedded vectors [15]. Paragraph vector includes two models: PV-DM (Distributed Memory Model of Paragraph Vectors) and PV-DBOW (Distributed Bag of Words version of Paragraph Vector).

Word embedded vector is also widely applied to other fields. In machine translation [26], word embedded vector can be used to mine the relationship between words in the two languages. After training word vectors of the two languages, it can be translated by a matrix transformation. From used sentences in image understanding [11]. Images and sentence knowledge are integrated by the neural network. Task extraction is built on semantic or knowledge relations. In the social network [19], network embedding method is widely used. DeepWalk [20] model is proposed based on word2vec. It is a new method of learning the potential representation of vertices in complex network. These potential representations encode social relations in continuous vector space and can be easily utilized by other machine learning models. Tang [28] proposed a novel embedding method-LINE, which embedded the large complex network into the low dimensional vector space. The classical stochastic gradient descent restriction is solved by using a sampling algorithm for edges. LINE is suitable for many type of information network.

In the paper, the idea of word embedded vector is introduced to abstract the temporal characteristics of financial time series. After discretizing original financial features, we consider each set of discretized features as a "word" of NLP, and the financial time series as the "paragraph". Then we apply the embedded vector models to financial series data and obtain financial embedded vectors. Next we adopt the financial embedded vectors as the input of the RBF neural network for single-step financial time series prediction.

To test the proposed financial embedded vector model, we use the daily data of S&P500 index for experiments. Compared with the RBF neural network without financial embedded vector, the numerical results have been improved greatly in the prediction accuracy.

2 Financial embedded vector model

Considering each set of discretized features of financial time series as a 'word' of NLP, we apply the embedded vector models in NLP to the financial time series. Appling word2vec and Paragraph vector, we can construct financial daily vector and financial weekly vector, respectively. In section 2.1, word2vec and Paragraph vector will be introduced briefly. And then our proposed financial embedded vector model will be stated in section 2.2.

2.1 Embedded vector model in NLP

1) Word2vec
Word2vec has two topology frameworks, namely Continuous Bag of Words (CBOW) and Skip-gram. The task of the former is to predict the word given its context, while that of latter is to predict the context given a word. On the other hand, there are two training methods to train the parameters for both of the topology frameworks. One is Hierarchical Softmax, the other is Negative Sampling. In this paper, CBOW framework and Hierarchical Softmax training method are used in our proposed method. Therefore word2vec with CBOW and Hierarchical Softmax will be described in the following. CBOW framework is a neural network model with three layers: input layer, projection layer and output layer (Fig.1). Unlike Neural Network Language Model(NNLM) [3], there are no nonlinear hidden layers in CBOW. The representation vectors of the context of a word are the inputs of the model. They are mapped to the projection layer by simply summation. The output layer represents the central word, which is expressed by a leaf node of the Huffman tree.

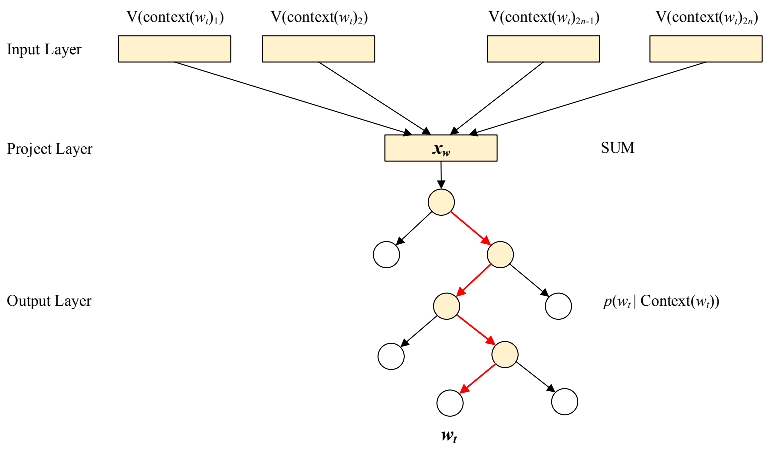


Figure 1: Diagram of CBOW model structure

Suppose the tth sample is the pair of $(Context(w_t), w_t)$,where $Context(w_t)$ is consisted of n words before and after the word w_t, $V(w)$ is the representation vector or embedded vector of word w. CBOW predict the probability of occurrence of the target word w_t by context words $Context(w_t)$. The target function is the following logarithmic likelihood function

$$L = \sum\nolimits_{w \in c} \log p(w|\text{context}(w)) \tag{1}$$

where p$(w_t|Context(w_t))$ is the probability of w_t on condition of $Context(w_t)$. The goal is to maximize the logarithmic likelihood function, which could be solved by Stochastic Gradient Ascent (SGA) [17].
2) Paragraph Vector

Based on word2vec model, Le and Mikolov added paragraph ID into the inputs and proposed Paragraph Vector model, which could learn the representation vectors for both word set and paragraph set [15]. Similar with word2vec, Paragraph vector has two frameworks, namely Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag of Words (PV-DBOW). Fig.2 gives the network structure of PV-DM, which could be considered as the CBOW framework of word2vec adding "Paragraph ID" in the input layer. The learning algorithm is the same with that of CBOW model. After being trained, the paragraph vectors can be used as features of the paragraph. There is one thing should be noticed that, it should be performed a short inference stage to get a representation vector for a new paragraph, by making gradient descended on the paragraph vector while holding the learned word embedded vectors fixed.

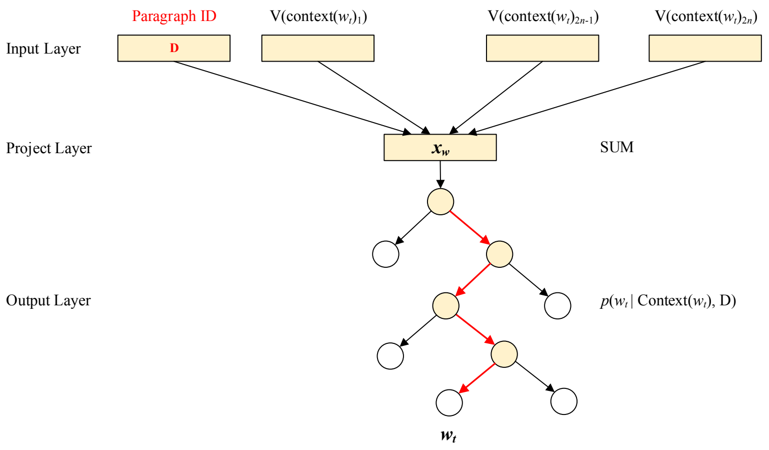


Figure 2: Diagram of PV-DM model structure

2.2 Financial embedded vector model

The key point of word2vec (or Paragraph vector) is to abstract the context relationships among different words by a neural network model and then obtain the embedding representation vectors of all the words in the dictionary. Similar with natural language, financial time series has obviously temporal characters, therefore based the embedding model in NLP, we propose a financial embedded vector model to represent the "context" relationships between different discretized feature sets of financial time series. According to word2vec and Paragraph vector models, "daily embedding vector" and "weekly embedding vector" are proposed, respectively. In the following of this section, the former will be described in detail, the latter is similar.
Assume the financial time series has m features, all of which are discretized. For example, it has 5 features and each of which is discretized into 3 sets. Then we could get the discretized feature sets from '11111' to '33333', $3^5 = 243$ sets in total. That is the "dictionary" size of financial embedding models. Firstly, we train the word2vec model to get the financial daily embedded vector by using CBOW model and Hierarchical Softmax framework. Assume the ith discretized feature set be '23213', that is to say w_i='23213', and denote the "context" set of w_i as $Context(i)$, then we will show the training procedure for the sample $(w_i, Context(i))$, which is shown in Fig.3.

The structure of Fig.3 is the same as Fig.1, each leaf node of Huffman tree represents a discretized feature set, the position of which is predetermined by its frequency in the training corpus. For example, '23213' is a leaf node of Huffman tree in Fig.3. There is a single path p^i from root node to the leaf node '23213', which includes 5 nodes and 4 branches. Each branch could be considered as a binary classification problem. Let the left branch be a positive class

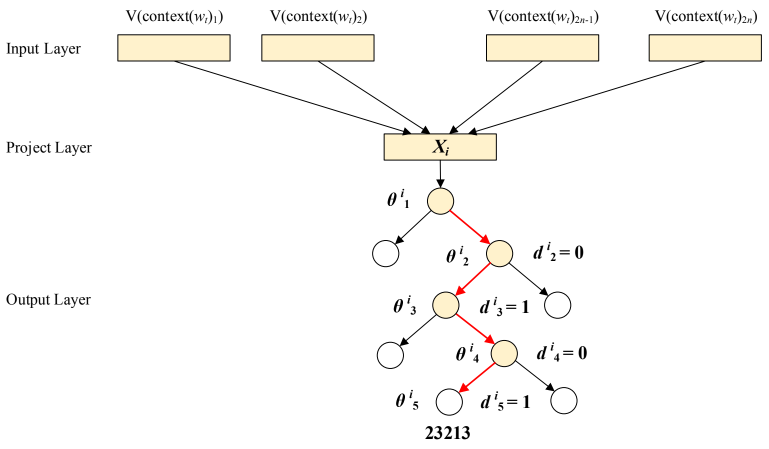


Figure 3: Diagram of CBOW model structure for the training of '23213'.

and the Huffman code be 1, while the right branch be a negative class and the Huffman code be 0. The Huffman code of w_i is the binary code of the nodes in p^i (except root node), namely that ${d_2}^i{d_3}^i{d_4}^i{d_5}^i$='0101'. It is easy to know the probability of the positive class is

$$\sigma(X_i^T\theta) = \frac{1}{1+e^{(-X_i^T\theta)}} \tag{2}$$

where θ is the undetermined non-leaf node vector, and X_i is the summation of input context vectors. The probability of the negative class is $1-\sigma\left({X_i}^T\theta\right)$. Therefore, the probability of the binary classification on the jth node of path p^i is

$$p(d_j^i|X_i,\theta_{j-1}^i) = \begin{cases} \sigma(X_i^T\theta_{j-1}^i), d_j^i = 1 \\ 1-\sigma(X_i^T\theta_{j-1}^i), d_j^i = 0 \end{cases} \tag{3}$$

where j=2,3,4,5. It can be equivalently written as

$$p(d_j^i|X_i,\theta_{j-1}^i) = [\sigma(X_i^T\theta_{j-1}^i)^{1-d_j^i}]\cdot[1-\sigma(X_i^T\theta_{j-1}^i)^{d_j^i}] \tag{4}$$

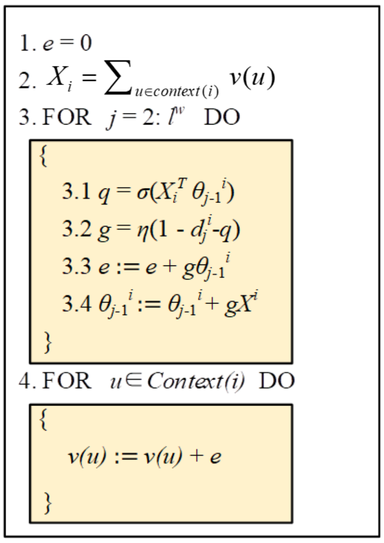
Then we could use the probability of the path p^i to represent how likely to generate w_i on condition of its context discretized feature set is $Context(i)$, which could be computed by

$$p(w_i|Context\,(i)) = \prod_{j=2}^{l^{w_i}} p(d_j^i|X_i,\theta_{j-1}^i) \tag{5}$$

where l^{w_i} is the node number of p^i , being 5 in this example. Of course, the probability of each sample pair is expected as large as possible. Then the objective functions of the whole model could be set as

$$\begin{aligned} L &= \sum_{w\in C}\log\left(p(w|Context(w))\right) \\ &= \sum_{w\in C}\log\left(\prod\nolimits_{j=2}^{l^w} p\left(d_j^w|X_i,\theta_{j-1}^i\right)\right) \\ &= \sum_{w\in C}\log\left(\prod\nolimits_{j=2}^{l^w} \sigma\left(X_i^T\theta_{j-1}^i\right)^{1-d_j^i}\left[1-\sigma^{1-d_j^i}\left(X_i^T\theta_{j-1}^i\right)\right]^{d_j^i}\right) \\ &= \sum_{w\in C}\sum\nolimits_{j=2}^{l^w}\left\{(1-d_j^i)\log\sigma\left(X_i^T\theta_{j-1}^i\right) + d_j^i\log\left[1-\sigma^{1-d_j^i}\left(X_i^T\theta_{j-1}^i\right)\right]\right\} \end{aligned} \tag{6}$$

The goal is to maximize the objective function by training parameters *V(X)* and θ. It could be solved by Stochastic Gradient Ascent (SGA), and finally the well trained V is the "daily financial embedded vector" set. For the detailed algorithm of SGA, one can refer to [17]. The pseudo codes of the method are shown in Fig.4.



```
1. $e = 0$
2. $X_i = \sum_{u \in context(i)} v(u)$
3. FOR $j = 2 : l^w$ DO
    {
        3.1 $q = \sigma(X_i^T \theta_{j-1}^i)$
        3.2 $g = \eta(1 - d_j^i - q)$
        3.3 $e := e + g\theta_{j-1}^i$
        3.4 $\theta_{j-1}^i := \theta_{j-1}^i + gX^i$
    }
4. FOR $u \in Context(i)$ DO
    {
        $v(u) := v(u) + e$
    }
```

Figure 4: Pseudo codes for SGA

Furthermore, we divide the financial series into weekly segments, and each weekly segment could be considered as a "paragraph" in NLP which consists of 5 discretized feature sets as the daily financial vector. Then by introducing the Paragraph vector model, we could get "weekly financial vector" for each week. The method is similar with that of "daily financial vector".

3 The algorithm framework

In Section 2, a financial embedded vector model is proposed, from which we could get "daily financial vector" and "weekly financial vector", respectively. And then the trained financial vectors could be set as the features for financial series analysis. Compared with the original features of financial series, the financial vectors abstract more temporal characters and are expected to get better results. In this section, we develop a financial series prediction framework based on financial embedded vectors and RBF neural network. There are three main processes, namely data discretization, financial embedded vector construction, and single step prediction based on RBF neural network, which will be described in the following of this section.

3.1 Data discretization

A financial time series is usually a set of time-dependent consecutive real numbers. In our financial embedded vector model, the financial vector at each time point is a "financial word", therefore, the real features should be discretized firstly and the feature set could be divided to a "financial vocabulary". Assume there are m real features of a financial time series, the ith feature is discretized into n_i segments, then the size of "financial vocabulary" is:

$$S = \prod_{i=1}^{m} n_i \tag{7}$$

In our method, K-means algorithm is adopted for the discretization. For all the values of the ith feature, they are clustered into n_i categories, which are labelled with the set $\{1, 2, ..., n_i\}$. Then the ith feature of series is discretized into n_i segments according to clustering results. K-means is an unsupervised clustering algorithm and often be used as discretization tool. Compared with other discretization algorithms, K-means algorithm has lower computational complexity and is advantageous for dealing with large-scale data.

3.2 Financial embedded vector construction

After data discretization, each time point of the financial series could be mapped to a "financial word" in the "financial vocabulary". And the whole financial series is a long "financial paragraph" consists of these "financial words". Take the context of a "financial word" as inputs, and the target "financial word" as output, the financial embedded vector model could be trained on CBOW framework and by SGA algorithm. Then we could obtain so called "daily financial vectors". Part of them are shown in Table1.

Table 1: Part of daily financial vectors

Discretization Results	Daily Financial Vectors
21232	[-0.0078 -0.0112 0.0211 0.0196 -0.0059 0.0211 0.0241 0.0091 0.0249 -0.0156 -0.0077 -0.0105 -0.0145 0.0141 0.0079 0.0021 -0.0110 0.0076 0.0153 0.0010]
33112	[-0.0182 -0.0200 0.0228 0.0003 -0.0129 -0.0152 -0.0154 0.0085 0.0010 0.0067 -0.0211 -0.0190 -0.0007 0.0057 0.0127 -0.0159 0.0204 0.0046 0.0056 -0.0030]
33113	[0.0077 -0.0190 -0.0205 0.0248 0.0190 0.0155 -0.0030 0.0201 -0.0025 0.0037 -0.0007 -0.0149 0.0066 -0.0067 -0.0191 0.0238 0.0219 -0.0055 -0.0011 0.0025]

We consider five days' "financial words" of a week as a weekly "financial sentence". Adding the sentence ID into the financial embedded vector model, it could be trained by stochastic gradient descent algorithm on PV-DM framework, and get "weekly financial vectors". Part of them are shown in Table2.

Table 2: Financial weekly vectors

Discretization Results (Weekly)	Weekly Financial Vectors
[33112 31232 21232 23113 31112]	[0.0429 -0.0725 -0.0783 0.2201 0.0254 -0.1170 0.0982 -0.0089 0.0813 -0.0245 -0.1699 -0.1943 -0.0463 -0.1262 0.0201 0.1168 -0.1370 0.0347 -0.1303 -0.1324]
[23113 33113 31212 33113 33113]	[0.0318 -0.0232 -0.1067 0.1555 0.0652 -0.1573 -0.0302 -0.1005 0.0855 0.0142 -0.0757 -0.1824 -0.0594 -0.0842 -0.0969 0.1414 -0.1923 0.0295 -0.0568 -0.1683]
[12313 31233 21233 23323 12132]	[0.0407 -0.0812 -0.1055 0.2603 -0.0162 -0.1914 0.1718 -0.0282 0.0654 0.0374 -0.2742 -0.1971 -0.0573 -0.1199 0.0504 0.1890 -0.1782 0.0816 -0.1934 -0.1976]

3.3 Single step prediction based on RBF neural network

To predict the financial series data, we select radial basis function (RBF) neural network as the forecasting model. RBF neural network is a particular type of Multilayer Perceptron (MLP) neural network, with input, hidden and output three layers (Fig.5) [26]. The hidden layer consists of RBF neurons, which use radial basis function as their active functions.

After getting the financial embedded vectors, we use RBF neural network to build the predicting models for one-step forecasting of the daily and weekly closing price, respectively. The original percentage data, the daily financial vector and the weekly financial vector are used as input data for comparison. Therefore we build six models for two goals with three types of inputs, which will be described in detail in section 4.2. In the hidden layer, the mostly used Gaussian function is selected as the active function. There is only one output node in all six models for our goal is one-step daily or weekly closing price predicting.

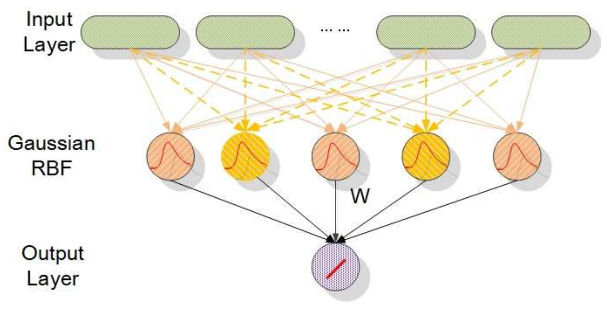


Figure 5: RBF network structure

4 Experiments

The S&P500 index has a longer trading history and enough sample which can reflect the laws of a mature securities market. The data are more stable and truthful. We collect the S&P500 index data from Yahoo Finance [33]. Date is from January 4, 1950 to November 14, 2016, a total of 16826 trading days. Every daily trading data includes five-dimensional data, namely the opening price, the highest price, the lowest price, the closing price and the volume. To eliminate the dimension influence of the five-dimensional data and improve the stability of the data, we will firstly perform the data pretreatment. Next, using RBF neural network and selecting different feature combinations, daily closing price and weekly closing price are predicted, respectively.

4.1 Data pretreatment

Firstly, Eq.8 is performed for percentage processing of the 5-dimensional data, respectively.

$$x'_t = (x_t - x_{t-1})/x_{t-1} \tag{8}$$

where x_t is the data for the day t,x_t-1 is the data for the day t-1, x'_t is the percentage of the data for the day t.

Secondly, the K-means algorithm will be adopted for the discretization of the original data. We take three clusters in the K-means algorithm. Finally, all the five features are clustered into three classes independently. That is to say, the discretized feature space size is $3^5 = 243$. Set the cluster labels as 1, 2, 3, a five-dimensional data of a time point will be discretized into $\{1, 2, 3\}^5$. Take for example, the clustering results of the five dimensions for the date 1950-01-04 are 2, 1, 2, 3, and 2, respectively. Therefore, the discretization result for the date is '21232', which is "financial word" of that day. After discretizing 5-dimensional time series data, we apply the financial embedded vector models to obtain "daily financial vectors" and "weekly financial vectors", which is described in section 3.2. For example, the daily financial vector of '21232' is shown in the first line of Table 1.

4.2 Daily closing price forecasting based on financial embedded vector

To test the performance of the obtained financial embedded vectors, they are set as the input features to predict daily closing price in this section. RBF network is applied as the forecasting model, and three types of feature combinations are arranged for comparison. Denote them as Model1, Model2, and Model3, the detail of which are described as follows:
Model1-Comparison daily model. The model output is the closing price for day t, that is, the closing price for day t will be forecasted, which is 1-dimensional data. The inputs are the

data three days before, the percentage data for day t-1, t-2, and t-3. The data for each day includes 5 percentage data of original features, namely that, the opening price, the highest price, the lowest price, the closing price and the volume. Therefore the dimension of the inputs is 15. The previous ten thousand data are used for training, the rest for testing.

Model2-Financial daily vector daily model. The model output is the same as Model1. The inputs are the financial daily vectors of three days before, namely day t-1, t-2, and t-3. In this model, we will take the dimension of each financial daily vector as 20. Then the dimension of input is 60. The division of training data and testing data is the same with Model1.

Model3-Financial daily and weekly vector daily model. The model output is the same as above two models. The inputs are the financial daily vectors for day t-1 and t-2, and the financial weekly vector for the last week of the date. In this model, we will take the dimension of each financial daily (and weekly) vector as 20, and then the dimension of input is also 60. The division of training data and testing data is the same with above two models.

RBF neural network training function is the built-in function in MATLAB. An important parameter is distribution density ('*Spread*' in the built-in function). A trade-off between over fitting and under fitting will be considered by different value of *Spread*. The value of *Spread* is set as 10 values in our experiments, which are 0.01, 0.05, 0.1, 0.3, 0.4, 0.7, 1, 5, 10, 20 and 50, respectively. Mean Square Error (MSE) is used as the evaluation index of forecasting effect, which is defined by

$$MSE = \frac{1}{n}\sum\nolimits_{i=1}^{n}(z_i - y_i)^2 \tag{9}$$

where z_i is the real value of the ith data, y_i is the forecasting value of the ith data, n is the sample number. The smaller the MSE value is, the more accurate the prediction is. The results of MSE for both training set and testing set are shown in Table 3.

From the results in Table 3, the MSE values of the three models for the training set are very similar. Model1 even gets the least average MSE among the three models, though all of them reach 10-4 magnitude. While for the testing set, the MSE values differ greatly in magnitude for different Spreads and different models. Model2 and Model3 are far better than Model1, and Model3 is superior to Model2. For example, the average MSE value of Model2 is 99.8% improved from Model1, and that of Model3 is 92.0% decreased to Model2. According to the big differences among the three models results of the testing data, it could be concluded that the financial embedded vector can improve the generalization ability of the models, especially for the combination of daily financial vector and weekly financial vector.

Fig.6 shows the comparing results of the predicted daily closing price curves of the three models. The curves in the top figure are the local enlarged of the ones in bottom figure. From the bottom figure, it is easy to find that both curves of Model2 and Model3 are obvious closer to actual data curve than that of Model1. When we focus on the enlarged curves, we could find that the performance of Model3 is better than that of Model2.

4.3 Weekly closing price forecasting based on financial embedded vector

To further examine the effect of financial embedded vectors, longer periods of data are adopted. In this section, experiments are performed to predict the weekly closing price. Similar with the above section, RBF is chosen as the forecasting model, and three types of feature combinations are arranged for comparison. They are denoted as Model4, Model5 and Model6, which are described below.

Model4-Comparison weekly model. Assume the closing day (Friday) of the target week is day t, the closing price of day t is the output of the model, which is a 1-dimensional value. The

Table 3: Prediction results of daily close price

	MSE for training				MSE for testing		
Spread	**Model1**	**Model2**	**Model3**	**Spread**	**Model1**	**Model2**	**Model3**
0.01	0	0.000029	0.000053	0.01	0.00410	0.000169	0.000198
0.05	0	0.000029	0.000054	0.05	0.10830	0.000168	0.000194
0.1	0.000007	0.000029	0.000054	0.1	287.808	0.000470	0.000191
0.3	0.000014	0.000029	0.000054	0.3	1225.83	0.000334	0.000266
0.4	0.000013	0.000029	0.000054	0.4	48.7838	0.005158	0.000216
0.7	0.000031	0.000029	0.000054	0.7	35.3886	1.243979	0.029582
1	0.000039	0.000029	0.000054	1	88.7457	2.229143	0.214848
5	0.000059	0.000030	0.000054	5	12.8574	0.003093	0.033577
10	0.000063	0.000042	0.000057	10	2.10287	0.000398	0.000277
20	0.000065	0.000053	0.000061	20	2.71368	0.000214	0.000193
50	0.000066	0.000065	0.000066	50	1.30023	0.000166	0.000161
average	**0.000032**	**0.000036**	**0.000056**	**average**	**155.058**	**0.316663**	**0.025428**

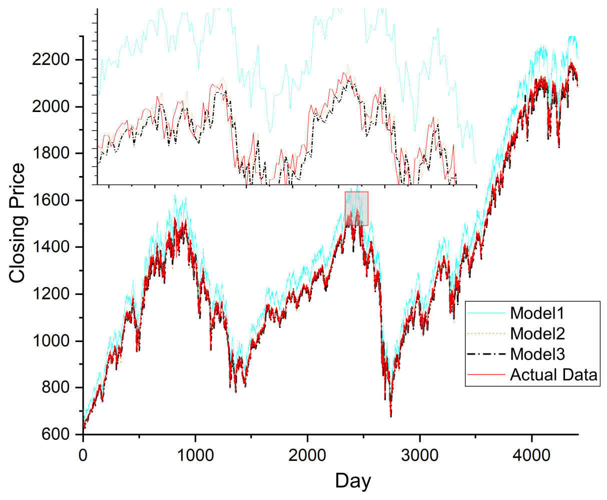


Figure 6: Forecasting and actual daily closing price in the training set

inputs are the percentage data of original features of 5 days before, namely from day t-1 to t-5. Therefore the dimension of the input is 25. Also, the previous ten thousand data are used for training, the rest for testing.

Model5-Financial daily vector weekly model. The output is the same as Model4. The inputs are the financial daily vectors of 5 days before, namely from day t-1 to t-5. For the financial daily vector is 20-dimensonal, the input dimension is 100. The division of training set and testing set is the same with Model4.

Model6-Financial weekly vector weekly model. The output is the same as Model4. The input is the weekly vector of the week before, which is 20 dimensions. The division of training set and testing set is the same with Model4.

The results of MSE for both training set and testing set are shown in Table 4. The results are very similar with those of above section. The MSE values of the three models for the training set are similar and very small. For the testing set, the MSE values differ greatly from different models. Model5 and Model6 are far better than Model4, and Model6 is superior to Model5. The average MSE value of Model5 is only 1.16 x 10-5% of that of Model4, and that of Model6 is 13.52% of that of Model5.

Table 4: Prediction results of weekly close price

MSE for training				**MSE for testing**			
Spread	**Model4**	**Model5**	**Model6**	**Spread**	**Model4**	**Model5**	**Model6**
0.01	0	0.000003	0.000004	0.01	0.00018	0.000149	0.000148
0.05	0	0.000003	0.000004	0.05	0.00140	0.000134	0.000211
0.1	0	0.000003	0.000004	0.1	0.00984	0.000126	0.000185
0.3	0	0.000003	0.000004	0.3	54.2725	0.002544	0.000455
0.4	0	0.000003	0.000004	0.4	30.7013	0.000280	0.000526
0.7	0	0.000003	0.000004	0.7	2779.44	0.001075	0.000735
1	0	0.000003	0.000004	1	211338	0.016169	0.000740
5	0.000021	0.000003	0.000045	5	6941.79	0.002522	0.000148
10	0.000034	0.000003	0.000051	10	2492.17	0.001514	0.000133
20	0.000043	0.000003	0.000052	20	756.955	0.001221	0.000133
50	0.000049	0.000010	0.000058	50	0.68581	0.000465	0.000124
Average	**0.000013**	**0.000004**	**0.000021**	**Average**	**20399.5**	**0.002382**	**0.000322**

Fig.7 shows comparing results of the predicted weekly closing price curves of the three models. The graph in the top of Fig.7 is the enlarged one of the local area of the bottom figure. From the figure, it is easy to find that both curves of Model5 and Model6 are obviously closer to the actual data curve than that of Model4, and Model6 is the best one. Also, the financial embedded vector can greatly improve the generalization ability of the models, especially for the weekly financial vector.

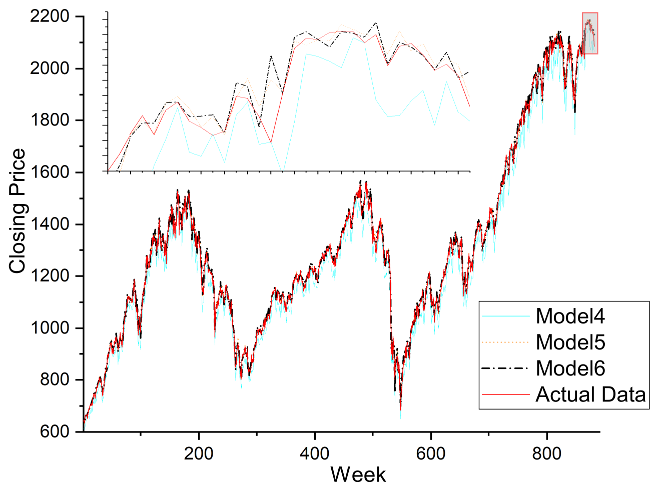


Figure 7: Forecasting and actual weekly closing price in the testing set

5 Conclusion

In the paper, inspired by the idea of embedded vector idea in NLP, we propose the financial embedded vector model to represent the financial time series, which could abstract the temporal features very well. Focused on the financial time series prediction, a framework is developed based on the financial embedded vector model and RBF neural network. To test the effectiveness of our method, it is applied to S&P500 index historical data for the daily and weekly closing price prediction. Comparing different kinds of input features, it could be concluded that the financial embedded vectors could improve the precision greatly to the original features.

Funding

The authors are grateful to the support of the National Natural Science Foundation of China (61373050), and Jilin Provincial Key Laboratory of Big Date Intelligent Computing (20180622002JC).

Bibliography

[1] Akbilgic, O.; Bozdoganm, H.; Balaban, M.E. (2014); A novel Hybrid RBF Neural Networks model as a forecaster, *Statistics and Computing*, 24(3), 365-375, 2014.

[2] Al-Ayyoub, M.; Nuseir, A.; Alsmearat, K.; Jararweh, Y.; Gupta, B. (2018); Deep learning for Arabic NLP: A survey, *Journal of Computational Science*, 26522-531, 2018.

[3] Bengio, Y.; Ducharme, J.; Vincent, P.; Janvin, C. (2003); A neural probabilistic language model, *The Journal of Machine Learning Research*, 3(2), 1137-1155, 2003.

[4] Blei, D.M.; Ng, A.Y.; Jordan, M.I. (2003); Latent dirichlet allocation, *Journal of Machine Learning Research*, 3(1), 993-1022, 2003.

[5] Cao, L. (2003); Support vector machines experts for time series forecasting, *Neurocomputing*, 51321-339, 2003.

[6] Cavalcante, R.C.; Brasileiro, R.C.; Souza, V.L.F.; Nobrega, J.P.; Oliveira, A.L.I. (2016); Computational Intelligence and Financial Markets: A Survey and Future Directions, *Expert Systems with Applications*, 55194-211, 2016.

[7] Chang, D.; Ma, Y.F.; Ding, X.L. (2016); Time Series Clustering Based on Singularity, *International Journal of Computers Communications & Control*, 12(6), 790-802, 2017.

[8] Dai, Y.; Wu, W.; Zhou, H.B.; Zhang, J.; Ma, F.Y. (2018); Numerical Simulation and Optimization of Oil Jet Lubrication for Rotorcraft Meshing Gears, *International Journal of Simulation Modelling*, 17(2), 318-326, 2018.

[9] Dai, Y.; Zhu, X.; Zhou, H.; Mao, Z.; Wu, W. (2018); Trajectory Tracking Control for Seafloor Tracked Vehicle By Adaptive Neural-Fuzzy Inference System Algorithm, *International Journal of Computers Communications & Control*, 13(4), 465-476, 2018.

[10] Deerwester, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K.; Harshman, R. (1990); Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, 41(6), 391-407, 1990.

[11] Frome, A.; Corrado, G.; Shlens, J.; Bengio, S.; J. Dean; Ranzato, M.A.; Mikolov, T. (2013); DeViSE: A Deep Visual-Semantic Embedding Model, *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 262121-2129, 2013.

[12] Jasemi, M.; Kimiagari, A.M.; Memariani, A. (2011); A modern neural network model to do stock market timing on the basis of the ancient investment technique of Japanese Candlestick, *Expert Systems with Applications*, 38(4), 3884-3890, 2011.

[13] Khan, W.; Daud, A.; Nasir, A.; Amjad, T. (2016); A survey on the state-of-the-art machine learning models in the context of NLP, *Kuwait Journal of Science*, 43(4), 95-113, 2016.

[14] Kuremoto, T.; Kimura, S.; Kobayashi, K.; Obayashi, M. (2014); Time series forecasting using a deep belief network with restricted Boltzmann machines, *Neurocomputing*, 13747-56, 2014.

[15] Le, Q.V.; Mikolov, T. (2014); Distributed Representations of Sentences and Documents, *the 31st International Conference on Machine Learning*, 32(2), 1188-1196, 2014.

[16] LeCun, Y.; Bengio, Y.; Hinton, G. (2015); Deep learning, *Nature*, 521(7553), 436-444, 2015.

[17] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. (2013); Efficient Estimation of Word Representations in Vector Space, *1st International Conference on Learning Representations (ICLR2013)*, 2013.

[18] Mikolov, T.; I. Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. (2013); Distributed Representations of Words and Phrases and their Compositionality, *the 26th International Conference on Neural Information Processing Systems*, 263111-3119, 2013.

[19] Moyano, L.G. (2017); Learning network representations, *The European Physical Journal Special Topics*, 226(3), 499-518, 2017.

[20] Perozzi, B.; Al-Rfou, R.; Skiena, S. (2014); DeepWalk: online learning of social representations, *the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701-710, 2014.

[21] Pulido, M.; Melin, P.; Castillo, O. (2014); Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican Stock Exchange, *Information Sciences*, 280188-204, 2014.

[22] Rehar, T.; Ogrizek, B.; Leber, M.; Pisnic, A.; Buchmeister, B. (2017); Product Lifecycle Forecasting Using System's Indicators, *International Journal of Simulation Modelling*, 16(1), 45-57, 2017.

[23] Scheffer, M.; Carpenter, S.R., ; Lenton, T.M.; Bascompte, J.; Brock, W.; Dakos, V.; van de Koppel, J.; van de Leemput, I.A.; Levin, S.A.; van Nes, E.H.; Pascual, M.; Vandermeer, J. (2012); Anticipating Critical Transitions, *Science*, 338(6105), 344-348, 2012.

[24] Shen, W.; X. Guo, X.; Wu, C.; Wu, D. (2011); Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, *Knowledge-Based Systems*, 24(3), 378-385, 2011.

[25] Shen, F.; Chao, J.; Zhao, J. (2015); Forecasting exchange rate using deep belief networks and conjugate gradient method, *Neurocomputing*, 167243-253, 2015.

[26] Singh, S.P.; Kumar, A.; Darbari, H.; Singh, L.; Rastogi, A.; Jain, S. (2017); Machine translation using deep learning: An overview, *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, 162-167, 2017.

[27] Sun, S.; Luo, C.; Chen, J. (2017); A review of natural language processing techniques for opinion mining systems, *Information Fusion*, 36(Supplement C), 10-25, 2017.

[28] Tang, J.; M. Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. (2015); LINE: Large-scale Information Network Embedding, *the 24th International Conference on World Wide Web*, 1067-1077, 2015.

[29] Tsai, C.-F.; Hsiao, Y.-C. (2010); Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches, *Decision Support Systems*, 50(1), 258-269, 2010.

[30] Turian, J.; Ratinov, L.; Bengio, Y. (2010); Word representations: a simple and general method for semi-supervised learning, *the 48th Annual Meeting of the Association for Computational Linguistics*, 384-394, 2010.

[31] Wang, J.; Hou, R.; Wang, C.; Shen, L. (2016); Improved v -Support vector regression model based on variable selection and brain storm optimization for stock price forecasting, *Applied Soft Computing*, 49164-178, 2016.

[32] Xiong, T.; Y. Bao, Y.; Hu, Z.; Chiong, R. (2015); Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms, *Information Sciences*, 30577-92, 2015.

[33] [Online]. http://finance.yahoo.com, Accesed on 1 Jan 2017.